



Red Hat Enterprise Linux 7

セキュリティーガイド

Red Hat Enterprise Linux 7 のセキュリティーに関するガイド

Red Hat Enterprise Linux 7 セキュリティーガイド

Red Hat Enterprise Linux 7 のセキュリティーに関するガイド

Mirek Jahoda
Red Hat Customer Content Services
mjahoda@redhat.com

Robert Krátký
Red Hat Customer Content Services

Martin Prpič
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Yoana Ruseva
Red Hat Customer Content Services

Miroslav Svoboda
Red Hat Customer Content Services

法律上の通知

Copyright © 2016 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書は、ユーザーおよび管理者が、ローカルまたはリモートからの侵入、悪用および悪意のある行為に対してワークステーションとサーバーを保護するプロセスと方法を習得する際の手助けとなります。本ガイドは Red Hat Enterprise Linux にフォーカスしたのですが、概念と手法はすべての Linux システムに適用できるものです。データセンター、勤務先および個人宅での安全なコンピューター環境の構築に必要なプランニングとツールについて詳細に説明しています。管理上の適切な知識と警戒体制およびツールを備えることで、Linux を実行しているシステムの機能をフルに活用し、かつこれらのシステムをほとんどの一般的な侵入や悪用の手法から保護することができます。

目次

第1章 セキュリティーの概要	4
1.1. コンピューターセキュリティとは	4
1.2. セキュリティーコントロール	5
1.3. 脆弱性のアセスメント	6
1.4. セキュリティーへの脅威	10
1.5. 一般的な不正使用と攻撃	13
第2章 インストール時におけるセキュリティのヒント	17
2.1. BIOS のセキュア化	17
2.2. ディスクのパーティション設定	17
2.3. 必要なパッケージの最小限のインストール	18
2.4. インストールプロセス時のネットワーク接続の制限	18
2.5. インストール後の手順	19
2.6. その他のリソース	19
第3章 システムを最新の状態に保つ	20
3.1. インストール済みソフトウェアのメンテナンス	20
3.2. RED HAT カスタマーポータルの使用	24
3.3. その他のリソース	25
第4章 ツールとサービスを使用したシステム強化	27
4.1. デスクトップのセキュリティ	27
4.2. ROOT アクセスの制御	36
4.3. サービスのセキュア化	43
4.4. ネットワークアクセスのセキュア化	63
4.5. ファイアウォールの使用	69
4.6. DNSSEC を使った DNS トラフィックのセキュア化	105
4.7. 仮想プライベートネットワーク (VPN) のセキュア化	114
4.8. OPENSSEL の使用	124
4.9. STUNNEL の使用	130
4.10. 暗号化	132
4.11. TLS 設定の強化	148
4.12. MACSEC (IEEE 802.1AE) の使用	156
第5章 システム監査	157
使用例	158
5.1. AUDIT システムのアーキテクチャー	158
5.2. AUDIT パッケージのインストール	159
5.3. AUDIT サービスの設定	160
5.4. AUDIT サービスの起動	161
5.5. AUDIT ルールの定義	161
5.6. AUDIT ログファイルについて	167
5.7. AUDIT ログファイルの検索	172
5.8. AUDIT レポートの作成	172
5.9. その他のリソース	173
第6章 コンプライアンスおよび OPENSCLAP を使った脆弱性のスキャン	175
6.1. RED HAT ENTERPRISE LINUX におけるセキュリティコンプライアンス	175
6.2. コンプライアンスポリシーの定義	175
6.3. SCAP WORKBENCH の使用	183
6.4. OSCAP の使用	192
6.5. DOCKER での OPENSCLAP の使用	199
6.6. ATOMIC での OPENSCLAP の使用	200

6.7. RED HAT SATELLITE での OPENSAP の使用	202
6.8. 実用的な使用例	202
6.9. その他のリソース	204
第7章 米連邦政府の標準および規制	205
7.1. 連邦情報処理標準 (FIPS: FEDERAL INFORMATION PROCESSING STANDARD)	205
7.2. NISPOM (NATIONAL INDUSTRIAL SECURITY PROGRAM OPERATING MANUAL)	207
7.3. PCI DSS (PAYMENT CARD INDUSTRY DATA SECURITY STANDARD)	207
7.4. セキュリティー技術導入ガイド (SECURITY TECHNICAL IMPLEMENTATION GUIDE)	207
付録A 暗号の標準	208
A.1. 同期式の暗号	208
A.2. 公開鍵暗号	208
付録B AUDIT システムのリファレンス	212
B.1. AUDIT イベントフィールド	212
B.2. AUDIT 記録のタイプ	217
付録C 改訂履歴	226

第1章 セキュリティーの概要

ビジネスの運営や個人情報の記録ではネットワーク化された強力なコンピューターへの依存度が高まっていることから、各種業界ではネットワークとコンピューターのセキュリティの実践に関心が向けられています。企業は、システム監査の適正な実施やソリューションが組織の運営要件を満たすようにするために、セキュリティ専門家の知識と技能を求めてきました。ほとんどの組織はますます動的になってきていることから、従業員は会社の重要な IT リソースにローカルまたはリモートでアクセスするようになっていきます。このため、セキュアなコンピューティング環境に対するニーズはより顕著になっています。

しかし残念なことに、多くの組織（個々のユーザーも含む）が、機能性や生産性、便利さ、使いやすさおよび予算面の懸念事項にばかり目を向けてセキュリティを付け足しと見なし、セキュリティのプロセスが見過ごされています。さらに、セキュリティの適切な実施については、無許可の侵入が発生した後にはじめて徹底されることも多くあります。多くの侵入の試みを阻止する効果的な方法は、インターネットなどの信頼できないネットワークにサイトを接続する前に、適切な措置を講じることです。



注記

本書では、**/lib** ディレクトリーにあるファイルにも言及しています。64 ビットのシステムを使用している場合は、それらのファイルの一部は **/lib64** にある可能性があります。

1.1. コンピューターセキュリティとは

コンピューターセキュリティは、コンピューティングと情報処理の幅広い領域で使用される一般的用語です。日常業務や重要な情報へのアクセスにおいてコンピューターシステムとネットワークに依存する業界は、企業データを総体的資産の重要な部分であると見なしています。総保有コスト (Total Cost of Ownership: TCO) や サービスの品質 (Quality of Service: QoS) などのいくつかの用語および評価指標は日常的なビジネス用語として用いられるようになっていますが、これらの評価指標を用いて、各種の業界はプランニングおよびプロセス管理コストの一環としてデータ保全性や可用性などを算出しています。電子商取引などを行う業界では、データの可用性と信頼性がビジネスの成否を決める可能性があります。

1.1.1. セキュリティーの標準化

すべての業界の企業は、米国医師会 (AMA: American Medical Association) や米国電気電子学会 (IEEE: Institute of Electrical and Electronics Engineers) などの標準化推進団体によって作成される規制やルールに従っています。情報セキュリティにも同じことが言えます。多くのセキュリティコンサルタントやベンダーは機密性 (Confidentiality)、保全性 (Integrity)、可用性 (Availability) の頭文字をとった CIA として知られる標準セキュリティモデルを採用しています。この 3 階層モデルは、機密情報のリスク評価やセキュリティ方針の確立において一般的に採用されているモデルです。以下でこの CIA モデルについて説明します。

- 機密性 — 機密情報は、事前に定義された個人に対してのみ利用可能とする必要があります。情報の許可されていない送信や使用は、制限する必要があります。例えば、情報に機密性があれば、権限のない個人が ID 盗難やクレジット詐欺などの悪意のある目的で顧客情報や財務情報を入手できません。
- 保全性 — 情報は、不完全または不正確になるように改ざんすべきではありません。承認されていないユーザーは、機密情報を変更したり破壊する機能を使用できないように制限される必要があります。
- 可用性 — 情報は、認証されたユーザーが必要な時にいつでもアクセスする必要があります。可用性は、情報が合意された頻度とタイミングで入手できることを保証します。これは、パー

センテージで測定されることが多く、ネットワークサービスプロバイダーやその企業顧客が使用するサービスレベルアグリーメント (SLA) で正式に合意されます。

1.2. セキュリティーコントロール

コンピューターセキュリティは多くの場合、以下の 3 つの異なるマスターカテゴリーに分類され、一般には **コントロール**と呼ばれています。

- 物理的
- 技術的
- 管理的

これら 3 つの大まかなカテゴリーは、セキュリティの適切な実施における主な目的を定義するものです。これらのコントロールには、コントロールおよびそれらの実装方法を詳細化するサブカテゴリーがあります。

1.2.1. 物理的コントロール

物理的コントロールは、機密資料への認証されていないアクセスの抑止または防止のために、明確な構造でセキュリティ対策を実施することです。物理的コントロールの例は以下の通りです。

- 有線監視カメラ
- 動作/温度感知アラームシステム
- 警備員
- 写真付き身分証明書
- 施錠された、デッドボルト付きのスチールドア
- バイオメトリクス (指紋、声、顔、虹彩、筆跡、および本人確認を行うためのその他の自動認識方法が含まれます)

1.2.2. 技術的コントロール

技術的コントロールでは、物理的な構造物やネットワークにおける機密データのアクセスや使用を制御するための基盤となる技術を使用します。技術的コントロールは広い範囲に及び、以下のような技術も含まれます。

- 暗号化
- スマートカード
- ネットワーク認証
- アクセス制御リスト (ACL: Access control lists)
- ファイル完全性監査ソフトウェア

1.2.3. 管理者コントロール

管理的コントロールは、セキュリティの人的要素を定義します。これらは組織内のあらゆるレベルの人員に関連するもので、次のような手段によって、誰がどのリソースや情報にアクセスするかを決定します。

- トレーニングおよび認識の向上
- 災害準備および復旧計画
- 人員採用と分離の戦略
- 人員登録とアカウンティング

1.3. 脆弱性のアセスメント

時間、リソースおよびやる気のある攻撃者は、ほとんどすべてのシステムに侵入することができます。現在利用できるすべてのセキュリティ手順と技術を駆使しても、すべてのシステムを侵入から完全に保護できる訳ではありません。ルーターはインターネットへのセキュアなゲートウェイの提供に役立ちます。ファイアウォールはネットワークの境界を保護します。仮想プライベートネットワーク (VPN) は、暗号化されたストリームにおいてデータを安全に通過させます。侵入検知システムは悪意のある活動について警告します。しかし、これらの技術が成功するかどうかは、以下を含む数多くの要因によって決まります。

- 技術の設定、監視および保守を行うスタッフの専門知識
- サービスとカーネルのパッチおよび更新を迅速かつ効率的に行う能力
- ネットワーク上での警戒を常に怠らない担当者の能力

データシステムと各種技術が動的であることを考えると、企業リソースのセキュア化は極めて複雑なタスクになり得ます。この複雑さゆえに、使用するすべてのシステムについての専門家リソースを見つけることは多くの場合、困難になります。高レベルの情報セキュリティの多くの分野に精通している人材を確保することはできても、複数分野における専門家スタッフを保持することは容易ではありません。これは主に、情報セキュリティの各専門分野では継続的な注意とフォーカスが必要とされるためです。情報セキュリティは常に変化しています。

脆弱性アセスメントは、お使いのネットワークとシステムのセキュリティについての内部監査です。このアセスメントの結果として、ネットワークの機密性、完全性および可用性の状態が明らかになります（「[セキュリティの標準化](#)」に詳述）。通常、脆弱性アセスメントは、対象システムとリソースに関する重要なデータを収集する調査フェーズから開始されます。この後にはシステム準備フェーズが続きます。基本的にこのフェーズでは、対象を絞り、それが持つすべての既知の脆弱性を検査します。準備フェーズの後には報告フェーズが続きます。ここでは、調査結果が高中低のカテゴリに分類され、対象のセキュリティを向上させる（または脆弱性のリスクを軽減する）方法が話し合われます。

自宅の脆弱性アセスメントを実施することを想定してみましょう。まずは自宅のドアが閉じられており、かつ施錠されていることを確認するために各ドアの点検を行うことでしょう。また、すべての窓が完全に閉じられていて掛け金が締められていることもチェックするでしょう。これと同じ概念がシステムやネットワーク、および電子データにも適用されます。悪意のあるユーザーはデータを盗み、これを破壊します。悪意のあるユーザーが使用するツールや考え方および動機に注目すると、彼らの行動にすばやく反応することが可能になります。

1.3.1. アセスメントとテストの定義

脆弱性アセスメントは、*外部からの視点*と*内部からの視点*の2種類に分類できます。

外部からの視点で脆弱性アセスメントを実施する場合、システムに外部から攻撃を試みます。会社の外部に立つことで、クラッカーの視点を得ることができます。一般にルーティング可能な IP アドレスや

DMZにあるシステムやファイアウォールの外部インターフェースなど、クラッカーが目をつけるものに着目します。DMZは「非武装地帯 (demilitarized zone)」を表し、企業のプライベート LAN などの信頼できる内部ネットワークと公的なインターネットなどの信頼できない外部ネットワークの間にあるコンピューターまたは小さなサブネットワークに相当します。通常、DMZには Web (HTTP) サーバー、FTP サーバー、SMTP (e-mail) サーバーおよび DNS サーバーなど、インターネットのトラフィックにアクセスできるデバイスが含まれます。

内部からの視点で脆弱性アセスメントを実施する場合、実行者は内部関係者であり、信頼されるステータスにあることから、有利な位置に立ちます。内部からの視点は、実行者やその同僚がシステムにログインした時点で得られるものです。プリントサーバーやファイルサーバー、データベースおよびその他のリソースを見ることができます。

これら 2 種類の脆弱性アセスメントには大きな違いがあります。会社の内部にいる場合は、部外者にはない多くの特権が与えられます。多くの組織では、侵入者を締め出すようにセキュリティーが構成されています。しかし、組織内の各分野に対しては、ほとんどセキュリティー対策が取られていません (部門内ファイアウォール、ユーザーレベルのアクセス制御および内部リソースに対する認証手順など)。また、一般的にほとんどのシステムは社内にあるので、内部からの方がより多くのリソースを確認できます。いったん社外に移動すると、ステータスは信頼されない状態になります。外部から利用できるシステムやリソースは、通常は非常に限られたものになります。

脆弱性アセスメントと侵入テストの違いを考えてみましょう。脆弱性テストを侵入テストの第一歩と捉えてください。アセスメントで得られる情報は、その後のテストで使用されます。アセスメントは抜け穴や潜在的な脆弱性を検査する目的で行われるのに対し、侵入テストでは調査結果を実際に使用する試みがなされます。

ネットワークインフラストラクチャーのアセスメントは動的なプロセスです。セキュリティー (情報セキュリティーと物理的なセキュリティー) は動的なものです。アセスメントを実施することで概要が明らかになり、誤検出 (False positives) と検出漏れ (False negatives) が示される場合があります。誤検出では、実際には存在しない脆弱性をツールが検出します。検出漏れでは、実際の脆弱性が除外されてしまいます。

セキュリティー管理者の力量は、使用するツールとその管理者が有する知識で決まります。現在使用できるアセスメントツールのいずれかを選び、それらをシステムに対して実行すると、ほぼ間違いなくいくつかの誤検出が見つかります。プログラム障害かユーザーエラーかに関わらず、結果は同じです。ツールは誤検出することもあれば、さらに悪い場合は、検出漏れをすることもあります。

脆弱性アセスメントと侵入テストの違いが定義されたところで、新たなベストプラクティスの一環として侵入テストを実施する前に、アセスメントの結果を注意深く確認、検討してみましょう。



警告

実稼働システム上で脆弱性を悪用する試みを行わないでください。システムとネットワークの生産性と効率に悪影響を与える可能性があります。

脆弱性アセスメントの実施には、以下のような利点があります。

- 情報セキュリティーに事前にフォーカスできる
- クラッカーが発見する前に潜在的な不正使用を発見できる
- システムを最新の状態に維持し、パッチを適用できる

- スタッフの成長と専門知識の開発を促す
- 経済的な損失とマイナスの評判を緩和する

1.3.2. 方法論の確立

脆弱性アセスメントの方法論を確立すると、脆弱性アセスメント用のツール選択に役立ちます。残念ながら、現時点では事前定義の方法論や業界で承認された方法論はありませんが、一般常識やベストプラクティスを適切なガイドとして活用することができます。

ターゲットは何か？ 1つのサーバーまたはネットワーク全体、およびネットワーク内のすべてが含まれるのか？ 会社の外部にいるのか、それとも内部にいるのか？ これらの質問に対する答えは、ツールの選択のみならず、ツールの使用方法を決定する際にも役立ちます。

方法論の確立についての詳細は、以下の Web サイトを参照してください。

- <https://www.owasp.org/> — 『The Open Web Application Security Project』

1.3.3. 脆弱性アセスメントのツール

アセスメントは、情報収集ツールを使用することから始まります。ネットワーク全体を評価する際は、最初にレイアウトを描いて実行されているホストを把握します。ホストの場所を確認したら、それぞれのホストを個別に検査します。これらのホストにフォーカスするには別のツールセットが必要になります。どのツールを使用すべきかを知っておくことは、脆弱性の発見において最も重要なステップである可能性があります。

日常生活のあらゆる状況と同様に、同じジョブを実行できる異なるツールは数多くあります。この概念は脆弱性アセスメントの実施にも当てはまります。ツールには、オペレーティングシステムやアプリケーションに固有のものや、(使用されるプロトコルに基づいて) ネットワークに固有のツールもあります。ツールには無料のものもあれば、そうでないものもあり、直感的で使いやすいツールもあれば、難解で適切に文書化されていないものの、他のツールにはない機能を持つツールもあります。

適切なツールを見つけることは困難なタスクとなる場合もあり、最終的には経験が重要になります。可能であれば、実験ラボをセットアップし、各ツールの長所と短所を発見できるようにできる限り多くのツールをテストするようにします。ツールの **README** ファイルや **man** ページも確認してください。さらに、インターネットからツールについての記事やステップバイステップのガイドまたはツール固有のメーリングリストなどの詳細情報を入手してください。

以下で説明するツールは、利用可能なツールのごくわずかなサンプルです。

1.3.3.1. Nmap を使用したホストのスキャン

Nmap はネットワークのレイアウトを決定するためによく使用されるツールです。**Nmap** は長年にわたって利用されており、情報収集を行う際におそらく最もよく使われるツールです。各種のオプションと使用法を詳述した優れたマニュアルページが含まれています。管理者は、ネットワーク上で **Nmap** を使用し、ホストシステムやそれらのシステム上で開かれているポートを見つけることができます。

Nmap は、脆弱性アセスメントにおける最初の有効なステップです。ネットワーク内にあるすべてのホストのマッピングができるほか、**Nmap** が特定ホスト上で実行中のオペレーティングシステムを特定できるようにするオプションを渡すこともできます。**Nmap** はセキュアなサービスの使用と不必要なサービスの停止についての方針を定める際の優れた土台を提供します。

Nmap をインストールするには、**root** で **yum install nmap** コマンドを実行します。

1.3.3.1.1. Nmap の使用

Nmap を実行するには、シェルプロンプトに **nmap** コマンドを入力し、その後にスキャン対象のマシンのホスト名または **IP** アドレスを入力します。

```
nmap <hostname>
```

例えば、ホスト名が **foo.example.com** のマシンをスキャンするには、シェルプロンプトに以下を入力します。

```
~]$ nmap foo.example.com
```

基本的なスキャン (ホストの位置や他のネットワーク条件によって数分の時間がかかる場合があります) の結果は以下のようになります。

```
Interesting ports on foo.example.com:
Not shown: 1710 filtered ports
PORT      STATE  SERVICE
22/tcp    open   ssh
53/tcp    open   domain
80/tcp    open   http
113/tcp   closed auth
```

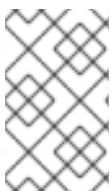
Nmap はリッスンしているまたはサービスを待機している最も一般的なネットワーク通信ポートをテストします。テストから得られる情報は、不必要または未使用のサービスの終了を希望している管理者にとって役に立つものです。

Nmap の使用方法に関する詳細情報は、以下の URL にある公式ホームページを参照してください。

<http://www.insecure.org/>

1.3.3.2. Nessus

Nessus は完全サービス型のセキュリティースキャナーです。**Nessus** のプラグインアーキテクチャを使うと、ユーザーは使用しているシステムやネットワーク用にスキャナーをカスタマイズすることができます。他のスキャナーと同様に、**Nessus** の性能はベースとなる署名データベースに左右されます。幸いなことに **Nessus** の更新は頻繁に行われており、完全なレポート作成機能やホストスキャンおよびリアルタイムの脆弱性検索機能があります。しかし、**Nessus** のように頻繁に更新される強力なツールであっても、誤検出や検出漏れの可能性があることに注意してください。



注記

Nessus クライアントおよびサーバーソフトウェアを使用するには、サブスクリプションが必要です。本ガイドでは、このアプリケーションに関心のあるユーザー向けに参考情報として説明しています。

Nessus に関する詳細は、以下の URL にある公式 Web サイトを参照してください。

<http://www.nessus.org/>

1.3.3.3. OpenVAS

OpenVAS (*Open Vulnerability Assessment System*) は、脆弱性および包括的な脆弱性管理のスキャンに使用可能なツールおよびサービスのセットです。**OpenVAS** フレームワークでは、ソリューションの様々なコンポーネントを制御するウェブベース、デスクトップ、およびコマンドラインツール

が多く提供されます。**OpenVAS** の中心となる機能は、セキュリティスキャナーが提供します。これは、毎日更新される 3 万 3 千以上の Network Vulnerability Tests (NVT) を活用します。**Nessus** (「[Nessus](#)」を参照) とは異なり、**OpenVAS** はサブスクリプションが必要ありません。

OpenVAS に関する詳細は、以下の URL にある公式 Web サイトを参照してください。

<http://www.openvas.org/>

1.3.3.4. Nikto

Nikto は、優れた 共通ゲートウェイインターフェース (CGI) のスクリプトスキャナーです。**Nikto** は、CGI の脆弱性をチェックするだけでなく、侵入検知システムを逃れるために回避的な方法でチェックを行います。**Nikto** には詳細の資料が同梱されており、プログラムの実行前に注意して確認してください。**Nikto** は、CGI スクリプトに対応する Web サーバーのセキュリティをチェックする優れたリソースになります。

Nikto の詳細については、以下の URL を参照してください。

<http://cirt.net/nikto2>

1.4. セキュリティへの脅威

1.4.1. ネットワークセキュリティへの脅威

ネットワークの以下の要素を設定する際に不適当なプラクティスが行われると、攻撃のリスクが増大します。

セキュリティが十分ではないアーキテクチャー

間違った構成のネットワークは、未承認ユーザーの主要のエントリーポイントになります。信頼ベースのオープンなローカルネットワークを安全性の非常に低いインターネットに対して無防備な状態にしておくことは、犯罪の多発地区でドアを半開きにしておくようなものです。すぐに何かが起きることはないかもしれませんが、**いずれ誰かがこの機会を悪用するでしょう。**

ブロードキャストネットワーク

システム管理者は、セキュリティ計画においてネットワーキングハードウェアの重要性を見落としがちです。ハブやルーターなどの単純なハードウェアは、ブロードキャストやノンスイッチの仕組みに基づいています。つまり、あるノードがネットワークを介して受信ノードにデータを送信するときは常に、受信ノードが受信してデータを処理するまで、ハブやルーターはデータパケットのブロードキャストを送信します。この方式は、外部侵入者やローカルホスト上の認証されていないユーザーが仕掛けるアドレス解決プロトコル (ARP) およびメディアアクセスコントロール (MAC) アドレスの偽装に対して最も脆弱です。

集中化サーバー

ネットワーキングのもうひとつの落とし穴は、集中化されたコンピューティングの使用にあります。多くの企業では、一般的なコスト削減手段として、すべてのサービスを 1 台の強力なマシンに統合しています。集中化は、複数サーバーの設定よりも管理がより簡単な上、コストを大幅に削減できるので便利な手段と言えます。しかし、集中化されたサーバーはネットワークにおける単一障害点となるため、集中化サーバーが攻撃されると、ネットワークは完全に使えなくなるか、またはデータの不正操作や盗難が起きやすくなる可能性があります。こうした状況では、1 つの集中化サーバーが侵入口となり、ネットワーク全体へのアクセスを許してしまうことになります。

1.4.2. サーバーセキュリティへの脅威

サーバーには組織の重要情報が数多く含まれることが多いため、サーバーのセキュリティはネットワークのセキュリティと同様に重要です。サーバーが攻撃されると、クラッカーがすべてのコンテン

ツを意のままに盗んだり、不正に操作したりできるようになることがあります。以下のセクションでは、主要な問題のいくつかを詳述します。

未使用のサービスと開かれたポート

Red Hat Enterprise Linux 7 を完全にインストールすると、1000 以上のアプリケーションとライブラリのパッケージが含まれます。しかし、ほとんどのサーバー管理者はディストリビューションに含まれるすべての個別パッケージをインストールすることはありません。その代わりに、複数のサーバーアプリケーションを含むパッケージのベースインストールを行います。インストールするパッケージ数を制限する理由および追加するリソースについての詳細は、「[必要なパッケージの最小限のインストール](#)」を参照してください。

システム管理者は、どのアプリケーションがインストールに含まれるかを気にせずにオペレーティングシステムをインストールしてしまうことがよくあります。必要のないパッケージがインストールされ、デフォルト設定でオンにしてしまうと、これが問題になる場合があります。これにより、管理者の気付かないところで、Telnet や DHCP、または DNS などの不要なサービスがサーバーやワークステーションで実行され、その結果、サーバーへの不要なトラフィックが発生したり、クラッカーにシステムへのパスが提供される可能性があります。ポートを閉じる方法や未使用のサービスを無効にする方法についての詳細は、「[サービスのセキュア化](#)」を参照してください。

パッチが適用されないサービス

デフォルトのインストールに含まれるほとんどのサーバーアプリケーションは、ソフトウェアの細部まで徹底的にテストされており、堅牢な作りになっています。何年も実稼働環境で使用される中で、それらのコードは入念に改良され、数多くのバグの発見や修正が行われるものです。

しかし、完璧なソフトウェアというものはなく、常に改良の余地があるものです。さらに、比較的に新しいソフトウェアは実稼働環境に最近導入されたか、または他のサーバーソフトウェアほど普及していない場合が多くあり、厳密なテストが期待通りに行われていない状況も多く見受けられます。

開発者やシステム管理者は、サーバーアプリケーションで悪用される可能性のあるバグを発見することが多くあり、関連する情報を Bugtraq メーリングリスト (<http://www.securityfocus.com>) や Computer Emergency Response Team (CERT) Web サイト (<http://www.cert.org>) などの、バグ追跡やセキュリティ関連の Web サイトに公開します。これらはセキュリティの脆弱性についてコミュニティに警告する効果的な方法ではありますが、システムに速やかにパッチを当てるかどうかは個々のシステム管理者次第となります。クラッカーも同じ脆弱性トラッキングサービスにアクセスし、可能な場合にはいつでもパッチが適用されていないシステムをクラッキングするために関連情報を利用できることを考慮すると、速やかな対応がとりわけ重要になります。優れたシステム管理には、よりセキュアなコンピューティング環境を確保するために、警戒を怠らず、バグ追跡を絶えず行い、適切なシステム保守を実行することが求められます。

システムを最新状態に維持する方法についての詳細は、[3章システムを最新の状態に保つ](#)を参照してください。

管理における不注意

システムにパッチを当てようとしない管理者は、サーバーセキュリティへの最大の脅威の 1 つとなります。*SysAdmin, Audit, Network, Security Institute (SANS)* によると、コンピューターのセキュリティ脆弱性の主な原因は、「訓練を受けていない人にセキュリティの保守を任せ、保守を行うために必要な訓練や時間を与えないこと」にあります。これは、自信過剰な管理者やモチベーションの高い管理者だけではなく、経験の少ない管理者にも起こり得ます。

サーバーやワークステーションにパッチを当てない管理者のほかに、システムのカーネルやネットワーク通信からのログメッセージを見落とす管理者もいます。またよく起こるエラーとして、サービスのデフォルトパスワードやキーを変更しないまま放置しておくことがあります。例えば、データベースにはデフォルトの管理パスワードが設定されているものがありますが、この設定では、システム管理者がインストール後すぐにデフォルトパスワードを変更することをデータベース開発者が想定しています。しかし、データベース管理者がこのパスワードの変更を忘れると、経験の浅いクラッカーでさえ、周知のデフォルトパスワードを使ってデータベースの管理者権限を得ることができます。これらは、管理者の不注意がサーバーを危険にさらすことになる数例に過ぎません。

本質的に安全ではないサービス

どんなに注意深い組織であっても、選択するネットワークサービスが本質的に安全でない限り、攻撃を受けやすくなります。例えば、多くのサービスは、信頼できるネットワークで使用されるとの仮定に基づいて開発されますが、これらのサービスが(本質的に信頼できない)インターネット上で利用可能になる時点で、この仮定は成立しなくなります。

安全ではないネットワークサービスのカテゴリの 1 つに、暗号化されていないユーザー名とパスワードを認証時に要求するサービスがあります。Telnet や FTP がこの種のサービスです。パケット盗聴ソフトウェアがリモートユーザーとこのようなサービス間のトラフィックをモニタリングしていれば、ユーザー名とパスワードは簡単に傍受される可能性があります。

また、基本的にこのようなサービスはセキュリティ業界で*中間者攻撃*と呼ばれる攻撃の餌食になりやすくなります。この種の攻撃では、クラッカーは、ネットワーク上のクラッキングされたネームサーバーをだまし、目標のサービスではなくクラッカーのマシンにポイントすることで、ネットワークトラフィックをリダイレクトします。誰かがサーバーへリモートセッションを開くと、攻撃者のマシンがリモートサービスと無防備なユーザー間に静かに存在する目に見えないパイプとして機能し、この間を流れる情報を取り込みます。このようにして、クラッカーはサーバーやユーザーに気付かれることなく、管理パスワードや生データを収集できるようになってしまいます。

安全ではないサービスの別のカテゴリは、NFS または NIS などのネットワークファイルシステムと情報サービスです。これらは、LAN 利用を目的として開発されましたが、残念なことに(リモートユーザー用の) WAN も対象に含まれるように拡張されました。NFS では、クラッカーによる NFS 共有のマウントやそこに格納されているものへのアクセスを防ぐ認証やセキュリティの仕組みがデフォルトで設定されていません。NIS も、プレーンテキスト ASCII または DBM (ASCII から派生) データベースの中に、パスワードやファイルパーミッションなどの、ネットワーク上のすべてのコンピューターに知らせる必要のある重要な情報を保持しています。クラッカーがこのデータベースのアクセス権を取得すると、管理者のアカウントを含む、ネットワーク上のすべてのユーザーアカウントにアクセスできるようになってしまいます。

Red Hat Enterprise Linux 7 は、上記のサービスをデフォルト時にすべて無効にしてリリースされます。しかし、管理者はこれらのサービスの使用を迫られる場合も多くあるため、注意して設定することが重要になります。安全な方法でサービスをセットアップする詳細情報は、「[サービスのセキュア化](#)」を参照してください。

1.4.3. ワークステーションおよび家庭用 PC のセキュリティに対する脅威

ワークステーションや家庭用 PC はネットワークやサーバーほど攻撃にさらされることはないかもしれませんが、クレジットカード情報のような機密データが含まれるのでシステムクラッカーの標的になります。ワークステーションは知らぬ間に攻撃者によって「スレーブ」マシンとして引き入れられ、一連の攻撃で攻撃者に使用される可能性もあります。このため、ユーザーはワークステーションの脆弱性を理解しておく、オペレーティングシステムの再インストールや、深刻な場合はデータ盗難からの回復といった問題から免れることができます。

不適切なパスワード

不適切なパスワードを設定すると、システムにアクセスする最も簡単な方法を攻撃者に提供してしまいます。パスワードを作成する際によくある落とし穴を避ける方法については、「[パスワードのセキュリティ](#)」を参照してください。

脆弱なクライアントアプリケーション

管理者がサーバーに十分な安全対策を施し、パッチを当てている場合でも、リモートユーザーによるアクセスが安全であるわけではありません。例えば、サーバーが公開ネットワーク上で Telnet や FTP サービスを提供している場合、攻撃者はネットワークを通過するプレーンテキストのユーザー名とパスワードを取り込み、アカウント情報を使用してリモートユーザーのワークステーションにアクセスすることが可能です。

SSH などのセキュアなプロトコルを使用している場合であっても、クライアントアプリケーションを

定期的に更新していないと、リモートユーザーは特定の攻撃を受けやすくなる可能性があります。例えば、v.1 SSH クライアントは、悪意のある SSH サーバーからの X 転送攻撃に対して脆弱です。クライアントがサーバーに接続すると、攻撃者はネットワーク上でクライアントによるキー入力やマウス操作をひそかに収集できます。この問題は v.2 SSH プロトコルで修正されましたが、ユーザーはどのアプリケーションにこのような脆弱性があるかを追跡し、必要に応じてアプリケーションを更新する必要があります。

「デスクトップのセキュリティ」では、管理者とホームユーザーがコンピューターのワークステーションの脆弱性を限定するために取るべき手順をより詳細に説明しています。

1.5. 一般的な不正使用と攻撃

表1.1「一般的な不正使用」では、侵入者が組織のネットワークリソースにアクセスするために使用する最も一般的な不正使用とエントリーポイントのいくつかについて詳しく説明しています。これらの一般的な不正使用については、それらがどのように実行され、管理者がそれらの攻撃からネットワークをどのように適切に保護できるかを理解していることが重要になります。

表1.1 一般的な不正使用

不正使用	説明	注意事項
空またはデフォルトのパスワード	管理パスワードを空白のままにしたり、製品ベンダーが設定したデフォルトパスワードをそのまま使用します。ルーターやファイアウォールなどのハードウェアで最もよく見られますが、Linux で実行するサービスにはデフォルトの管理者パスワードが入っているものがあります（ただし Red Hat Enterprise Linux 7 はパスワードなしで出荷されます）。	ルーター、ファイアウォール、VPN および Network Attached Storage (NAS) アプライアンスなどのネットワークハードウェアに一般的に関連するものです。 数多くのレガシーオペレーティングシステム、特にサービスをバンドルしたオペレーティングシステム (UNIX や Windows など) によくあります。 管理者が急いで特権ユーザーアカウントを作成したためにパスワードが空白のままになっていることがありますが、これは、このアカウントを発見した悪意のあるユーザーにとっては、絶好のエントリーポイントとなります。
デフォルトの共有鍵	セキュアなサービスでは、開発や評価テスト用にデフォルトのセキュリティ鍵がパッケージ化されていることがあります。これらの鍵を変更せずにインターネット上の実稼働環境に置いた場合、同じデフォルトの鍵を持つすべてのユーザーがその共有鍵のリソースや、そこにあるすべての機密情報にアクセスできるようになります。	無線アクセスポイントや事前設定済みのセキュアなサーバー機器に最も多く見られます。

不正使用	説明	注意事項
IP スプーフィング	リモートマシンがローカルネットワーク上でノードとして動作し、サーバーに脆弱性を見つけるとバックドアプログラムまたはトロイの木馬をインストールして、ネットワークリソース全体へのコントロールを得ようとしています。	<p>スプーフィングは、攻撃者が標的となるシステムへの接続を調整するのに TCP/IP シーケンス番号を予測しなければならないのでかなり難しいことですが、クラッカーの脆弱性の攻撃を支援する利用可能なツールがいくつかあります。</p> <p>標的となるシステムで実行される、ソーススペース認証技術を使用するサービス (rsh、telnet、FTP その他) によって異なりますが、このようなサービスは、ssh または SSL/TLS で使用される PKI やその他の形式の暗証化認証と比較すると推奨できるものではありません。</p>
盗聴	2 つのノード間の接続を盗聴することにより、ネットワーク上のアクティブなノード間を行き交うデータを収集します。	<p>この種類の攻撃には大抵、Telnet、FTP、および HTTP 転送などのプレーンテキストの転送プロトコルが使われます。</p> <p>このような攻撃を仕掛けるには、リモートの攻撃者は LAN 上で攻撃するシステムへのアクセス権を持っていないければなりません。通常、クラッカーは LAN 上にあるシステムを危険にさらすために活発な攻撃 (IP スプーフィングや中間者攻撃など) を仕掛けます。</p> <p>パスワードのなりすましを防ぐ予防策としては、暗号化鍵交換、ワンタイムパスワードまたは暗号化された認証によるサービス使用が挙げられます。通信中は強力な暗号化を実施することをお勧めします。</p>

不正使用	説明	注意事項
サービスの脆弱性	<p>攻撃者はインターネット上で実行されているサービスの欠陥や抜け穴を見つけます。この脆弱性を利用する攻撃者は、システム全体と格納されているデータを攻撃するだけでなく、ネットワーク上の他のシステムも攻撃する可能性があります。</p>	<p>CGI などの HTTP ベースのサービスは、リモートのコマンド実行やインタラクティブなシェルアクセスに対しても脆弱です。HTTP サービスが「nobody」などの権限のないユーザーとして実行される場合でも、設定ファイルやネットワークマップなどの情報が読み取られる可能性があります。または、攻撃者がサービス拒否攻撃を開始して、システムのリソースを流出させたり、他のユーザーが利用できないようにする可能性があります。</p> <p>開発時およびテスト時には気付かない脆弱性がサービスに含まれることがあります。このような脆弱性（攻撃者が任意の値を使用してアプリケーションのメモリーバッファ領域をあふれさせ、攻撃者が任意のコマンドを実行できるようなインタラクティブなコマンドプロンプトを与えて、サービスをクラッシュさせるバッファオーバーフローなど）は完全な管理コントロールを攻撃者に与えるものとなる可能性があります。</p> <p>管理者は、サービスが root ユーザーとして実行されないようにし、ベンダーまたは CERT や CVE などのセキュリティー組織からのアプリケーション用のパッチやエラータ更新がないか常に注意する必要があります。</p>
アプリケーションの脆弱性	<p>攻撃者はデスクトップやワークステーションのアプリケーション（電子メールクライアントなど）に欠陥を見つけ出し、任意のコードを実行したり、将来のシステム侵害のためにトロイの木馬を移植したり、システムを破壊したりします。攻撃を受けたワークステーションがネットワークの残りの部分に対して管理特権を持っている場合は、さらなる不正使用が起こる可能性があります。</p>	<p>ワークステーションとデスクトップは、ユーザーが侵害を防いだり検知するための専門知識や経験を持たないため、不正使用の対象になりやすくなります。認証されていないソフトウェアをインストールしたり、要求していないメールの添付ファイルを開く際には、それに伴うリスクについて個々に通知することが必須です。</p> <p>電子メールクライアントソフトウェアが添付ファイルを自動的に開いたり、実行したりしないようにするといった、予防手段を取ることが可能です。さらに、Red Hat Network や他のシステム管理サービスなどからワークステーションのソフトウェアを自動更新することにより、マルチシートのセキュリティーデプロイメントの負担を軽減することができます。</p>

不正使用	説明	注意事項
サービス拒否攻撃 (DoS: Denial of Service)	単独の攻撃者または攻撃者のグループは、目標のホスト（サーバー、ルーター、ワークステーションのいずれか）に認証されていないパケットを送ることにより、組織のネットワークまたはサーバーのリソースに対して攻撃を仕掛けます。これにより、正当なユーザーはリソースを使用できなくなります。	<p>2000 年に発生した米国内での DoS で最も多く報告されたケースとして、通信量の非常に多い民間および政府サイトのいくつかが利用不可能になりました。ゾンビ(zombie) やリダイレクトされたブロードキャストノードとして動作する、高帯域幅接続を有する複数の攻撃対象のシステムを使って、調整された ping フラッド攻撃が行われたためです。</p> <p>通常ソースパケットは、攻撃の本当のものを調査するのが難しくなるよう、偽装（または再ブロードキャスト）されています。</p> <p>iptables を使用したイングレスフィルタリング (IETF rfc2267) や snort などのネットワーク侵入検知システムにおける進歩は、管理者が分散型サービス拒否攻撃を追跡し、これを防止するのに役立っています。</p>

第2章 インストール時におけるセキュリティーのヒント

セキュリティーは、Red Hat Enterprise Linux 7 をインストールするために CD や DVD をディスクドライブに入れた時から始まります。最初からシステムを安全に設定することで、追加のセキュリティー設定を後で実装することがより簡単になります。

2.1. BIOS のセキュア化

BIOS (もしくは BIOS に相当するもの) およびブートローダーのパスワード保護により、システムに物理的にアクセス可能な未承認ユーザーがリムーバブルメディアを使って起動したり、シングルユーザーモードで root 権限を取得したりすることを防ぐことができます。このような攻撃に対するセキュリティー対策は、ワークステーション上の情報の機密性とマシンの場所によって異なります。

例えば、マシンが見本市で使われていて機密情報を含んでいない場合、このような攻撃を防ぐことは重要ではないかもしれません。しかし、同じ見本市で企業ネットワーク用のプライベートの暗号化されていない SSH 鍵のあるノートパソコンが誰の監視下にもなく置かれていた場合、重大なセキュリティー侵害につながり、その影響は企業全体に及ぶ可能性があります。

ただし、ワークステーションが権限のあるユーザーもしくは信頼できるユーザーのみがアクセスできる場所に置かれてるのであれば、BIOS もしくはブートローダーの安全確保は必要ない可能性もあります。

2.1.1. BIOS パスワード

コンピューターの BIOS をパスワードで保護する主な理由は、以下の 2 つです^[1]。

1. **BIOS 設定の変更を防止する** — 侵入者が BIOS にアクセスした場合、CD-ROM やフラッシュドライブから起動するように設定できます。このようにすると、侵入者がレスキューモードやシングルユーザーモードに入ることが可能になり、システム上で任意のプロセスを開始したり、機密性の高いデータをコピーできるようになってしまいます。
2. **システムの起動を防止する** — BIOS の中には起動プロセスをパスワードで保護できるものもあります。これを有効にすると、攻撃者は BIOS がブートローダーを開始する前にパスワード入力を求められます。

BIOS パスワードの設定方法はコンピューターメーカーで異なるため、具体的な方法についてはコンピューターのマニュアルを参照してください。

BIOS パスワードを忘れた場合は、マザーボード上のジャンパーでリセットするか、CMOS バッテリーを外します。このため、可能な場合はコンピューターのケースをロックすることがよい方法です。ただし、CMOS バッテリーを外す前にコンピューターもしくはマザーボードのマニュアルを参照してください。

2.1.1.1. 非 BIOS ベースのシステム

他のシステムやアーキテクチャーでは、異なるプログラムを使用して x86 システム上の BIOS とほぼ同等の低レベルのタスクを実行します。*Unified Extensible Firmware Interface (UEFI)* シェルなどがこの例になります。

BIOS と同様のプログラムをパスワード保護する方法については、メーカーの指示を参照してください。

2.2. ディスクのパーティション設定

Red Hat は、**/boot/**、**/**、**/home//tmp/**、および **/var/tmp/** の各ディレクトリーに個別のパーティションを作成することを推奨しています。各パーティションの目的は異なります。以下でこれらのパーティションについて説明します。

/boot

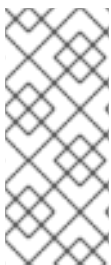
このパーティションは、ブート時にシステムが最初に読み込むパーティションです。システムを Red Hat Enterprise Linux 7 にブートするために使われるブートローダーとカーネルイメージはこのパーティションに保存されます。このパーティションは暗号化すべきではありません。このパーティションが **/** に含まれていて、そのパーティションが暗号化されているかまたは他の理由で利用できなくなると、システムをブートすることができなくなります。

/home

ユーザーデータ (**/home**) が別個のパーティションではなく **/** に保存されていると、このパーティションが一杯になり、オペレーティングシステムが不安定になる可能性があります。また、システムを次のバージョンの Red Hat Enterprise Linux 7 にアップグレードする際には、**/home** パーティションでデータを保存できると、このデータはインストール時に上書きされないので、アップグレードが非常に簡単になります。root パーティション (**/**) が破損すると、ユーザーのデータは完全に失われてしまいます。個別のパーティションを使うことで、データ損失に対する保護が少しは高まります。また、このパーティションを頻繁にバックアップする対象とすることも可能です。

/tmp および **/var/tmp**

/tmp と **/var/tmp** ディレクトリーは、どちらも長期保存の必要がないデータを保存するために使われます。しかし、これらのディレクトリーのいずれかが多くのデータであふれると、ストレージスペースがすべて消費されてしまう可能性があります。こうした状態が発生して、かつこれらのディレクトリーが **/** に保管されていると、システムが不安定になり、クラッシュする可能性があります。そのため、これらのディレクトリーを個別のパーティションに移動するとよいでしょう。



注記

インストールのプロセス中には、パーティションを暗号化するオプションがユーザーに示されます。ユーザーは、パスフレーズを提供する必要があります。これは、パーティションのデータを保護するために使われるバルク暗号鍵を解除する鍵として使用されます。LUKS についての詳細情報は、「[LUKS のディスク暗号化の使用](#)」を参照してください。

2.3. 必要なパッケージの最小限のインストール

コンピューター上の各ソフトウェアには脆弱性が潜んでいる可能性があるため、実際に使用するパッケージのみをインストールすることがベストプラクティスになります。インストールを DVD から行う場合は、インストールしたいパッケージのみを選択するようにします。他のパッケージが必要になる場合は、いつでも後でシステムに追加することができます。

最小限のインストール 環境に関する詳細情報は、Red Hat Enterprise Linux 7 インストールガイドの「[ソフトウェアの選択](#)」の章を参照してください。最小限のインストールは、Kickstart ファイルで **-nobase** オプションを使用して実行することもできます。Kickstart インストールについての詳細情報は、Red Hat Enterprise Linux 7 インストールガイドの「[パッケージの選択](#)」のセクションを参照してください。

2.4. インストールプロセス時のネットワーク接続の制限

Red Hat Enterprise Linux をインストールする際に、特定のタイミングで作成したスナップショットをインストールメディアとして使用します。そのため、セキュリティ修正が最新ののではなく、こ

のインストールメディアで設定するシステムが公開されてから修正された特定の問題に対して安全性に欠ける場合があります。

脆弱性が含まれる可能性のあるオペレーティングシステムをインストールする場合には、必ず、必要最小限のネットワークゾーンだけに公開レベルを制限してください。最も安全な選択肢は、インストールプロセス中はマシンをネットワークから切断した状態にする「ネットワークなし」のゾーンです。インターネット接続からのリスクが最も高くなっていますが、LAN またはイントラネット接続で十分な場合もあります。ベストなセキュリティーの慣習に従い、ネットワークから Red Hat Enterprise Linux をインストールする場合はお使いのリポジトリに最も近いゾーンを選択するようにしてください。

ネットワーク接続の設定に関する詳しい情報は、『Red Hat Enterprise Linux 7 インストールガイド』の「[ネットワーク & ホスト](#)」の章を参照してください。

2.5. インストール後の手順

以下のステップは、Red Hat Enterprise Linux のインストール直後に実行する必要があるセキュリティー関連の手順です。

1. システムを更新します。root で以下のコマンドを実行します。

```
~]# yum update
```

2. ファイアウォールサービスの **firewalld** は Red Hat Enterprise Linux のインストールで自動的に有効になっていますが、kickstart 設定などで明示的に無効となっている場合もあります。このような場合は、ファイアウォールを再度有効にすることが推奨されます。

firewalld を開始するには、root で以下のコマンドを実行します。

```
~]# systemctl start firewalld
~]# systemctl enable firewalld
```

3. セキュリティーを強化するために、不要なサービスは無効にしてください。たとえば、使用中のコンピューターにプリンターがインストールされていなければ、以下のコマンドを使って **cups** サービスを無効にします。

```
~]# systemctl disable cups
```

アクティブなサービスを見直すには、以下のコマンドを実行します。

```
~]$ systemctl list-units | grep service
```

2.6. その他のリソース

インストールに関する全般的な情報は、『[Red Hat Enterprise Linux 7 インストールガイド](#)』を参照してください。

[1]。システム BIOS はメーカーによって異なるため、パスワード保護をサポートしないものもあれば、あるタイプのパスワード保護のみをサポートするものもあります。

第3章 システムを最新の状態に保つ

本章では、システムを最新の状態に保つプロセスについて説明します。セキュリティ更新のインストール方法の立案や設定、新たに更新されたパッケージが導入する変更の適用、Red Hat カスタマーポータルを利用したセキュリティアドバイザリーの追跡などが関連してきます。

3.1. インストール済みソフトウェアのメンテナンス

セキュリティの脆弱性が発見された場合、セキュリティ上のリスクを最小限に抑えるために、影響を受けるソフトウェアを更新する必要があります。そのソフトウェアが現在サポートされている Red Hat Enterprise Linux ディストリビューション内のパッケージの一部である場合、Red Hat は脆弱性を修正する更新パッケージをできるだけ迅速にリリースするように全力を尽くします。

多くの場合、セキュリティ上の特定の不正使用に関する発表にはパッチ（または問題を解決するソースコード）が伴われています。このパッチは Red Hat Enterprise Linux パッケージに適用されます。パッチはテストされ、エラータ更新としてリリースされるものです。しかし、発表にパッチが含まれていない場合には、Red Hat の開発者はまず、問題の解決に向けてそのソフトウェアの保守担当者と協力します。問題が解決されると、パッケージがテストされ、エラータ更新としてリリースされます。

システムで使用しているソフトウェアのエラータ更新がリリースされている場合、システムが攻撃される可能性のある期間を最短にするため、影響を受けるパッケージをできる限り早く更新することが強く推奨されます。

3.1.1. セキュリティの更新の立案と設定

すべてのソフトウェアにはバグが含まれます。これらのバグは、悪意のあるユーザーにシステムをさらしてしまう可能性のある脆弱性につながりかねません。更新されていないパッケージは、コンピューターへの侵入を招く一般的な原因となります。脆弱性の悪用を防ぐには、セキュリティパッチをタイミングよくインストールする計画を実行して、発見された脆弱性をすばやく排除してください。

セキュリティ更新が利用可能になったらそれらをテストして、インストールするスケジュールを立てます。更新のリリースからシステムにインストールするまでの間、システムを保護するための新たな制御が必要になります。これらの制御は脆弱性の内容によって異なりますが、ファイアウォールの追加ルールや、外部ファイアウォールの使用、またはソフトウェア設定の変更などがこれらに含まれることがあります。

サポート対象のパッケージ内のバグは、エラーのメカニズムを使用して修正されます。エラータは、1 つ以上の RPM パッケージとそのエラータが対処する問題の簡単な説明で構成されています。エラータはすべて、アクティブなサブスクリプションを持つお客様に **Red Hat サブスクリプション管理** サービスで配布されます。セキュリティ問題に対処するエラータは、Red Hat セキュリティアドバイザリーと呼ばれます。

セキュリティエラータを使った作業についての詳細情報は、[「カスタマーポータルでセキュリティ更新を見る」](#) を参照してください。**RHN Classic** からの移行を含む **Red Hat サブスクリプション管理** サービスについての詳細情報は、[Red Hat サブスクリプション管理](#) にある関連ドキュメントを参照してください。

3.1.1.1. Yum のセキュリティ機能の使用

Yum パッケージマネジャーには、セキュリティエラータの検索、一覧、表示、インストールに使用可能なセキュリティ関連の機能がいくつか含まれています。これらの機能を使うと、**Yum** を使ってセキュリティ更新のみをインストールすることもできます。

使用中のシステムで利用可能なセキュリティ関連の更新を確認するには、**root** で以下のコマンドを実行します。


```
~]# yum check-update --security
Loaded plugins: langpacks, product-id, subscription-manager
rhel-7-workstation-rpms/x86_64 | 3.4 kB 00:00:00
No packages needed for security; 0 packages available
```

上記のコマンドは非対話モードで実行されるので、更新が利用可能かどうかの自動確認のスクリプトに使用することができます。このコマンドは、利用可能なセキュリティ更新がある場合には 100 を返し、更新がない場合には 0 を返します。エラーが発生すると、1 が返されます。

同様に、以下のコマンドを使用するとセキュリティ関連の更新のみがインストールされます。

```
~]# yum update --security
```

updateinfo サブコマンドを使うと、リポジトリが提供する利用可能な更新についての情報を表示したり、それについてアクションを取ることができます。この **updateinfo** サブコマンド自体は、多くのコマンドを受け付け、この中にはセキュリティ関連の使用も含まれます。これらのコマンドの概要については、表3.1「**yum updateinfo で使用可能なセキュリティ関連のコマンド**」を参照してください。

表3.1 **yum updateinfo** で使用可能なセキュリティ関連のコマンド

コマンド	説明
advisory [advisories]	1 つ以上のアドバイザリーについての情報を表示します。 <i>advisories</i> をアドバイザリー番号または番号に置き換えます。
cves	CVE (<i>Common Vulnerabilities and Exposures</i>) に関連する情報のサブセットを表示します。
security または sec	セキュリティ関連の情報をすべて表示します。
severity [severity_level] or sev [severity_level]	提供された <i>severity_level</i> のセキュリティ関連パッケージについての情報を表示します。

3.1.2. パッケージの更新とインストール

システムのソフトウェアを更新する際には、信頼できるソースから更新をダウンロードすることが重要です。攻撃者は、問題を解決するはずのバージョンと同じバージョン番号のパッケージを簡単に再構築し、そのパッケージからセキュリティ上の別の不正使用を可能にした上で、それをインターネット上にリリースする場合があります。こうした事態が発生すると、元の RPM に対するファイル検証などのセキュリティ対策を講じて、不正アクセスを検知することができません。このため、RPM は Red Hat などの信頼できるソースからのみダウンロードし、その安全性を検証するためにパッケージの署名を確認することが極めて重要になります。

Yum パッケージマネジャーの使用法に関する詳細情報は、Red Hat Enterprise Linux 7 システム管理者のガイドの [Yum](#) の章を参照してください。

3.1.2.1. 署名パッケージの検証

Red Hat Enterprise Linux のパッケージはすべて、Red Hat **GPG** 鍵を使って署名されています。**GPG** は **GNU Privacy Guard** または **GnuPG** の略で、配信ファイルの信頼性を保証するために使用されるフリーソフトウェアのパッケージです。パッケージ署名の検証が失敗すると、パッケージは改ざんされている可能性があるため信頼できません。

Yum パッケージマネジャーを使うと、インストールまたはアップグレード対象の全パッケージを自動的に検証できます。この機能は、デフォルトで有効になっています。使用中のシステムでこのオプションを設定する場合は、`/etc/yum.conf` 設定ファイル内で **gpgcheck** 設定ディレクティブが **1** に設定されていることを確認してください。

以下のコマンドを使用すると、ファイルシステム上のパッケージファイルを手動で検証できます。

```
rpmkeys --checksig package_file.rpm
```

Red Hat パッケージの署名のプラクティスについての追加情報は、Red Hat カスタマーポータルの [Red Hat GPG キー](#) の記事を参照してください。

3.1.2.2. 署名パッケージのインストール

ファイルシステムから検証済みのパッケージ (パッケージの検証方法については、「[署名パッケージの検証](#)」を参照) をインストールするには、以下のように **root** で **yum install** コマンドを実行します。

```
yum install package_file.rpm
```

複数のパッケージを一度にインストールするには、シェル glob を使用します。たとえば、以下のコマンドを実行すると、現行ディレクトリーにすべての **.rpm** パッケージがインストールされます。

```
yum install *.rpm
```



重要

セキュリティに関するエラーをインストールする前に、エラーレポートに記載されているすべての特別な指示をよく読み、指示に従ってインストールしてください。エラー更新に基づく変更の適用についての全般的な指示は、「[インストールされた更新による変更の適用](#)」を参照してください。

3.1.3. インストールされた更新による変更の適用

セキュリティに関するエラーと更新をダウンロードしてインストールした後は、古いソフトウェアの使用を中止して新しいソフトウェアの使用を開始することが重要です。これを実際に行う方法は、更新済みのソフトウェアの種類によって異なります。以下では、ソフトウェアの一般的なカテゴリを示し、パッケージのアップグレード後に更新バージョンを使用する方法について説明します。



注記

一般的に、システムの再起動は、ソフトウェアパッケージの最新バージョンが使用されていることを確認する最も確実な方法です。ただし、このオプションは常に必要というわけではなく、システム管理者が常に利用できるわけでもありません。

アプリケーション

ユーザースペースのアプリケーションとは、ユーザーが開始できるすべてのプログラムのことです。通常このようなアプリケーションは、ユーザー、スクリプトまたは自動化されたタスクユーティリティがそれらを起動する場合にのみ使用されるものです。

ユーザースペースのアプリケーションが更新されると、システムにあるアプリケーションのすべてのインスタンスが停止し、更新バージョンを使用するためにプログラムが再起動されます。

カーネル

カーネルは、Red Hat Enterprise Linux 7 オペレーティングシステムの中心的なソフトウェアコンポーネントです。カーネルはメモリー、プロセッサおよび周辺機器へのアクセスを管理し、すべてのタスクをスケジュールします。

カーネルは中心的な役割を担うので、カーネルの再起動にはコンピューターの再起動が伴います。つまりカーネルの更新バージョンは、システムの再起動後に初めて使用できるようになります。

KVM

qemu-kvm および libvirt のパッケージが更新されると、すべてのゲスト仮想マシンを停止して、関連の仮想化モジュールをリロードし（またはホストシステムを再起動し）、仮想マシンを再起動する必要があります。

kvm、**kvm-intel**、または **kvm-amd** のどのモジュールがロードされているかを確認するには、**lsmod** コマンドを使用します。その後に **modprobe -r** コマンドを使用してロードされているモジュールを削除し、**modprobe -a** コマンドで影響を受けたモジュールをリロードします。以下に例を示します。

```
~]# lsmod | grep kvm
kvm_intel          143031  0
kvm                460181  1 kvm_intel
~]# modprobe -r kvm-intel
~]# modprobe -r kvm
~]# modprobe -a kvm kvm-intel
```

共有ライブラリ

共有ライブラリは、**glibc** のように、多くのアプリケーションやサービスにより使用されるコードの集合です。通常、共有ライブラリを使用しているアプリケーションは、アプリケーションが初期化される際に共有コードを読み込みます。そのため、更新されたライブラリを使用しているすべてのアプリケーションは、まず停止してから再起動する必要があります。

特定ライブラリにリンクしている実行中のアプリケーションを判別するには、以下の例のように **lsof** コマンドを使用します。

lsof library

たとえば、**libwrap.so** ライブラリにリンクしている実行中のアプリケーションを判別するには、以下を入力します。

```
~]# lsof /lib64/libwrap.so.0
COMMAND      PID USER  FD   TYPE DEVICE SIZE/OFF      NODE NAME
pulseaudi 12363 test  mem    REG  253,0    42520 34121785
/usr/lib64/libwrap.so.0.7.6
gnome-set 12365 test  mem    REG  253,0    42520 34121785
/usr/lib64/libwrap.so.0.7.6
gnome-she 12454 test  mem    REG  253,0    42520 34121785
/usr/lib64/libwrap.so.0.7.6
```

このコマンドは、ホストのアクセス制御に **TCP Wrapper** を使用する実行中のプログラムの一覧を返します。そのため **tcp_wrappers** パッケージが更新されると、リストにあるすべてのプログラムは停止してから再起動する必要があります。

systemd サービス

systemd サービスは、通常ブートプロセス中に開始される永続的なサーバープログラムです。systemd サービスの例としては、**sshd** や **vsftpd** などがあります。

通常、これらのプログラムはマシンが起動している限りはメモリ内に残るので、パッケージのアップグレード後には、更新された systemd サービスは停止してから再起動する必要があります。これは、**root** で **systemctl** コマンドを使用すると実行できます。

```
systemctl restart service_name
```

`service_name` を **sshd** などの再起動するサービスの名前に置き換えます。

他のソフトウェア

以下のアプリケーションの更新は、リンク先のリソースで示された指示にしたがってください。

- **Red Hat Directory Server** — https://access.redhat.com/site/documentation/en-US/Red_Hat_Directory_Server/ で該当する Red Hat Directory Server バージョンの『Release Notes』を参照してください。
- **Red Hat Enterprise Virtualization Manager** — https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Virtualization/ で該当する Red Hat Enterprise Virtualization のバージョンの『インストールガイド』を参照してください。

3.2. RED HAT カスタマーポータルの使用

Red Hat カスタマーポータルは <https://access.redhat.com/> にあり、これは Red Hat 製品に関する公式情報のお客様向け主要リソースになります。ドキュメンテーションの検索やサブスクリプションの管理、製品および更新のダウンロード、サポートケースの開始、セキュリティ更新についての情報収集などができます。

3.2.1. カスタマーポータルでセキュリティ更新を見る

アクティブなサブスクリプションに関連するセキュリティアドバイザリー（エラータ）を見るには、<https://access.redhat.com/> でカスタマーポータルにログインして、メインページの **Download Products & Updates** ボタンをクリックします。**Software & Download Center** ページに入り、**Errata** ボタンをクリックすると、登録済みシステムに関連するアドバイザリーが一覧表示されます。

アクティブな Red Hat 全製品のセキュリティ更新をすべて表示するには、ページのトップにあるナビゲーションメニューを使って **セキュリティ** → **セキュリティアップデート** → **アクティブな製品** に移動します。

テーブルの左側にあるエラータコードをクリックすると、個別のアドバイザリーについての詳細情報が表示されます。次のページには、特定のエラータの原因や結果、必要となる修正だけでなく、その特定のエラータが更新するパッケージ全一覧と更新の適用方法が示されています。このページには、関連する CVE などの関連参照情報のリンクも含まれています。

3.2.2. カスタマーポータルページでの CVE への移動

CVE (Common Vulnerabilities and Exposures) プロジェクトは MITRE Corporation がメンテナンスを行なっているもので、脆弱性とセキュリティエクスポージャーの標準名を一覧提供しています。Red Hat 製品に関連する CVE の一覧をカスタマーポータルで確認するに

は、<https://access.redhat.com/> でアカウントにログインして、ページトップにあるナビゲーションメニューを利用して **セキュリティ** → **リソース** → **CVE データベース** に移動します。

テーブルの左側にある CVE コードをクリックすると、個別の脆弱性についての詳細情報が表示されます。次のページには、特定の CVE 説明だけでなく、影響を受けている Red Hat 製品の一覧と関連する Red Hat エラータのリンクが含まれています。

3.2.3. 問題の影響度の分類について

Red Hat 製品で発見されたすべてのセキュリティ問題には、*Red Hat 製品セキュリティ* がその問題の重大度に応じて影響度を割り当てます。これは、低、中、重要、重大の 4 段階評価で判断されます。また、セキュリティ問題はすべて *Common Vulnerability Scoring System (CVSS)* ベーススコアを使って評価されます。

これらを合わせると、セキュリティ問題の影響度が理解できます。そうすることで、使用中のシステムのアップグレード戦略のスケジュールを立てたり、優先順位を決めたりすることができます。これらの評価は特定の脆弱性の潜在的なリスクを反映するもので、現行の脅威レベルではなく、バグの技術的分析に基づいています。つまり、セキュリティの影響度評価は、特定の弱点に対してエクスプロイトがリリースされても変更されません。

カスタマーポータルで影響度評価の個別レベルの詳細情報を確認するには、[重大度レベル](#) のページを参照してください。

3.3. その他のリソース

セキュリティ更新やその適用方法、Red Hat カスタマーポータル、および関連するトピックについての詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- `yum(8)` — The manual page for the **Yum** package manager provides information about the way **Yum** can be used to install, update, and remove packages on your systems.
- `rpmkeys(8)` — The manual page for the **rpmkeys** utility describes the way this program can be used to verify the authenticity of downloaded packages.

オンラインのドキュメント

- [Red Hat Enterprise Linux 7 システム管理者ガイド](#) — Red Hat Enterprise Linux 7 の『システム管理者ガイド』では、**Yum** および **rpm** コマンドを使って Red Hat Enterprise Linux 7 システム上でパッケージをインストール、更新、削除する方法について説明しています。
- [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) — Red Hat Enterprise Linux 7 の『SELinux User's and Administrator's Guide』では、**SELinux mandatory access control** メカニズムの設定について説明しています。

Red Hat カスタマーポータル

- [Red Hat カスタマーポータル、セキュリティ](#) — カスタマーポータルのセキュリティセクションには、Red Hat CVE データベースや Red Hat 製品セキュリティの連絡先など、最も重要なリソースへのリンクが含まれています。
- [Red Hat セキュリティブログ](#) — Red Hat セキュリティ専門家による最新のセキュリティ関連問題に関する記事。

関連項目

- [2章インストール時におけるセキュリティーのヒント](#)では、追加のセキュリティー設定を後で簡単に実装できるようにシステムを安全に一から設定する方法について説明しています。
- 「[GPG 鍵の作成](#)」では、個人の **GPG** 鍵セットを作成し、通信を認証する方法について説明しています。

第4章 ツールとサービスを使用したシステム強化

4.1. デスクトップのセキュリティー

Red Hat Enterprise Linux 7 は、不正アクセスの防止や攻撃から守るためのデスクトップ強化に複数の手段を提供しています。本項では、ユーザーパスワード、セッション、アカウントのロック、取り外し可能なメディアの安全な取り扱いについて推奨の方法を紹介します。

4.1.1. パスワードのセキュリティー

パスワードは、Red Hat Enterprise Linux 7 がユーザーの ID を確認する第一の手段です。ユーザーやワークステーション、ネットワークの保護にパスワードのセキュリティーが重要であるのは、このためです。

インストールプログラムではセキュリティー目的で、システムが **セキュアハッシュアルゴリズム 512 (SHA512)** とシャドウパスワードを使うように設定します。この設定は、変更しないことが強く推奨されます。

インストール中にシャドウパスワードが検出されると、すべてのパスワードは全ユーザーが読み取り可能な **/etc/passwd** ファイルに一方方向のハッシュとして保存されます。この場合、システムはオフラインのパスワードクラッキング攻撃に脆弱なものとなってしまいます。侵入者が通常のユーザーとしてマシンにアクセスすると、**/etc/passwd** ファイルを侵入者自身のマシンにコピーして、パスワードクラッキングプログラムをいくつでも実行できるようになります。このファイルに安全でないパスワードがあれば、パスワードクラッカーがこれを見つけるのは時間の問題となります。

シャドウパスワードは、パスワードハッシュを **/etc/shadow** ファイルに保存することで、このタイプの攻撃を排除します。このファイルは、root ユーザーのみが読み取り可能なためです。

この場合、攻撃者は SSH や FTP などのマシン上のネットワークサービスにログインしてリモートからパスワードクラッキングを試みるのが強いられます。このようなブルートフォース攻撃は時間がかかり、何百回ものログイン失敗がシステムファイルに書き込まれるので、明らかな痕跡が残ります。もちろん、クラッカーが脆弱なパスワードがあるシステムに真夜中から攻撃を開始した場合、夜明け前にはアクセスに成功し、痕跡を隠すためにログファイルを編集してしまう、という可能性もあります。

フォーマットやストレージに加えて考慮すべき点は、コンテンツの問題です。パスワードクラッキングからアカウントを保護するためにユーザーがなし得る最重要事項は、強固なパスワードを作成することです。

注記

Red Hat は、Red Hat Identity Management (IdM) などの集約型の認証ソリューションの使用を推奨します。ローカルのパスワードよりも集約型のソリューションの使用が推奨されます。詳しい情報は、以下のリンクをご覧ください。

- [『LINUX ドメイン ID、認証、およびポリシーガイド』](#)
- [「パスワードポリシーの定義」](#)

4.1.1.1. 強固なパスワードの作成

安全なパスワードを作成するには、長いパスワードの方が短くかつ複雑なものよりも強固であることをユーザーは認識する必要があります。あるパスワードが数字や特殊文字、大文字のアルファベットを含んでいても、それが 8 文字の長さしかなければ、強固なパスワードとは言えません。「John The

Ripper」のようなパスワード解析ツールは、人間が記憶するには困難なパスワードを分解するために最適なものです。

情報の理論上では、エントロピーはランダムな変数に関連付けられる不確実性のレベルで、ビット表示されます。エントロピーの値が高ければ高いほど、パスワードはより安全と言えます。NIST (米標準技術研究所) の SP 800-63-1 文書によると、一般的に選択される 5 万件のパスワードからなる辞書にないパスワードには、少なくとも 10 ビットのエントロピーがあります。このため、ランダムな 4 単語からなるパスワードには、およそ 40 ビットのエントロピーがあることになります。安全性を高めるために複数の単語で構成される長いパスワードは、パスフレーズとも呼ばれます。例を示します。

```
randomword1 randomword2 randomword3 randomword4
```

大文字や数字、特殊文字の使用が強制されるシステムでは、上記の推奨事項に合致するパスフレーズは簡単に修正することができます。たとえば、最初の文字を大文字にして、"**1!**" を追加します。このような修正は、パスフレーズの安全性を大幅に高めるもの **ではない** ことに注意してください。

パスワードを自分で作成するもうひとつの方法は、パスワードジェネレーターを使用することです。**pwmake** は、大文字、小文字、数字、特殊文字の 4 種類の文字からなるランダムなパッケージを生成するコマンドラインツールです。このユーティリティを使うと、パスワード生成に使用されるエントロピービットの数を指定することができます。エントロピーは、**/dev/urandom** から引き出されます。指定可能な最小ビット数は 56 で、これはブルートフォース攻撃が滅多に仕掛けられないシステムやサービスのパスワードには十分なものです。攻撃者がパスワードハッシュファイルに直接アクセスできないアプリケーションであれば、64 ビットで十分です。攻撃者がパスワードハッシュへの直接アクセスを取得する可能性がある場合やパスワードが暗号化鍵として使用される場合は、80 ビットや 128 ビットを使うべきです。無効なエントロピービット数を指定すると、**pwmake** はデフォルトのビット数を使用します。128 ビットのパッケージを作成するには、以下のコマンドを実行します。

```
pwmake 128
```

安全なパスワードの作成にはいくつかのアプローチがありますが、以下の悪い点は必ず避けてください。

- 辞書の 1 単語を使用する。外国語の 1 単語を使用する。反転した単語を使用する。数字のみを使用する。
- パスワードまたはパスフレーズを 10 文字未満にする。
- キーボードのレイアウトにおけるキーの配列を使用する。
- パスフレーズを書き留める。
- 誕生日や記念日、家族の名前、ペットの名前などの個人情報をパスフレーズに使用する。
- 複数のマシンで同じパスワードまたはパスフレーズを使用する。

セキュアなパスワードの作成は不可欠ですが、パスワードの適切な管理も特に大きな組織のシステム管理者にとっては重要です。次のセクションでは、組織内でのユーザーパスワードの作成および管理で推奨される方法を説明します。

4.1.1.2. 強固なパスワードの強制

組織内に多くのユーザーがいる場合、システム管理者は 2 つの基本的な方法で強固なパスワードの使用を強制できます。つまり、管理者がパスワードを作成してユーザーに渡すか、パスワードに十分な強度があるかを検証しながらユーザー自身がパスワードを作成する方法です。

前者の方法だとパスワードは強固なものになりますが、組織が拡大するにつれてタスクが重荷になります。また、ユーザーが自分のパスワードを書き留め、それをさらしてしまうリスクも高まります。

これらの理由で、多くのシステム管理者は後者を選択しますが、パスワードが強固かどうかを積極的に検証します。場合によっては、管理者はパスワードエージングで定期的にユーザーがパスワードを変更するように強制することもできます。

ユーザーは、パスワードの作成や変更を求められると、**passwd** コマンドラインユーティリティーを使用することができます。これは **PAM (Pluggable Authentication Modules)** を認識し、パスワードが短すぎないか、短くない場合は簡単にクラックされないかを確認します。この確認は、**pam_pwquality.so** PAM モジュールが実行します。



注記

Red Hat Enterprise Linux 7 では、**pam_pwquality** PAM モジュールが **pam_cracklib** の代わりとなっています。後者は、Red Hat Enterprise Linux 6 でパスワードの品質確認にデフォルトのモジュールとして使われていました。新たなモジュールは、**pam_cracklib** と同じバックエンドを使用します。

pam_pwquality モジュールは、ルールセットに対してパスワードの強度を確認するために使用されます。確認の手順は 2 段階になります。まず、該当パスワードが辞書にあるかどうかをチェックします。辞書にない場合は、さらにいくつかのチェックを行います。**pam_pwquality** は **/etc/pam.d/passwd** ファイルの **password** コンポーネント内に他の PAM モジュールと並んでいます。ルールのカスタムセットは、**/etc/security/pwquality.conf** 設定ファイル内で指定されます。これらのチェック項目の完全一覧については、**pwquality.conf (8)** man ページを参照してください。

例4.1 pwquality.conf 内でのパスワード強度チェックの設定

pam_quality の使用を有効にするには、以下の行を **/etc/pam.d/passwd** ファイルの **password** スタックに追加します。

```
password    required    pam_pwquality.so retry=3
```

チェック項目のオプションは、1 行に 1 つずつ指定します。たとえば、パスワードを 8 文字以上にして、4 種類すべての文字を含めることを要件とするには、以下の行を **/etc/security/pwquality.conf** ファイルに追加します。

```
minlen = 8
minclass = 4
```

文字シーケンスと連続した同じ文字についてパスワード強度チェックを設定するには、以下の行を **/etc/security/pwquality.conf** に追加します。

```
maxsequence = 3
maxrepeat = 3
```

この例では、入力するパスワードに、**abcd** などの単純なシーケンスの 3 文字を超える文字と、**1111** などの 3 文字を超える同一連続文字を含めることができません。

**注記**

root ユーザーはパスワード作成ルールを強制している本人なので、警告メッセージが出ても自分または通常ユーザーにどんなパスワードでも設定することができます。

4.1.1.3. パスワードエージングの設定

パスワードエージングは、システム管理者が組織内での脆弱なパスワードに対する防御として用いるもうひとつのテクニックです。パスワードエージングとは、一定期間後（通常 90 日）にユーザーが新たなパスワードを作成するように求められるシステムのことです。理論上は、ユーザーに定期的なパスワード変更を強制すれば、クラックされたパスワードは侵入者にとって一定期間しか有効でないことになります。しかし、パスワードエージングのマイナス面は、ユーザーがパスワードを書き留める可能性が高くなるという点です。

Red Hat Enterprise Linux 7 でパスワードエージングを指定するには、**chage** コマンドを使用します。

**重要**

Red Hat Enterprise Linux 7 では、デフォルトでシャドウパスワードが有効になっています。詳細は、[Red Hat Enterprise Linux 7 システム管理者のガイド](#)を参照してください。

chage コマンドの **-M** オプションでは、パスワードが有効となる最長日数を指定します。例えば、ユーザーのパスワードが 90 日間で期限切れとなるように設定するには、以下のコマンドを実行します。

```
chage -M 90 username
```

上記のコマンドでは、*username* をユーザーの名前に置き換えます。パスワードの有効期限を無効にするには、**-M** オプションの後に **-1** の値を指定します。

chage コマンドで使用可能なオプションの詳細情報については、以下の表を参照してください。

表4.1 chage のコマンドラインオプション

オプション	説明
-d days	1970 年 1 月 1 日から最後にパスワードを変更した日までの日数を指定します。
-E date	アカウントがロックされる日付を YYYY-MM-DD の形式で指定します。日付の代わりに、1970 年 1 月 1 日からの日数を指定することも可能です。
-I days	パスワードが失効してからアカウントがロックされるまでのアクティブでない日数を指定します。値を 0 とすると、パスワード失効後にアカウントはロックされません。
-l	現在のアカウントエージングの設定を一覧表示します。
-m days	パスワード変更が必要となる間隔の最短日数を指定します。値を 0 とすると、パスワードは失効しません。

オプション	説明
-M days	パスワードの有効最長日数を指定します。このオプションで指定した日数と -d オプションで指定した日数の合計が、1970 年 1 月 1 日から数えた現在までの日数より少ない場合は、ユーザーはアカウントを使用する前にパスワードを変更する必要があります。
-W days	パスワードが失効する前にユーザーに警告を発する日数を指定します。

インタラクティブモードで **chage** を使用して、複数のパスワードエージングおよびアカウントの詳細を修正することもできます。以下のコマンドでインタラクティブモードに入ります。

chage <username>

以下は、このコマンドを使用したインタラクティブセッションの例です。

```
~]# chage juan
Changing the aging information for juan
Enter the new value, or press ENTER for the default
Minimum Password Age [0]: 10
Maximum Password Age [99999]: 90
Last Password Change (YYYY-MM-DD) [2006-08-18]:
Password Expiration Warning [7]:
Password Inactive [-1]:
Account Expiration Date (YYYY-MM-DD) [1969-12-31]:
```

ユーザーの初回ログイン時にパスワードが失効するように設定できます。これにより、ユーザーはパスワードを直ちに変更するよう強制されます。

1. 初期パスワードを設定します。デフォルトのパスワードを割り当てるには、シェルプロンプトで **root** として以下のコマンドを実行します。

passwd username



警告

passwd ユーティリティには、null パスワードを設定するオプションがあります。null パスワードの使用は便利ですが、安全性は極めて低くなります。これは、いかなる第三者でも、セキュアでないユーザー名を使用して最初にシステムにログインし、アクセスできるためです。可能な場合は、null パスワードの使用は避けてください。どうしても避けられない場合は、null パスワードでアカウントのロック解除を行う前に、ユーザーがログインできる状態にあることを常に確かめてください。

2. パスワードを直ちに強制的に失効させるには、**root** として以下のコマンドを実行します。

chage -d 0 username

■

このコマンドは、パスワードが最後に変更された日付の値を 1970 年 1 月 1 日に設定します。パスワードエージングのポリシーが設定されていても、この値はパスワードを直ちに強制的に期限切れにします。

これで、ユーザーは初回ログイン時に新規パスワードを入力するよう要求されます。

4.1.2. アカウントのロック

Red Hat Enterprise Linux 7 では、**pam_faillock** PAM モジュールを使うとシステム管理者は特定回数のログイン失敗の後にユーザーアカウントをロックアウトすることができます。ユーザーログインの試行回数を制限する主な目的はセキュリティ措置であり、ユーザーアカウントのパスワード獲得を目的とする総当たり攻撃を防ぐためのものです。

pam_faillock モジュールを使うと、ログイン失敗はユーザーごとに **/var/run/faillock** ディレクトリー内の個別ファイルに保存されます。



注記

ログイン失敗のログファイルにおける行の順番は重要なものです。**even_deny_root** オプションが使用されている場合、この順番が変更されると **root** アカウントを含むすべてのユーザーアカウントがロックされます。

アカウントのロックを設定するには、以下の手順を実行します。

1. **root** 以外のユーザーがログインに 3 回失敗した後にロックアウトし、その 10 分後にこのユーザーのロックアウトを解除するようにするには、2 つの行を **/etc/pam.d/system-auth** および **/etc/pam.d/password-auth** の各ファイルの **auth** セクションに追加します。編集後に、両方のファイルの **auth** セクション全体は以下のようになります。

```
1 auth      required      pam_env.so
2 auth      required      pam_faillock.so preauth silent audit
deny=3 unlock_time=600
3 auth      sufficient    pam_unix.so nullok try_first_pass
4 auth      [default=die] pam_faillock.so authfail audit deny=3
unlock_time=600
5 auth      requisite     pam_succeed_if.so uid >= 1000
quiet_success
6 auth      required      pam_deny.so
```

行番号 2 および 4 が追加されました。

2. 以下の行を上記の手順の両ファイルの **account** セクションに追加します。

```
account      required      pam_faillock.so
```

3. アカウントのロックアウトを **root** ユーザーにも適用するには、**/etc/pam.d/system-auth** および **/etc/pam.d/password-auth** の各ファイルの **pam_faillock** エントリーに **even_deny_root** オプションを追加します。

```
auth      required      pam_faillock.so preauth silent audit
deny=3 even_deny_root unlock_time=600
auth      sufficient    pam_unix.so nullok try_first_pass
auth      [default=die] pam_faillock.so authfail audit deny=3
```

```
even_deny_root unlock_time=600

account      required      pam_faillock.so
```

ユーザー **john** がログインに 3 回失敗した後に再度ログインしようとする、このユーザーのアカウントはこの 4 回目のログイン試行でロックされます。

```
[yruseva@localhost ~]$ su - john
Account locked due to 3 failed logins
su: incorrect password
```

複数回のログイン失敗後でもユーザーがロックアウトされないようにするには、以下の行を **/etc/pam.d/system-auth** および **/etc/pam.d/password-auth** の各ファイルで **pam_faillock** が最初に呼び出される行のすぐ上に追加します。また、**user1**、**user2**、および **user3** を実際のユーザー名で置き換えます。

```
auth [success=1 default=ignore] pam_succeed_if.so user in
user1:user2:user3
```

ユーザーあたりの実際の失敗回数を表示するには、**root** で以下のコマンドを実行します。

```
[root@localhost ~]# faillock
john:
When                Type  Source
Valid
2013-03-05 11:44:14 TTY    pts/0
V
```

ユーザーのアカウントをロック解除するには、**root** で以下のコマンドを実行します。

```
faillock --user <username> --reset
```

authconfig でのカスタム設定の保持

authconfig ユーティリティーを使って認証設定を修正すると、**system-auth** および **password-auth** の各ファイルは **authconfig** ユーティリティーからの設定で上書きされます。これを避けるには、設定ファイルの代わりにシンボリックリンクを作成します。このリンクを **authconfig** が認識し、上書きが避けられます。設定ファイル内のカスタム設定と **authconfig** を同時に使うには、以下の手順でアカウントロックを設定します。

1. **system-auth** および **password-auth** ファイルがすでに **system-auth-ac** および **password-auth-ac** を参照しているシンボリックリンクであるかどうかを確認します (これはシステムデフォルト)。

```
~]# ls -l /etc/pam.d/{password,system}-auth
```

出力が以下のような内容である場合、シンボリックリンクは問題なく設定されているため、手順 3 に進むことができます。

```
lrwxrwxrwx. 1 root root 16 24. Feb 09.29 /etc/pam.d/password-auth ->
password-auth-ac
lrwxrwxrwx. 1 root root 28 24. Feb 09.29 /etc/pam.d/system-auth ->
system-auth-ac
```

system-auth および **password-auth** ファイルがシンボリックリンクでない場合は、次の手順に進んでください。

2. 設定ファイルの名前を変更します。

```
~]# mv /etc/pam.d/system-auth /etc/pam.d/system-auth-ac
~]# mv /etc/pam.d/password-auth /etc/pam.d/password-auth-ac
```

3. カスタマー設定を含む設定ファイルを作成します。

```
~]# vi /etc/pam.d/system-auth-local
```

/etc/pam.d/system-auth-local ファイルに以下の行を記載します。

```
auth            required          pam_faillock.so preauth silent audit
deny=3 unlock_time=600
auth            include            system-auth-ac
auth            [default=die]      pam_faillock.so authfail silent audit
deny=3 unlock_time=600

account         required          pam_faillock.so
account         include            system-auth-ac

password        include            system-auth-ac

session        include            system-auth-ac
```

```
~]# vi /etc/pam.d/password-auth-local
```

/etc/pam.d/password-auth-local ファイルに以下の行を記載します。

```
auth            required          pam_faillock.so preauth silent audit
deny=3 unlock_time=600
auth            include            password-auth-ac
auth            [default=die]      pam_faillock.so authfail silent audit
deny=3 unlock_time=600

account         required          pam_faillock.so
account         include            password-auth-ac

password        include            password-auth-ac

session        include            password-auth-ac
```

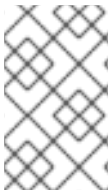
4. 以下のシンボリックリンクを作成します。

```
~]# ln -sf /etc/pam.d/system-auth-local /etc/pam.d/system-auth
~]# ln -sf /etc/pam.d/password-auth-local /etc/pam.d/password-auth
```

For more information on various **pam_faillock** configuration options, see the **pam_faillock(8)** manual page.

4.1.3. セッションのロック

日常の業務中にユーザーがワークステーションから離れなければならない時もあります。こういう場合は、特に十分な物理的セキュリティ対策がとられていない環境（「[物理的コントロール](#)」を参照）では、攻撃者にマシンに物理的にアクセスする機会を与えてしまいます。ノートパソコンの場合は特に、持ち運びが便利なので物理的な安全性が脅かされます。このリスクは、正しいパスワードが入力されないといシステムにアクセスできないようにするセッションロック機能を使うことで緩和できます。



注記

画面のロックがログアウトよりも優れている点は、ロックの場合は（ファイル転送といった）ユーザーのプロセスを続行できるという点です。ログアウトしてしまうと、このようなプロセスは中断してしまいます。

4.1.3.1. vlock を使った仮想コンソールのロック

仮想コンソールをロックする必要がある場合は、**vlock** というユーティリティを使って実行できます。このユーティリティをインストールするには、以下のコマンドを **root** で実行します。

```
~]# yum install vlock
```

インストール後は、新たなパラメーターなしで **vlock** コマンドを使えば、コンソールセッションはすべてロックできます。このコマンドでは、アクティブな仮想コンソールをロックしますが、他のセッションへのアクセスは可能です。ワークステーション上のすべての仮想コンソールへのアクセスを防止するには、以下のコマンドを実行します。

```
vlock -a
```

この場合、**vlock** がアクティブなコンソールをロックし、**-a** オプションが他の仮想コンソールへのスイッチを防ぎます。

詳細は **vlock(1)** man ページを参照してください。



重要

Red Hat Enterprise Linux 7 で現在利用可能な **vlock** のバージョンに関する既知の問題がいくつかあります。

- このプログラムでは現在、**root** パスワードを使ったコンソールのロック解除ができません。詳細は [BZ#895066](#) を参照してください。
- コンソールをロックしても、画面およびスクロールバックバッファを削除しないので、それまでのコマンドやコンソールで表示された出力が、ワークステーションに物理的アクセスできれば誰でも見られることになります。詳細については、[BZ#807369](#) を参照してください。

4.1.4. リムーバブルメディアの読み取り専用マウントの強制

リムーバブルメディア（USB フラッシュディスクなど）の読み取り専用マウントを強制するために、管理者は **udev** ルールを使用してリムーバブルメディアを検出し、**blockdev** ユーティリティを使用してリムーバブルメディアを読み取り専用でマウントするよう設定できます。物理メディアの読み取り専用マウントの強制に必要な作業はこれだけです。

blockdev を使用したリムーバブルメディアの読み取り専用マウントの強制

すべてのリムーバブルメディアを読み取り専用でマウントするには、`/etc/udev/rules.d/` ディレクトリ内に新しい **udev** 設定ファイル (**80-readonly-removables.rules** など) を以下の内容で作成します。

```
SUBSYSTEM=="block",ATTRS{removable}=="1",RUN{program}="/sbin/blockdev --setro %N"
```

上記の **udev** ルールにより、**blockdev** ユーティリティーを使用して、新しく接続されたすべてのリムーバブルブロック (ストレージ) デバイスが自動的に読み取り専用で設定されます。

新しい udev 設定の適用

これらの設定を有効にするには、新しい **udev** ルールを適用する必要があります。**udev** サービスは設定ファイルの変更を自動的に検出しますが、新しい設定は既存のデバイスに適用されません。新しく接続されたデバイスのみが新しい設定の影響を受けます。したがって、新しい設定をリムーバブルディスクに適用するには、接続されたすべてのリムーバブルメディアをアンマウントし、取り外す必要があります。

すべてのルールを既存のデバイスに強制的に再適用するよう **udev** を設定するには、**root** で以下のコマンドを実行します。

```
~# udevadm trigger
```

上記のコマンドを使用して **udev** によってすべてのルールを強制的に再適用しても、すでにマウントされているストレージデバイスは影響を受けません。

udev によってすべてのルールを強制的にリロードするには (何らかの理由で新しいルールが自動的に検出されない場合)、以下のコマンドを使用します。

```
~# udevadm control --reload
```

4.2. ROOT アクセスの制御

ホームとなるマシンを管理する際に、ユーザーは **root** ユーザーとして、または **sudo** や **su** などの **setuid** プログラムで有効な **root** 権限を取得して一部のタスクを実行する必要があります。**setuid** プログラムは、プログラムを実行しているユーザーではなく、プログラムの所有者のユーザー ID (**UID**) で実行されるプログラムです。これらのプログラムは、以下の例のように、ロング形式のリストの所有者セクションにある **s** で示されます。

```
~]$ ls -l /bin/su
-rwsr-xr-x. 1 root root 34904 Mar 10 2011 /bin/su
```



注記

s は大文字または小文字の場合があります。大文字の場合は、基になるパーミッションビットが設定されていないことを示します。

しかし、組織のシステム管理者は、組織のユーザーが自身のマシンにどの程度の管理アクセスを持つかを決定しなくてはなりません。**pam_console.so** と呼ばれる **PAM** モジュールを使うと、再起動やリムーバブルメディアのマウントなど通常は **root** ユーザーにのみ許可されているアクティビティーを、物理コンソールに最初にログインしたユーザーが実行できるようになります。しかし、ネットワーク設定の変更、新たなマウスの設定、ネットワークデバイスのマウントといったシステム管理者の他の重要なタスクは、管理者権限なしでは実行できません。したがって、システム管理者はネットワーク上のユーザーにどの程度のアクセスを許可するかを判断する必要があります。

4.2.1. Root アクセスの拒否

上記またはその他の理由でユーザーが **root** でログインすることに管理者が不安を感じる場合、**root** パスワードは秘密にし、ランレベル 1 へのアクセスやシングルユーザーモードへのアクセスをブートローダーパスワード保護で拒否してください（このトピックに関する詳細については「[ブートローダーのセキュア化](#)」を参照）。

管理者は、以下の 4 つの方法で **root** ログインを拒否できます。

Changing the root shell

ユーザーが直接 **root** としてログインしないように、システム管理者は **/etc/passwd** ファイルで **root** アカountのシェルを **/sbin/nologin** に設定できます。

表4.2 Root シェルの無効化

影響あり	影響なし
<p>root シェルへのアクセスを阻止し、そのようなアクセス試行をログに記録します。以下のプログラムは root アカountにアクセスできません。</p> <ul style="list-style-type: none"> • login • gdm • kdm • xdm • su • ssh • scp • sftp 	<p>FTP クライアント、メールクライアント、多くの setuid プログラムなどのシェルを必要としないプログラム。以下のプログラムは root アカountにアクセスできません。</p> <ul style="list-style-type: none"> • sudo • FTP クライアント • Email クライアント

Disabling root access via any console device (tty)

root アカountへのアクセスをさらに制限するために、管理者は **/etc/securetty** ファイルを編集してコンソールでの **root** ログインを無効にすることができます。このファイルには、**root** ユーザーがログイン可能なすべてのデバイスが一覧表示されます。このファイル自体がない場合、**root** ユーザーはシステム上のどんな通信デバイス（コンソールか raw ネットワークインターフェースか）に関係なく）からでもログインできます。この状態は危険です。ユーザーは Telnet 経由で **root** としてマシンにログインでき、プレーンテキストのパスワードがネットワーク上で送信されてしまうからです。

デフォルトでは、Red Hat Enterprise Linux 7 の **/etc/securetty** ファイルにより、**root** ユーザーのみがマシンに物理的に接続されたコンソールでログインできます。**root** ユーザーによるログインを防ぐには、**root** でシェルプロンプトから以下のコマンドを入力して、このファイルの内容を削除します。

```
echo > /etc/securetty
```

KDM、GDM、XDM のログインマネージャーでの **securetty** サポートを有効にするには、以下の行を追加します。

```
auth [user_unknown=ignore success=ok ignore=ignore default=bad]
pam_securetty.so
```

追加対象のファイルは以下のとおりです。

- /etc/pam.d/gdm
- /etc/pam.d/gdm-autologin
- /etc/pam.d/gdm-fingerprint
- /etc/pam.d/gdm-password
- /etc/pam.d/gdm-smartcard
- /etc/pam.d/kdm
- /etc/pam.d/kdm-np
- /etc/pam.d/xdm



警告

/etc/securetty ファイルを空白にしても、**root** ユーザーがリモートでツールの OpenSSH スイートを使用してログインすることは**止められません**。これは、認証が終わるまでコンソールが開けないためです。

表4.3 Root ログインの無効化

影響あり	影響なし
<p>コンソールまたはネットワーク経由での root アカウントへのアクセスを阻止します。以下のプログラムは root アカウントにアクセスできません。</p> <ul style="list-style-type: none">• login• gdm• kdm• xdm• tty を開く その他のネットワークサービス	<p>root としてログインしないが、setuid または別のメカニズムで管理タスクを実行するプログラム。以下のプログラムは root アカウントにアクセスできません。</p> <ul style="list-style-type: none">• su• sudo• ssh• scp• sftp

Disabling root SSH logins

SSH プロトコル経由での **root** ログインを防ぐには、SSH デーモンの設定ファイルである **/etc/ssh/sshd_config** を編集し、

```
#PermitRootLogin yes
```

の行を以下のように変更します。

```
PermitRootLogin no
```

表4.4 Root SSH ログインの無効化

影響あり	影響なし
<p>ツールの OpenSSH スイート経由での root アクセスを防ぎます。以下のプログラムは root アカウントにアクセスできません。</p> <ul style="list-style-type: none"> • ssh • scp • sftp 	<p>ツールの OpenSSH スイートの一部ではないプログラム</p>

Using PAM to limit root access to services

PAM (`/lib/security/pam_listfile.so` モジュール経由) を使用すると、特定アカウントを柔軟に拒否できます。管理者はこのモジュールを使用してログインが許可されないユーザーリストを参照できます。システムサービスへの **root** アクセスを制限するには、`/etc/pam.d/` ディレクトリ内の対象サービスのファイルを編集し、認証に **pam_listfile.so** モジュールが必要となるようにします。

以下の例では、`/etc/pam.d/vsftpd` PAM 設定ファイルの **vsftpd** FTP サーバーにこのモジュールを使用しています（一行目の末にある `\` の文字は、ディレクティブが一行の場合は必要ありません）。

```
auth    required    /lib/security/pam_listfile.so    item=user \
sense=deny file=/etc/vsftpd.ftpusers onerr=succeed
```

これにより、PAM が `/etc/vsftpd.ftpusers` ファイルを見て、リストに載っているユーザーにサービスへのアクセスを拒否するように指示します。管理者はこのファイル名を変更し、各サービスごとに別個のリストを維持したり、複数サービスへのアクセスを拒否する集中リストを使用したりすることができます。

管理者が複数サービスへのアクセスを拒否したい場合、メールクライアントでは `/etc/pam.d/pop` や `/etc/pam.d/imap` ファイルに、SSH クライアントでは `/etc/pam.d/ssh` ファイルのような PAM 設定ファイルに同様の行を追加することができます。

PAM についての詳細情報は、『The Linux-PAM System Administrator's Guide』を参照してください。`/usr/share/doc/pam-<version>/html/` ディレクトリにあります。

表4.5 PAM を使った root の無効化

影響あり	影響なし
<p>PAM 対応のネットワークサービスへの root アクセスを防ぎます。以下のサービスは root アカウントにアクセスできません。</p> <ul style="list-style-type: none"> • login • gdm • kdm • xdm • ssh • scp • sftp • FTP クライアント • Email クライアント • すべての PAM 対応サービス 	<p>PAM 非対応のプログラムおよびサービス</p>

4.2.2. Root アクセスの許可

組織内のユーザーが信頼できコンピューターの知識を持っていれば、**root** アクセスを許可することは問題ではありません。ユーザーに **root** アクセスを許可すれば、デバイスの追加やネットワークインターフェースの設定などの重要性の低いアクティビティを個別のユーザーが行うことができるので、システム管理者はネットワークセキュリティや他の重要な問題に専念できます。

一方で、個別のユーザーに **root** アクセスを許可すると、以下のような問題が発生することがあります。

- **マシンの誤った設定** — **root** アクセスを持つユーザーは自身のマシンを間違えて設定する可能性があり、この問題を解決するには助けを必要とします。ひどい場合には、知らないうちにセキュリティホールを開いてしまう可能性もあります。
- **安全でないサービスの実行** — **root** アクセスを持つユーザーは、FTP や Telnet といった安全でないサービスを自身のマシン上で実行して、ユーザー名やパスワードを危険にさらす可能性があります。これらのサービスは、ユーザー名やパスワードをプレーンテキストでネットワーク上で送信します。
- **Email の添付ファイルを root として実行** — 滅多にないことですが、Linux に影響を及ぼす Email ウイルスも存在します。悪意のあるプログラムは、**root** ユーザーとして実行された場合に脅威となります。
- **監査証跡がそのままになる** — 多くの場合、**root** アカウントは複数のユーザーが共有し、複数のシステム管理者がシステムのメンテナンスをできるようになっているため、ある時点でこのうちのどのユーザーが **root** だったかを見分けることは不可能です。別のログインを使用すると、ユーザーがログインしたアカウントとセッション追跡目的の一意の番号がタスク構造に組み込まれ、ユーザーが開始する各プロセスによって継承されます。同時ログインを使用すると、一意の番号は特定ログインのアクションの追跡に使用できます。アクションが監査イベントを生成する際には、ログインアカウントとその一意の番号に関連付けられているセッション

とともに記録されます。これらのログインとセッションを表示するには、**aulast** コマンドを使用します。**aulast** コマンドの **--proof** オプションを使うと、特定のセッションで生成された監査可能なイベントを切り離す特定の **ausearch** クエリーの提示が可能になります。監査システムの詳細については、[5章システム監査](#)を参照してください。

4.2.3. Root アクセスの制限

root ユーザーへのアクセスを完全に拒否するのではなく、管理者が **su** や **sudo** などの **setuid** プログラム経由でのアクセスのみを許可したい場合もあるかもしれません。**su** および **sudo** の詳細については、Red Hat Enterprise Linux 7 システム管理者のガイドの章 [権限の取得](#) と、**su(1)** および **sudo(8)** の man ページを参照してください。

4.2.4. 自動ログアウトの有効化

root としてログインしている場合に、このセッションを無人状態にしておく、重大なセキュリティリスクをもたらす恐れがあります。このリスクを軽減するために、一定期間が経過してからアイドル状態のユーザーを自動的にログアウトするようにシステムを設定することができます。

1. **root** として **/etc/profile** ファイルの先頭に以下の行を追加して、このファイルの処理が中断されないようにします。

```
trap "" 1 2 3 15
```

2. **root** として、**/etc/profile** ファイルに以下の行を挿入して、120 秒後に自動的にログアウトされるようにします。

```
export TMOUT=120
readonly TMOUT
```

TMOUT 変数は、指定した期間 (秒) に活動がない場合にはシェルを中断します (上記の例では **120**)。特定のインストールのニーズに応じて時間制限を変更してください。

4.2.5. ブートローダーのセキュア化

Linux ブートローダーをパスワードで保護する主要な理由は以下のとおりです。

1. シングルユーザーモードへのアクセスを防ぐー 攻撃者がシングルユーザーモードでシステムを起動できると、攻撃者は **root** パスワードを求められずに自動的に **root** としてログインします。



警告

/etc/sysconfig/init ファイルで **SINGLE** パラメーターを編集してシングルユーザーモードへのアクセスを保護する方法は推奨されません。攻撃者は、GRUB 2 のカーネルコマンドライン上で (**init=** を使って) カスタム初期コマンドを特定することでパスワードを迂回できます。Red Hat Enterprise Linux 7 システム管理者のガイドの [GRUB 2 パスワードの保護](#) の章にあるように、GRUB 2 ブートローダーをパスワードで保護することが推奨されます。

2. **GRUB 2 コンソールへのアクセスを防ぐ**— マシンがブートローダーに GRUB 2 を使用している場合、攻撃者は GRUB 2 エディターインターフェースを使って設定を変更したり、**cat** コマンドを使用して情報収集ができるようになります。
3. **安全でないオペレーティングシステムへのアクセスを防ぐ**— デュアルブートシステムの場合、攻撃者はアクセス制御やファイルパーミッションを無視する (例えば、DOS などの) OS を起動時に選択できます。

Intel 64 および AMD64 プラットホームの Red Hat Enterprise Linux 7 では、GRUB 2 ブートローダーが同梱されています。GRUB 2 の詳細については、Red Hat Enterprise Linux 7 システム管理者のガイドの [GRUB 2 ブートローダーの操作](#) の章を参照してください。

4.2.5.1. インタラクティブスタートアップの無効化

起動順序の最初に **I** キーを押すと、インタラクティブな方法でシステムを起動できます。この方法では、システムがユーザーにサービスを 1 つずつ開始するように求めます。しかし、システムに物理的なアクセスがある攻撃者の場合は、この方法だとセキュリティ関連のサービスを無効にし、システムへのアクセスを許すことになってしまいます。

ユーザーがシステムを対話的に起動しないようにするには、**root** として **/etc/sysconfig/init** ファイルの **PROMPT** パラメーターを以下のように無効にします。

```
PROMPT=no
```

4.2.6. ハードリンクおよびシンボリックリンクの保護

保護されていないハードリンクおよびシンボリックリンクにより引き起こされた脆弱性を悪意のあるユーザーが利用しないようにするために、Red Hat Enterprise Linux 7 には、特定の条件を満たす場合のみリンクの作成または実行を許可する機能が含まれます。

ハードリンクの場合は、以下のいずれかに該当する必要があります。

- ユーザーがリンク先のファイルを所有します。
- ユーザーがリンク先のファイルへの読み取りアクセスおよび書き込みアクセスをすでに持っています。

シンボリックリンクの場合、プロセスは、スティッキービットが設定された誰でも書き込みが可能なディレクトリーの外にいるとき、または以下のいずれかの条件を満たすときにのみリンクを実行することが許可されます。

- シンボリックリンクを実行するプロセスはシンボリックリンクの所有者です。
- ディレクトリーの所有者はシンボリックリンクの所有者と同じです。

この保護はデフォルトで有効になり、**/usr/lib/sysctl.d/50-default.conf** ファイルの以下のオプションで制御されます。

```
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
```

デフォルトの設定を上書きし、保護を無効にするには、**51-no-protect-links.conf** などの名前の新しい設定ファイルを以下の内容で **/etc/sysctl.d/** ディレクトリーに作成します。

```
fs.protected_hardlinks = 0
fs.protected_symlinks = 0
```



注記

デフォルトのシステム設定を上書きするには、新しい設定ファイルの拡張子を **.conf** にし、デフォルトのシステムファイルの**後**にその設定ファイルを読み取る必要があります (ファイルは辞書の順序で読み取られるため、ファイル名の最初の番号が大きいファイルに含まれる設定が優先されます)。

See the `sysctl.d(5)` manual page for more detailed information about the configuration of kernel parameters at boot using the **sysctl** mechanism.

4.3. サービスのセキュア化

組織内のシステム管理者にとっては管理機能へのユーザーアクセスは重要な問題ですが、どのネットワークサービスがアクティブかを監視することは、Linux システムのすべての管理、運営担当者にとっての最重要事項です。

Red Hat Enterprise Linux 7 における多くのサービスは、ネットワークサーバーとして動作します。マシン上でネットワークサービスが稼働中であれば、(デーモンと呼ばれる) サーバーアプリケーションが 1 つ以上のネットワークポート上での接続をリッスンします。これらの各サーバーは、潜在的な攻撃の接近手段として扱われる必要があります。

4.3.1. サービスへのリスク

ネットワークサービスは Linux システムに多くのリスクを与える可能性があります。主な問題を以下に挙げます。

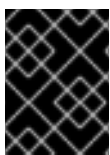
- **サービス拒否攻撃 (DoS)** — サービス拒否攻撃は、サービスに対して要求を大量に送信することでシステムがすべての要求をログ記録・応答しようとし、使用不可能になります。
- **分散型サービス拒否攻撃 (DDoS)** — これは DoS 攻撃の一種で、複数の脆弱なマシンを使用し (通常、数千以上)、サービスに対して一斉攻撃を仕掛けます。大量の要求が送信され、サービスを使用不可能にしてしまいます。
- **スクリプトの脆弱性への攻撃** — Web サーバーが通常行うように、サーバー全体のアクションにサーバーがスクリプトを使用している場合、攻撃者は誤って書かれたスクリプトを狙うことができます。このスクリプトの脆弱性に対する攻撃により、バッファがオーバーフロー状態になるか、攻撃者がシステム上のファイルを変更できる可能性があります。
- **バッファオーバーフロー攻撃** — ポート 1~1023 でリッスンするサービスを起動するには、管理権限を使用するか、**CAP_NET_BIND_SERVICE** 機能を設定する必要があります。プロセスがポートにバインドされ、そのポートでリッスンしている場合は、権限または機能が破棄されることがよくあります。権限または機能が破棄されず、アプリケーションに利用可能なバッファオーバーフローがある場合、攻撃者はデーモンを実行中のユーザーとしてシステムにアクセスすることができます。利用可能なバッファオーバーフローが存在するため、クラッカーは自動ツールを使って脆弱性のあるシステムを特定でき、アクセスを確保した後に自動ルートキットを使ってシステムへのアクセスを維持します。



注記

Red Hat Enterprise Linux 7 では、*ExecShield* と呼ばれる x86 互換のシングルおよびマルチプロセッサのカーネルがサポートする実行可能メモリのセグメント化および保護技術により、バッファオーバーフローの脆弱性における脅威は緩和されています。*ExecShield* は仮想メモリを実行可能なセグメントと非実行可能セグメントに分けることでバッファオーバーフローのリスクを下げます。(バッファオーバーフローエクスプロイトから注入された悪意のあるコードなど) 実行可能セグメント外で実行を試みるすべてのプログラムコードは、セグメント化の失敗を発生させ、終了します。

Execshield には AMD64 プラットフォーム上の *No eXecute (NX)* と Intel® 64 システムのサポートが含まれます。これらのテクノロジーが *ExecShield* と組み合わせることで、4 KB の実行可能コードという粒度で仮想メモリの実行可能な部分での悪意のあるコードの実行を防ぎます。これで、バッファオーバーフローエクスプロイトからの攻撃リスクを減らします。



重要

ネットワーク上での攻撃への露出を限定するために、使用していないサービスはすべてオフにすることが推奨されます。

4.3.2. サービスの特定および設定

セキュリティを強化するために、Red Hat Enterprise Linux 7 でインストールされているネットワークサービスのほとんどはデフォルトでオフとなっています。ただし、以下のものは例外となります。

- **cups** — Red Hat Enterprise Linux 7 のデフォルトのプリントサーバー
- **cups-lpd** — 代替プリントサーバー
- **xinetd** — **gssftp** や **telnet** などの下位サーバーへの接続を管理するスーパーサーバー
- **sshd** — Telnet の安全な代替となる OpenSSH サーバー

これらサービスの稼働を継続しておくかどうかを判断する際は、常識にしたがってリスクを避けるのが最善策です。例えば、プリンターが利用できない場合は **cups** を無効にします。同じことは **portreserve** についても言えます。NFSv3 ボリュームをマウントしていなかったり、NIS(y²bind サービス)を使用しないのであれば、**rpcbind** を無効にすべきです。ブート時にどのネットワークサービスが開始可能になっているかを確認するだけでは、十分ではありません。どのポートがオープンでリッスンしているかを確認することが推奨されます。詳細情報は、「[リッスンしているポートの確認](#)」を参照してください。

4.3.3. 安全でないサービス

潜在的にはどのネットワークサービスも安全ではありません。未使用のサービスをオフにすることが重要なのは、このためです。サービスのエクスプロイトは定期的に発見され修正プログラムが提供されているので、ネットワークサービスはどんなものでも関連するパッケージを定期的に更新することが非常に重要です。詳細は [3章 システムを最新の状態に保つ](#) を参照してください。

ネットワークプロトコルの中には、もともと他のものよりも安全性が低いものがあります。以下の動作を実行するサービスがそれに当たります。

- 暗号化されていないネットワークでユーザー名やパスワードを送信する — Telnet や FTP など多くの古いプロトコルは認証セッションを暗号化しないので、できるだけ避けてください。

- 暗号化されていないネットワークで機密性の高いデータを送信するー Telnet や FTP、HTTP、SMTP など多くのプロトコルでは暗号化されていないネットワークでデータを送信します。また、NFS や SMB などの多くのネットワークファイルシステムでも暗号化されていないネットワークで情報を送信します。これらのプロトコルを使用する際に送信するデータの種別を制限することは、ユーザーの責任になります。

もともと安全性が低いサービスには、**rlogin**、**rsh**、**telnet**、**vsftpd** などがあります。

リモートログインおよびシェルプログラム (**rlogin**、**rsh**、**telnet**) はすべて避けて、**SSH** を使用するようにしてください。**sshd** に関する詳細については、「[SSH のセキュア化](#)」を参照してください。

FTP は、システムのセキュリティにとってリモートシェルほど危険ではありませんが、問題を回避するには **FTP** サーバーを慎重に設定し、監視する必要があります。**FTP** サーバーのセキュア化に関する詳細については、「[FTP のセキュア化](#)」を参照してください。

実装時に注意が必要で、ファイアウォールの背後に配置する必要があるサービスは以下のものです。

- **auth**
- **nfs-server**
- **smb** および **nbm** (Samba)
- **yppasswdd**
- **ypserv**
- **ypxfrd**

ネットワークサービスの安全性を高めるための詳細情報は、「[ネットワークアクセスのセキュア化](#)」を参照してください。

4.3.4. rpcbind のセキュア化

rpcbind サービスは、NIS や NFS などの RPC サービス用の動的なポート割り当てデーモンです。この認証メカニズムは脆弱なもので、制御対象のサービスに幅広いポートを割り当てる機能があります。このため、セキュア化は困難になります。



注記

NFSv4 は **rpcbind** を必要としなくなったので、**portmap** のセキュア化が影響するのは NFSv2 と NFSv3 のみです。NFSv2 もしくは NFSv3 サーバーの導入を計画する場合は、**rpcbind** が必要となり、以下のセクションが適用されます。

RPC サービスを実行している場合は、以下の基本的なルールにしたがってください。

4.3.4.1. TCP Wrapper による rpcbind の保護

rpcbind にはビルトインの認証がないので、TCP Wrapper を使用して **rpcbind** サービスにアクセスするネットワークやホストを制限することが重要です。

さらに、サービスへのアクセスを制限する際には、IP アドレス **のみ** を使用してください。ホスト名は DNS ポイズニングやその他の方法で偽造される恐れがあるので、使用しないでください。

4.3.4.2. firewallld による rpcbind の保護

rpcbind サービスへのアクセスをさらに制限するには、サーバーに**firewalld** ルールを追加し、特定ネットワークへのアクセスを制限するとよいでしょう。

以下は、**firewalld** リッチ言語コマンドの 2 つの例です。最初のコマンドは 192.168.0.0/24 ネットワークからポート 111 (**rpcbind** サービスが使用) への TCP 接続を許可します。2 つ目のコマンドは、ローカルホストからの同一ポートへの接続を許可します。他のパケットはすべて遮断されます。

```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" port port="111"
protocol="tcp" source address="192.168.0.0/24" invert="True" drop'
~]# firewall-cmd --add-rich-rule='rule family="ipv4" port port="111"
protocol="tcp" source address="127.0.0.1" accept'
```

UDP トラフィックを同様に制限するには、以下のコマンドを使用します。

```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" port port="111"
protocol="udp" source address="192.168.0.0/24" invert="True" drop'
```



注記

設定を永続的にするには、**--permanent** を **firewalld** リッチ言語コマンドに追加します。ファイアウォールの実装に関する詳細情報は、「[ファイアウォールの使用](#)」を参照してください。

4.3.5. rpc.mountd のセキュア化

rpc.mountd デーモンは、NFS バージョン 2 (『RFC 1904』) と NFS バージョン 3 (『RFC 1813』) で使用されているプロトコルである NFS MOUNT プロトコルのサーバーサイドを実装します。

RPC サービスを実行している場合は、以下の基本的なルールにしたがってください。

4.3.5.1. TCP Wrappers による rpc.mountd の保護

rpc.mountd には認証が組み込まれていないため、TCP Wrapper を使用して**rpc.mountd** サービスにアクセスするネットワークやホストを制限することが重要です。

さらに、サービスへのアクセスを制限する際には、**IP アドレスのみ** を使用してください。ホスト名は DNS ポイズニングやその他の方法で改ざんできるため、使用しないでください。

4.3.5.2. firewalld による rpc.mountd の保護

rpc.mountd サービスへのアクセスをさらに制限するには、サーバーに**firewalld** リッチ言語ルールを追加し、特定のネットワークへのアクセスを制限します。

以下は、**firewalld** リッチ言語コマンドの 2 つの例です。最初のコマンドでは、**192.168.0.0/24** ネットワークからの **mountd** 接続を許可します。2 つ目のコマンドでは、ローカルホストからの **mountd** 接続を許可します。他のパケットはすべて遮断されます。

```
~]# firewall-cmd --add-rich-rule 'rule family="ipv4" source NOT
address="192.168.0.0/24" service name="mountd" drop'
~]# firewall-cmd --add-rich-rule 'rule family="ipv4" source
address="127.0.0.1" service name="mountd" accept'
```



注記

設定を永続的にするには、**--permanent** を **firewalld** リッチ言語コマンドに追加します。ファイアウォールの実装に関する詳細情報は、「[ファイアウォールの使用](#)」を参照してください。

4.3.6. NIS のセキュア化

ネットワーク情報サービス (NIS) は **ypserv** と呼ばれる RPC サービスの一つで、**rpcbind** および他の関連サービスと一緒に使用することで、ドメイン内にあると主張するすべてのコンピューターに、ユーザー名やパスワード、他の機密性のある情報のマップを配布します。

NIS サーバーは、以下のものを含むいくつかのアプリケーションで構成されています。

- **/usr/sbin/rpc.yppasswdd** — **yppasswdd** サービスとも呼ばれます。このデーモンを使用することで、ユーザーは NIS パスワードを変更できます。
- **/usr/sbin/rpc.ypxfrd** — **ypxfrd** サービスとも呼ばれます。このデーモンは、ネットワーク上での NIS マップ転送を担当します。
- **/usr/sbin/ypserv** — これは NIS サーバーデーモンです。

NIS は今日の基準ではあまり安全なものではありません。ホスト認証メカニズムがなく、パスワードハッシュを含むすべての情報をネットワーク上で暗号化せずに送信します。このため、NIS を使用するネットワークの設定時には、非常に注意深い作業が必要になります。さらに、NIS のデフォルト設定がもともと安全でないことで複雑性が増してしまいます。

NIS サーバーの実装を予定している場合は、「[rpcbind のセキュア化](#)」に概説があるようにまず **rpcbind** サービスのセキュア化を図ることが推奨されます。その後に、以下のようなネットワークプログラミングの問題などに対処してください。

4.3.6.1. ネットワークの注意深いプログラミング

NIS は機密性の高い情報を暗号化せずにネットワーク上で送信するので、ファイアウォールの背後で、またセグメント化された安全なネットワーク上で実行することが重要です。NIS 情報が安全でないネットワーク上で送信される際は、常に傍受される危険があります。ネットワークを注意深く設計することで、重大なセキュリティ侵害の防止に役立ちます。

4.3.6.2. パスワードのような NIS ドメイン名およびホスト名を使用する

ユーザーが NIS サーバーの DNS ホスト名と NIS ドメイン名を知っていれば、NIS ドメイン内のマシンはどれもコマンドを使用して認証なしにサーバーから情報を引き出すことができます。

例えば、だれかがノートパソコンをネットワークに接続するか、外部からネットワークに侵入すると (そして内部 IP アドレスにスプーフィングできたとすると)、以下のコマンドが **/etc/passwd** マップを公開します。

```
ypcat -d <NIS_domain> -h <DNS_hostname> passwd
```

この攻撃者が root ユーザーであった場合、以下のコマンドを入力して **/etc/shadow** ファイルを入手することが可能です。

```
ypcat -d <NIS_domain> -h <DNS_hostname> shadow
```



注記

Kerberos を使用していれば、**/etc/shadow** ファイルは NIS マップ内に保存されません。

攻撃者による NIS マップへのアクセスをより困難にするには、**o7hfawtgmhgw.domain.com** のようなランダムな文字列を DNS ホスト名にします。同様に、異なるランダムな NIS ドメイン名を作成します。これにより、攻撃者は NIS サーバーへのアクセスが非常に困難になります。

4.3.6.3. /var/yp/securenets ファイルを編集する

/var/yp/securenets ファイルが空白もしくは存在しない場合 (デフォルトインストールの後の場合のように)、NIS はすべてのネットワークをリッスンします。最初にするもののひとつは、ネットマスクとネットワークのペアをファイルに置くことで、これにより **ypserv** は適正なネットワークからの要求のみに応答するようになります。

以下は **/var/yp/securenets** ファイルからのエントリのサンプルです。

```
255.255.255.0      192.168.0.0
```



警告

/var/yp/securenets ファイルを作成せずに NIS サーバーを初回起動することは、絶対にしないでください。

このテクニックは IP スプーフィングからの保護は提供しませんが、少なくとも NIS サーバーが対応するネットワークに対して制限をかけます。

4.3.6.4. 静的ポートの割り当てとリッチ言語ルールの使用

NIS に関連付けられているサーバーは、**rpc.yppasswdd** を除いて特定のポートの割り当てが可能です。このデーモンを使うと、ユーザーは自身のログインパスワードを変更できるようになります。**rpc.ypxfrd** と **ypserv** の 2 つの NIS サーバーデーモンにポートを割り当てると、NIS サーバーデーモンをさらに侵入者から保護するためのファイアウォールルールが作成できます。

これを実行するには、以下の行を **/etc/sysconfig/network** に追加します。

```
YPSERV_ARGS="-p 834"
YPXFRD_ARGS="-p 835"
```

以下のリッチ言語の **firewalld** ルールを使って、これらのポートでサーバーがリッスンするネットワークを強制できます。

```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" source
address="192.168.0.0/24" invert="True" port port="834-835" protocol="tcp"
drop'
```



```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" source
address="192.168.0.0/24" invert="True" port port="834-835" protocol="udp"
drop'
```

つまり、**192.168.0.0/24** ネットワークからの要求であれば、サーバーはポート 834 および 835 への接続のみを許可することになります。最初のルールは **TCP** のもので、2 つ目は **UDP** になります。



注記

iptables コマンドによるファイアウォール実装についての詳細情報は、「[ファイアウォールの使用](#)」を参照してください。

4.3.6.5. Kerberos 認証を使用する

認証に NIS を使用する際に考慮すべきことの一つは、ユーザーがマシンにログインする際は常に、**/etc/shadow** マップからのパスワードハッシュがネットワーク上で送信されるということです。侵入者が NIS ドメインへアクセスしてネットワークトラフィックを傍受した場合、ユーザー名とパスワードハッシュを取得できることになります。さらに時間があれば、攻撃者はパスワードクラッキングプログラムで脆弱なパスワードを推測し、ネットワーク上の有効なアカウントへのアクセスを取得できるようになります。

Kerberos は秘密鍵の暗号作成方法を使用するので、パスワードハッシュがネットワーク上で送信されることはなく、システムを大幅に安全なものとしします。Kerberos についての詳細は、Linux Domain Identity, Authentication, and Policy Guide の [Authentication: Kerberos KDC](#) のセクションを参照してください。

4.3.7. NFS のセキュア化



重要

NFS トラフィックは全バージョンで TCP を使用して送信することが可能で、UDP ではなく NFSv3 を使って送信してください。NFS のバージョンはすべて、**RPCSEC_GSS** カーネルモジュールの一部として Kerberos ユーザーおよびグループ認証をサポートしています。Red Hat Enterprise Linux 7 では **rpcbind** を使用する NFSv3 がサポートされているので、**rpcbind** に関する情報も引き続き含まれています。

4.3.7.1. ネットワークの注意深いプランニング

NFSv2 と NFSv3 ではこれまで、データの受け渡しは安全に行われていませんでした。今では NFS の全バージョンで Kerberos を使った通常のファイルシステム操作の認証（およびオプションで暗号化）ができます。NFSv4 では、すべての操作で Kerberos の使用が可能です。v2 または v3 では、ファイルロックとマウントにまだ Kerberos が使用されていません。NFSv4 使用時は、クライアントが NAT もしくはファイアウォールの背後にあるのであれば、委任はオフにすることができます。NFSv4.1 を使って NAT およびファイアウォールを通じた委任による操作を可能にする方法については、Red Hat Enterprise Linux 7 ストレージ管理ガイドの [pNFS](#) のセクションを参照してください。

4.3.7.2. NFS マウントオプションのセキュア化

/etc/fstab ファイル内での **mount** コマンド使用については、Red Hat Enterprise Linux 7 ストレージ管理ガイドの [mount コマンドの使い方](#) の章で説明されています。セクション管理の観点からは、NFS マウントオプションは **/etc/nfsmount.conf** でも指定可能であることは注目に値します。これを使うと、カスタムのデフォルトオプションを設定することが可能です。

4.3.7.2.1. NFS サーバーのレビュー



警告

ファイルシステムをエクスポートする際は、全体のエクスポートのみを行なってください。ファイルシステムのサブディレクトリーをエクスポートすると、セキュリティ問題につながる可能性があります。場合によってはクライアントがファイルシステムのエクスポートされた部分から抜け出し、エクスポートされていない部分に至ることもあります (**exports(5)** man ページのサブツリーチェックのセクションを参照)。

マウント済みファイルシステムへの書き込み可能なユーザー数を減らすためには、可能な場合は常に **ro** オプションを使用してファイルシステムを読み取り専用としてエクスポートしてください。**rw** オプションの使用は、明確に必要な場合のみとしてください。詳細は **exports(5)** man ページを参照してください。書き込みアクセスを許可すると、シンボリックリンク攻撃などのリスクが高まります。これには、**/tmp** や **/usr/tmp** などの一時ディレクトリーが含まれます。

ディレクトリーを **rw** オプションでマウントする必要がある場合は、リスク低減のためにできる限り全ユーザー書き込み可能としないようにします。アプリケーションのなかにはパスワードをクリアテキストで保存したり暗号化が弱いものもあるので、ホームディレクトリーのエクスポートもリスクとみなされます。このリスクは、アプリケーションコードがレビューされ、改善されることで軽減されてきています。SSH キーにパスワードを設定しないユーザーもいるので、ホームディレクトリーがリスクをもたらすことになります。パスワードや Kerberos の使用を強制することで、このリスクは緩和されます。

エクスポートはアクセスを必要とするクライアントのみとしてください。NFS サーバーで **showmount -e** コマンドを使用して、サーバーが何をエクスポートしているかを確認します。特に必要でないものはエクスポートしないでください。

no_root_squash オプションは使用しないでください。また、既存のインストールのレビューを行い、これが使用されていないことを確認してください。詳細は「[no_root_squash オプションは使用しないでください](#)」を参照してください。

secure オプションはサーバー側のエクスポートオプションで、「予約済み」ポートへのエクスポートを制限する際に使用します。デフォルトでは、サーバーは「予約済み」ポート (ポート番号 1024 未満のもの) からのクライアント通信のみを許可します。これは、クライアントが通常これらのポートの使用を許可するのは「信頼できる」コード (カーネル内の NFS クライアントなど) のみだったためです。しかし、多くのネットワークではクライアント上で root になるのは難しいことではないので、予約済みポートからの通信が権限を伴うものであると仮定するのは安全ではありません。このため、予約済みポートへの制限は限定的な価値しかありません。Kerberos やファイアウォール、エクスポートを特定のクライアントに制限するという方法を信頼する方がより安全です。

ほとんどのクライアントは、未だに予約済みポートを使用しています (可能な場合)。ただし、予約済みポートは限定的なリソースなので、クライアント (特に NFS マウント数が多いもの) はより大きい番号のポートを使う選択をする場合もあります。Linux クライアントは、「nfs_vport」マウントオプションを使用してこれを行うことができます。エクスポートでこれを許可したい場合は、「insecure」エクスポートオプションで行うことができます。

ユーザーがサーバーにログインできないようにしておくのは、よい方法です。上記の NFS サーバー設定を確認する間に、誰および何がサーバーにアクセス可能かを確認してください。

4.3.7.2.2. NFS クライアントのレビュー

setuid プログラムを使用できないようにするには、**nosuid** オプションを使用します。**nosuid** オプションは **set-user-identifier** または **set-group-identifier** ビットを無効にします。これにより、リモートユーザーが **setuid** プログラムを実行してより高い権限を取得することを防ぎます。このオプションは、クライアントおよびサーバー側で使用してください。

noexec オプションはクライアント上のすべての実行可能ファイルを無効にします。共有しているファイルシステムにあるファイルをユーザーが不注意で実行しないようにこのオプションを使用します。**nosuid** および **noexec** オプションは、ほとんどのファイルシステムの標準オプションです。

nodev オプションを使うと、クライアントが「device-files」をハードウェアデバイスとして処理することを防ぎます。

resvport オプションはクライアント側のマウントオプションで、**secure** はこれに対応するサーバー側のエクスポートオプションです (上記の説明を参照)。これは「予約済みポート」への通信を制限します。予約済みまたは「よく知られた」ポートは、root ユーザーなどの権限のあるユーザーやプロセス用に確保されています。このオプションを設定すると、クライアントが予約済みソースのポートを使ってサーバーと通信するようになります。

NFS の全バージョンですでに Kerberos 認証に対応しています。これを有効にするマウントオプションは、次の通りです。**sec=krb5**

NFSv4 は、整合性には **krb5i** を、プライバシー保護には **krb5p** を使用して Kerberos によるマウントをサポートします。これらは **sec=krb5** でのマウント時に使用されますが、NFS サーバー上での設定が必要です。詳細はエクスポートに関する man ページ (**man 5 exports**) を参照してください。

NFS man ページ (**man 5 nfs**) には「SECURITY CONSIDERATIONS」セクションがあり、ここでは NFSv4 のセキュリティ強化の説明と NFS の特定のマウントオプションすべてが含まれています。

4.3.7.3. 構文エラーに注意

NFS サーバーは **/etc/exports** ファイルを参照して、エクスポートするファイルシステムとこれらのディレクトリーをエクスポートするホストを決定します。このファイルを編集する際は、無関係な領域を追加しないように注意してください。

例えば、**/etc/exports** ファイル内の以下の行は読み取り/書き込みパーミッションでディレクトリー **/tmp/nfs/** をホスト **bob.example.com** と共有します。

```
/tmp/nfs/      bob.example.com(rw)
```

一方、**/etc/exports** ファイルは同じディレクトリーを読み取り専用パーミッションでホスト **bob.example.com** と共有し、ホスト名の後ろに一文字分の空白があるので読み取り/書き込みパーミッションで **world** と共有します。

```
/tmp/nfs/      bob.example.com (rw)
```

showmount コマンドを使って何が共有されているかを検証するのは、設定済み NFS 共有をチェックするよい方法です。

```
showmount -e <hostname>
```

4.3.7.4. no_root_squash オプションは使用しないでください

デフォルトで NFS 共有は、root ユーザーを権限のないユーザーアカウントである **nfsnobody** ユーザーに変更します。これにより、root で作成された全ファイルの所有者は **nfsnobody** に変更されます。この変更で setuid ビットが設定されたプログラムのアップロードが防止されます。

no_root_squash を使用すると、リモートの root ユーザーは共有ファイルシステム上のどのファイルも変更できるようになり、トロイの木馬に感染したアプリケーションを他のユーザーが間違えて実行できる状態にしてしまいます。

4.3.7.5. NFS ファイアウォールの設定

Red Hat Enterprise Linux 7 のデフォルトの NFS は NFSv4 で、この場合は TCP でポート 2049 が開いていれば問題ありません。NFSv3 を使用していると、以下の説明にあるように、さらに 4 つのポートが必要になります。

NFSv3 用のポート設定

NFS に使用されるポートは rpcbind が動的に割り当てますが、ファイアウォールルールの作成時に問題を起こす恐れがあります。このプロセスを簡素化するには、**/etc/sysconfig/nfs** ファイルを使って使用するポートを特定します。

- **MOUNTD_PORT** — mountd (rpc.mountd) 用の TCP および UDP ポート
- **STATD_PORT** — status (rpc.statd) 用の TCP および UDP ポート
- **LOCKD_TCP** — nlockmgr (rpc.lockd) 用 TCP ポート
- **LOCKD_UDP** — nlockmgr (rpc.lockd) 用 UDP ポート

指定されたポート番号は、他のサービスが使用してはいけません。ポート番号の指定と TCP および UDP ポート 2049 (NFS) を許可するようにファイアウォールを設定してください。

NFS サーバーで **rpcinfo -p** コマンドを実行し、どのポートと RPC プログラムが使用されているかを確認します。

4.3.7.6. Red Hat Identity Management でのセキュアな NFS

Red Hat Enterprise Linux に含まれる Red Hat Identity Management を使用する環境では、Kerberos 対応の NFS 設定は大幅に簡素化することができます。

『[LINUX ドメイン ID、認証、およびポリシーガイド](#)』に従ってください。特に[Kerberos 対応の NFS サーバーの設定](#)を参照して、Red Hat Identity Management を使用して Kerberos で NFS のセキュリティを確保する方法を確認してください。

4.3.8. Apache HTTP サーバーのセキュア化

Apache HTTP サーバーは Red Hat Enterprise Linux 7 に同梱されているサービスのなかで最も安定性があり安全なものの一つです。Apache HTTP サーバーを安全にするには多くのオプションとテクニックがあり、ここで詳述するには多すぎるほどです。以下のセクションでは、Apache HTTP サーバー稼働時に実行できる優れた方法を簡単に説明します。

スクリプトを実稼働環境で実行する **前**に、常にそのシステムがシステム上で意図したとおりに稼働していることを確認してください。また、スクリプトもしくは CGI を含むディレクトリーに書き込みパーミッションを持っているのは root ユーザーのみであることを確認してください。これを行うには、root ユーザーで以下のコマンドを実行します。

```
chown root <directory_name>
```

```
chmod 755 <directory_name>
```

以下の設定オプションを (`/etc/httpd/conf/httpd.conf` 内の設定で) 使用する際は、システム管理者は注意してください。

FollowSymLinks

このディレクティブはデフォルトで有効となっているので、Web サーバーのドキュメントルートへのシンボリックリンク作成時には注意してください。例えば、`/` へのシンボリックリンクを提供することはよい方法ではありません。

Indexes

このディレクティブはデフォルトで有効となっていますが、これが最適ではない可能性があります。ビジターがサーバー上のファイル閲覧をできないようにするには、このディレクティブを削除します。

UserDir

UserDir はシステム上でのユーザーアカウントの有無を確認できるので、デフォルトでは無効となっています。サーバー上のユーザーディレクトリーのブラウジングを有効にするには、以下のディレクティブを使用します。

```
UserDir enabled
UserDir disabled root
```

これらのディレクティブは、`/root/` 以外のすべてのユーザーディレクトリーのブラウジングを有効にします。無効アカウントリストにユーザーを追加するには、**UserDir disabled** 行に空白で区切ったユーザーのリストを追加します。

サーバートークン

サーバートークン ディレクティブは、クライアントに返信されるサーバー応答ヘッダーフィールドを制御します。これには、以下のパラメーターを使用してカスタマイズできる種々の情報が含まれます。

- **ServerTokens Full** (デフォルトのオプション) — (OS の種類や使用されるモジュールなど) 利用可能なすべての情報を提供します。例えば、

```
Apache/2.0.41 (Unix) PHP/4.2.2 MyMod/1.2
```

- **ServerTokens Prod** または **ServerTokens ProductOnly** — 以下の情報を提供します。

```
Apache
```

- **ServerTokens Major** — 以下の情報を提供します。

```
Apache/2
```

- **ServerTokens Minor** — 以下の情報を提供します。

```
Apache/2.0
```

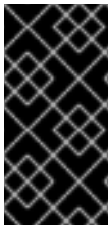
- **ServerTokens Min** または **ServerTokens Minimal** — 以下の情報を提供します。

Apache/2.0.41

- **ServerTokens OS** — 以下の情報を提供します。

Apache/2.0.41 (Unix)

潜在的な攻撃者がユーザーのシステムについて価値ある情報を取得できないようにするには、**ServerTokens Prod** の使用が推奨されます。



重要

IncludesNoExec ディレクティブは削除しないでください。デフォルトでは、*Server-Side Includes (SSI)* モジュールはコマンドの実行ができません。絶対に必要な場合以外は、この設定を変更しないことが推奨されます。変更すると、攻撃者がシステム上でコマンドを実行できるようになる可能性があります。

httpd モジュールの削除

特定のシナリオでは、特定の **httpd** モジュールを削除して HTTP サーバーの機能を制限した方がよい場合もあります。これは、`/etc/httpd/conf/httpd.conf` ファイルで削除したいモジュールをロードし行全体をコメントアウトするだけで行えます。たとえば、プロキシモジュールを削除するには、以下の行の先頭にハッシュ記号を加えてコメントアウトします。

```
#LoadModule proxy_module modules/mod_proxy.so
```

`/etc/httpd/conf.d/` ディレクトリーにもモジュールの読み込みに使われる設定ファイルが含まれていることに注意してください。

httpd および SELinux

詳細情報は、Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide の [The Apache HTTP Server and SELinux](#) の章を参照してください。

4.3.9. FTP のセキュア化

File Transfer Protocol (FTP) は旧式の TCP プロトコルで、ネットワーク上でファイル転送するために設計されています。ユーザー認証を含むサーバーとのトランザクションがすべて暗号化されないの、安全でないプロトコルとみなされ、慎重に設定する必要があります。

Red Hat Enterprise Linux 7 は以下の 2 つの FTP サーバーを提供します。

- **Red Hat Content Accelerator (tux)** — FTP 機能のあるカーネル空間の Web サーバーです。
- **vsftpd** — スタンドアロンでセキュリティ重視の FTP サービス実装です。

以下のセキュリティガイドラインは **vsftpd** FTP サービス設定のためのものです。

4.3.9.1. FTP グリーティングバナー

ユーザー名とパスワードの送信前に、すべてのユーザーにグリーティングバナーが示されます。デフォルトでは、このバナーにはシステムの脆弱性を特定しようとしているクラッカーに有益なバージョン情報が含まれています。

vsftpd のグリーティングバナーを変更するには、以下のディレクティブを **/etc/vsftpd/vsftpd.conf** ファイルに追加します。

```
ftpd_banner=<insert_greeting_here>
```

上記のディレクティブの **<insert_greeting_here>** をグリーティングメッセージのテキストで置き換えます。

複数行のバナーの場合、バナーファイルの使用が最善の方法となります。複数のバナーの管理を簡素化するには、**/etc/banners/** という新規ディレクトリーにすべてのバナーを格納します。この例では、FTP 接続のバナーファイルは、**/etc/banners/ftp.msg** となります。以下はこのファイルのサンプルになります。

```
##### Hello, all activity on ftp.example.com is logged. #####
```



注記

「[TCP Wrapper](#)および [xinetd](#) によるサービスのセキュア化」にあるように、各行を **220** で始める必要はありません。

vsftpd でこのバナーを参照するようにするには、以下のディレクティブを **/etc/vsftpd/vsftpd.conf** ファイルに追加します。

```
banner_file=/etc/banners/ftp.msg
```

「[TCP Wrapper](#) と接続バナー」にあるように、**TCP Wrapper** を使って着信接続に新たなバナーを送信することも可能です。

4.3.9.2. 匿名のアクセス

/var/ftp/ ディレクトリーが存在すると、匿名アカウントがアクティベートされます。

このディレクトリーを作成する最も簡単な方法は、**vsftpd** パッケージをインストールすることです。このパッケージは、匿名ユーザー向けのディレクトリーツリーを確立し、匿名ユーザーによる読み取り専用のパーミッションをディレクトリーに設定します。

デフォルトでは、匿名ユーザーはどのディレクトリーにも書き込みできません。



警告

FTP サーバーへの匿名アクセスを有効にする場合は、機密性の高いデータの保存場所に注意してください。

4.3.9.2.1. 匿名のアップロード

匿名ユーザーによるファイルのアップロードを許可する場合は、**/var/ftp/pub/** 内に書き込み専用のディレクトリーを作成することが推奨されます。これを行うには、以下のコマンドを **root** で実行します。

```
~]# mkdir /var/ftp/pub/upload
```

次に、パーミッションを変更して、匿名ユーザーがディレクトリーのコンテンツを閲覧できないようにします。

```
~]# chmod 730 /var/ftp/pub/upload
```

ディレクトリーのロング形式での一覧は以下のようになります。

```
~]# ls -ld /var/ftp/pub/upload
drwx-wx---. 2 root ftp 4096 Nov 14 22:57 /var/ftp/pub/upload
```

管理者が匿名ユーザーによるディレクトリー内での書き込みや読み取りを許可すると、そのサーバーが盗難ソフトウェアのレポジトリになってしまう場合があります。

vsftpd で、以下の行を **/etc/vsftpd/vsftpd.conf** ファイルに追加します。

```
anon_upload_enable=YES
```

4.3.9.3. ユーザーアカウント

FTP は安全でないネットワーク上で認証用のユーザー名とパスワードを暗号化せずに送信するので、ユーザーアカウントからサーバーへのシステムユーザーアクセスを拒否することはよい方法です。

vsftpd のすべてのユーザーアカウントを無効にするには、以下のディレクティブを **/etc/vsftpd/vsftpd.conf** に追加します。

```
local_enable=NO
```

4.3.9.3.1. ユーザーアカウントの制限

root ユーザーや **sudo** 権限を持つユーザーなど、特定のアカウントや特定のアカウントグループの FTP アクセスを無効にする最も簡単な方法は、「[Root アクセスの拒否](#)」にあるように PAM リストファイルを使用することです。**vsftpd** の PAM 設定ファイルは、**/etc/pam.d/vsftpd** です。

また、各サービス内で直接ユーザーアカウントを無効にすることもできます。

vsftpd で特定のユーザーアカウントを無効にするには、ユーザー名を **/etc/vsftpd/ftpusers** に追加します。

4.3.9.4. TCP Wrapper を使用してアクセスを制御する

FTP デモンへのアクセスを制御するには、「[TCP Wrapperおよび xinetd によるサービスのセキュア化](#)」にあるように TCP Wrapper を使用します。

4.3.10. Postfix のセキュア化

Postfix は メール転送エージェント (MTA) で、他の MTA や Email クライアント、配信エージェント間で電子メッセージを配信するために Simple Mail Transfer Protocol (SMTP) を使用します。多くの MTA には MTA 間のトラフィックを暗号化する機能がありますが、ほとんど使用されていないので、公開ネットワーク上での Email 送信はもともと安全でない通信方法とみなされています。Postfix は Red Hat Enterprise Linux 7 のデフォルトの MTA として Sendmail に代わるものです。

Postfix サーバーの実装を計画している場合は、以下の問題に対処することが推奨されます。

4.3.10.1. サービス拒否攻撃を制限する

Email の性質上、攻撃者が本気になるとサーバーに大量のメールを送信し、サービス拒否を発生させることが簡単にできます。`/etc/postfix/main.cf` ファイル内のディレクティブに制限を設定すると、このような攻撃の有効性が制限されます。既存のディレクティブの値を変更するか、以下の形式で希望する値の必要なディレクティブを追加することもできます。

```
<directive> = <value>
```

サービス拒否攻撃の制限に使用できるディレクティブを以下に示します。

- **smtpd_client_connection_rate_limit** — 一定の時間単位内 (下記を参照) にクライアントが当該サーバーに接続を試みることができる最大回数。デフォルト値は 0 で、この場合クライアントは Postfix が受付可能な回数内で時間単位当たり何回でも接続を試みることができます。デフォルトでは、信頼できるネットワーク内のクライアントは除外されます。
- **anvil_rate_time_unit** — この時間単位は割合制限の計算に使用されます。デフォルト値は 60 秒です。
- **smtpd_client_event_limit_exceptions** — 接続および割合制限のコマンドから除外されるクライアントです。デフォルトでは、信頼できるネットワーク内のクライアントは除外されます。
- **smtpd_client_message_rate_limit** — 時間単位内でクライアントが要求可能な最大メッセージ配信数です (Postfix が実際にこの数のメッセージを受け付けるかどうかは別問題です)。
- **default_process_limit** — あるサービスを提供する Postfix の子プロセスの最大デフォルト数です。この制限は、**master.cf** ファイル内の特定サービスによって無効にされる場合があります。デフォルト値は 100 です。
- **queue_minfree** — メール受信に必要なキューファイルシステム内での空き領域の最低バイト数です。これは現在、Postfix SMTP サーバーがメールを受信するかどうかを判断するために使用しています。デフォルトでは、Postfix SMTP サーバーは、空き領域が **message_size_limit** の 1.5 倍未満であれば **MAIL FROM** コマンドを拒否します。空き領域の最低限度をより大きくするように指定するには、**queue_minfree** の値が少なくとも **message_size_limit** の 1.5 倍になるように指定します。デフォルトの **queue_minfree** 値は 0 です。
- **header_size_limit** — メッセージヘッダー保存に使用するメモリの最大バイト数です。ヘッダーサイズがこの制限を超える場合は、超過分が廃棄されます。デフォルト値は 102400 です。
- **message_size_limit** — メッセージの最大バイト数で、これにはエンベロープ情報も含まれます。デフォルト値は 10240000 です。

4.3.10.2. NFS と Postfix

メールスプールディレクトリー `/var/spool/postfix/` を NFS 共有ボリュームに配置しないでください。NFSv2 と NFSv3 ではユーザー ID とグループ ID の制御を維持しないので、2 人以上のユーザーが同一 UID を持つ可能性があり、それぞれがお互いのメールを受信、閲覧してしまう可能性があります。



注記

Kerberos を使用する NFSv4 ではこういうことはありません。これは **SECRPC_GSS** カーネルモジュールが UID ベースの認証を使用しないためです。ただし、それでも NFS 共有ボリュームにメールスプールディレクトリーを **置かない** 方がよいと考えられます。

4.3.10.3. メール専用ユーザー

ローカルユーザーによる Postfix サーバーの悪用を避けるには、メールユーザーが Email プログラムを使用して Postfix サーバーにアクセスするだけにするのが最善の方法です。メールサーバー上のシェアルアカウントを許可せず、**/etc/passwd** ファイル内のすべてのユーザーシェルを **/sbin/nologin** に設定します (root ユーザーは例外とする場合もある)。

4.3.10.4. Postfix ネットワークリスニングの無効化

デフォルトでは、Postfix はローカルのループバックアドレスのみをリッスンするように設定されています。これは、**/etc/postfix/main.cf** ファイルを表示すると確認できます。

/etc/postfix/main.cf ファイルを閲覧して、以下の **inet_interfaces** 行のみが表示されることを確認してください。

```
inet_interfaces = localhost
```

これにより、Postfix は ネットワークからではなく、(cron ジョブレポートなどの) ローカルシステムからのメールメッセージのみを受信することが確認できます。これがデフォルト設定で、Postfix を ネットワーク攻撃から守ります。

ローカルホストの制限を取り除き、Postfix がすべてのインターフェースをリッスンできるようにするには、**inet_interfaces = all** と設定します。

4.3.10.5. Postfix が SASL を使用する設定

Postfix の Red Hat Enterprise Linux 7 バージョンは、**SMTP 認証** (または **SMTP AUTH**) に **Dovecot** もしくは **Cyrus SASL** 実装を使用できます。SMTP 認証は、**Simple Mail Transfer Protocol** の拡張機能です。これを有効にすると、SMTP クライアントはサーバーとクライアントとの両方でサポートされ、受け入れられている認証方法を使って SMTP サーバーを認証しなくてはなりません。本セクションでは、**Dovecot SASL** 実装を利用するように **Postfix** を設定する方法を説明します。

Dovecot POP/IMAP サーバーをインストールして、**Dovecot SASL** 実装を使用するシステム上で利用可能とするには、**root** で以下のコマンドを実行します。

```
~]# yum install dovecot
```

Postfix SMTP サーバーは、UNIX ドメインソケットか TCP ソケットのいずれかを使って **Dovecot SASL** 実装と通信します。後者の方法は、**Postfix** と **Dovecot** アプリケーションが別個のマシンで実行中の場合にのみ、必要となります。UNIX ドメインソケットの方がよりすぐれたプライバシーを提供するので、本ガイドではこちらを推奨しています。

Postfix が **Dovecot SASL** 実装を使用するように指示するには、両方のアプリケーションで多くの設定変更が必要になります。以下の手順にしたがってください。

Dovecot のセットアップ

1. **Dovecot** のメインの設定ファイルである **/etc/dovecot/conf.d/10-master.conf** に以下の行を含めます (デフォルトの設定ファイルには、ほとんどの関連セクションがすでに記載されており、これらの行はコメント解除するだけです)。

```
service auth {
    unix_listener /var/spool/postfix/private/auth {
        mode = 0660
        user = postfix
        group = postfix
    }
}
```

上記の例では、**Postfix** と **Dovecot** の通信に UNIX ドメインソケットを使用することを仮定しています。また、**Postfix SMTP** サーバーの設定はデフォルトとしています。この設定では、メールキューが **/var/spool/postfix/** ディレクトリに配置され、アプリケーションは **postfix** のユーザーおよびグループで実行されることになります。この方法では、読み取りおよび書き込み許可は、**postfix** ユーザーおよびグループに限定されます。

別の方法では、以下の設定を使うと **Dovecot** は TCP 経由で **Postfix** 認証リクエストをリスンするようになります。

```
service auth {
    inet_listener {
        port = 12345
    }
}
```

上記の例では、**12345** を、実際に使用するポート番号に置き換えます。

2. **/etc/dovecot/conf.d/10-auth.conf** 設定ファイルを編集して、**Dovecot** が **Postfix SMTP** サーバーに **plain** および **login** の認証メカニズムを提供するようにします。

```
auth_mechanisms = plain login
```

Postfix のセットアップ

Postfix の場合、メインの設定ファイルである **/etc/postfix/main.cf** を修正するだけです。以下の設定ディレクティブを追加もしくは編集します。

1. **Postfix SMTP** サーバーの SMTP 認証を有効にします。

```
smtpd_sasl_auth_enable = yes
```

2. **Postfix** が SMTP 認証に **Dovecot SASL** 実装を使用するように指示します。

```
smtpd_sasl_type = dovecot
```

3. **Postfix** キューディレクトリーの認証相対パスを提供します (相対パスを使用することで、**Postfix** サーバーが **chroot** で稼働しているかどうかにかかわらず、この設定が確実に機能するようになります)。

```
smtpd_sasl_path = private/auth
```

この手順では、**Postfix** と **Dovecot** 間の通信に UNIX ドメインソケットを使用することを前提します。通信に TCP ソケットを使用する場合で、**Postfix** が別のマシンにある

Dovecot を探すように設定するには、以下のような設定値を使用します。

```
smtpd_sasl_path = inet:127.0.0.1:12345
```

上記の例では、**127.0.0.1** を **Dovecot** マシンの **IP** アドレスに、**12345** を **Dovecot** の **/etc/dovecot/conf.d/10-master.conf** 設定ファイルで指定されているポート番号に置き換えます。

4. **Postfix SMTP** サーバーがクライアントに対して利用可能とする **SASL** メカニズムを指定します。暗号化セッションと非暗号化セッションでは、異なるメカニズムを設定できることに注意してください。

```
smtpd_sasl_security_options = noanonymous, noplaintext
smtpd_sasl_tls_security_options = noanonymous
```

上記の例では、非暗号化セッション中は匿名認証が許可されず、また暗号化されていないユーザー名やパスワードを送信するメカニズムも許可されません。(TLS を使った) 暗号化セッションでは、匿名でない認証メカニズムのみが許可されます。

許可される **SASL** メカニズムを制限する対応ポリシー全一覧は、http://www.postfix.org/SASL_README.html#smtpd_sasl_security_options を参照してください。

その他のリソース

以下のオンラインのリソースでは、**SASL** による **Postfix SMTP** 認証の設定に便利な追加情報が提供されています。

- <http://wiki2.dovecot.org/HowTo/PostfixAndDovecotSASL> — SMTP 認証に **Dovecot SASL** 実装を使用するための **Postfix** の設定方法が説明されています。
- http://www.postfix.org/SASL_README.html#server_sasl — SMTP 認証に **Dovecot** または **Cyrus SASL** 実装を使用するための **Postfix** の設定方法が説明されています。

4.3.11. SSH のセキュア化

Secure Shell (SSH) は、セキュアなチャネル上で別のシステムと通信するために使用される強力なネットワークプロトコルです。**SSH** 上の送信は暗号化され、傍受から保護されます。暗号化ログインを使用して、従来のユーザー名とパスワードよりも優れた認証方法を提供することもできます。**SSH** プロトコルと Red Hat Enterprise Linux 7 における **SSH** サービスの使用方法についての一般的な情報は、Red Hat Enterprise Linux 7 システム管理者のガイドの [OpenSSH](#) の章を参照してください。



重要

本セクションでは、**SSH** 設定のセキュア化における最も一般的な方法にフォーカスしてきました。ただし、ここで提案された方法が網羅的または決定的であるというわけではありません。**sshd** デーモンの動作修正に利用可能な設定ディレクティブすべてについての説明は **sshd_config(5)** の man ページを、基本的な **SSH** の概念については **ssh(1)** の man ページを参照してください。

4.3.11.1. 暗号化ログイン

SSH は、コンピューターにログインするための暗号化鍵の使用をサポートしています。これはパスワードのみの使用よりもはるかに安全です。この方法を他の認証方法と組み合わせると、マルチファクター認証 (multifactor authentication) と見なすことができます。マルチ認証方法についての詳細は、「[複数の認証方法](#)」を参照してください。

認証における暗号鍵の使用を有効にするには、`/etc/ssh/sshd_config` ファイルの **PubkeyAuthentication** 設定ディレクティブを **yes** に設定する必要があります。これがデフォルト設定であることに注意してください。 **PasswordAuthentication** ディレクティブを **no** に設定すると、ログインでのパスワード使用が無効になります。

SSH 鍵は **ssh-keygen** コマンドを使って生成できます。引数なしでこれを実行すると、2048 ビットの RSA 鍵のセットが作成されます。デフォルトでは、この鍵は `~/.ssh` ディレクトリーに保存されます。 **-b** スイッチを使うと、鍵のビット強度を修正することができます。通常は、2048 ビット鍵で十分な強度が提供されます。鍵のペアの生成に関する詳細情報は、Red Hat Enterprise Linux 7 システム管理者のガイドの [OpenSSH の設定](#) の章を参照してください。

`~/.ssh` ディレクトリーに、2 つの鍵が見つかるはずですが、**ssh-keygen** コマンドの実行時にデフォルトを受け入れると、生成されるファイル名は **id_rsa** と **id_rsa.pub** になり、それぞれに秘密鍵と公開鍵が含まれます。秘密鍵はファイルの所有者のみが読み取り可能として、公開されないように常に保護してください。ただし公開鍵は、ログインするシステムに送信する必要があります。 **ssh-copy-id** コマンドを使用すると、サーバーにこの鍵を移動することができます。

```
~]$ ssh-copy-id -i [user@]server
```

このコマンドを使用すると、`server` 上の `~/.ssh/authorized_keys` ファイルに公開鍵が自動的に追加されます。このファイルは、ユーザーがサーバーへのログインを試みる際に **sshd** デーモンによってチェックされます。

パスワードやその他の認証メカニズムと同様に、SSH 鍵は定期的に変更する必要があります。その際、**authorized_keys** ファイルから使用されていない鍵を必ず削除してください。

4.3.11.2. 複数の認証方法

マルチファクター認証とも呼ばれる複数の認証方式を使用すると、権限のないアクセスに対する保護のレベルが高まります。このため、システムのセキュリティを強化する際には、マルチファクター認証を検討してください。マルチファクター認証を使用するシステムにログインしようとする、ユーザーは指定されたすべての認証方式で成功しないと、アクセスが認められません。

使用する認証方式を指定するには、`/etc/ssh/sshd_config` ファイル内の **AuthenticationMethods** 設定ディレクティブを使用します。このディレクティブでは、必要となる認証方式の複数のリストを定義することに注意してください。複数を定義する場合は、各方式を少なくとも 1 つのリストで定義する必要があります。各リストは空白で区切ります。リスト内の各認証方式の名前はコンマ区切りとします。例を示します。

```
AuthenticationMethods publickey,gssapi-with-mic publickey,keyboard-interactive
```

上記の **AuthenticationMethods** ディレクティブを使って設定された **sshd** デーモンは、ユーザーが **publickey** 認証の後に **gssapi-with-mic** または **keyboard-interactive** 認証でログインを完了した場合にのみ、アクセスを許可します。必要とされる認証方式はそれぞれ、`/etc/ssh/sshd_config` ファイル内の対応する設定ディレクティブ (**PubkeyAuthentication** など) で明示的に有効となっている必要があることに注意してください。利用可能な認証方式の全般的なリストは、**ssh(1)** man ページの『AUTHENTICATION』セクションを参照してください。

4.3.11.3. SSH の他のセキュア化

プロトコルのバージョン

Red Hat Enterprise Linux 7 で提供される SSH プロトコルは SSH-1 と SSH-2 の両バージョンをサポートしますが、可能な場合は常に後者のみを使用してください。SSH-2 バージョンには、旧式の

SSH-1 の多くの改善点が含まれており、高度な設定オプションの多くは SSH-2 でのみ使用可能となっています。

SSH プロトコルによる認証と対象となる通信の保護を最大限まで活用するために、ユーザーは SSH-2 を使用することが推奨されます。**sshd** デーモンがサポートするプロトコルのバージョンは、**/etc/ssh/sshd_config** ファイル内で **Protocol** 設定ディレクティブを使って指定できます。デフォルト設定は **2** になります。

鍵のタイプ

ssh-keygen コマンドはデフォルトで SSH-2RSA 鍵のペアを生成しますが、**-t** オプションを使うと、DSA または ECDSA 鍵を生成するように指示することもできます。ECDSA (Elliptic Curve Digital Signature Algorithm) は、同じ対称鍵の長さでより優れたパフォーマンスを提供します。また、より短い鍵も生成します。

デフォルト以外のポート

デフォルトでは、**sshd** デーモンは TCP ポート **22** をリッスンします。ポートを変更すると、自動ネットワークスキャンに基づく攻撃に対するシステムの露出が減り、セキュリティが強化されます。ポートは、**/etc/ssh/sshd_config** 設定ファイルの **Port** ディレクティブで指定できます。デフォルト以外のポート使用を可能にするには、デフォルトの SELinux ポリシーの変更も必要になることに注意してください。これは、**root** で以下のコマンドを実行して **ssh_port_t** SELinux タイプを修正することで可能になります。

```
~]# semanage -a -t ssh_port_t -p tcp port_number
```

上記のコマンドでは、*port_number* を **Port** ディレクティブで指定する新たなポート番号に置き換えます。

Root 以外のログイン

特定のユースケースで **root** ユーザーとしてのログインが必要ない場合は、**/etc/ssh/sshd_config** ファイルで **PermitRootLogin** 設定ディレクティブを **no** に設定することを検討してください。**root** ユーザーとしてのログインの可能性をなくすことで、どのユーザーが通常のユーザーとしてログインした後に **root** 権限を獲得し、権限のあるコマンドを実行したかを管理者が監査できるようになります。

4.3.12. PostgreSQL のセキュリティ確保

PostgreSQL はオブジェクト関係データベース管理システム (DBMS) です。Red Hat Enterprise Linux 7 では **postgresql-server** パッケージは **PostgreSQL** を提供します。インストールされていない場合は、以下のコマンドを **root** ユーザーとして実行してインストールしてください。

```
~]# yum install postgresql-server
```

PostgreSQL の使用を開始する前に、ディスクにあるデータベースのストレージ領域を初期化する必要があります。これは、データベースのクラスターと呼ばれます。データベースのクラスターを初期化するには、**PostgreSQL** と合わせてインストールされる **initdb** のコマンドを使用します。データベースクラスターの希望のファイルシステムの場合は、**-D** オプションで指定することができます。以下に例を示します。

```
~]$ initdb -D /home/postgresql/db1
```

initdb コマンドでは、ディレクトリが存在しない場合にはそのディレクトリの作成が試行されます。今回の例では、**/home/postgresql/db1** という名前を使用します。**/home/postgresql/db1** ディレクトリには、データベースに保存するデータすべてと、クライアント認証の設定ファイルが含まれます。

```
~]$ cat pg_hba.conf
```

```
# PostgreSQL Client Authentication Configuration File
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access. Records take one of these forms:
#
# local      DATABASE  USER  METHOD  [OPTIONS]
# host       DATABASE  USER  ADDRESS METHOD  [OPTIONS]
# hostssl    DATABASE  USER  ADDRESS METHOD  [OPTIONS]
# hostnossl  DATABASE  USER  ADDRESS METHOD  [OPTIONS]
```

pg_hba.conf ファイルに以下の行を追加すると、認証済みのローカルユーザーが自分のユーザー名を使用してどのデータベースにもアクセスできるようになります。

```
local    all                                all                                trust
```

データベースユーザーを作成して、ローカルユーザーは作成しない階層式のアプリケーションを使用する場合には問題となる可能性があります。システム上のユーザー名をすべて明示的に制御しない場合には、**pg_hba.conf** ファイルからこの行を削除してください。

4.3.13. Docker のセキュリティ確保

Docker は Linux のコンテナ内のアプリケーションのデプロイメントを自動化して、ランタイムの依存関係とアプリケーションをコンテナにパッケージ化する機能を提供するオープンソースのプロジェクトです。**Docker** のワークフローのセキュリティを強化する方法は、『[Red Hat Enterprise Linux Atomic Host 7 Container Security Guide](#)』を参照してください。

4.4. ネットワークアクセスのセキュア化

4.4.1. TCP Wrapper および xinetd によるサービスのセキュア化

TCP Wrappers are capable of much more than denying access to services. This section illustrates how they can be used to send connection banners, warn of attacks from particular hosts, and enhance logging functionality. See the `hosts_options(5)` man page for information about the TCP Wrapper functionality and control language. See the `xinetd.conf(5)` man page for the available flags, which act as options you can apply to a service.

4.4.1.1. TCP Wrapper と接続バナー

システム管理者が警戒していることを潜在的な攻撃者に知らせるには、ユーザーがサービスに接続する際に適切なバナーを表示させるのがよい方法です。また、ユーザーに対してシステムのどの情報を表示させるかを制御することもできます。サービスに TCP Wrapper のバナーを実装するには、**banner** オプションを使用します。

以下の例では、**vsftpd** にバナーを導入します。最初にバナーファイルを作成します。これはシステム上のどこでも構いませんが、デーモンと同じ名前にする必要があります。例えば、ファイル名 `/etc/banners/vsftpd` には以下の行が含まれます。

```
220-Hello, %c
220-All activity on ftp.example.com is logged.
220-Inappropriate use will result in your access privileges being removed.
```

%c トークンは、ユーザー名およびホスト名、またはユーザー名および IP アドレスなどの幅広いクライアント情報を提供して接続に対して威嚇します。

このバナーを受信接続に表示させるには、以下の行を **/etc/hosts.allow** ファイルに追加します。

```
vsftpd : ALL : banners /etc/banners/
```

4.4.1.2. TCP Wrapper と攻撃警告

特定のホストまたはネットワークによるサーバーへの攻撃が検出された場合、TCP Wrapper は **spawn** ディレクティブを使ってそのホストまたはネットワークからのその後の攻撃について管理者に警告することができます。

以下の例では、206.182.68.0/24 ネットワークからのクラッカーがサーバーに攻撃を仕掛けようとしていることが検出されたとします。**/etc/hosts.deny** ファイルに以下の行を挿入すると、そのネットワークからの接続の試みが拒否され、特別ファイルにログ記録されます。

```
ALL : 206.182.68.0 : spawn /bin/echo `date` %c %d >>  
/var/log/intruder_alert
```

%d トークンは、攻撃者がアクセスを試みるサービス名を提供します。

接続とログインを許可するには、**spawn** ディレクティブを **/etc/hosts.allow** ファイル内に置きます。



注記

spawn ディレクティブはどんなシェルコマンドも実行するので、特定のクライアントがサーバーに接続しようとする際に、管理者に通知したり一連のコマンドを実行したりする特別スクリプトを作成するとよいでしょう。

4.4.1.3. TCP Wrapper とロギングの強化

特定の種類の接続が他のものよりも懸念される場合は、**severity** オプションを使うと該当サービスに対するログレベルを高めることができます。

以下の例では、FTP サーバーのポート 23 (Telnet ポート) への接続はクラッカーによるものとし、これを示すために、デフォルトのフラグである **info** の代わりに **emerg** フラグをログファイルに置いて接続を拒否します。

これを実行するには、以下の行を **/etc/hosts.deny** に挿入します。

```
in.telnetd : ALL : severity emerg
```

これはデフォルトの **authpriv** ロギング機能を使用しますが、優先順位をデフォルト値の **info** から **emerg** に引き上げ、ログメッセージを直接コンソールに投稿します。

4.4.2. リッスンしているポートの確認

ポートを必要以上に開くとシステムの攻撃対象領域を増やすことになるので、避けるべきです。システムがサービスを開始した後で、予期せず開放されたポートがリッスン状態になっている場合は、侵入の形跡である可能性があり、調査が必要になります。

root で以下のコマンドを実行して、ネットワークからの接続をリッスンしているポートを調べます。

```
~]# netstat -pan -A inet,inet6 | grep -v ESTABLISHED
```



```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
1/systemd
tcp        0      0 192.168.124.1:53        0.0.0.0:*               LISTEN
1975/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
1362/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
1355/cupsd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
1802/master
tcp6       0      0 :::1:111                :::*                    LISTEN
1/systemd
tcp6       0      0 :::22                   :::*                    LISTEN
1362/sshd
tcp6       0      0 :::1:631                 :::*                    LISTEN
1355/cupsd
tcp6       0      0 :::1:25                  :::*                    LISTEN
1802/master
raw6       0      0 :::58                    :::*                    7
791/NetworkManager
```

netstat コマンドの **-l** オプションを使用して、リスニングするサーバーソケットのみを表示できます。

```
~]# netstat -tlnw
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 192.168.124.1:53        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp6       0      0 :::1:111                :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::1:631                 :::*                    LISTEN
tcp6       0      0 :::1:25                  :::*                    LISTEN
raw6       0      0 :::58                    :::*                    7
```

本書執筆時は、**-l** オプションを使用しても SCTP サーバーが一覧表示されないことに注意してください。

リッスン状態の開放ポートを表示するには、**ss** ユーティリティーを使用することもできますが、本書執筆時はこのオプションを使用しても SCTP サーバーは表示されません。

```
~]# ss -tlw
Netid State      Recv-Q Send-Q Local Address:Port      Peer
Address:Port
udp     UNCONN    0      0          :::ipv6-icmp           :::*
tcp     LISTEN    0      128          *:sunrpc                *:~
tcp     LISTEN    0      5          192.168.124.1:domain    *:~
tcp     LISTEN    0      128          *:ssh                   *:~
tcp     LISTEN    0      128          127.0.0.1:ipp           *:~
tcp     LISTEN    0      100         127.0.0.1:smtp          *:~
```

```

tcp    LISTEN    0      128             :::1:sunrpc      :::*
tcp    LISTEN    0      128             :::ssh           :::*
tcp    LISTEN    0      128             :::1:ipp         :::*
tcp    LISTEN    0      100            :::1:smtp        :::*

```

システムで必要なサービスをコマンドの出力で確認して、特に必要でないものや権限が与えられていないものをオフにしてから、繰り返し確認します。次に、ネットワーク経由で最初のシステムに接続している別のシステムから **nmap** を使用して、外部チェックを行います。これは、**firewalld** のルールの確認に使用できます。

以下は、ネットワークからの TCP 接続をリッスンしているポートを判断するために、別のシステムのコンソールから発行するコマンドの例です。

```
~]# nmap -sT -O 192.168.122.1
```

See the `nmap(1)` and `services(5)` manual pages for more information.

4.4.3. ソースルーティングの無効化

ソースルーティングはインターネットプロトコルメカニズムで、IP パケットがアドレス一覧の情報を持ち運べるようにします。このアドレスは、パケットが通過する必要があるパスをルーターに知らせるものです。ルートを移動する際にホップを記録するオプションもあります。「ルート記録」と呼ばれるホップの記録は、宛先にソースまでの帰りのパスを提供します。これによりソースは（つまり送信ホスト）、すべてもしくは一部のルーターのルーティングテーブルを無視して、ルートを厳密もしくは緩やかに特定することができます。これによりユーザーは、不正目的でネットワークトラフィックをリダイレクトすることが可能になります。このため、ソーススペースのルーティングは無効にする必要があります。

accept_source_route オプションを使用すると、ネットワークインターフェースが厳密なソースルート (SSR) もしくは緩やかなソースルーティング (LSR) オプションセットのあるパケットを受け付けるようになります。ソースルーティングパケットの受け付けは `sysctl` 設定で制御します。以下のコマンドを `root` で実行し、SSR または LSR オプションセットのあるパケットを遮断します。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.accept_source_route=0
```

パケット転送の無効化は、可能な場合は上記のコマンドと合わせて行うべきです（転送の無効化は仮想化に影響する場合があります）。以下のコマンドを `root` で発行します。

以下のコマンドはすべてのインターフェースで IPv4 および IPv6 パケットの転送を無効にします。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.forwarding=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.forwarding=0
```

以下のコマンドはすべてのインターフェースですべてのマルチキャストパケットの転送を無効にします。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.mc_forwarding=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.mc_forwarding=0
```

ICMP リダイレクトの受信が正当に使用される場合はほとんどありません。ICMP リダイレクトパケットは特に必要でなければ、その受信と送信を無効にしてください。

以下のコマンドはすべてのインターフェースですべての ICMP リダイレクトパケットの受信を無効にします。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.accept_redirects=0
```

```
~]# /sbin/sysctl -w net.ipv6.conf.all.accept_redirects=0
```

以下のコマンドはすべてのインターフェースでセキュアな ICMP リダイレクトパケットの受信を無効にします。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.secure_redirects=0
```

以下のコマンドはすべてのインターフェースですべての IPv4 ICMP リダイレクトパケットの受信を無効にします。

```
~]# /sbin/sysctl -w net.ipv4.conf.all.send_redirects=0
```

IPv4 リダイレクトパケットの送信を無効にできるのはディレクティブのみです。IPv4 と IPv6 の差異を生み出している「IPv6 Node Requirements」の説明については、[RFC4294](#) を参照してください。

設定を永続的なものにするには、その設定を **/etc/sysctl.conf** に追加する必要があります。

詳細は `sysctl` の man ページ **sysctl(8)** を参照してください。ソーススペースのルーティングおよびそのバリエーションに関連するインターネットオプションの説明については、[RFC791](#) を参照してください。



警告

イーサネットネットワークは、ARP や MAC アドレススプーフィング、権限のない DHCP サーバー、IPv6 ルーターまたは近隣アドバタイズメントといったトラフィックをリダイレクトする新たな方法を提供します。さらに、ユニキャストトラフィックはブロードキャストの場合もたまにあり、情報の漏洩を引き起こします。これらの脆弱性は、ネットワークオペレーター導入する特定の対策によってのみ、対処可能になります。ホストベースの対応策の有効性は、完全なものではありません。

4.4.4. 逆方向パス転送

逆方向パス転送は、あるインターフェースから着信したパケットが異なるインターフェース経由で去ってしまうことを防ぐために使用されます。送信ルートと着信ルートが異なる場合は、**非対称ルーティング**と呼ばれる場合もあります。ルーターがパケットをこの方法でルート設定することはよくありますが、ほとんどのホストはこのようなことをする必要はないはずです。例外として挙げられるのは、トラフィックをあるリンクで送信し、異なるサービスプロバイダーから別のリンクでトラフィックを受け取るアプリケーションです。例えば、xDSL との組み合わせで専用回線を使っている場合や、3G モデムを使ったサテライトリンクなどの場合です。このようなシナリオが該当する場合は、着信インターフェースで逆方向パス転送をオフにすることが必要になります。つまり、これが必要だと分かっている場合を除いて、有効にしておくのが最善の方法です。これは、ローカルサブネットからユーザーが IP アドレスをスプーフィングすることを防ぎ、DDoS 攻撃の機会を減らすためです。



注記

Red Hat Enterprise Linux 7 はデフォルトで 厳密な逆方向パス転送 (RPF: Reverse Path Forwarding) を使用します。これは、[RFC 3704, Ingress Filtering for Multihomed Networks](#) の厳密な逆方向パスに関する推奨事項に準拠します。



警告

転送が有効になっていれば、(**iptables** ルールなど) ソースアドレス確認に他の方法がある場合にのみ、逆方向パス転送を無効にしてください。

rp_filter

逆方向パス転送は **rp_filter** ディレクティブで有効にします。**sysctl** ユーティリティーを使用すると、稼働中のシステムに変更を加えることができます。永続的な変更は、**/etc/sysctl.conf** ファイルに行を追加して行えます。**rp_filter** オプションは、カーネルが 3 つのいずれかのモードから選択することを指示するために使用されます。

一時的なグローバル変更を行うには、**root** で以下のコマンドを入力します。

```
sysctl -w net.ipv4.conf.default.rp_filter=integer
sysctl -w net.ipv4.conf.all.rp_filter=integer
```

ここで、*integer* は、以下のいずれかになります。

- **0** — ソース確認なし
- **1** — RFC 3704 で定義された厳密なモード。
- **2** — RFC 3704 で定義された緩慢なモード。

この設定は、以下のように **net.ipv4.conf.interface.rp_filter** コマンドを使用してネットワークインターフェースごとに上書きできます。

```
sysctl -w net.ipv4.conf.interface.rp_filter=integer
```

これらの設定を再起動しても保持するには、**/etc/sysctl.conf** ファイルを変更します。たとえば、すべてのインターフェースのモードを変更するには、**root** ユーザーとして実行しているエディターで **/etc/sysctl.conf** ファイルを開き、以下のような行を追加します。

```
net.ipv4.conf.all.rp_filter=2
```

IPv6_rpfilter

IPv6 プロトコルの場合は、**firewalld** デーモンはデフォルトで逆方向パス転送が適用されます。この設定は、**/etc/firewalld/firewalld.conf** ファイルで確認できます。**IPv6_rpfilter** オプションを設定すると **firewalld** の動作を変更することができます。

逆方向パス転送のカスタム設定が必要な場合には、

```
ip6tables -t raw -I PREROUTING -m rpfilter --invert -j DROP
```

■

のように **ip6tables** コマンドを使用することで **firewalld** なし で実行できます。このルールは、すべてのトラフィックに適用されるように、特にステートフルの一致ルールの前で **raw/PREROUTING** チェーンの開始部分の近くに挿入する必要があります。**iptables** および **ip6tables** サービスに関する詳しい情報は「[iptables サービスの使用](#)」を参照してください。

4.4.4.1. その他のリソース

以下のリソースでは、逆方向パス転送について詳細な説明が提供されています。

- インストール済みドキュメンテーション

/usr/share/doc/kernel-doc-version/Documentation/networking/ip-sysctl.txt - このファイルには、**/proc/sys/net/ipv4/** ディレクトリーで使用可能なファイルおよびオプションの完全な一覧が含まれています。初めてカーネルドキュメンテーションにアクセスする場合は、**root** で以下のコマンドを入力します。

```
~]# yum install kernel-doc
```

- オンラインドキュメンテーション

Ingress Filtering for Multihomed Networks の説明については、[RFC 3704](#) を参照してください。

4.5. ファイアウォールの使用

動的ファイアウォールデーモンである **firewalld** は、動的に管理されるファイアウォールを提供し、信頼レベルをネットワークと関連する接続およびインターフェースに割り当てるネットワーク「ゾーン」をサポートします。これは、**IPv4** および **IPv6** のファイアウォール設定をサポートします。また、イーサネットブリッジをサポートし、実行時と永続的な設定オプションを別々にすることができます。ファイアウォールルールを直接追加するためのサービスやアプリケーション向けのインターフェースもあります。firewalld で完全な通信を行うには **D-Bus** を使用します。



注記

専門知識をさらに得るために、[Red Hat Server Hardening \(RH413\)](#) トレーニングコースが用意されています。

4.5.1. firewalld の概要

ファイアウォールデーモンは、デフォルトで **iptables**、**ip6tables**、**ebtables** のリストアコマンドを使用し、ルールセットを変更する全ファイアウォールアクションを高速化します。**firewalld.conf** ファイルで構成設定の **IndividualCalls** が **yes** に設定されている場合、またはルールがフォールバックの解決策としてリストアコマンドで適用できない場合には、通常のコマンドを使用します。通常のコマンドを使用すると、大幅に操作が遅くなります。

グラフィカルな **firewall-config** ツールを使用するには、**Super** キーを押してアクティビティーを開き、**firewall** と入力してから **Enter** を押します。**firewall-config** ツールが表示され、管理者パスワードの入力が求められます。

左側のサイドバーは、アクティブゾーンの **アクティブバインディング** が表示されます。これらは接続別にグループ化され、**NetworkManager**、**インターフェース** および **ソース** で処理されます。

firewall-config ツールには、**設定** のラベルが付いたドロップダウンメニューがあります。ここで、**実行時** と **永続** のモードが選べます。**永続** を選ぶと、左下に新たなアイコンの列が表示されます。サービスのパラメーターは **実行時** モードでは変更できないため、これらのアイコンは永続設定モードでのみ表示されます。この設定は、**アクティブバインディング** のサイドバーには影響ありません。

firewalld が提供するファイアウォールのサービスは、設定の変更はいつでも可能で即座に実行されるので、静的ではなく動的なものです。変更は保存したり適用したりする必要がありません。ファイアウォールはリロードの必要がないので、既存のネットワーク接続が意図せずに中断されることがありません。

コマンドラインクライアントである **firewall-cmd** が提供され、これを使うと **man firewall-cmd(1)** で説明されている永続および非永続の実行時の変更を行うことができます。永続的な変更は、**firewalld(1)** **man** ページの説明にしたがってください。**firewall-cmd** コマンドは、**root** ユーザーと管理者ユーザーといういわゆる **wheel** グループのメンバーのみが実行できることに注意してください。管理者ユーザーの場合は、**polkit** メカニズム経由でコマンドが承認されます。

コマンドラインクライアント **firewall-offline-cmd** は、**root** ユーザーのみが使用して永続的に環境を変更することができます。このクライアントは、**firewalld** と対話せずに、**firewalld** コアおよび I/O バックエンドの一部を使用して設定を変更します。**firewalld** がアクティブな状態でこのツールを使用することは推奨していません。使用は可能ですが、**firewall-offline-cmd** での変更は **firewalld** に即座に適用されません。変更は、**firewalld** がファイルシステムのファイルの変更を検出できてから永続的な環境に適用されます。たとえば、**firewall-offline-cmd** コマンドはファイアウォールの設定のインストール中に使用されます。また、ファイアウォール設定を変更するためにインストール後の設定で使用してから、新規インストールされたシステムを起動することもできます。

firewall-applet アプリケーションにより、使用中の接続を **NetworkManager** 設定タブにすぐに表示することもできます。**全般** タブを使用して割り当てたファイアウォールゾーンに変更を加えることができます。Red Hat Enterprise Linuxでは、このアプレットはデフォルトでインストールされていません。

firewalld の設定は、様々な XML ファイルで **/usr/lib/firewalld/** と **/etc/firewalld/** に保存されます。これらのファイルは修正、書き込み、バックアップ、他のインストールへのテンプレートとしての使用などができるので、柔軟性が高まります。**/usr/lib/firewalld/** の設定はデフォルトでフォールバックの設定ですが、**/etc/firewalld/** の設定はシステム固有の設定になっています。

アプリケーションはすべて、**D-Bus** インターフェースを使って **firewalld** と通信できます。

4.5.1.1. firewalld と system-config-firewall および iptables との比較

firewalld と **iptables** (and **ip6tables**) の本質的な違いは、以下の通りです。

- **iptables service** は設定を **/etc/sysconfig/iptables** および **/etc/sysconfig/ip6tables** に保存しますが、**firewalld** は様々な XML ファイルで **/usr/lib/firewalld/** および **/etc/firewalld/** に保存します。Red Hat Enterprise Linux では **firewalld** がデフォルトでインストールされるので、**/etc/sysconfig/iptables** ファイルがないことに注意してください。
- **iptables service** では変更のひとつひとつで古いルールがフラッシュされ新しいルールが **/etc/sysconfig/iptables** から読み込まれますが、**firewalld** ではそのようなすべてのルールの再生はなく、差異のみが適用されます。このため、**firewalld** では既存の接続が中断されることなく実行時に設定変更ができます。

これら両方が **iptables tool** を使用してカーネルパケットフィルターと通信します。



注記

firewalld は、`/etc/sysconfig/ip*tables` ファイルからファイアウォールの設定をインポートできません。**lokkit** または **system-config-firewall** の設定をインポートするには、**firewall-offline-cmd** と `/etc/sysconfig/system-config-firewall` ファイルを使用します。カスタムのルールファイルは、**firewalld** にインポートできません。インポートした設定は、デフォルトゾーンに適用されます。

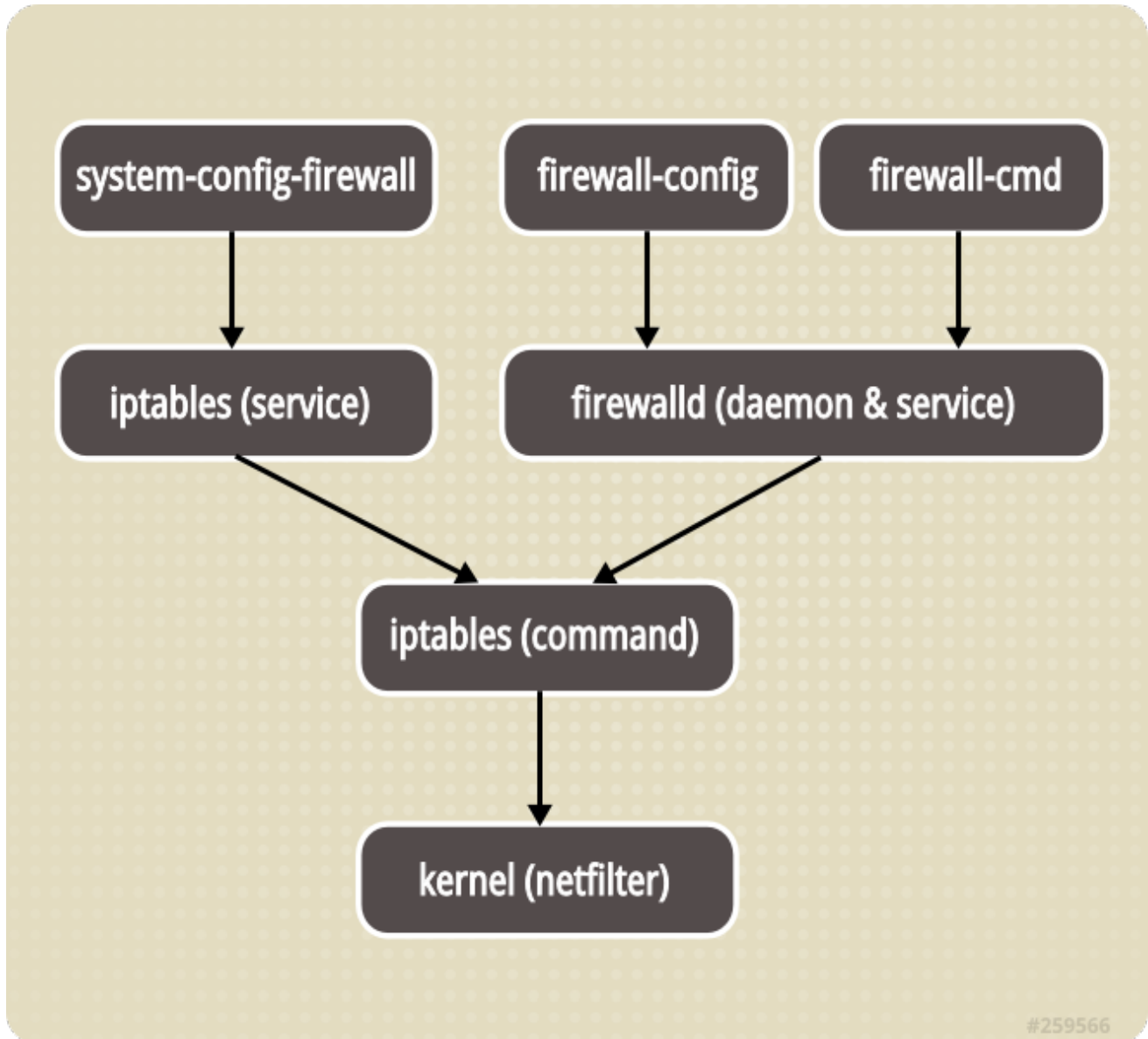


図4.1 ファイアウォールスタック

4.5.1.2. ネットワークゾーンについて

firewalld を使うと、ユーザーがネットワーク内のインターフェースとトラフィックに配置する信頼レベルに基づいて、ネットワークを異なるゾーンに分けることができます。**NetworkManager** が **firewalld** にインターフェースが所属するゾーンを通知します。インターフェースに割り当てられたゾーンは **NetworkManager** で、または関連する **NetworkManager** ウィンドウを開く **firewall-config** ツール経由で変更できます。**firewall-cmd** コマンドラインツールも使用できます。インターフェースが **NetworkManager** で制御されていて、ユーザーが **firewall-cmd**、**firewall-offline-cmd** または **firewall-config** を使用してインターフェースのゾーンを変更した場合には、この要求は **NetworkManager** に転送され、**firewalld** では処理されません。

/etc/firewalld/ におけるゾーン設定は、ネットワークインターフェースにすぐに適用可能な事前設定の範囲です。以下でこれらを簡単に説明します。

drop

着信ネットワークパケットはすべて遮断され、返信されません。送信ネットワーク接続のみが可能です。

block

IPv4 では `icmp-host-prohibited` メッセージで、**IPv6** では `icmp6-adm-prohibited` メッセージですべての着信ネットワーク接続が拒否されます。システム内で開始されたネットワーク接続のみが可能です。

public

公開エリア用です。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターを信頼しません。選択された着信接続のみが許可されます。

external

マスカレードを特別にルーター用に有効にした外部ネットワーク上での使用向けです。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターを信頼しません。選択された着信接続のみが許可されます。

dmz

公開アクセスが可能ではあるものの、内部ネットワークへのアクセスには制限がある非武装地帯にあるコンピューター用。選択された着信接続のみが許可されます。

work

作業エリア用です。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターをほぼ信頼します。選択された着信接続のみが許可されます。

home

ホームエリア用です。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターをほぼ信頼します。選択された着信接続のみが許可されます。

internal

内部ネットワーク用です。自分のコンピューターを保護するため、ネットワーク上の他のコンピューターをほぼ信頼します。選択された着信接続のみが許可されます。

trusted

すべてのネットワーク接続が許可されます。

これらのゾーンのいずれかをデフォルトに指定することができます。インターフェース接続が **NetworkManager** に追加されると、デフォルトのゾーンに割り当てられます。インストール時には、**firewalld** のデフォルトは **public** ゾーンに設定されます。

ネットワークゾーンの選択

ネットワークゾーンの名前は、すぐに分かりユーザーが妥当な決定をすばやく下せるように付けられています。ただし、ユーザーのニーズおよびリスク評価に合わせて、デフォルト設定の見直しを行ったり、不要なサービスを無効にしてください。

ゾーン名および設定は一例で、必要に応じて変更できます。組み込まれているゾーンは削除できませんが、**firewall-config** または **firewall-cmd** の永続設定にゾーンのデフォルト値を読みこませて、初期デフォルト値に設定を戻すことは可能です。

4.5.1.3. 事前定義のサービスについて

サービスは、ローカルポート、プロトコル、セキュアなポートと宛先のリストの場合や、サービスが有効であれば自動的に読み込まれるファイアウォールヘルパーのモジュールのリストの場合があります。事前定義サービスを使用すると、ユーザーによるサービスへのアクセスの有効、無効の切り替えが容易になります。ポートやポート範囲を開くのではなく、事前定義のサービスまたはカスタム定義のサービスを使用すると、管理が容易になる場合があります。サービスの設定オプションと一般的なファイル情報は、**firewalld.service(5)** man ページで説明されています。サービスは個別の XML 設定ファイルで指定されており、これらのファイル名の形式は、**service-name.xml** となります。**firewalld** では、プロトコル名よりもサービスまたはアプリケーション名が推奨されます。

グラフィカルな **firewall-config** ツールを使用してサービス一覧を表示するには、**Super** キーを押してアクティビティを開き、**firewall** と入力してから **Enter** を押します。**firewall-config** ツールが表示され、管理者パスワードの入力が求められます。サービスタブでサービスの一覧を確認できます。

システム上で利用可能な全サービスを一覧表示するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --get-services
```

サービスの設定を取得するには、以下のコマンドを使用します。

```
~]$ firewall-cmd --info-service=service-name
```

コマンドラインを使用して、デフォルトで事前定義されていて利用可能なサービスのみを表示するには、以下のコマンドを実行します。

```
~]$ ls /usr/lib/firewalld/services/
```



注記

/usr/lib/firewalld のファイルを表示するには、root 権限は必要ありません。カスタムのプライベートファイルを追加した後に、属性を適宜変更するようにしてください。

/usr/lib/firewalld/services/ にあるファイルは編集しないでください。**/etc/firewalld/services/** にあるファイルのみを編集してください。

システムもしくはユーザーが作成したサービスを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# ls /etc/firewalld/services/
```

サービスの追加もしくは削除は、グラフィカルな **firewall-config** ツール、**firewall-cmd**、**firewall-offline-cmd** を使用するか、**/etc/firewalld/services/** にある XML ファイルを編集することで実行できます。ユーザーがサービスを追加もしくは変更していない場合には、そのサービスに対応する XML ファイルは **/etc/firewalld/services/** にありません。サービスの追加または変更には、**/usr/lib/firewalld/services/** のファイルをテンプレートとして使用してください。

ターミナルで新規サービスを追加するには **firewall-cmd** を使用してください。**firewalld** がアクティブでない場合には **firewall-offline-cmd** を使用してください。新規サービスや空のサービスを追加するには以下のコマンドを実行します。

```
~]$ firewall-cmd --permanent --new-service=service-name
```

ローカルファイルを使用して新規サービスを追加するには以下のコマンドを使用します。

```
~]$ firewall-cmd --permanent --new-service-from-file=service-name.xml
```

--name=service-name オプションを指定して、サービス名を変更できます。

サービス設定が変更されたらすぐに、サービスの更新コピーが **/etc/firewalld/services/** に配置されます。

root として、以下のコマンドを実行しサービスを手動でコピーします。

```
~]# cp /usr/lib/firewalld/services/service-name.xml  
/etc/firewalld/services/service-name.xml
```

firewalld は、**/usr/lib/firewalld/services** からファイルを読み込みます。ファイルが **/etc/firewalld/services** に配置されて有効な場合には、**/usr/lib/firewalld/services** にある対応のファイルを上書きします。**/etc/firewalld/services** にある対応のファイルが削除されるか、サービスのデフォルトを読みこむように **firewalld** に要求が出されるとすぐに、**/usr/lib/firewalld/services** の上書きされたファイルが使用されます。これが該当するのは永続環境のみで、ランタイム環境でフォールバックさせるには、再読み込みが必要です。

4.5.1.4. ダイレクトインターフェースについて

firewalld には、ルールを直接 **iptables**、**ip6tables**、および **ebtables** に渡せるようにするダイレクトインターフェースが含まれます。これは主にアプリケーションによる使用が対象です。**iptables** に精通していないと意図せずにファイアウォール侵害を引き起こす可能性があるため、ダイレクトインターフェースを使用することは危険です。追跡のあるインターフェースパーツを使用している限り、**firewalld** をクエリーでき、このモードでアプリケーションが加えた変更を確認することができます。追跡のないパススルーモードは、**libvirt** および **docker** など、独自のルールセットに完全に対応するサービスのみを対象にしています。ダイレクトインターフェースを使用するには、**--direct** オプションを **firewall-cmd** コマンドに追加します。

直接インターフェースモードは、サービスまたはアプリケーションが実行時に特定のファイアウォールルールを追加するためのものです。**--permanent** オプションを追加して **firewall-cmd --permanent --direct** コマンドを使用するか、**/etc/firewalld/direct.xml** を修正することで、ルールを永続的なものにできます。ルールが永続的になっていないと、**D-Bus** を使用して **firewalld** から開始、再開、リロードのメッセージを受け取るたびにルールを毎回適用する必要があります。ダイレクトインターフェースでは、チェーン、ルール、追跡あり/なしのパススルールールを追加できます。また、ゾーン固有のチェーンにダイレクトルールを使用することもできます。

4.5.2. firewalld のインストール

Red Hat Enterprise Linux 7 では、**firewalld** はデフォルトでインストールされます。必要な場合は、確実にインストールするために **root** で以下のコマンドを入力します。

```
~]# yum install firewalld
```

グラフィカルユーザーインターフェース設定ツール **firewall-config** は一部のバージョンの Red Hat Enterprise Linux 7 でデフォルトでインストールされます。必要な場合は、**root** で以下のコマンドを入力して、確実に **firewall-config** がインストールされるようにします。

```
~]# yum install firewall-config
```

オプションの **firewalld** をインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install firewall-applet
```

次のログインでデスクトップのパネルに、アプレットは表示されます。アプレットの **GNOME** ルールによって、**GNOME** でのアプレットの使用には制約があります。

4.5.2.1. firewalld の停止

firewalld を停止するには、**root** で以下のコマンドを入力します。

```
~]# systemctl stop firewalld
```

firewalld がシステム起動時に自動的に起動しないようにするには、**root** で以下のコマンドを発行します。

```
~]# systemctl disable firewalld
```

firewalld D-Bus インターフェースにアクセスしても **firewalld** が起動されないようにする場合、他のサービスが **firewalld** を必要とする場合には、以下のコマンドを **root** として実行します。

```
~]# systemctl mask firewalld
```

4.5.2.2. firewalld の起動

firewalld を起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl unmask firewalld
~]# systemctl start firewalld
```

firewalld をシステム起動時に自動的に起動するには、**root** で以下のコマンドを入力します。

```
~]# systemctl enable firewalld
```

4.5.2.3. firewalld の稼働確認

firewalld が稼働しているかどうかを確認するには、以下のコマンドを実行します。

```
~]$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
  vendor preset: enabled)
   Active: active (running) since Tue 2016-10-11 09:15:58 CEST; 2 days ago
     Docs: man:firewalld(1)
  Main PID: 721 (firewalld)
    CGroup: /system.slice/firewalld.service
```

```
└─721 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
Oct 11 09:15:57 localhost.localdomain systemd[1]: Starting firewalld -
dynam...
Oct 11 09:15:58 localhost.localdomain systemd[1]: Started firewalld -
dynamic...
Hint: Some lines were ellipsized, use -l to show in full.
```

さらに、**firewall-cmd** がデーモンに接続可能かどうかを確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --state
running
```

4.5.3. firewalld の設定

firewalld デーモンにより実装されるファイアウォールサービスは、**firewall-config** のグラフィカルユーザーインターフェースツールまたは **firewall-cmd** および **firewall-offline-cmd** のコマンドラインインターフェースツールを使用するか、XML 設定ファイルを編集して設定できます。これらの方法について順番に説明します。

4.5.3.1. グラフィカルユーザーインターフェースを使った firewalld の設定

4.5.3.1.1. グラフィカルファイアウォール設定ツールの起動

グラフィカルの **firewall-config** ツールを起動するには、**Super** キーを押してアクティビティを開き、**firewall** と入力してから **Enter** を押します。**firewall-config** ツールが表示され、管理者パスワードの入力が求められます。

コマンドラインを使ってグラフィカルなファイアウォール設定ツールを起動するには、以下のコマンドを実行します。

```
~]$ firewall-config
```

ファイアウォールの設定 ウィンドウが開きます。このコマンドは一般ユーザーとしても実行できますが、管理者パスワードが求める場合があります。

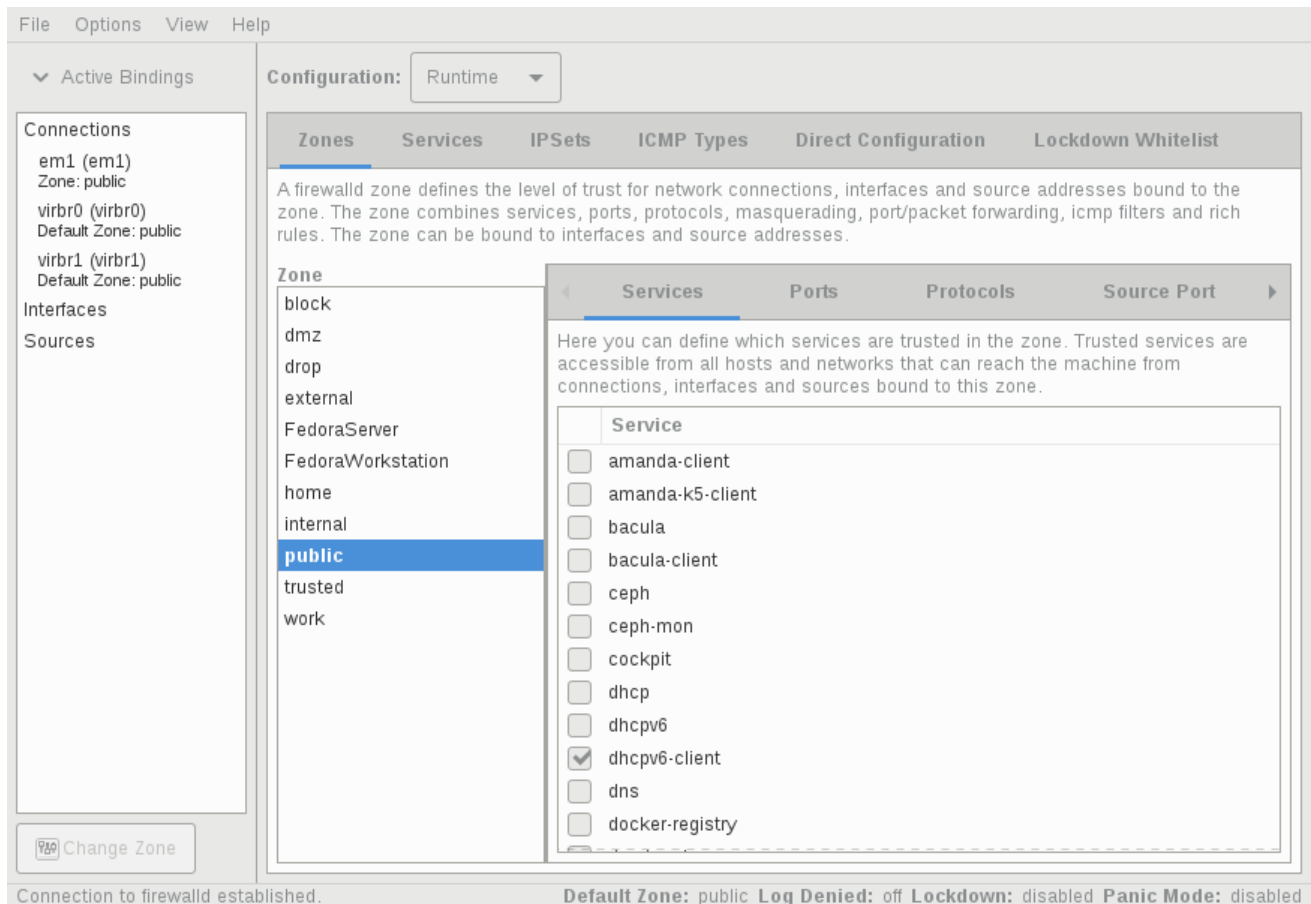


図4.2 ファイアウォール設定ツール

左下に表示されている「Connection to firewalld established」とのメッセージを探します。これにより、**firewall-config** ツールが **firewalld** に接続されていることが分かります。**ICMP タイプ**、**IPSets**、**ダイレクト設定**、**ロックダウンホワイトリスト** タブは、**表示** のドロップダウンメニューから選択した後にのみ、表示されることに注意してください。左側の **アクティブバインディング** のサイドバーはデフォルトで表示されます。

4.5.3.1.2. ファイアウォールの設定変更

現行のファイアウォール設定を直ちに変更するには、現在の表示が **実行時** に設定されていることを確認してください。次のシステム起動時またはファイアウォールの再読み込み時に設定が適用されるようにするには、ドロップダウンリストから **永続** を選択します。



注記

実行時 モードでファイアウォール設定を変更する際には、サービスに関連するチェックボックスにチェックを入れたり消したりすると、その選択が直ちに反映されます。他のユーザーが使用している可能性のあるシステムでこの作業を行う場合は、この点を考慮してください。

永続 モードでファイアウォール設定を変更すると、ファイアウォールを再読み込みした場合か、またはシステムの再起動時にのみ変更が反映されます。**オプション** メニューを選択して **Firewalld の再読み込み** を選択します。

ゾーンは左側のコラムで選択できます。ゾーンには有効になっているサービスがいくつかあり、すべてを閲覧するにはウィンドウのサイズを変更するか、スクロールする必要がある場合があります。サービスを選択および選択解除することで、設定をカスタマイズできます。

4.5.3.1.3. ゾーンへのインターフェースの追加

ゾーンに接続（接続に使用するインターフェース）を追加するには、**firewall-config** を起動します。左側のゾーン一覧のゾーンをクリックして、右側の **インターフェース** タブを選択します。**追加** ボタンをクリックするとインターフェース追加の新しいダイアログが表示されます。

インターフェースのゾーン設定を変更するには、**アクティブバインディング** のサイドバーのインターフェースで適切な接続をダブルクリックします。以下のダイアログのドロップダウンメニューから新しいファイアウォールゾーンを選択して、**OK** をクリックして確定します。

または、接続インターフェースのゾーンへの追加や再度割り当ては、**firewall-config** を開始して、メニューバーから **オプション** を選び、ドロップダウンメニューから **接続のゾーンの変更** を選択します。すると、**ネットワーク接続**、**インターフェース**、**ソース** のリストが表示されます。ここで、再度割り当てる接続を選択します。**接続のゾーンを選択する** のウィンドウが表示されます。ドロップダウンメニューから新規のファイアウォールゾーンを選択して **OK** をクリックします。

NetworkManager で処理される接続は、ゾーン変更の要求は **NetworkManager** に転送されます。ゾーンのインターフェース設定は、**firewalld** には保存されません。

接続、インターフェース、ソースのゾーンを変更する際に、**firewall-cmd** コマンドラインツールまたは **firewall-applet** アプレットも使用できます。

4.5.3.1.4. デフォルトゾーンの設定

新たなインターフェースに割り当てるデフォルトのゾーンを設定するには、**firewall-config** を開始して、メニューバーから **オプション** を選び、ドロップダウンメニューから **標準のゾーンの変更** を選択します。**標準ゾーン** のウィンドウが表示されるので、デフォルトのゾーンとして使用するゾーンをリストから選択して **OK** をクリックします。または、以下のコマンドを実行します。

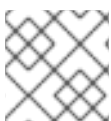
```
~]$ firewall-cmd --set-default-zone=zone-name
```

4.5.3.1.5. サービスの設定

事前定義またはカスタマイズサービスを有効、無効にするには、**firewall-config** ツールを起動して、設定するサービスのネットワークゾーンを選択します。**サービス** タブを選択し、信頼するサービスの各タイプのチェックボックスを選択します。サービスをブロックするには、チェックボックスを外します。

サービスを編集するには、**firewall-config** ツールを起動してから **設定** ラベルのあるドロップダウン選択メニューで **永続** モードを選びます。**サービス** ウィンドウの下部に新たなアイコンとメニューボタンが表示されます。設定するサービスを選びます。

ポート、**プロトコル**、**ソースポート** のタブでは、選択されたサービスのポート、プロトコルおよびソースポートの追加、変更、削除ができます。モジュールタブでは、**Netfilter** ヘルパーモジュールの設定を行います。**送信先** タブでは、特定の送信先アドレスとインターネットプロトコル **IPv4** または **IPv6** へのトラフィックが制限できます。



注記

実行時 モードのサービス設定は変更できません。

4.5.3.1.6. ファイアウォールのポート開放

特定のポートへのトラフィックがファイアウォールを通過できるようにするには、**firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。**ポート** タブを選んで**追加** ボタンをクリックします。**ポートとプロトコル** ウィンドウが開きます。

許可するポート番号またはポートの範囲を入力します。ドロップダウンリストから **tcp** または **udp** を選択します。

4.5.3.1.7. ファイアウォールのプロトコル開放

特定のプロトコルへのトラフィックがファイアウォールを通過できるようにするには、**firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。**プロトコル** タブを選んで**追加** ボタンをクリックします。**プロトコル** ウィンドウが開きます。

ドロップダウンリストからプロトコルを選択するか、**他のプロトコル** のチェックボックスを選択して、フィールドにプロトコルを入力します。

4.5.3.1.8. ファイアウォールのソースポート開放

特定のポートへのトラフィックがファイアウォールを通過できるようにするには、**firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。**ソースポート** タブを選んで**追加** ボタンをクリックします。**ソースポート** ウィンドウが開きます。

許可するポート番号またはポートの範囲を入力します。ドロップダウンリストから **tcp** または **udp** を選択します。

4.5.3.1.9. IPv4 アドレスのマスカレーディングの有効化

IPv4 アドレスを単一の外部アドレスに変換するには、**firewall-config** ツールを起動して、変換するアドレスのネットワークゾーンを選択します。**マスカレーディング** タブを選択し、チェックボックスを選択して **IPv4** アドレスの単一アドレスへの変換を有効にします。



注記

IPv6 のマスカレーディングを有効にするには、リッチルールを使用します。

4.5.3.1.10. ポート転送 (フォワーディング) の設定

特定のポートに向けられた受信ネットワークトラフィック または「パケット」を、内部アドレスまたは別のポートに転送するには、まず IP アドレスのマスカレーディングを有効にしてから、**ポート転送** タブを選択します。

受信トラフィックのプロトコルとポートもしくはポート範囲をウィンドウの上部で選択します。

ローカルポートにトラフィックを転送する、つまり同一システム上のポートに転送するには、**ローカル転送** のチェックボックスを選択します。トラフィックの送信先となるローカルポートまたはポート範囲を入力します。

トラフィックを別の **IPv4** アドレスに転送するには、**他のポートへの転送** チェックボックスを選択します。送信先の IP アドレスとポートまたはポート範囲を入力します。ポートのフィールドに入力がないと、デフォルトで同一ポートに送信されます。**OK** をクリックして変更を適用します。

4.5.3.1.11. ICMP フィルターの設定

ICMP フィルターを有効、無効にするには、**firewall-config** ツールを起動して、フィルターにかけるメッセージのネットワークゾーンを選択します。**ICMP フィルター** タブを選択し、フィルターにかけ

る **ICMP** メッセージの各タイプのチェックボックスを選択します。フィルターを無効にするには、チェックボックスの選択を外します。これは方向ごとに設定され、デフォルトではすべてが許可されます。

ICMP タイプを編集するには、**firewall-config** ツールを起動してから **設定** ラベルのあるドロップダウンメニューで **永続** モードを選びます。サービス ウィンドウの下部に新たなアイコンが表示されます。以下のダイアログで **はい** を選択して、マスカレーディングを有効化し、別の機能するマシンに転送します。

ICMP フィルター の反転を有効化するには、右側の **フィルターの反転** チェックボックスをクリックします。マークが付いた **ICMP** タイプが受け入れられ、その他はすべて拒否されます。DROP ターゲットを使用するゾーンでは、ドロップされます。

4.5.3.2. firewall-cmd コマンドラインツールを使用したファイアウォールの設定

firewall-cmd コマンドラインツールは、デフォルトでインストールされる **firewalld** アプリケーションの一部です。これがインストールされているかどうかを確認するには、バージョンを確認するか、ヘルプの出力を表示させます。バージョンのチェックは以下のコマンドで実行します。

```
~]$ firewall-cmd --version
```

ヘルプの出力を表示するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --help
```

以下でコマンドをいくつか紹介しています。完全一覧については、**man firewall-cmd(1)** man ページを参照してください。

注記

コマンドを永続的にするには、**--direct** コマンド（これらは元々一時的なもの）を除いてすべてのコマンドに **--permanent** オプションを追加します。こうすると変更が永続的になるだけでなく、変更はファイアウォールのリロード、サービスの再起動、もしくはシステムの再起動後にのみ適用されることに注意してください。**--permanent** オプションなしで **firewall-cmd** を使って設定すると、変更は即座に適用されますが、有効なのは次のファイアウォールのリロード、システム再起動、または **firewalld** サービスの再起動が行われるまでです。ファイアウォールのリロード自体は接続を切断しませんが、一時的な変更が破棄されることに注意してください。

コマンドを永続的にし、すぐに有効にするには、コマンドを **--permanent** を使用して 1 回、オプションなしで 1 回の合計 2 回入力します。これは、ファイアウォールのリロードには単にコマンドを繰り返すよりも時間がかかるためです（すべての設定ファイルをリロードし、ファイアウォール設定全体を再作成する必要があります）。リロード中に、安全上の理由により組み込みチェーンのポリシーは **DROP** に設定され、最終的に **ACCEPT** に再設定されます。したがって、リロード中にサービスが破棄されることがあります。



重要

--permanent --add-interface オプションは、**NetworkManager** ユーティリティによって管理されないインターフェースのみに使用します。これは、**NetworkManager** (つまり、従来のネットワークサービス) では、**ifcfg** インターフェース設定ファイルの **ZONE=** ディレクティブに従ってインターフェースがゾーンに自動的に追加されるためです。**NetworkManager** と **ifcfg** ファイルの使用については、『[Red Hat Enterprise Linux 7 ネットワークガイド](#)』を参照してください。

4.5.3.3. コマンドラインインターフェース (CLI) を使ったファイアウォール設定の表示

firewalld の状態をテキスト表示するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --state
```

アクティブなゾーンと、それらに割り当てられているインターフェースの一覧を表示するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --get-active-zones
public
interfaces: em1
```

たとえば、インターフェース **em1** が現在割り当てられているゾーンを確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --get-zone-of-interface=em1
public
```

public ゾーンなど、ゾーンに割り当てられているすべてのインターフェースを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=public --list-interfaces
em1 wlan0
```

この情報は **NetworkManager** から得られるもので、接続ではなくインターフェースのみを表示します。

public ゾーンなどのゾーンの全設定を確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=public --list-all
public
interfaces:
services: mdns dhcpv6-client ssh
ports:
forward-ports:
icmp-blocks: source-quench
```

現在読み込まれているサービスを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --get-services
cluster-suite pop3s bacula-client smtp ipp radius bacula ftp mdns samba
dhcpv6-client dns openvpn imaps samba-client http https ntp vnc-server
telnet libvirt ssh ipsec ipp-client amanda-client tftp-client nfs tftp
libvirt-tls
```

ここで表示されるのは、`/usr/lib/firewalld/services/` から読み込まれた事前定義サービスの名前と、現在読み込まれているカスタムサービスになります。設定ファイル自体は **service-name.xml** と命名されることに注意してください。

カスタムが作成されていても読み込まれていない場合は、以下のコマンドで確認できます。

```
~]# firewall-cmd --permanent --get-services
```

ここでは、サービスが読み込まれていなくても、`/etc/firewalld/services/` で設定されたカスタムサービスを含めたすべてのサービスが一覧表示されます。

4.5.3.4. コマンドラインインターフェース (CLI) を使ったファイアウォール設定の変更

4.5.3.4.1. 全パケットの遮断 (パニックモード)

すべての送受信パケットの遮断を開始するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --panic-on
```

これですべての送受信パケットが遮断されます。アクティブな接続は、一定期間アクションがないと切断されます。この期間は、個別のセッションタイムアウト値によって異なります。

すべての送受信パケットの受け渡しを再開するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --panic-off
```

パニックモードの有効期間が短かった場合、パニックモードを無効にすると、確立されていた接続が再度機能する可能性があります。

パニックモードの有効、無効を確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --query-panic
```

有効な場合は終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。

4.5.3.4.2. コマンドラインインターフェース (CLI) を使ったファイアウォール設定のリロード

ユーザー接続を切断せず (状態情報を失わずに) ファイアウォールをリロードするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --reload
```

ファイアウォールをリロードするには、すべての設定ファイルをリロードし、ファイアウォール設定全体を再作成する必要があります。リロード中に、安全上の理由により組み込みチェーンのポリシーは **DROP** に設定され、最終的に **ACCEPT** に再設定されます。したがって、リロード中にサービスが破棄されることがあります。

ユーザー接続を切断し、状態情報を破棄してファイアウォールをリロードするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --complete-reload
```

このコマンドは通常、重大なファイアウォール問題の場合にのみ使用されます。たとえば、状態情報に問題があって接続が確立されないものの、ファイアウォールルール自体は適正な場合などには、このコマンドを使用します。

4.5.3.4.3. コマンドラインインターフェース (CLI) を使用してインターフェースをゾーンに追加する手順

インターフェースをゾーンに追加するには (例: `em1` を **public** ゾーンに追加する場合など)、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=public --add-interface=em1
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

4.5.3.4.4. インターフェース設定ファイルを編集してインターフェースをゾーンに追加する

ifcfg-em1 設定ファイル (例: **work** ゾーンへの `em1` の追加) を編集してゾーンにインターフェースを追加するには、**root** として以下の行を **ifcfg-em1** に追加します。

```
ZONE=work
```

ZONE オプションを省略するか、**ZONE=** または **ZONE=''** を使用すると、デフォルトのゾーンが使用される点に注意してください。

NetworkManager が自動的に再接続を行い、ゾーンがそれに応じて設定されます。

4.5.3.4.5. ファイアウォール設定ファイルを編集してデフォルトゾーンを設定する手順

root で `/etc/firewalld/firewalld.conf` ファイルを開き、以下のように編集します。

```
# default zone
# The default zone used if an empty zone string is used.
# Default: public
DefaultZone=home
```

root で以下のコマンドを実行して、ファイアウォールをリロードします。

```
~]# firewall-cmd --reload
```

この結果、状態情報を失わずにファイアウォールがリロードされます (TCP セッションが中断されませんが、リロード中にサービスが破棄されることがあります)。

4.5.3.4.6. コマンドラインインターフェース (CLI) を使用したデフォルトゾーンの設定

public などにデフォルトのゾーンを設定するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --set-default-zone=public
```

この変更は即座に有効になり、今回の場合はファイアウォールのリロードは不要です。

4.5.3.4.7. コマンドラインインターフェース (CLI) を使用したファイアウォールのポート開放

dmz など、ゾーンで開放されている全ポートを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=dmz --list-ports
```

--add-services コマンドで開いたポートはここで表示されないことに注意してください。

(**dmz** ゾーンに対して、ポート **8080** への **TCP** トラフィックを許可するなど) ゾーンにポートを追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=dmz --add-port=8080/tcp
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

ポート範囲をゾーンに追加するには、たとえばポート **5060** から **5061** を **public** ゾーンで許可するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=public --add-port=5060-5061/udp
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

4.5.3.4.8. コマンドラインインターフェース (CLI) を使用したゾーンへのサービスの追加

SMTP を **work** ゾーンで許可するなど、サービスをゾーンに追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=work --add-service=smtp
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

4.5.3.4.9. コマンドラインインターフェース (CLI) を使用したゾーンからのサービスの削除

SMTP を **work** ゾーンから削除するなど、サービスをゾーンから削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=work --remove-service=smtp
```

この変更を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。この変更を加えても、確立されている接続は切断されません。切断する場合は **--complete-reload** オプションを使えますが、この場合切断されるのは削除したサービスだけでなく、確立されている接続がすべて切断されることに注意してください。

4.5.3.4.10. XML ファイルを編集してゾーンにサービスを追加する手順

デフォルトゾーンのファイルを表示するには、**root** で以下のコマンドを実行します。

```
~]# ls /usr/lib/firewalld/zones/  
block.xml  drop.xml      home.xml      public.xml  work.xml  
dmz.xml    external.xml  internal.xml  trusted.xml
```

これらのファイルは、編集しないでください。これらのファイルは、**/etc/firewalld/zones/** ディレクトリーに同等のファイルがない場合に、デフォルトで使用されます。

デフォルトから変更されたゾーンファイルを表示するには、**root** で以下のコマンドを実行します。

```
~]# ls /etc/firewalld/zones/
external.xml  public.xml  public.xml.old
```

上記の例では、work ゾーンのファイルは存在しません。**work** ゾーンのファイルを追加するには、**root** で以下のコマンドを実行します。

```
~]# cp /usr/lib/firewalld/zones/work.xml /etc/firewalld/zones/
```

これで **/etc/firewalld/zones/** ディレクトリー内のファイルを編集できます。ファイルを削除してしまった場合は、**firewalld** は **/usr/lib/firewalld/zones/** にあるデフォルトファイルを使用してフォールバックします。

SMTP を **work** ゾーンで許可するなど、サービスをゾーンに追加するには、**root** で **/etc/firewalld/zones/work.xml** に以下の行を追加します。

```
<service name="smtp"/>
```

4.5.3.4.11. XML ファイルを編集してサービスをゾーンから削除する手順

XML ゾーンファイルを編集するには、**root** 権限でエディターを実行する必要があります。以前に設定されたゾーンのファイルを表示するには、**root** で以下のコマンドを実行します。

```
~]# ls /etc/firewalld/zones/
external.xml  public.xml  work.xml
```

work ゾーンから **SMTP** を削除するなど、ゾーンからサービスを削除するには **root** 権限で **/etc/firewalld/zones/work.xml** ファイルを編集して以下の行を削除します。

```
<service name="smtp"/>
```

work.xml ファイルにそれ以外の変更がなされない場合には、これでこの行は削除され、**firewalld** は次回のリロード時またはシステム起動時にデフォルトの **/usr/lib/firewalld/zones/work.xml** 設定ファイルを使用します。

4.5.3.4.12. IP アドレスのマスカレードの設定

external ゾーンなどで IP マスカレーディングが有効かどうかを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=external --query-masquerade
```

有効な場合は終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。**zone** が省略されると、デフォルトのゾーンが使用されます。

IP マスカレーディングを有効にするには、**root** で以下のコマンドを発行します。

```
~]# firewall-cmd --zone=external --add-masquerade
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

IP マスカレーディングを無効にするには、**root** で以下のコマンドを発行します。


```
~]# firewall-cmd --zone=external --remove-masquerade
```

この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

4.5.3.4.13. コマンドラインインターフェース (CLI) を使ったポート転送 (フォワーディング) の設定

受信ネットワークパケットをポートから別のポートやアドレスに転送するには、まず **root** で以下のコマンドを実行して、ゾーン (例: **external**) の IP アドレスマスカレーディングを有効にします。

```
~]# firewall-cmd --zone=external --add-masquerade
```

パケットをローカルポート (同一システム上の別のポート) に転送するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=external --add-forward-port=port=22:proto=tcp:toport=3753
```

この例では、ポート **22** に向けられていたパケットがポート **3753** に転送されます。元の宛先のポートは、**port** オプションで指定します。このオプションでは、プロトコルとともにポートまたはポート範囲を指定できます。プロトコルが指定された場合、プロトコルは **tcp**、または **udp** のいずれかにする必要があります。トラフィックの転送先となるポートまたはポート範囲である新しいローカルポートは、**toport** オプションで指定します。この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

宛先ポートを変更せずに、パケットを別の **IPv4** アドレス (通常は内部アドレス) に転送するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=external --add-forward-port=port=22:proto=tcp:toaddr=192.0.2.55
```

この例では、ポート **22** に向けられていたパケットがポート **toaddr** で指定したアドレスと同じポートに転送されます。元の宛先のポートは、**port** オプションで指定します。このオプションでは、プロトコルとともにポートまたはポート範囲を指定できます。プロトコルが指定された場合、プロトコルは **tcp**、または **udp** のいずれかにする必要があります。トラフィックの転送先となるポートまたはポート範囲である新しい宛先ポートは、**toport** オプションで指定します。この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

パケットを別の **IPv4** アドレス (通常は内部アドレス) の別のポートに転送するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --zone=external \
--add-forward-port=port=22:proto=tcp:toport=2055:toaddr=192.0.2.55
```

この例では、ポート **22** に向けられていたパケットがポート **toaddr** オプションで指定したアドレスの **2055** ポートに転送されます。元の宛先のポートは、**port** オプションで指定します。このオプションでは、プロトコルとともにポートまたはポート範囲を指定できます。プロトコルが指定された場合、プロトコルは **tcp**、または **udp** のいずれかにする必要があります。トラフィックの転送先となるポートまたはポート範囲である新しい宛先ポートは、**toport** オプションで指定します。この設定を永続的にするには、**--permanent** オプションを追加してコマンドを繰り返します。

4.5.3.5. XML ファイルを使ったファイアウォールの設定

firewalld の設定は、XML ファイルで **/etc/firewalld/** ディレクトリーに保存されています。

す。`/usr/lib/firewalld/` ディレクトリー内のファイルは (デフォルト設定なので)、変更しないでください。XML ファイルの閲覧と変更には、**root** ユーザー権限が必要になります。XML ファイルは、以下の 3 つの man ページで説明されています。

- **firewalld.icmptype(5)** man ページ —**ICMP** フィルタリングの XML 設定ファイルについて説明しています。
- **firewalld.service(5)** man ページ —**firewalld service** の XML 設定ファイルについて説明しています。
- **firewalld.zone(5)** man ページ —**firewalld** ゾーン設定の XML 設定ファイルについて説明しています。

XML ファイルは直接作成、編集するか、グラフィカルおよびコマンドラインツールを使って間接的に作成することができます。組織内では RPM ファイルでこれらを配布することで、管理とバージョンコントロールが容易になります。**Puppet** のようなツールを使うと、このような設定ファイルの配布が可能になります。

4.5.3.6. ダイレクトインターフェースの使用

firewall-cmd ツールで **--direct** オプションを使うと、ランタイム時にチェーンの追加、削除が可能になります。ここではいくつかの例を紹介していますが、詳細は **firewall-cmd(1)** man ページを参照してください。

ダイレクトインターフェースの使用は意図せずにファイアウォール侵害を引き起こす可能性があるので、**iptables** に精通していない場合には危険です。

ダイレクトインターフェースモードは、サービスもしくはアプリケーションが実行時に特定のファイアウォールルールを追加するためのものです。**--permanent** オプションを追加して **firewall-cmd -permanent --direct** コマンドを使用するか、`/etc/firewalld/direct.xml` を修正することで、ルールを永続的なものにできます。`/etc/firewalld/direct.xml` ファイルの詳細は、**firewalld.direct(5)** の man ページを参照してください。

4.5.3.6.1. ダイレクトインターフェースを使用したカスタムルールの追加

「IN_public_allow」チェーンにカスタムルールを追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --direct --add-rule ipv4 filter IN_public_allow \
    0 -m tcp -p tcp --dport 666 -j ACCEPT
```

--permanent オプションを追加して設定を永続化します。

4.5.3.6.2. ダイレクトインターフェースを使用したカスタムルールの削除

「IN_public_allow」チェーンからカスタムルールを削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --direct --remove-rule ipv4 filter IN_public_allow \
    0 -m tcp -p tcp --dport 666 -j ACCEPT
```

--permanent オプションを追加して設定を永続化します。

4.5.3.6.3. ダイレクトインターフェースを使用したカスタムルールの一覧表示

「IN_public_allow」のルールを表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --direct --get-rules ipv4 filter IN_public_allow
```

このコマンド (**--get-rules** オプション) は、**--add-rule** オプションを使用して以前に追加されたルールのみを表示します。他の手段で追加された **iptables** ルールは表示されません。

4.5.3.7. 「リッチ言語」構文を使った複雑なファイアウォールルールの設定

「リッチ言語」構文を使用すると、ダイレクトインターフェースよりも理解しやすい方法で複雑なファイアウォールルールが作成できます。さらに、設定を永続的にすることができます。この言語は値の付いたキーワードを使用するもので、**iptables** ルールの抽象表現です。ゾーンはこの言語を使って設定することができ、現行の設定方式もそのままサポートされます。

4.5.3.7.1. リッチ言語コマンドの形式

このセクションのコマンドはすべて **root** で実行する必要があります。ルールを追加するコマンド形式は以下のとおりです。

```
firewall-cmd [--zone=zone] --add-rich-rule='rule' [--timeout=timeval]
```

これでリッチ言語ルール (*rule*) がゾーン (*zone*) に追加されます。このオプションは複数回指定することができます。ゾーンが省略されると、デフォルトのゾーンが使用されます。タイムアウトが指定されていれば、ルールは指定の秒数の間、アクティブになり、その後自動的に削除されます。時間の値の後に **s** (秒)、**m** (分) または **h** (時間) を追加して、時間の単位を指定することができます。デフォルトは秒です。

ルールの削除

```
firewall-cmd [--zone=zone] --remove-rich-rule='rule'
```

これでゾーン (*zone*) のリッチ言語のルール (*rule*) が削除されます。このオプションは複数回指定することができます。ゾーンが省略されると、デフォルトのゾーンが使用されます。

ルールが存在するかの確認

```
firewall-cmd [--zone=zone] --query-rich-rule='rule'
```

このコマンドは、リッチ言語ルールの *rule* がゾーン *zone* に追加されたかどうかを返します。有効な場合は終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。ゾーンが省略されると、デフォルトのゾーンが使用されます。

For information about the rich language representation used in the zone configuration files, see the `firewalld.zone(5)` man page.

4.5.3.7.2. リッチルールの構造について

リッチルールコマンドの形式または構造は、以下のようになります。

```
rule [family="rule family"]
    [ source address="address" [invert="True"] ]
    [ destination address="address" [invert="True"] ]
    [ element ]
    [ log [prefix="prefix text"] [level="log level"] [limit
```

```
value="rate/duration"] ]
  [ audit ]
  [ action ]
```

ルールは特定のゾーンに関連付けられます。ゾーンには複数のルールを関連付けることができます。いくつかのルールが相互に作用する、または矛盾する場合は、パケットに適合する最初のルールが適用されます。

4.5.3.7.3. リッチルールのコマンドオプションについて

family

ルール family が提供されている場合は、**ipv4** か **ipv6** になり、ルールを **IPv4** または **IPv6** に制限します。ルール family が提供されていない場合は、このルールは **IPv4** と **IPv6** の両方に追加されます。ルール内でソースまたは宛先のアドレスが使用されていると、ルール family を提供する必要があります。これは、ポート転送の場合でも同じです。

ソースおよび宛先のアドレス

source

ソースのアドレスを指定すると、接続試行の元を指定したソースアドレスに限定することができます。ソースアドレスまたはアドレスの範囲は、**IPv4** または **IPv6** のマスクを伴う IP アドレスかネットワーク IP アドレスになります。ネットワーク family (**IPv4** または **IPv6**) が自動的に見つけられます。**IPv4** のマスクは、ネットワークマスクか単純な番号になります。**IPv6** のマスクは、単純な番号になります。ホスト名の使用はサポートされていません。**invert="true"** または **invert="yes"** を追加すると、ソースアドレスコマンドの意味が逆になり、提供されたアドレス以外のものがマッチすることになります。

destination

宛先のアドレスを指定すると、ターゲットを指定した宛先アドレスに限定することができます。宛先アドレスでは、ソースアドレスと同様の構文を使用します。ソースおよび宛先アドレスの使用はオプションで、宛先アドレスの使用はすべての要素と使用できるわけではありません。これは、サービスエントリーにおける宛先アドレスの使用などによります。

要素

要素は、以下のいずれか **ひとつのみ** のタイプになります。**service**、**port**、**protocol**、**masquerade**、**icmp-block** または **forward-port** です。

service

service 要素は、**firewalld** が提供するサービスのひとつです。事前定義のサービス一覧を入手するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --get-services
```

サービスが宛先アドレスを提供する場合、ルール内の宛先アドレスと競合し、エラーが発生します。内部で宛先アドレスを使用するサービスのほとんどは、マルチキャストを使用するサービスです。コマンドは以下の形式になります。

```
service name=service_name
```

port

port 要素は、単一のポート番号か **5060-5062** のようなポート範囲のどちらかで、その後に **tcp** か **udp** のプロトコルが続きます。コマンドは以下の形式になります。

```
port port=number_or_range protocol=protocol
```

protocol

プロトコルの値は、プロトコル ID 番号かプロトコル名になります。利用可能なプロトコルエントリーについては、**/etc/protocols** を参照してください。コマンドは以下の形式になります。

```
protocol value=protocol_name_or_ID
```

icmp-block

ひとつ以上の **ICMP** タイプをブロックするには、このコマンドを使用します。**ICMP** タイプは、**firewalld** がサポートする **ICMP** タイプのひとつになります。サポートされる **ICMP** タイプの一覧を入手するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --get-icmptypes
```

ここではアクションの特定はできません。**icmp-block** は **reject** のアクションを内部で使用します。コマンドは以下の形式になります。

```
icmp-block name=icmptype_name
```

masquerade

ルール内の IP マスカレードを有効にします。ソースアドレスを提供するとこの分野へのマスカレードを制限できますが、宛先アドレスは制限できません。ここではアクションの特定はできません。

forward-port

tcp または **udp** として指定されたプロトコルのローカルポートから別のローカルポート、別のマシン、または別のマシン上の別のポートにパケットを転送します。**port** および **to-port** は、単一のポート番号もしくはポート範囲のどちらでも構いません。宛先アドレスは、単純な IP アドレスになります。ここではアクションの特定はできません。**forward-port** コマンドは **accept** のアクションを内部で使用します。コマンドは以下の形式になります。

```
forward-port port=number_or_range protocol=protocol /
to-port=number_or_range to-addr=address
```

ロギング

log

syslog などのカーネルロギングでルールへの新たな接続試行を記録します。ログメッセージに接頭辞として追加される接頭辞テキストを定義することができます。ログレベルは、**emerg**、**alert**、**crit**、**error**、**warning**、**notice**、**info** または **debug** のいずれかになります。ログの使用はオプションです。ログの使用は以下のように制限できます。

```
log [prefix=prefix text] [level=log level] limit value=rate/duration
```

rate は正の自然数 [1, ..] で、**s**、**m**、**h**、**d** は時間の長さになります。**s** は秒数、**m** は分数、**h** は時間数、**d** は日数を表します。制限の最大値は **1/d** で、これは 1 日あたり最大 1 ログエントリーになります。

audit

Audit は、サービス **auditd** に送信された監査記録を使ってロギングの別の方法を提供します。audit タイプは **ACCEPT**、**REJECT** または **DROP** のいずれかになりますが、これはルールアクションから自動的に獲得されるので、**audit** コマンドの後では指定されません。Audit にはそれ自体のパラメーターはありませんが、オプションで制限を加えることができます。Audit の使用はオプションになります。

アクション

accept|reject|drop

アクションは **accept**、**reject** または **drop** のいずれかになります。ルールに含めることができるのは、単一の要素またはソースのみになります。ルールに要素が含まれている場合、その要素のマッチする新たな接続はそのアクションで処理されます。ルールにソースが含まれている場合、そのソースアドレスからのものがすべて指定されたアクションで処理されます。

```
accept | reject [type=reject type] | drop
```

accept を使うと、新たな接続試行がすべて許可されます。**reject** を使うとそれらは拒否され、そのソースは拒否メッセージを受け取ります。拒否のタイプは、別の値を使用するように設定できます。**drop** を使うと、すべてのパケットが即座に切断され、ソースにはなにも情報が送られません。

4.5.3.7.4. リッチルールログコマンドの使用

ロギングは **Netfilter** ログターゲットおよび audit ターゲットを使用して実行できます。「zone_log」という形式の名前で新たなチェーンがすべてのゾーンに追加されます。ここでの zone はゾーン名になります。適切な順序にするために、これは **deny** チェーンの前に処理されます。ルールまたはルールの部分は以下のようにルールのアクションにしたがって、別個のチェーンに置かれます。

```
zone_log
zone_deny
zone_allow
```

ロギングのルールはすべて「zone_log」チェーンに置かれ、これが最初に解析されます。**reject** および **drop** ルールはすべて「zone_deny」チェーンに置かれ、これは log チェーンの後には解析されます。**accept** ルールはすべて「zone_allow」チェーンに置かれ、これは **deny** チェーンの後には解析されます。ルールに **log** と **deny** または **allow** アクションが含まれる場合、これらのアクションを指定しているルールの部分は一致するチェーンに置かれます。

4.5.3.7.4.1. リッチルールログコマンドの使用例 1

認証ヘッダープロトコル **AH** 用に新たな **IPv4** および **IPv6** 接続を有効にします：

```
rule protocol value="ah" accept
```

4.5.3.7.4.2. リッチルールログコマンドの使用例 2

プロトコル **FTP** および audit を使用した 1 分あたり 1 件のログ用に新たな **IPv4** および **IPv6** 接続を許可します：

```
rule service name="ftp" log limit value="1/m" audit accept
```

4.5.3.7.4.3. リッチルールログコマンドの使用例 3

プロトコル **TFTP** と **syslog** を使用した毎分 1 件のログ用にアドレス **192.168.0.0/24** からの新たな **IPv4** 接続を許可します。

```
rule family="ipv4" source address="192.168.0.0/24" service name="tftp" log
prefix="tftp" level="info" limit value="1/m" accept
```

4.5.3.7.4.4. リッチルールログコマンドの使用例 4

プロトコル **RADIUS** 用の **1:2:3:4:6::** からの新たな **IPv6** 接続はすべて拒否されます。他のソースからの新たな **IPv6** 接続は許可されます。

```
rule family="ipv6" source address="1:2:3:4:6::" service name="radius" log
prefix="dns" level="info" limit value="3/m" reject
rule family="ipv6" service name="radius" accept
```

4.5.3.7.4.5. リッチルールログコマンドの使用例 5

プロトコル **TCP** を使ってポート 4011 で **1:2:3:4:6::** から受信した **IPv6** パケットを、ポート 4012 上の **1::2:3:4:7** に転送します。

```
rule family="ipv6" source address="1:2:3:4:6::" forward-port to-
addr="1::2:3:4:7" to-port="4012" protocol="tcp" port="4011"
```

4.5.3.7.4.6. リッチルールログコマンドの使用例 6

ソースアドレスをホワイトリスト化してそのソースからの接続すべてを許可します。

```
rule family="ipv4" source address="192.168.2.2" accept
```

他の例については、**firewalld.richlanguage(5)** man ページを参照してください。

4.5.3.8. ファイアウォールのロックダウン

ローカルのアプリケーションやサービスは、**root** で実行していれば (たとえば **libvirt**) ファイアウォール設定を変更することができます。この機能を使うと、管理者はファイアウォール設定をロックして、どのアプリケーションもファイアウォール変更を要求できなくするか、ロックダウンのホワイトリストに追加されたアプリケーションのみがファイアウォール変更を要求できるようにすることが可能になります。ロックダウン設定はデフォルトで無効になっています。これを有効にすると、ローカルのアプリケーションやサービスによるファイアウォールの望ましくない変更を確実に防止することができます。

4.5.3.8.1. ファイアウォールロックダウンの設定

root でエディターを使用し、以下の行を **/etc/firewalld/firewalld.conf** ファイルに追加します。

```
Lockdown=yes
```

root で以下のコマンドを実行して、ファイアウォールをリロードします。

```
~]# firewall-cmd --reload
```


管理ユーザーとして、つまり **wheel** グループのユーザーとして (通常、システムの最初のユーザー) 以下のコマンドを実行し、デフォルトゾーンで **imaps** サービスの有効化を図ります。ユーザーパスワードが求められます。

```
~]$ firewall-cmd --add-service=imaps
Error: ACCESS_DENIED: lockdown is enabled
```

firewall-cmd の使用を有効にするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python -Es /usr/bin/firewall-cmd*'
```

この設定を永続的にするには、**--permanent** オプションを追加します。

root でファイアウォールをリロードします。

```
~]# firewall-cmd --reload
```

管理ユーザーとして以下のコマンドを実行し、再度、デフォルトゾーンで **imaps** サービスの有効化を図ります。ユーザーパスワードが求められます。

```
~]$ firewall-cmd --add-service=imaps
```

これでコマンド実行が成功します。

4.5.3.8.2. コマンドラインクライアントを使用したロックダウンの設定

ロックダウンが有効になっているかどうかを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --query-lockdown
```

ロックダウンが有効な場合は終了ステータスが **0** で **yes** が出力され、無効の場合は終了ステータスが **1** で **no** が出力されます。

ロックダウンを有効にするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --lockdown-on
```

ロックダウンを無効にするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --lockdown-off
```

4.5.3.8.3. コマンドラインを使用したロックダウンホワイトリストオプションの設定

ロックダウンのホワイトリストには、コマンドやセキュリティのコンテキスト、ユーザー、およびユーザー ID を含めることができます。ホワイトリストのコマンドエントリーがアスタリスク「*」で終わっている場合、そのコマンドで始まるすべてのコマンドラインがマッチすることになります。「*」がなければ、引数を含めたコマンドが完全に一致する必要があります。

ここでのコンテキストは、実行中のアプリケーションやサービスのセキュリティ (SELinux) コンテキストです。実行中のアプリケーションのコンテキストを確認するには、以下のコマンドを実行します。

```
~]$ ps -e --context
```

このコマンドで、実行中のアプリケーションすべてが返されます。**grep** ツールを使用して、出力から目的のアプリケーションを以下のようにパイプ処理します。

```
~]$ ps -e --context | grep example_program
```

ホワイトリストにあるコマンドラインすべてを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --list-lockdown-whitelist-commands
```

ホワイトリストにコマンド *command* を追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-lockdown-whitelist-command='/usr/bin/python -Es /usr/bin/command'
```

ホワイトリストからコマンド *command* を削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --remove-lockdown-whitelist-command='/usr/bin/python -Es /usr/bin/command'
```

ホワイトリストにコマンド *command* があるかどうかを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --query-lockdown-whitelist-command='/usr/bin/python -Es /usr/bin/command'
```

True の場合は終了ステータスが **0** で **yes** が出力され、False の場合は終了ステータスが **1** で **no** が出力されます。

ホワイトリストにあるセキュリティコンテキストすべてを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --list-lockdown-whitelist-contexts
```

ホワイトリストにコンテキスト *context* を追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-lockdown-whitelist-context=context
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストからコンテキスト *context* を削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --remove-lockdown-whitelist-context=context
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストにコンテキスト *context* があるかどうかを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --query-lockdown-whitelist-context=context
```

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

ホワイトリストにあるユーザー ID すべてを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --list-lockdown-whitelist-uids
```

ホワイトリストにユーザー ID *uid* を追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-lockdown-whitelist-uid=uid
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストからユーザー ID *uid* を削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --remove-lockdown-whitelist-uid=uid
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストにユーザー ID *uid* があるかどうかを確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --query-lockdown-whitelist-uid=uid
```

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

ホワイトリストにあるユーザー名すべてを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --list-lockdown-whitelist-users
```

ホワイトリストにユーザー名 *user* を追加するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-lockdown-whitelist-user=user
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストからユーザー名 *user* を削除するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --remove-lockdown-whitelist-user=user
```

この設定を永続的にするには、**--permanent** オプションを追加します。

ホワイトリストにユーザー名 *user* があるかどうかを確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --query-lockdown-whitelist-user=user
```

コマンドがある場合は終了ステータスが **0** で **yes** が出力され、ない場合は終了ステータスが **1** で **no** が出力されます。

4.5.3.8.4. 設定ファイルを使用したロックダウンホワイトリストオプションの設定

デフォルトのホワイトリスト設定ファイルには、**NetworkManager** コンテキストと **libvirt** のデフォルトのコンテキストが含まれています。また、ユーザー ID 0 がリストに含まれています。

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist>
  <selinux context="system_u:system_r:NetworkManager_t:s0"/>
  <selinux context="system_u:system_r:virttd_t:s0-s0:c0.c1023"/>
  <user id="0"/>
</whitelist>
```

以下のホワイトリスト設定ファイルの例では、**firewall-cmd** ユーティリティーのコマンドと、ユーザー ID が **815** である **user** のコマンドすべてを有効にしています。

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist>
  <command name="/usr/bin/python -Es /bin/firewall-cmd*"/>
  <selinux context="system_u:system_r:NetworkManager_t:s0"/>
  <user id="815"/>
  <user name="user"/>
</whitelist>
```

上記の例では **user id** と **user name** の両方が使われていますが、実際にはどちらか一方のみのオプションが必要になります。インタプリタは Python なので、コマンドラインに追加されています。または、以下のような明確なコマンドを使用することもできます。

```
/usr/bin/python /bin/firewall-cmd --lockdown-on
```

この例では、**--lockdown-on** コマンドのみが許可されます。

注記

Red Hat Enterprise Linux 7 では、すべてのユーティリティーが **/usr/bin/** ディレクトリーに格納され、**/bin/** ディレクトリーは **/usr/bin/** ディレクトリーにシンボリックリンクされています。つまり、**root** で **firewall-cmd** のパスを実行すると **/bin/firewall-cmd** に対して解決しますが、**/usr/bin/firewall-cmd** が使用できるようになっています。新たなスクリプトはすべて新しい格納場所を使うべきですが、**root** で実行するスクリプトが **/bin/firewall-cmd** のパスを使用するようになっていれば、通常 **root** ユーザー以外のみを使用される **/usr/bin/firewall-cmd** パスに加えてこのコマンドパスもホワイトリストに加える必要があります。

コマンドの名前属性の最後にある「*」は、それで始まるすべてのコマンドが一致することを意味します。「*」がなければ、引数を含めたコマンドが完全に一致する必要があります。

4.5.4. iptables サービスの使用

firewalld の代わりに **iptables** および **ip6tables** の各サービスを使用するには、まず **root** で以下のコマンドを実行して **firewalld** を無効にします。

```
~]# systemctl disable firewalld
~]# systemctl stop firewalld
```

そして **root** で以下のコマンドを実行して、**iptables-services** パッケージをインストールします。

```
~]# yum install iptables-services
```

iptables-services パッケージには、**iptables** と **ip6tables** のサービスが含まれています。

インストール後に **root** で以下のコマンドを実行して、**iptables** および **ip6tables** のサービスを開始します。

```
~]# systemctl start iptables
~]# systemctl start ip6tables
```

システム起動時にサービスの起動を有効にするには、以下のコマンドを入力します。

```
~]# systemctl enable iptables
~]# systemctl enable ip6tables
```

4.5.4.1. IPTables および IP セット

ipset ユーティリティーは、Linux カーネルで *IP セット* を管理するために使用します。IP セットは IP アドレス、ポート番号、IP と MAC アドレスのペア、または IP アドレスとポート番号のペアを格納するフレームワークです。IP セットが非常に大きい場合であっても IP セットに対する照合が非常に高速に行われるよう IP セットにはインデックスが作成されます。IP セットにより、設定が簡素化され、管理しやすくなり、**iptables** を使用した場合にパフォーマンスが向上します。セットを参照する **iptables** のマッチとターゲットにより、カーネルの該当するセットを保護するリファレンスが作成されます。セットを参照するリファレンスが 1 つでもある限り、セットを破棄することはできません。

ipset を使用すると、以下のような **iptables** コマンドをセットに置き換えることができます。

```
~]# iptables -A INPUT -s 10.0.0.0/8 -j DROP
~]# iptables -A INPUT -s 172.16.0.0/12 -j DROP
~]# iptables -A INPUT -s 192.168.0.0/16 -j DROP
```

セットは以下のように作成されます。

```
~]# ipset create my-block-set hash:net
~]# ipset add my-block-set 10.0.0.0/8
~]# ipset add my-block-set 172.16.0.0/12
~]# ipset add my-block-set 192.168.0.0/16
```

セットは、以下のように **iptables** コマンドで参照されます。

```
~]# iptables -A INPUT -m set --set my-block-set src -j DROP
```

セットが複数回使用される場合は、設定時間が短縮されます。セットに多くのエントリーが含まれる場合は、処理時間が短縮されます。

4.5.4.1.1. firewalld での IP セットの使用

firewalld で IP セットを使用するには、セットを参照するために永続的な直接ルールが必要であり、各 **ipset** に対して **firewalld** が起動する前にカスタムサービスを作成および起動する必要があります。永続的な直接ルールは、**/etc/firewalld/direct.xml** ファイルを使用して追加できます。

手順4.1 IP セットのカスタムサービスの設定

firewalld が起動する前に、カスタムサービスが IP セット構造を作成およびロードするよう設定します。

1. **root** でエディターを使用して、以下のようにファイルを作成します。

```
~]# vi /etc/systemd/system/ipset_name.service
[Unit]
Description=ipset_name
Before=firewalld.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/ipset_name.sh start
ExecStop=/usr/local/bin/ipset_name.sh stop

[Install]
WantedBy=basic.target
```

2. **firewalld** で IP セットを永続的に使用します。

```
~]# vi /etc/firewalld/direct.xml
<?xml version="1.0" encoding="utf-8"?>
<direct>
  <rule ipv="ipv4" table="filter" chain="INPUT" priority="0">-m set
  --match-set <replaceable>ipset_name</replaceable> src -j DROP</rule>
</direct>
```

3. 変更を反映するには **firewalld** をリロードする必要があります。

```
~]# firewall-cmd --reload
```

この結果、状態情報を失わずにファイアウォールがリロードされます (TCP セッションが中断されません) が、リロード中にサービスが破棄されることがあります。

4.5.4.1.2. ipset のインストール

ipset ユーティリティーをインストールするには、以下のコマンドを **root** で実行します。

```
~]# yum install ipset
```

使用法のメッセージを表示するには、以下のコマンドを実行します。

```
~]$ ipset --help
ipset v6.11

Usage: ipset [options] COMMAND出力省略
```

4.5.4.1.3. ipset コマンド

ipset コマンドの形式は以下のようになります。

```
ipset [options] command [command-options]
```

Where *command* is one of:

```
create | add | del | test | destroy | list | save | restore | flush |
rename | swap | help | version | -
```

Allowed *options* are:

```
-exist | -output [ plain | save | xml ] | -quiet | -resolve | -sorted | -
name | -terse
```

create コマンドは、IP データのセットを格納する新しいデータ構造を作成するために使用します。**add** コマンドを使用すると、セットに新しいデータが追加され、追加されたデータはセットの要素として参照されます。

-exist オプションを使用すると、要素がすでに存在する場合に、エラーメッセージが表示されず、タイムアウト値を更新する特別なロールが提供されます。タイムアウトを変更するには、**ipset add** コマンドを使用し、必要に応じてタイムアウト値のみを変更して **-exist** オプションを使用することにより要素のすべてのデータを再び指定します。

test オプションは、要素がすでにセット内に存在する場合のテスト用です。

create コマンドの形式は以下のようになります。

```
ipset create set-name type-name [create-options]
```

set-name は、ユーザーにより選択された適切な名前であり、*type-name* はセットを構成するデータを格納するために使用するデータ構造の名前です。*type-name* の形式は以下のとおりです。

```
method:datatype[,datatype[,datatype]]
```

データの格納に許可される方法は以下のとおりです。

```
bitmap | hash | list
```

許可されるデータタイプは以下のとおりです。

```
ip | net | mac | port | iface
```

セット内でエントリー追加、削除、またはテストする場合は、セット内で 1 つのエントリー (つまり要素) を構成するデータに、カンマで区切られた同じデータを使用する必要があります。以下に例を示します。

```
ipset add set-name ipaddr,portnum,ipaddr
```



注記

セットには **IPv4** アドレスと **IPv6** アドレスを同時に含めることはできません。セットは作成されたときにファミリーにバインドされます (**IPv4** の場合は **inet**、**IPv6** の場合は **inet6**。デフォルト値は **inet**)。

例4.2 IP セットの作成

ソース IP アドレス、ポート、および宛先 IP アドレスで構成される IP セットを作成するには、コマンドを以下のように実行します。

```
~]# ipset create my-set hash:ip,port,ip
```

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-set 192.168.1.2,80,192.168.2.2
~]# ipset add my-set 192.168.1.2,443,192.168.2.2
```

セットタイプには共通して以下のオプションのパラメーターがあります。これらのパラメーターを使用するには、セットの作成時にパラメーターを指定する必要があります。

- **timeout** — **create** コマンドで提供された値は、作成されたセットのデフォルト値になります。**add** コマンドで提供された値は要素の非デフォルト初期値になります。
- **counters** — オプションが**create** コマンドで提供された場合は、セット内の各要素に対してパケットおよびバイトカウンターが作成されます。値が **add** コマンドで提供されない場合、カウンターはゼロから始まります。
- **comment** — オプションが**create** コマンドで提供された場合は、追加される要素の目的を文書化するために **add** コマンドで引用符で囲まれたテキスト文字列を渡すことができます。引用符は文字列内で許可されず、エスケープ文字は IP セット内で無効になることに注意してください。

例4.3 IP セットのリスト

特定の IP セットである **my-set** の内容をリストするには、以下のようにコマンドを実行します。

```
~]# ipset list my-set
Name: my-set
Type: hash:ip,port,ip
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 8360
References: 0
Members:
192.168.1.2,tcp:80,192.168.2.2
192.168.1.2,tcp:443,192.168.2.2
```

すべてのセットをリストするにはセット名を省略します。

例4.4 IP セットの要素のテスト

大きいセットの内容をリストするには時間がかかります。以下のように、要素があるかどうかテストできます。

```
~]# ipset test my-set 192.168.1.2,80,192.168.2.2
192.168.1.2,tcp:80,192.168.2.2 is in set my-set.
```

4.5.4.1.4. IP セットタイプ

bitmap:ip

セットの作成時に **netmask** オプションが使用される場合は、IPv4 ホストアドレス、ネットワーク範囲、またはプレフィックス長が CIDR 表記の IPv4 ネットワークアドレスを格納します。オプションでタイムアウト値、カウンター値、およびコメントを格納できます。最大で **65536** のエントリーを格納できます。**bitmap:ip** セットを作成するコマンドの形式は以下のとおりです。


```
ipset create set-name range start_ipaddr-end_ipaddr |ipaddr/prefix-length [netmask prefix-length] [timeout value] [counters] [comment]
```

例4.5 プレフィックス長を使用したアドレス範囲の IP セットの作成

プレフィックス長を使用してアドレス範囲の IP セットを作成するには、以下のように **bitmap:ip** セットタイプを使用します。

```
~]# ipset create my-range bitmap:ip range 192.168.33.0/28
```

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-range 192.168.33.1
```

リストのメンバーを確認します。

```
~]# ipset list my-range
Name: my-range
Type: bitmap:ip
Header: range 192.168.33.0-192.168.33.15
Size in memory: 84
References: 0
Members:
192.168.33.1
```

アドレス範囲を追加するには、以下のようにコマンドを実行します。

```
~]# ipset add my-range 192.168.33.2-192.168.33.4
```

リストのメンバーを確認します。

```
~]# ipset list my-range
Name: my-range
Type: bitmap:ip
Header: range 192.168.33.0-192.168.33.15
Size in memory: 84
References: 0
Members:
192.168.33.1
192.168.33.2
192.168.33.3
192.168.33.4
```

例4.6 ネットマスクを使用したアドレス範囲の IP セットの作成

ネットマスクを使用してアドレス範囲の IP セットを作成するには、以下のように **bitmap:ip** セットタイプを使用します。

```
~]# ipset create my-big-range bitmap:ip range 192.168.124.0-192.168.126.0 netmask 24
```

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-big-range 192.168.124.0
```

アドレスを追加しようとする、そのアドレスを含む範囲が追加されます。

```
~]# ipset add my-big-range 192.168.125.150
~]# ipset list my-big-range
Name: my-big-range
Type: bitmap:ip
Header: range 192.168.124.0-192.168.126.255 netmask 24
Size in memory: 84
References: 0
Members:
192.168.124.0
192.168.125.0
```

bitmap:ip,mac

IPv4 アドレスと MAC アドレスをペアとして格納します。最大で **65536** のエントリーを格納できます。

```
ipset create my-range bitmap:ip,mac range start_ipaddr-end_ipaddr |
ipaddr/prefix-length [timeout value ] [counters] [comment]
```

例4.7 IPv4 MAC アドレスペアの範囲の IP セットの作成

IPv4 MAC アドレスペアの範囲の IP セットを作成するには、以下のように **bitmap:ip,mac** セットタイプを使用します。

```
~]# ipset create my-range bitmap:ip,mac range 192.168.1.0/24
```

セットを作成する場合に、MAC アドレスを指定する必要はありません。

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-range 192.168.1.1,12:34:56:78:9A:BC
```

bitmap:port

ポート範囲を格納します。最大で **65536** のエントリーを格納できます。

```
ipset create my-port-range bitmap:port range start_port-end_port
[timeout value ] [counters] [comment]
```

設定されたマッチおよび SET ターゲット netfilter カーネルモジュールは、格納された番号を TCP または UDP ポート番号として解釈します。プロトコルはオプションでポートともに指定できます。サービス名が使用され、その名前が TCP サービスとして存在しない場合は、**proto** のみを指定する必要があります。

例4.8 ポート範囲の IP セットの作成

ポート範囲の IP セットを作成するには、以下のように **bitmap:port** セットタイプを使用します。

```
~]# ipset create my-permitted-port-range bitmap:port range 1024-49151
```

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-permitted-port-range 5060-5061
```

hash:ip

ホストまたはネットワークアドレスをハッシュ形式で格納します。デフォルトでは、ネットワークプレフィックス長なしで指定されたアドレスがホストアドレスになります。すべてがゼロの IP アドレスを格納することはできません。

```
ipset create my-addresses hash:ip [family[ inet | inet6 ]] [hashsize  
value] [maxelem value ] [netmask prefix-length] [timeout value ]
```

inet ファミリーがデフォルト値になります。**family** が省略されると、アドレスは IPv4 アドレスとして解釈されます。**hashsize** 値は使用する初期ハッシュサイズであり、デフォルトで**1024**に設定されます。**maxelem** 値はセットに格納できる要素の最大数であり、デフォルトで**65536**に設定されます。

netfilter ツールは最も明確なネットワークプレフィックスを検索し、一致するアドレスの最も小さいブロックを見つけようとします。

例4.9 IP アドレスの IP セットの作成

IP アドレスの IP セットを作成するには、以下のように **hash:ip** セットタイプを使用します。

```
~]# ipset create my-addresses hash:ip
```

セットが作成されると、エントリーは以下のように追加できます。

```
~]# ipset add my-addresses 10.10.10.0
```

ネットマスクやタイムアウトなどの追加のオプションが必要な場合は、セットの作成時にオプションを指定する必要があります。例を以下に示します。

```
~]# ipset create my-busy-addresses hash:ip maxelem 24 netmask 28 timeout  
100
```

maxelem オプションを使用すると、セット内の要素の合計数に制限され、メモリー使用量が節約されます。

timeout オプションは、指定された秒数の間にだけ要素がセット内に存在することを意味します。以下に示します。

```
~]# ipset add my-busy-addresses 192.168.60.0 timeout 100
```

以下の出力は時間のカウントダウンを示しています。

```
~]# ipset list my-busy-addresses
Name: my-busy-addresses
Type: hash:ip
Header: family inet hashsize 1024 maxelem 24 netmask 28 timeout 100
Size in memory: 8300
References: 0
Members:
192.168.60.0 timeout 90
~]# ipset list my-busy-addresses
Name: my-busy-addresses
Type: hash:ip
Header: family inet hashsize 1024 maxelem 24 netmask 28 timeout 100
Size in memory: 8300
References: 0
Members:
192.168.60.0 timeout 83
```

タイムアウト期間が終了すると、要素がセットから削除されます。

他の例については、**ipset(8)** man ページを参照してください。

4.5.5. その他のリソース

以下の情報ソースでは、**firewalld** に関する追加リソースが提供されています。

4.5.5.1. インストールされているドキュメント

- **firewalld(1)** man ページ — **firewalld** のコマンドオプションについて説明しています。
- **firewalld.conf(5)** man ページ — **firewalld** の設定に関する情報が含まれています。
- **firewall-cmd(1)** man ページ — **firewalld** コマンドラインクライアントのコマンドオプションについて説明しています。
- **firewalld.icmptype(5)** man ページ — **ICMP** フィルタリングの XML 設定ファイルについて説明しています。
- **firewalld.service(5)** man ページ — **firewalld service** の XML 設定ファイルについて説明しています。
- **firewalld.zone(5)** man ページ — **firewalld** ゾーン設定の XML 設定ファイルについて説明しています。
- **firewalld.direct(5)** man ページ — **firewalld** ダイレクトインターフェース設定ファイルについて説明しています。
- **firewalld.lockdown-whitelist(5)** man ページ — **firewalld** ロックダウンホワイトリストの設定ファイルについて説明しています。
- **firewall.richlanguage(5)** man ページ — **firewalld** リッチ言語ルール of 構造について説明しています。
- **firewalld.zones(5)** man ページ — ゾーンについての全般的な説明とそれらの設定方法が説明されています。

4.6. DNSSEC を使った DNS トラフィックのセキュア化

4.6.1. DNSSEC の概要

DNSSEC は、*Domain Name System Security Extensions* (DNSSEC: ドメイン名システムセキュリティ拡張機能) のセットです。DNS クライアントがDNS ネームサーバーからの応答の整合性を認証、検査して、応答元を検証し、転送中に不正に変更されなかったかどうかを判別できるようにします。

4.6.2. DNSSEC について

今日では、インターネット接続で多くのウェブサイトが **HTTPS** を使った安全な接続機能を提供しています。しかし、IP アドレスを直接入力していなければ、**HTTPS** ウェブサーバーに接続する前に **DNS** ルックアップが実行される必要があります。この **DNS** ルックアップは安全に実行されず、認証がないことから *中間者攻撃* の対象となります。つまり、ある **DNS** ネームサーバーから送信されたようにみえる応答が本物で、不正に変更されていないことに **DNS** クライアントは確信を持ってません。さらに重要なのは、再帰問い合わせネームサーバーは、他のネームサーバーから得られる記録が本物かどうかを判断できません。**DNS** プロトコルは、クライアントが中間者攻撃の対象とならないようなメカニズムを提供していませんでした。DNSSEC は、**DNS** を使用したドメインネームを解決する際の認証および完全性のチェックの欠如に対処するために導入されました。DNSSEC は、機密性の問題には対処しません。

DNSSEC 情報の公開では、**DNS** リソース記録のデジタル署名と、**DNS** リゾルバーが信頼の階層チェーンを構築できる方法での公開鍵の配布が行われます。すべての **DNS** リソースレコードのデジタル署名は、デジタル署名のリソース記録 (RRSIG) として生成され、ゾーンに追加されます。ゾーンの公開鍵は、DNSKEY リソースレコードとして追加されます。階層チェーンを構築するために、DNSKEY のハッシュが *Delegation of Signing* (DS) リソースレコードとして親ゾーンに公開されます。存在しない証拠を活用するため、*NextSECure* (NSEC) および NSEC3 リソースレコードが使用されます。DNSSEC 署名ゾーンでは、各 リソースレコードセット (RRset) に対応する RRSIG リソースレコードがあります。子ゾーンへの委任用で使用されるレコード (NS および glue レコード) には署名されないことに注意してください。それらのレコードは子ゾーンに表示され、そこで署名されます。

DNSSEC 情報の処理は、root ゾーンの公開鍵で設定されたリゾルバーが行います。この鍵を使ってリゾルバーは root ゾーンで使用された署名を検証することができます。たとえば、root ゾーンが **.com** の DS レコードを署名しているとします。root ゾーンは **.com** ネームサーバーの NS および glue レコードも処理します。リゾルバーはこの委任を追いかけ、これらの委任されたネームサーバーを使用して **.com** の DNSKEY レコードを問い合わせます。獲得された DNSKEY レコードのハッシュは、root ゾーンの DS レコードとマッチするはずで、マッチする場合、リゾルバーは獲得された **.com** の DNSKEY を信頼します。**.com** ゾーンでは、RRSIG レコードは **.com** DNSKEY が作成します。このプロセスは、**redhat.com** などの **.com** 内の委任で同様に繰り返されます。この方法を使用することで、検証を行う **DNS** リゾルバーは 1 つの root 鍵での設定のみが必要で、多くの DNSKEY を通常の操作中に世界中から収集します。暗号検査が失敗すると、リゾルバーはアプリケーションに **SERVFAIL** を返します。

DNSSEC は、DNSSEC をサポートしていないアプリケーションには全く見えないように設計されています。DNSSEC 非対応のアプリケーションが DNSSEC 対応リゾルバーを問い合わせると、RRSIG のような新たなリソースレコードタイプのない応答を受け取ります。ただし、DNSSEC 対応リゾルバーはすべての暗号検査を実行し、悪意のある **DNS** 応答を検出するとアプリケーションに **SERVFAIL** エラーを返します。DNSSEC は、**DNS** サーバー間 (権威と再帰) でデータの整合性を保護しますが、アプリケーションとリゾルバー間のセキュリティは提供しません。このため、アプリケーションにリゾルバーへの安全なトランスポートが提供されることが重要になります。一番簡単な方法は、DNSSEC 対応リゾルバーを **localhost** 上で稼働して **/etc/resolv.conf** 内の **127.0.0.1** を使用することで。代わりに、リモートの **DNS** サーバーに VPN 接続することもできます。

ホットスポットの問題について

Wi-Fi ホットスポットまたは VPN は「DNS 偽装」に依存: キャプティブポータルは、ユーザーを Wi-Fi サービス認証 (または支払い) する必要があるページにリダイレクトするために、**DNS** を乗っ取る傾向があります。VPN に接続するユーザーは、企業ネットワークの外に存在しないリソースを見つけるために、「内部のみ」の **DNS** サーバーを使用することがよくあります。これには、ソフトウェアによる追加処理が必要になります。たとえば、**dnssec-trigger** を使って、ホットスポットが **DNS** クエリーを乗っ取っているかを検出し、**unbound** は プロキシネームサーバーとして DNSSEC クエリーを処理するように動作することが可能です。

DNSSEC 対応のリカーシブリゾルバー

DNSSEC 対応のリカーシブリゾルバーを実装するには、**BIND** または **unbound** を使用することができます。両方ともデフォルトで DNSSEC を有効にし、DNSSEC root キーで設定されています。サーバー上で DNSSEC を有効にするにはどちらでも可能ですが、ノートパソコンなどのモバイルデバイスでは、**unbound** が推奨されます。これは、**dnssec-trigger** 使用時にはホットスポット、**Libreswan** 使用時には VPN に必要となる DNSSEC 上書きの再構成をローカルユーザーが動的に行えるようになるためです。**unbound** デーモンはさらに、**etc/unbound/*.d/** ディレクトリーにリスト化されている DNSSEC 例外の実装もサポートします。これは、サーバーとモバイルデバイスの両方で役立ちます。

4.6.3. Dnssec-trigger について

unbound が **/etc/resolv.conf** にインストールされ設定ができれば、アプリケーションからのすべての **DNS** クエリーは **unbound** によって処理されます。**dnssec-trigger** が **unbound** リゾルバーを再構成するのは、そうするようにトリガーされた場合のみです。これは主に、異なる Wi-Fi ネットワークに接続するノートパソコンのような、ローミングを行うクライアントマシンに当てはまります。プロセスは以下のようになります。

- 新規 **DNS** サーバーが **DHCP** 経由で獲得されると、**NetworkManager** が **dnssec-trigger** を「トリガー (開始)」します。
- **Dnssec-trigger** がサーバーに対して多くのテストを実行し、サーバーが DNSSEC を適切にサポートしているかどうかを判断します。
- サポートしている場合は、**dnssec-trigger** が **unbound** を再構成して、その **DNS** サーバーをすべてのクエリーに対するフォワーダーとして使用します。
- テストが失敗した場合は、**dnssec-trigger** は新規 **DNS** サーバーを無視して利用可能なフォールバックの方法をいくつか試行します。
- 無制限ポート 53 (**UDP** および **TCP**) が利用可能だと判断されたら、フォワーダーを使用せずに **unbound** が完全なリカーシブ **DNS** サーバーになるように指示します。
- たとえば、ポート 53 がネットワークの **DNS** サーバー自体以外にはファイアウォールでブロックされているなどしてこれが可能でない場合は、ポート 80 へは **DNS** を、ポート 443 へは **TLS** カプセル化された **DNS** の使用を試みます。ポート 80 および 443 で **DNS** を稼働しているサーバーは、**/etc/dnssec-trigger/dnssec-trigger.conf** で設定できます。デフォルトの設定ファイルでは、コメントアウトされた例が利用可能になっています。
- これらのフォールバックの方法も失敗する場合は、**dnssec-trigger** は DNSSEC を完全に迂回してセキュアでない状態で機能するか、「cache only」モードで実行します。後者の場合、新規の **DNS** クエリーは試行されませんが、キャッシュにすでにあるものについてはすべて応答します。

Wi-Fi ホットスポットでは、ユーザーはインターネットへのアクセスを許可される前にサインオンページにリダイレクトされる傾向が強まっています。上記で示された調査手順の間にリダイレクトが検出されると、ユーザーはインターネットアクセスにログインが必要かどうかを尋ねるようにプロンプトが示

されます。**dnssec-trigger** デーモンは 10 分ごとに DNSSEC リゾルバーをプローブし続けます。**dnssec-trigger** グラフィカルユーティリティーの使用に関する情報は、「[Dnssec-trigger の使用](#)」を参照してください。

4.6.4. VPN が提供されるドメインサーバーおよびネームサーバー

VPN 接続の中には、ドメインとネームサーバー一覧を伝達して、そのドメインを VPN トンネル設定の一部として使用できるタイプもあります。**Red Hat Enterprise Linux** では、**NetworkManager** がこれをサポートしています。つまり、**unbound**、**dnssec-trigger**、および **NetworkManager** の組み合わせは、VPN ソフトウェアが提供するドメインとネームサーバーを適切にサポートできるということです。VPN トンネルができれば、ローカルの **unbound** キャッシュは、受け取られたドメインネームのエントリすべてについてフラッシュされるので、そのドメインネーム内のネームに関するクエリーは VPN 経由で届いた内部ネームサーバーから新たにフェッチされます。VPN トンネルが終了すると、**unbound** キャッシュが再度フラッシュされ、ドメインへのクエリーが以前に獲得されたプライベート IP アドレスではなく、公開 IP アドレスを返すようにします。詳細は、「[接続によって提供されるドメイン用の DNSSEC 検証の設定](#)」を参照してください。

4.6.5. 推奨される命名プラクティス

Red Hat では、静的および一時的なネームの両方が **host.example.com** のように DNS 内のマシンに使用されている **完全修飾ドメインネーム (FQDN)** に合致することを推奨しています。

ICANN (The Internet Corporation for Assigned Names and Numbers) は、**(.yourcompany** などの) トップレベルの未登録ドメインを公開登録簿に追加することがあります。このため、Red Hat では、プライベートネットワーク上であっても委任されていないドメイン名を使用しないことを強く推奨しています。これは、ネットワーク設定によっては異なる解決をしてしまうドメインネームになってしまう可能性があるからです。その結果、ネットワークリソースは利用不可能になってしまいます。また、委任されていないドメイン名を使うと、DNSSEC の実装および維持がより困難になります。これは、ドメインネームが競合すると DNSSEC 検証に手動の設定が必要となるからです。この問題に関する詳細情報は、[ICANN FAQ on domain name collision](#)を参照してください。

4.6.6. トラストアンカーについて

階層暗号化システムでは、トラストアンカーは信頼できるとされる権威あるエンティティです。たとえば、X.509 アーキテクチャーではルート証明書はトラストチェーンの元となるトラストアンカーとなっています。トラストアンカーは、パスの検証ができるように事前に信頼できる団体が所有しておく必要があります。

DNSSEC のコンテキストでは、トラストアンカーは **DNS** ネームとそのネームに関連づけられた公開鍵 (または公開鍵のキャッシュ) で構成されます。ベース 64 のエンコード化された鍵として表示されます。これは、**DNS** レコードの検証と認証に使用可能な公開鍵を含む情報の交換方法という意味で、証明書と同様のものになります。[RFC 4033](#) では、設定済みの **DNSKEY RR** または **DNSKEY RR** の **DS RR** ハッシュとしてトラストアンカーを定義しています。有効なセキュリティ対応のリゾルバーは、この公開鍵またはハッシュを、署名済みの **DNS** の応答に対して認証チェーンを構築するための開始地点として使用します。一般的に、有効なリゾルバーはトラストアンカーの初期値をセキュアまたは信頼できる手段で **DNS** プロトコルの外部から取得する必要があります。トラストアンカーがあると、リゾルバーは、トラストアンカーがポイントするゾーンは署名されているはずとの扱いをすると意味合いがあります。

4.6.7. DNSSEC のインストール

4.6.7.1. unbound のインストール

マシン上でローカルに DNSSEC を使用する **DNS** を検証するためには、**DNS** リゾルバー **unbound** (または **bind**) をインストールする必要があります。モバイルデバイスでは、**dnssec-trigger** のインス

ツールのみが必要です。サーバーには **unbound** で十分ですが、サーバーの場所 (LAN もしくはインターネット) によってはローカルドメイン用の転送設定が必要となる場合もあります。**dnssec-trigger** は現在、グローバルの公開 DNS ゾーンのみで機能します。**NetworkManager**、**dhclient**、および VPN アプリケーションは自動でドメイン一覧 (およびネームサーバー一覧も) を収集することが多くありますが、**dnssec-trigger** や **unbound** ではこれは行われません。

unbound をインストールするには、**root** ユーザーで以下のコマンドを実行します。

```
~]# yum install unbound
```

4.6.7.2. unbound の稼働確認

unbound デーモンが稼働しているかどうかを確認するには、以下のコマンドを実行します。

```
~]$ systemctl status unbound
unbound.service - Unbound recursive Domain Name Server
   Loaded: loaded (/usr/lib/systemd/system/unbound.service; disabled)
   Active: active (running) since Wed 2013-03-13 01:19:30 CET; 6h ago
```

unbound サービスが稼働していないと **systemctl status** コマンドは **unbound** を **Active: inactive (dead)** とレポートします。

4.6.7.3. unbound の起動

現行セッション用に **unbound** を起動するには、**root** ユーザーで以下のコマンドを実行します。

```
~]# systemctl start unbound
```

systemctl enable コマンドを実行すると **unbound** がシステム起動時に毎回起動します。

```
~]# systemctl enable unbound
```

unbound デーモンは、以下のディレクトリーを使用してローカルデータの設定または上書きを可能にします。

- **/etc/unbound/conf.d** ディレクトリーは、特定のドメインネーム用の設定追加に使用されます。ドメインネームのクエリーを特定の **DNS** サーバーにリダイレクトするために使用されません。企業 WAN 内にのみ存在するサブドメインに多く使用されます。
- **/etc/unbound/keys.d** ディレクトリーは、特定のドメインネーム用のトラストアンカーの追加に使用されます。これは、内部のみのネームが **DNSSEC** 署名されているものの、トラストのパスを構築するための **DS** レコードが公開で存在しない場合に必要になります。別のユースケースは、企業 WAN 外で公開で入手可能なネームではない別の **DNSKEY** を使って署名された、内部バージョンのドメインの場合になります。
- **/etc/unbound/local.d** ディレクトリーは、ローカルで優先させる特定の **DNS** データを追加するために使用されます。このデータを使用すると、ブラックリストを構築したり、手動で特定の **DNS** を優先させたりできます。このデータは **unbound** によってクライアントに返されますが、**DNSSEC** 署名としてマークされません。

一部の VPN ソフトウェアと同様に **NetworkManager** では設定を動的に変更することが可能です。これらの設定ディレクトリーには、コメントアウトしたサンプルエントリーが含まれています。詳細については **unbound.conf(5)** man ページを参照してください。

4.6.7.4. Dnssec-trigger のインストール

dnssec-trigger アプリケーションは **dnssec-triggerd** デーモンとして稼働します。**dnssec-trigger** をインストールするには、**root** ユーザーで以下のコマンドを実行します。

```
~]# yum install dnssec-trigger
```

4.6.7.5. Dnssec-trigger デーモンの稼働確認

dnssec-triggerd が稼働しているかどうかを確認するには、以下のコマンドを実行します。

```
~]$ systemctl status dnssec-triggerd
systemctl status dnssec-triggerd.service
dnssec-triggerd.service - Reconfigure local DNS(SEC) resolver on network
change
    Loaded: loaded (/usr/lib/systemd/system/dnssec-triggerd.service;
    enabled)
    Active: active (running) since Wed 2013-03-13 06:10:44 CET; 1h 41min
    ago
```

systemctl status コマンドは、**dnssec-triggerd** デーモンが稼働していなければ **Active: inactive (dead)** とレポートします。**dnssec-triggerd** を現行セッションで起動するには、**root** ユーザーで以下のコマンドを実行します。

```
~]# systemctl start dnssec-triggerd
```

systemctl enable コマンドを実行すると **dnssec-triggerd** がシステム起動時に毎回起動します。

```
~]# systemctl enable dnssec-triggerd
```

4.6.8. Dnssec-trigger の使用

dnssec-trigger アプリケーションには、DNSSEC プローブの結果を表示し、DNSSEC プローブ要求をオンデマンドで実行するための GNOME パネルユーティリティがあります。このユーティリティを起動するには、**Super** キーを押してアクティビティ画面に入り、**DNSSEC** と入力して **Enter** を押します。船の錨に似たアイコンが画面下部のメッセージトレイに追加されます。画面右下の青く丸い通知アイコンを押してこれを表示させます。錨アイコンを右クリックすると、ポップアップメニューが表示されます。

通常の操作では、**unbound** はローカルでネームサーバーとして使用され、**resolv.conf** は **127.0.0.1** をポイントします。**Hotspot Sign-On** パネルの **OK** をクリックすると、これが変更されます。**DNS** サーバーが **NetworkManager** からクエリーを受け、**resolv.conf** に置かれます。これで **Hotspot** のサインオンページで認証が可能になります。錨アイコンは赤い大きな感嘆符を表示して、**DNS** クエリーが安全でない状態で行われていることを警告します。認証が行われると、**dnssec-trigger** は自動的にこれを検出し、セキュアモードに戻します。ただし、場合によってはこれができず、ユーザーが手動で **Reprobe** を選択する必要があることもあります。

Dnssec-trigger は通常、ユーザーとのやりとりを必要としません。一旦開始するとバックグラウンドで稼働し、問題が発生したらポップアップのテキストボックスでユーザーに通知します。また、**resolv.conf** ファイルへの変更を **unbound** に知らせます。

4.6.9. DNSSEC における dig の使用

DNSSEC が稼働しているかどうかを確認するには、様々なコマンドラインツールの使用が可能です。最適なのは、bind-utils パッケージからの **dig** コマンドです。ldns パッケージからの **drill** と unbound パッケージからの **unbound-host** も有用です。**nslookup** および **host** という古い DNS ユーティリティーは旧式なので使用しないでください。

dig を使用して DNSSEC データを要求するクエリーを送信するには、以下のように **+dnssec** オプションをコマンドに追加します。

```
~]$ dig +dnssec whitehouse.gov
; <<>> DiG 9.9.3-rl.13207.22-P2-RedHat-9.9.3-4.P2.el7 <<>> +dnssec
whitehouse.gov
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21388
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;whitehouse.gov.      IN A

;; ANSWER SECTION:
whitehouse.gov.  20 IN A 72.246.36.110
whitehouse.gov.  20 IN RRSIG A 7 2 20 20130825124016 20130822114016 8399
whitehouse.gov.  BB8VHWEkIaKpaLprt3hq1GkjDR0vkmjYTBxiGhuki/BJn3PoIGyrftxR
HH0377I0Lsybj/uZv5hL4UwWd/lw6Gn8GPikqhztAkqMxddMQ2IARP6p
wbMOKbSUuV6NGUT1WwWpbi+LelFMqQcAq3Se66iyH0Jem7HtgPEUE1Zc 3oI=

;; Query time: 227 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Aug 22 22:01:52 EDT 2013
;; MSG SIZE rcvd: 233
```

A レコードに加えて RRSIG レコードが返されます。これには DNSSEC 署名と署名の取得および失効時間が含まれています。**unbound** サーバーは、上部の **flags:** セクションで **ad** を返して、データが DNSSEC 認証されていることを示しています。

DNSSEC 検証が失敗すると、**dig** は SERVFAIL エラーを返します。

```
~]$ dig badsign-a.test.dnssec-tools.org
; <<>> DiG 9.9.3-rl.156.01-P1-RedHat-9.9.3-3.P1.el7 <<>> badsign-
a.test.dnssec-tools.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 1010
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:;; udp: 4096
;; QUESTION SECTION:
;badsign-a.test.dnssec-tools.org. IN A

;; Query time: 1284 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Aug 22 22:04:52 EDT 2013
;; MSG SIZE rcvd: 60]
```

失敗に関する詳細情報を要求するには、**dig** コマンドに**+cd** オプションを指定して DNSSEC 検査を無効にすることができます。

```
~]$ dig +cd +dnssec badsign-a.test.dnssec-tools.org
; <<>> DiG 9.9.3-rl.156.01-P1-RedHat-9.9.3-3.P1.el7 <<>> +cd +dnssec
badsign-a.test.dnssec-tools.org
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 26065
;; flags: qr rd ra cd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;badsign-a.test.dnssec-tools.org. IN A

;; ANSWER SECTION:
badsign-a.test.dnssec-tools.org. 49 IN A 75.119.216.33
badsign-a.test.dnssec-tools.org. 49 IN RRSIG A 5 4 86400 20130919183720
20130820173720 19442 test.dnssec-tools.org.
E572dLKMvYB4cgTRyAHIKKEvd0P7tockQb7hXFNZKVbfXbZJ0IDREJrr
zCgAfJ2hykfY0yJHAlnuQvM0s6x0nNBSvc2xLIybJdfTaN6kSR0YFdYZ
n2NpPctn2kUBn5UR1BJRin3Gqy20LZlZx2KD7cZBtieMsU/IunyhCSc0 kYw=

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Aug 22 22:06:31 EDT 2013
;; MSG SIZE rcvd: 257
```

DNSSEC は、間違った取得または失効時間によってマニフェスト自体を誤解することが多くあります。ただし上記の例では、www.dnssec-tools.org のユーザーはわざとこの RRSIG 署名を分割しており、この出力を見るだけでは検出できません。このエラーは **systemctl status unbound** の出力で表示され、**unbound** デーモンがこれらのエラーを**syslog**に以下のように記録します。

```
Aug 22 22:04:52 laptop unbound: [3065:0] info: validation failure badsign-
a.test.dnssec-tools.org. A IN
```

以下は、**unbound-host** を使用した例です。

```
~]$ unbound-host -C /etc/unbound/unbound.conf -v whitehouse.gov
whitehouse.gov has address 184.25.196.110 (secure)
whitehouse.gov has IPv6 address 2600:1417:11:2:8800::fc4 (secure)
whitehouse.gov has IPv6 address 2600:1417:11:2:8000::fc4 (secure)
whitehouse.gov mail is handled by 105 mail1.eop.gov. (secure)
whitehouse.gov mail is handled by 110 mail5.eop.gov. (secure)
whitehouse.gov mail is handled by 105 mail4.eop.gov. (secure)
whitehouse.gov mail is handled by 110 mail6.eop.gov. (secure)
whitehouse.gov mail is handled by 105 mail2.eop.gov. (secure)
whitehouse.gov mail is handled by 105 mail3.eop.gov. (secure)
```

4.6.10. Dnssec-trigger 用のホットスポット検出インフラストラクチャーの設定

ネットワークに接続する際に、**dnssec-trigger** はホットスポットの検出を試みます。ホットスポットは通常、ユーザーがネットワークリソースを利用する前にウェブページとのやりとりを強制するデバイスです。この検出は既知のコンテンツがある特定の固定ウェブページのダウンロード試行で行われま

す。ホットスポットがある場合は、既知のコンテンツ以外のものがダウンロードされます。

dnssec-trigger によるホットスポット検出に使用可能な、既知のコンテンツがある固定ウェブページを設定するには、以下の手順にしたがいます。

1. インターネット上で公開されているマシン上にウェブサーバーを設定します。Red Hat Enterprise Linux 7 システム管理者のガイドの [Web サーバー](#) の章を参照してください。
2. サーバーが稼働したら、既知のコンテンツがある静的ページを公開します。このページは、有効な HTML ページである必要はありません。たとえば、**hotspot.txt** という名前のプレーンテキストファイルで **OK** という文字列の内容だけでも構いません。サーバーが **example.com** にあり、そのサーバーのサブディレクトリー **document_root/static/** に **hotspot.txt** ファイルを公開したとすると、この静的ページのアドレスは、**example.com/static/hotspot.txt** になります。Red Hat Enterprise Linux 7 システム管理者のガイドの [Web サーバー](#) の章にある **DocumentRoot** ディレクティブを参照してください。
3. 以下の行を **/etc/dnssec-trigger/dnssec-trigger.conf** ファイルに追加します。

```
url: "http://example.com/static/hotspot.txt OK"
```

このコマンドは、**HTTP** (ポート 80) 経由でプローブされる URL を追加します。最初の部分は解決される URL とダウンロードされるページで、後の部分はダウンロードしたウェブページに含まれることになる文字列です。

設定オプションに関する詳細情報は、man ページ **dnssec-trigger.conf(8)** を参照してください。

4.6.11. 接続によって提供されるドメイン用の DNSSEC 検証の設定

デフォルトでは、適切なネームサーバーのある転送ゾーンは、Wi-Fi 以外の接続が提供するドメインすべてについて **dnssec-trigger** が自動的に **NetworkManager** で **unbound** に追加します。デフォルトでは、**unbound** に追加されるすべての転送ゾーンは DNSSEC で検証されます。

転送ゾーンの検証におけるデフォルト動作は変更可能なので、すべての転送ゾーンがデフォルトで DNSSEC 検証されるわけは **ありません**。すべてを検証するようにするには、**dnssec-trigger** 設定ファイルである **/etc/dnssec.conf** にある **validate_connection_provided_zones** 変数を変更します。**root** ユーザーでファイルを開き、以下のように行を編集します。

```
validate_connection_provided_zones=no
```

この変更は既存の転送ゾーンではなく、将来の転送ゾーンにのみ有効になります。このため、現在提供されているドメインについて DNSSEC を無効にするには、再接続が必要になります。

4.6.11.1. Wi-Fi が提供するドメイン用の DNSSEC 検証の設定

Wi-Fi が提供するゾーンへの転送ゾーンの追加を有効にすることができます。これを行うには、**dnssec-trigger** の設定ファイル **/etc/dnssec.conf** にある **add_wifi_provided_zones** 変数を変更します。**root** ユーザーでファイルを開き、以下のように行を編集します。

```
add_wifi_provided_zones=yes
```

この変更は既存の転送ゾーンではなく、将来の転送ゾーンにのみ有効になります。このため、現在 Wi-Fi が提供するドメインについて DNSSEC を有効にするには、Wi-Fi の再接続 (再起動) が必要になります。



警告

Wi-Fi 提供ドメインを転送ゾーンとして **unbound** に追加することをオンにすると、セキュリティ面で以下のような影響があります。

1. Wi-Fi アクセスポイントは意図的に、**DHCP** 経由でドメインを提供することが可能です。このドメインは認証されていないもので、ユーザーの **DNS** クエリーすべてをこのアクセスポイントの **DNS** サーバーにルーティングしてしまいます。
2. 転送ゾーンの **DNSSEC** 検証を **オフ** にすると、Wi-Fi 提供の **DNS** サーバーは、ユーザーが知らない間に提供されたドメインからドメインネームの **IP** アドレスのなりすましができるようになります。

4.6.12. その他のリソース

以下のリソースでは、DNSSEC について詳細な説明が提供されています。

4.6.12.1. インストールされているドキュメント

- **dnssec-trigger(8)** man ページ — **dnssec-triggerd**、**dnssec-trigger-control** および **dnssec-trigger-panel** のコマンドオプションについて説明しています。
- **dnssec-trigger.conf(8)** man ページ — **dnssec-triggerd** の設定オプションについて説明しています。
- **unbound(8)** man ページ — **unbound**、**DNS** 検証リゾルバーのコマンドオプションについて説明しています。
- **unbound.conf(5)** man ページ — **unbound** の設定方法に関する情報が含まれています。
- **resolv.conf(5)** man ページ — リゾルバールーチンが読み取る情報が含まれています。

4.6.12.2. オンラインのドキュメント

<http://tools.ietf.org/html/rfc4033>

RFC 4033 DNS Security Introduction and Requirements.

<http://www.dnssec.net/>

DNSSEC リソースに関する多くのリンクがあるウェブサイトです。

<http://www.dnssec-deployment.org/>

米国土安全保障省が後援する DNSSEC Deployment Initiative です。DNSSEC に関する多くの情報と DNSSEC 導入に関する問題を話し合うメーリングリストが含まれています。

<http://www.internetsociety.org/deploy360/dnssec/community/>

Internet Society の「Deploy 360」イニシアチブで、DNSSEC 導入の活性化と調整を図ります。世界中のコミュニティと DNSSEC アクティビティを探る便利なリソースになります。

<http://www.unbound.net/>

unbound DNS サービスに関する全般的な情報が含まれています。

<http://www.nlnetlabs.nl/projects/dnssec-trigger/>

dnssec-trigger に関する全般的な情報が含まれています。

4.7. 仮想プライベートネットワーク (VPN) のセキュア化

Red Hat Enterprise Linux 7 では、仮想プライベートネットワーク (VPN) は、**Libreswan** アプリケーションがサポートしている **IPsec** トンネリングプロトコルを使って設定できます。**Libreswan** は **Openswan** アプリケーションの分岐で、ドキュメント内の例は交換可能なものです。**NetworkManager IPsec** プラグインは、**NetworkManager-libreswan** と呼ばれます。GNOME Shell のユーザーは、**NetworkManager-libreswan-gnome** パッケージをインストールしてください。これには、依存関係として **NetworkManager-libreswan** が含まれています。**NetworkManager-libreswan-gnome** は、Optional チャンネルからのみ入手可能です。[Enabling Supplementary and Optional Repositories](#) を参照してください。

Libreswan は、Red Hat Enterprise Linux 7 で利用可能なオープンソースのユーザースペース **IPsec** 実装です。インターネット鍵交換 (IKE) プロトコルを使用します。**IKE** バージョン 1 および 2 は、ユーザーレベルのデーモンとして実装されます。**ip xfrm** コマンドを使って、手動で鍵を確立することも可能ですが、これは推奨されません。**Libreswan** は、**netlink** を使って Linux カーネルとインターフェース接続し、暗号化鍵を送信します。パケットの暗号化と暗号解読は、Linux カーネルで行われます。

Libreswan は ネットワークセキュリティサービス (NSS) 暗号化ライブラリーを使用します。これは、*Federal Information Processing Standard: 米連邦情報処理規格 (FIPS)* のセキュリティ準拠に必要なものです。



重要

Libreswan が実施する **IPsec** は、Red Hat Enterprise Linux 7 での使用が推奨される唯一の VPN 技術です。他の VPN は、そのリスクを理解することなく使用しないでください。

4.7.1. Libreswan を使った IPsec VPN

Libreswan をインストールするには、以下のコマンドを **root** で実行します。

```
~]# yum install libreswan
```

Libreswan がインストールされているかどうかを確認するには、以下のコマンドを実行します。

```
~]$ yum info libreswan
```

Libreswan を新規インストールすると、NSS データベースがインストールプロセスの一部として初期化されます。ただし、新規データベースを起動する必要がある場合は、以下のようにまず、古いデータベースを削除してください。

```
~]# rm /etc/ipsec.d/*db
```

その後に以下のコマンドを **root** で実行して、新規 NSS データベースを初期化します。

■


```
~]# ipsec initnss
Initializing NSS database
See 'man pluto' if you want to protect the NSS database with a password
```

NSS にパスワードを使用しない場合は、パスワードが要求されたときに **Enter** を 2 回押します。パスワードを入力した場合は、システム起動時など、**Libreswan** の起動ごとに毎回パスワードの再入力が必要になります。

Libreswan が提供する **ipsec** デーモンを起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start ipsec
```

デーモンが稼働していることを確認します。

```
~]$ systemctl status ipsec
ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; disabled)
   Active: active (running) since Wed 2013-08-21 12:14:12 CEST; 18s ago
```

システム起動時に **Libreswan** が起動するようにするには、**root** で以下のコマンドを実行します。

```
~]# systemctl enable ipsec
```

中間およびホストベースのファイアウォールが **ipsec** サービスを許可するように設定します。ファイアウォールおよび特定サービスの通過を許可することについての詳細情報は、「[ファイアウォールの使用](#)」を参照してください。**Libreswan** の使用には、以下のパケットがファイアウォールを通過できるようにしておく必要があります。

- **Internet Key Exchange (IKE)** プロトコル用に **UDP** ポート 500
- **IKE NAT-Traversal** 用に **UDP** ポート 4500
- **Encapsulated Security Payload (ESP) IPsec** パケット用に プロトコル 50
- **Authenticated Header (AH) IPsec** パケット用にプロトコル 51 (一般的でない)

Libreswan を使って **IPsec VPN** を設定する例を 3 つ紹介します。ひとつ目は、2 つのホストを接続してセキュアな通信ができるようにします。2 つ目は、2 つのサイトを接続して 1 つのネットワークを形成します。3 つ目は、このコンテキストでは **ロードウォリアー** と呼ばれるローミングユーザーをサポートします。

4.7.2. Libreswan を使った VPN 設定

Libreswan では、「ソース」や「宛先」といった用語を使用しません。代わりに、エンドポイント(ホスト)に向かって「左」および「右」という用語を使用します。これにより、ほとんどのケースで両方のエンドポイントで同じ設定が使えるようになります。ただし、ほとんどの管理者はローカルホストに「左」を、リモートホストに「右」を使用します。

エンドポイントの認証には、3 つの一般的な方法が使用されます。

- **事前共有鍵 (PSK)** は、最もシンプルな認証方法です。PSK は任意の 20 以上の文字で構成されます。PSK が任意でなくかつ短いものになる危険があることから、システムが **FIPS** モードで稼働している際は、この方法は利用できません。

- 生 RSA 鍵は、静的なホスト間またはサブネット間で一般的に使用される **IPsec** 設定です。ホストは、それぞれの公開 RSA 鍵で手動で設定されます。この方法は、12 以上のホストで相互に **IPsec** トンネルを設定する必要がある場合には、うまく拡張できません。
- X.509 証明書は、共通の **IPsec** ゲートウェイに接続する必要のあるホストが多くある、大型の導入案件でよく使用されます。ホストまたはユーザーの RSA 証明書の署名には、中央 認証機関 (CA) が使用されます。この中央 CA は、個別ホストまたはユーザーの取り消しを含む信頼の中継を担当します。

4.7.3. Libreswan を使用したホスト間の VPN

Libreswan が「左」および「右」と呼ばれる 2 つのホスト間で **IPsec** VPN を作成するよう設定するには、両方のホスト上 (「left」および「right」) で **root** として以下のコマンドを実行し、新たな生 RSA 鍵のペアを作成します。

```
~]# ipsec newhostkey --configdir /etc/ipsec.d \
--output /etc/ipsec.d/www.example.com.secrets
Generated RSA key pair using the NSS database
```

これでホストの RSA 鍵のペアが生成されます。エントロピーが低い仮想マシンでは特に、RSA 鍵の生成プロセスは時間が長くなる場合があります。

公開鍵を表示するには、どちらかのホスト上で **root** として以下のコマンドを実行します。たとえば、「left」のホストで公開鍵を表示するには以下を実行します。

```
~]# ipsec showhostkey --left
ipsec showhostkey loading secrets from "/etc/ipsec.secrets"
ipsec showhostkey loading secrets from
"/etc/ipsec.d/www.example.com.secrets"
ipsec showhostkey loaded private key for keyid: PPK_RSA:AQ0jAKLlL
# rsakey AQ0jAKLlL
lefttrsasigkey=0sAQ0jAKLlL4a7YBv [...]
```

以下で説明するように、この鍵を設定ファイルに追加する必要があります。

秘密の部分は、「NSS データベース」とも呼ばれる **/etc/ipsec.d/*.db** ファイルに保存されます。

このホスト間のトンネル用に設定ファイルを作成するには、上記の **lefttrsasigkey=** と **righttrsasigkey=** の各行を **/etc/ipsec.d/** ディレクトリー内のカスタム設定ファイルに記載します。**Libreswan** がカスタム設定ファイルを読み取れるようにするには、**root** でエディターを実行し、メイン設定ファイルである **/etc/ipsec.conf** で以下の行の **#** 文字を削除します。これで、以下の行が有効になります。

```
include /etc/ipsec.d/*.conf
```

root でエディターを使用して以下の形式で適切な名前のファイルを作成します。

```
/etc/ipsec.d/my_host-to-host.conf
```

以下のようにファイルを編集します。

```
conn mytunnel
    leftid=@west.example.com
```



```

left=192.1.2.23
lefttrsasigkey=0sAQ0rlo+h0afUZDlCQmXFrje/oZm [...]
W2n417C/4urYHQkCvuIQ==
rightid=@east.example.com
right=192.1.2.45
righttrsasigkey=0sAQ03fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
authby=rsasig
# load and initiate automatically
auto=start

```

左右両方のホストに同一の設定ファイルを使用することができます。ホストは「左」か「右」かを自動検出します。ホストのいずれかがモバイルホストであり、**IP** アドレスが事前に分からない場合は、モバイルホストの **IP** アドレスに **%defaultroute** を使用します。これで動的 **IP** アドレスが自動的に拾われます。モバイルホストからの接続を受け付ける静的ホストでは、**IP** アドレスに **%any** を指定します。

lefttrsasigkey の値が「左」のホストから、**righttrsasigkey** の値が「右」のホストからであることを確認してください。

ipsec を再起動して、変更された設定を読み込みます。

```
~]# systemctl restart ipsec
```

root で以下のコマンドを実行して、**IPsec** トンネルを読み込みます。

```
~]# ipsec auto --add mytunnel
```

トンネルを UP にするには、**root** で以下のコマンドを実行します。

```
~]# ipsec auto --up mytunnel
```

4.7.3.1. Libreswan を使ったホスト間 VPN の検証

IKE 交渉は、**UDP** ポート 500 で行われます。**IPsec** パケットは、**Encapsulated Security Payload (ESP)** パケットとして現れます。VPN 接続が NAT ルーターを通過する必要がある場合は、**ESP** パケットはポート 4500 上の **UDP** パケットにカプセル化されます。

パケットが VPN トンネル経由で送信されていることを確認するには、**root** で以下の形式のコマンドを実行します。

```

~]# tcpdump -n -i interface esp or udp port 500 or udp port 4500
00:32:32.632165 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1a),
length 132
00:32:32.632592 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1a),
length 132
00:32:32.632592 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489,
seq 7, length 64
00:32:33.632221 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1b),
length 132
00:32:33.632731 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1b),
length 132
00:32:33.632731 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489,
seq 8, length 64
00:32:34.632183 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1c),

```

```
length 132
00:32:34.632607 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1c),
length 132
00:32:34.632607 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489,
seq 9, length 64
00:32:35.632233 IP 192.1.2.45 > 192.1.2.23: ESP(spi=0x63ad7e17,seq=0x1d),
length 132
00:32:35.632685 IP 192.1.2.23 > 192.1.2.45: ESP(spi=0x4841b647,seq=0x1d),
length 132
00:32:35.632685 IP 192.0.2.254 > 192.0.1.254: ICMP echo reply, id 2489,
seq 10, length 64
```

ここでの *interface* は、監視するインターフェースになります。**tcpdump** での表示を終了するには、**Ctrl+C** を押します。



注記

tcpdump コマンドと **IPsec** のインタラクションはやや予想外のものになります。見えるのは暗号化された送信パケットのみで、プレーンテキストの送信パケットは見えません。受信パケットは、暗号化および暗号解読された両方を表示します。可能であれば、**tcpdump** コマンドはどちらかのエンドポイント上ではなく、2つのマシン間にあるルーター上で実行してください。

4.7.4. Libreswan を使ったサイト間の VPN

Libreswan が2つのネットワークを結合させるサイト間の**IPsec** VPNを作成するようにするには、エンドポイントとなる2つのホスト間に **IPsec** トンネルを作成します。これらのホストは、1つ以上のサブネットからのトラフィック通過を許可するよう設定します。このため、これらはネットワークのリモート部分にはゲートウェイのように見えます。サイト間 VPN とホスト間 VPN の唯一の違いは、前者では1つ以上のネットワークまたはサブネットを設定ファイルで指定する必要があるという点です。

サイト間の **IPsec** VPN を作成するように **Libreswan** を設定するには、まず「[Libreswan を使用したホスト間の VPN](#)」にあるようにホスト間の**IPsec** VPNを設定し、その設定ファイルを `/etc/ipsec.d/my_site-to-site.conf` などの適切なファイル名にコピーまたは移動します。**root** としてエディターを使用して、カスタム設定ファイルである `/etc/ipsec.d/my_site-to-site.conf` を以下のように編集します。

```
conn mysubnet
    also=mytunnel
    leftsubnet=192.0.1.0/24
    rightsubnet=192.0.2.0/24

conn mysubnet6
    also=mytunnel
    connaddrfamily=ipv6
    leftsubnet=2001:db8:0:1::/64
    rightsubnet=2001:db8:0:2::/64

conn mytunnel
    auto=start
    leftid=@west.example.com
    left=192.1.2.23
    lefttrsasigkey=0sAQ0rlo+h0afUZDlCQmXFrje/oZm [...]
W2n417C/4urYHQkCvuIQ==
```

```

rightid=@east.example.com
right=192.1.2.45
rightrsasigkey=0sAQ03fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
authby=rsasig

```

トンネルを表示するには、**Libreswan** を再起動するか、**root** で以下のコマンドを実行して手動ですべての接続を読み込み、開始します。

```
~]# ipsec auto --add mysubnet
```

```
~]# ipsec auto --add mysubnet6
```

```
~]# ipsec auto --add mytunnel
```

```

~]# ipsec auto --up mysubnet
104 "mysubnet" #1: STATE_MAIN_I1: initiate
003 "mysubnet" #1: received Vendor ID payload [Dead Peer Detection]
003 "mytunnel" #1: received Vendor ID payload [FRAGMENTATION]
106 "mysubnet" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "mysubnet" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "mysubnet" #1: received Vendor ID payload [CAN-IKEv2]
004 "mysubnet" #1: STATE_MAIN_I4: ISAKMP SA established
{auth=OAKLEY_RSA_SIG cipher=aes_128 prf=oakley_sha group=modp2048}
117 "mysubnet" #2: STATE_QUICK_I1: initiate
004 "mysubnet" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel
mode {ESP=>0x9414a615 <0x1a8eb4ef xfrm=AES_128-HMAC_SHA1 NATOA=none
NATD=none DPD=none}

```

```

~]# ipsec auto --up mysubnet6
003 "mytunnel" #1: received Vendor ID payload [FRAGMENTATION]
117 "mysubnet" #2: STATE_QUICK_I1: initiate
004 "mysubnet" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel
mode {ESP=>0x06fe2099 <0x75eaa862 xfrm=AES_128-HMAC_SHA1 NATOA=none
NATD=none DPD=none}

```

```

~]# ipsec auto --up mytunnel
104 "mytunnel" #1: STATE_MAIN_I1: initiate
003 "mytunnel" #1: received Vendor ID payload [Dead Peer Detection]
003 "mytunnel" #1: received Vendor ID payload [FRAGMENTATION]
106 "mytunnel" #1: STATE_MAIN_I2: sent MI2, expecting MR2
108 "mytunnel" #1: STATE_MAIN_I3: sent MI3, expecting MR3
003 "mytunnel" #1: received Vendor ID payload [CAN-IKEv2]
004 "mytunnel" #1: STATE_MAIN_I4: ISAKMP SA established
{auth=OAKLEY_RSA_SIG cipher=aes_128 prf=oakley_sha group=modp2048}
117 "mytunnel" #2: STATE_QUICK_I1: initiate
004 "mytunnel" #2: STATE_QUICK_I2: sent QI2, IPsec SA established tunnel
mode {ESP=>0x9414a615 >0x1a8eb4ef xfrm=AES_128-HMAC_SHA1 NATOA=none
NATD=none DPD=none}

```

4.7.4.1. Libreswan を使ったサイト間 VPN の検証

パケットが VPN トンネル経由で送信されていることを確認する方法は、「[Libreswan を使ったホスト間 VPN の検証](#)」で説明されている手順と同じになります。

4.7.5. Libreswan を使ったサイト間の単一トンネル VPN

サイト間のトンネルを構築する際には、ゲートウェイは公開 **IP** アドレスではなく、内部の **IP** アドレスを使って相互に通信する必要があります。これは、単一トンネルを使用することで実行できます。ホスト名が **west** の左のホストの内部 **IP** アドレスが **192.0.1.254** で、ホスト名が **east** の右のホストの **IP** アドレスが **192.0.2.254** の場合、単一トンネルを使った以下の設定が使用できます。

```
conn mysubnet
    leftid=@west.example.com
    lefttrsasigkey=0sAQ0rlo+h0afUZDlCQmXFrje/oZm [...]
W2n417C/4urYHQkCvuIQ==
    left=192.1.2.23
    leftsourceip=192.0.1.254
    leftsubnet=192.0.1.0/24
    rightid=@east.example.com
    righttrsasigkey=0sAQ03fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
    right=192.1.2.45
    rightsourceip=192.0.2.254
    rightsubnet=192.0.2.0/24
    auto=start
    authby=rsasig
```

4.7.6. Libreswan を使ったサブネット押し出し

IPsec は、ハブおよびスポークのアーキテクチャーで導入されることがよくあります。各リーフノードには、広い範囲の一部である **IP** 範囲があります。各リーフはハブ経由で相互に通信します。これは、**サブネット押し出し**と呼ばれます。下記の例では、ヘッドオフィスで **10.0.0.0/8** で設定し、2つのブランチでは小型の **/24** サブネットを使用します。

ヘッドオフィスでは以下ようになります。

```
conn branch1
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    lefttrsasigkey=0sA[...]
#
    right=5.6.7.8
    rightid=@branch1
    rightsubnet=10.0.1.0/24
    righttrsasigkey=0sAXXXX[...]
#
    auto=start
    authby=rsasigkey

conn branch2
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    lefttrsasigkey=0sA[...]
#
    right=10.11.12.13
    rightid=@branch2
    rightsubnet=10.0.2.0/24
    righttrsasigkey=0sAYYYY[...]
```

```
#
auto=start
authby=rsasigkey
```

「branch1」 オフィスでは、同一の接続を使用します。さらに、パススルー (pass-through) 接続を使用して、ローカル LAN トラフィックをトンネル経由の送信から除外します。

```
conn branch1
    left=1.2.3.4
    leftid=@headoffice
    leftsubnet=0.0.0.0/0
    leftrsasigkey=0sA[...]
#
    right=10.11.12.13
    rightid=@branch2
    rightsubnet=10.0.1.0/24
    rightrsasigkey=0sAYYYY[...]
#
    auto=start
    authby=rsasigkey

conn passthrough
    left=1.2.3.4
    right=0.0.0.0
    leftsubnet=10.0.1.0/24
    rightsubnet=10.0.1.0/24
    authby=never
    type=passthrough
    auto=route
```

4.7.7. Libreswan を使ったロードウォリアーアプリケーション

ロードウォリアーとは、ノート PC などのモバイルクライアントを使用する移動ユーザーのことで、これらのクライアントには動的に IP アドレスが割り当てられます。これは、証明書を使って認証します。

サーバー上では以下の設定になります。

```
conn roadwarriors
    left=1.2.3.4
    # if access to the LAN is given, enable this
    #leftsubnet=10.10.0.0/16
    leftcert=gw.example.com
    leftid=%fromcert
    right=%any
    # trust our own Certificate Agency
    rightca=%same
    # allow clients to be behind a NAT router
    rightsubnet=vhost:%priv,%no
    authby=rsasigkey
    # load connection, don't initiate
    auto=add
    # kill vanished roadwarriors
```

```
dpddelay=30
dpdtimeout=120
dpdaction=%clear
```

ロードウォリアーのデバイスであるモバイルクライアント上では、上記の設定に多少変更を加えて使用します。

```
conn roadwarriors
# pick up our dynamic IP
left=%defaultroute
leftcert=myname.example.com
leftid=%fromcert
# right can also be a DNS hostname
right=1.2.3.4
# if access to the remote LAN is required, enable this
#rightsubnet=10.10.0.0/16
# trust our own Certificate Agency
rightca=%same
authby=rsasigkey
# Initiate connection
auto=start
```

4.7.8. Libreswan および X.509 証明書による XAUTH を使ったロードウォリアーアプリケーション

Libreswan は、接続確立の際に **XAUTH IPsec** 拡張機能を使って、ローミング VPN クライアントに対してネイティブに **IP** アドレスと **DNS** 情報を割り当てる方法を提供します。**XAUTH** は、**PSK** または **X.509** 証明書を使って導入することができます。**X.509** を使った導入の方がより安全です。クライアントの証明書は、証明書失効リストまたは *Online Certificate Status Protocol (OCSP)* で失効させることができます。**X.509** 証明書を使うと、個別のクライアントはサーバーを偽装することができません。グループパスワードとも呼ばれる **PSK** を使うと、これは理論上は可能になります。

XAUTH は、それ自体とユーザー名およびパスワードを新たに確認するために、VPN クライアントを必要とします。**Google** 認証システムや **RSA SecureID** トークンなどのワンタイムパスワード (**OTP**) では、ユーザーパスワードにワンタイムトークンが付けられます。

XAUTH では、以下の 3 つのバックエンドが考えられます。

xauthby=pam

これは、**/etc/pam.d/pluto** にある設定を使用してユーザーを認証します。**Pam** は、それ自体で様々なバックエンドを使用するように設定できます。システムアカウントのユーザーパスワードスキームや **LDAP** ディレクトリー、**RADIUS** サーバー、カスタムパスワード認証モジュールなどが使用可能です。

xauthby=file

これは、設定ファイル **/etc/ipsec.d/passwd** (**/etc/ipsec.d/nsspassword** と混同しないこと) を使用します。このファイルの形式は **Apache httpasswd** ファイルと同様のものです。**Apache httpasswd** コマンドはこのファイルのエントリー作成に使用できます。ただし、ユーザー名とパスワードの後に、使用する **IPsec** 接続の接続名が 3 番目のコラムに必要になります。たとえば、**conn remoteusers** を使用してリモートユーザーに VPN を提供する場合、パスワードファイルのエントリーは以下のようになります。

```
user1:$apr1$MIwQ3DHb$1I69LzTnZhnCT2DPQmAOK.:remoteusers
```

注記: **htpasswd** コマンドを使用する場合は、各行の *user:password* 部分の後ろに、接続名を手動で追加する必要があります。

xauthby=alwaysok

サーバーは常に、XAUTH ユーザーとパスワードの組み合わせが適切であるように装います。サーバーはユーザー名とパスワードを無視しますが、クライアントはこれらを指定する必要があります。これはユーザーが X.509 証明書で既に特定されている場合、もしくは XAUTH バックエンドが不要な VPN をテストしている場合にのみ使用します。

X.509 証明書を使った設定例を以下に示します。

```
conn xauth-rsa
    auto=add
    authby=rsasig
    pfs=no
    rekey=no
    left=ServerIP
    leftcert=vpn.example.com
    #leftid=%fromcert
    leftid=vpn.example.com
    leftsendcert=always
    leftsubnet=0.0.0.0/0
    rightaddresspool=10.234.123.2-10.234.123.254
    right=%any
    rightrsasigkey=%cert
    modecfgdns1=1.2.3.4
    modecfgdns2=8.8.8.8
    modecfgdomain=example.com
    modecfgbanner="Authorized Access is allowed"
    leftxauthserver=yes
    rightxauthclient=yes
    leftmodecfgserver=yes
    rightmodecfgclient=yes
    modecfgpull=yes
    xauthby=pam
    dpddelay=30
    dpdtimeout=120
    dpdaction=clear
    ike_frag=yes
    # for walled-garden on xauth failure
    # xauthfail=soft
    #leftupdown=/custom/_updown
```

xauthfail を **hard** ではなく **soft** に設定すると認証失敗は無視され、VPN はユーザーが適切に認証したかのように設定されます。カスタムのアップダウンスクリプトを使用すると、環境変数 **XAUTH_FAILED** をチェックできます。このようなユーザーは、たとえば **iptables** DNAT を使って「walled garden」にリダイレクトすることが可能です。リダイレクト先では管理者への問い合わせ、サービスの有料サブスクリプションの更新などができます。

VPN クライアントは、**modecfgdomain** の値と DNS エントリーを使って特定ドメインに対するクエリーを指定されたネームサーバーにリダイレクトします。これにより、ローミングユーザーは内部 DNS ネームを使って内部のみのリソースにアクセスできるようになります。

leftsubnet が **0.0.0.0/0** でない場合は、分割トンネリング設定要求が自動的にクライアントに送信されます。たとえば、**leftsubnet=10.0.0.0/8** を使用すると、VPN クライアントは

10.0.0.0/8 のトラフィックのみを VPN 経由で送信します。

4.7.9. その他のリソース

以下のドキュメントは、**Libreswan** および **ipsec** デーモンに関する追加リソースを提供します。

4.7.9.1. インストールされているドキュメント

- **ipsec(8)** man ページ — **ipsec** のコマンドオプションを説明しています。
- **ipsec.conf(5)** man ページ — **ipsec** の設定情報が含まれています。
- **ipsec.secrets(5)** man ページ — **ipsec.secrets** ファイルの形式を説明しています。
- **ipsec_auto(8)** man ページ — 鍵の自動交換を使用して確立された **Libreswan IPsec** 接続を操作する **auto** コマンドラインクライアントの使用方法を説明しています。
- **ipsec_rsasigkey(8)** man ページ — RSA 署名鍵の生成に使用するツールを説明しています。
- **/usr/share/doc/libreswan-version/README.nss** — **Libreswan pluto** デーモンの NSS 暗号化ライブラリーで使用する生 RSA 鍵と証明書に関するコマンドを説明しています。

4.7.9.2. オンラインのドキュメント

<https://libreswan.org>

アップストリームプロジェクトの Web サイトです。

<http://www.mozilla.org/projects/security/pki/nss/>

Network Security Services (NSS) プロジェクトです。

4.8. OPENSSL の使用

OpenSSL は、アプリケーションに暗号化プロトコルを提供するライブラリーです。**openssl** コマンドラインユーティリティーを使うと、シェルから暗号化機能が使えるようになります。これには、インタラクティブモードが含まれています。

openssl コマンドラインユーティリティーには多くの擬似コマンドがあり、システムにインストールされた **openssl** のバージョンが対応しているコマンドに関する情報を提供します。擬似コマンド **list-standard-commands**、**list-message-digest-commands**、および **list-cipher-commands** はそれぞれ、すべての標準コマンド一覧、メッセージダイジェストコマンド一覧、暗号コマンド一覧を出力します。これは、現行の **openssl** ユーティリティーで利用可能なものです。

擬似コマンド **list-cipher-algorithms** および **list-message-digest-algorithms** は、すべての暗号およびメッセージダイジェストネームを一覧表示します。擬似コマンド **list-public-key-algorithms** は、対応するすべての公開鍵アルゴリズムを一覧表示します。たとえば、対応するすべての公開鍵アルゴリズムを一覧表示するには、以下のコマンドを実行します。

```
~]$ openssl list-public-key-algorithms
```


The pseudo-command `no-command-name` tests whether a *command-name* of the specified name is available. Intended for use in shell scripts. See `man openssl(1)` for more information.

4.8.1. 暗号鍵の作成と管理

OpenSSL では、公開鍵は対応する秘密鍵から生成されます。このため、アルゴリズムを決定した後最初にすることは、秘密鍵の生成になります。以下の例では、秘密鍵を *privkey.pem* とします。たとえば、デフォルトのパラメーターを使用して RSA 秘密鍵を作成するには、以下のコマンドを実行します。

```
~]$ openssl genpkey -algorithm RSA -out privkey.pem
```

RSA アルゴリズムは以下のオプションに対応しています。

- **rsa_keygen_bits:numbits** — 生成される鍵のビット数です。指定されない場合は、**1024** が使われます。
- **rsa_keygen_pubexp:value** — RSA 公開指数の値です。これは大きな十進数か、**0x** で始まる場合は十六進数にすることができます。デフォルト値は **65537** です。

たとえば、公開指数として **3** を使用して 2048 ビットの RSA 秘密鍵を作成するには、以下のコマンドを実行します。

```
~]$ openssl genpkey -algorithm RSA -out privkey.pem -pkeyopt
rsa_keygen_bits:2048 \ -pkeyopt rsa_keygen_pubexp:3
```

128 ビットの AES とパスフレーズ「hello」を使用してこの秘密鍵を出力する際に暗号化するには、以下のコマンドを実行します。

```
~]$ openssl genpkey -algorithm RSA -out privkey.pem -aes-128-cbc -pass
pass:hello
```

See `man genpkey(1)` for more information on generating private keys.

4.8.2. 証明書の生成

OpenSSL を使って証明書を生成するには、秘密鍵が利用可能である必要があります。以下の例では、秘密鍵を *privkey.pem* とします。秘密鍵をまだ生成していない場合は、「[暗号鍵の作成と管理](#)」を参照してください。

証明書に *認証局* (CA) による署名を受けるには、証明書を生成して認証局に送信する必要があります。これは、証明書署名要求と呼ばれます。詳細情報は、「[証明書署名要求の作成](#)」を参照してください。別の方法では、自己署名の証明書を作成します。詳細情報は、「[自己署名証明書の作成](#)」を参照してください。

4.8.2.1. 証明書署名要求の作成

CA に提出する証明書を作成するには、コマンドを以下の形式で実行します。

```
~]$ openssl req -new -key privkey.pem -out cert.csr
```

これで、デフォルトの *privacy-enhanced electronic mail* (PEM) 形式でエンコードされた **cert.csr** と呼ばれる X.509 証明書が作成されます。PEM という名前は、「Privacy Enhancement

for Internet Electronic Mail」に由来し、RFC 1424 で説明されています。別の DER 形式で証明書ファイルを生成するには、**-outform DER** コマンドオプションを使用します。

上記のコマンドを発行すると、証明書の識別名 (DN) を作成するために、ユーザー自身と組織の情報が求められます。以下の情報が必要になります。

- 2 文字の国コード
- 州または県の名前
- 市または自治体
- 組織の名前
- 組織内の部署名
- ユーザー名もしくはシステムのホスト名
- Email アドレス

The req(1) man page describes the PKCS# 10 certificate request and generating utility. Default settings used in the certificate creating process are contained within the `/etc/pki/tls/openssl.cnf` file. See `man openssl.cnf(5)` for more information.

4.8.2.2. 自己署名証明書の作成

366 日間有効の自己署名証明書を生成するには、以下の形式でコマンドを実行します。

```
~]$ openssl req -new -x509 -key privkey.pem -out selfcert.pem -days 366
```

4.8.2.3. Makefile を使った証明書の作成

`/etc/pki/tls/certs` ディレクトリーには **Makefile** が格納されており、これに **make** コマンドを使用すると証明書が作成できます。使用方法を確認するには、以下のコマンドを実行します。

```
~]$ make -f /etc/pki/tls/certs/Makefile
```

または、ディレクトリーに移動して以下のように **make** コマンドを実行することもできます。

```
~]$ cd /etc/pki/tls/certs/  
~]$ make
```

See the `make(1)` man page for more information.

4.8.3. 証明書の検証

CA 署名がされている証明書は、信頼できる証明書と呼ばれます。このため、自己署名証明書は、信頼されない証明書になります。検証ユーティリティーは、**OpenSSL** が通常の工程で使用するものと同じ SSL および S/MIME 機能を使用します。エラーが見つかると報告され、他のエラーを見つけるためにテストを継続する試みがなされます。

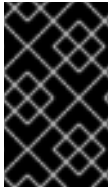
PEM 形式の複数の X.509 証明書を検証するには、以下の形式のコマンドを実行します。

```
~]$ openssl verify cert1.pem cert2.pem
```

証明書チェーンを検証するには、リーフ証明書が **cert.pem** にあり、信頼していない中間証明書が **untrusted.pem** 内で直接連結している必要があります。信頼できるルート CA 証明書は、**/etc/pki/tls/certs/ca-bundle.crt** または **cacert.pem** ファイルでリスト表示されているデフォルトの CA 内にある必要があります。この状態でチェーンを検証するには、以下の形式のコマンドを実行します。

```
~]$ openssl verify -untrusted untrusted.pem -CAfile cacert.pem cert.pem
```

See man verify(1) for more information.



重要

MD5 ハッシュアルゴリズムを使用した署名の検証は、このアルゴリズムの強度が十分でないために Red Hat Enterprise Linux 7 で無効になっています。SHA 256 などの強度の高いアルゴリズムを常に使用するようにしてください。

4.8.4. ファイルの暗号化および暗号化解除

OpenSSL でファイルを暗号化 (および復号化) する場合は、**pkeyutl** または **enc** 組み込みコマンドを使用できます。**pkeyutl** では、暗号化および復号化を実行するために **RSA** 鍵が使用されますが、**enc** では、シンメトリックアルゴリズムが使用されます。

RSA 鍵の使用

ファイル **plaintext** を暗号化するには、以下のコマンドを実行します。

```
~]$ openssl pkeyutl -in plaintext -out cyphertext -inkey privkey.pem
```

鍵および証明書のデフォルトの形式は PEM です。必要に応じて **-keyform DER** オプションを使用して、DER 鍵形式を指定します。

暗号化エンジンを指定するには、**-engine** オプションを以下のように使用します。

```
~]$ openssl pkeyutl -in plaintext -out cyphertext -inkey privkey.pem -  
engine id
```

ここで、**id** は、暗号化エンジンの ID です。エンジンが利用可能かどうかを確認するには、以下のコマンドを実行します。

```
~]$ openssl engine -t
```

plaintext という名前のデータファイルに署名するには、以下のコマンドを実行します。

```
~]$ openssl pkeyutl -sign -in plaintext -out sigtext -inkey privkey.pem
```

署名されたデータファイルを検証し、データを抽出するには、以下のコマンドを実行します。

```
~]$ openssl pkeyutl -verifyrecover -in sig -inkey key.pem
```

DSA 鍵などを使用した署名を検証するには、以下のコマンドを実行します。

```
~]$ openssl pkeyutl -verify -in file -sigfile sig -inkey key.pem
```

The `pkeyutil(1)` manual page describes the public key algorithm utility.

シンメトリックアルゴリズムの使用

利用可能なシンメトリック暗号化アルゴリズムをリスト表示するには、**-l** などの未サポートのオプションを使用して **enc** コマンドを実行します。

```
~]$ openssl enc -l
```

アルゴリズムを指定するには、その名前をオプションとして使用します。たとえば、**aes-128-cbc** アルゴリズムを使用するには、以下の構文を使用します。

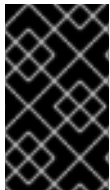
```
openssl enc -aes-128-cbc
```

aes-128-cbc アルゴリズムを使用して **plaintext** という名前のファイルを暗号化するには、以下のコマンドを実行します。

```
~]$ openssl enc -aes-128-cbc -in plaintext -out plaintext.aes-128-cbc
```

前の例で取得したファイルを復号化するには、以下のように **-d** オプションを使用します。

```
~]$ openssl enc -aes-128-cbc -d -in plaintext.aes-128-cbc -out plaintext
```



重要

enc コマンドは **AEAD** 暗号を適切にサポートしないため、**ecb** モードは安全と見なされません。最良の結果を得るために、**cbc**、**cfb**、**ofb**、または **ctr** 以外のモードを使用しないでください。

4.8.5. メッセージダイジェストの生成

dgst コマンドは、十六進数形式で提供されたファイルのメッセージダイジェストを作成します。このコマンドは、デジタル署名および検証にも使用できます。メッセージダイジェストコマンドの形式は以下のとおりです。

```
openssl dgst algorithm -out filename -sign private-key
```

ここで、*algorithm* は、**md5** | **md4** | **md2** | **sha1** | **sha** | **mdc2** | **ripemd160** | **dss1** のいずれかになります。本書執筆時点では、SHA1 アルゴリズムが推奨されます。DSA を使用した署名もしくは検証が必要な場合は、**-rand** オプションで指定したランダムなデータを含むファイルとともに **dss1** オプションを使用する必要があります。

sha1 アルゴリズムを使用してデフォルトの Hex 形式でメッセージダイジェストを作成するには、以下のコマンドを実行します。

```
~]$ openssl dgst sha1 -out digest-file
```

秘密鍵 *privekey.pem* を使用してダイジェストにデジタル署名するには、以下のコマンドを実行します。

```
~]$ openssl dgst sha1 -out digest-file -sign privkey.pem
```

See `man dgst(1)` for more information.

4.8.6. パスワードハッシュの生成

passwd コマンドは、パスワードのハッシュを計算します。コマンドラインでパスワードのハッシュを計算するには、以下のコマンドを実行します。

```
~]$ openssl passwd password
```

デフォルトでは、**-crypt** アルゴリズムが使用されます。

BSD アルゴリズム **1** に基づく MD5 を使用して、標準入力からパスワードのハッシュを計算するには、以下のコマンドを実行します。

```
~]$ openssl passwd -1 password
```

-apr1 オプションが BSD アルゴリズムの Apache バリエーションを指定します。

salt xx を使用してファイルに保存されているパスワードのハッシュを計算するには、以下のコマンドを実行します。

```
~]$ openssl passwd -salt xx -in password-file
```

パスワードは標準出力に送信され、出力ファイルを指定する **-out** オプションはありません。**-table** は、パスワードハッシュの表とそれに対応するクリアテキストのパスワードを生成します。

See man sslpasswd(1) for more information and examples.

4.8.7. ランダムデータの生成

シードファイルを使用してランダムなデータを含むファイルを生成するには、以下のコマンドを実行します。

```
~]$ openssl rand -out rand-file -rand seed-file
```

ランダムデータプロセスをシードするための複数ファイルは、コロン : 区切りのリストを使用して指定できます。

See man rand(1) for more information.

4.8.8. システムのベンチマーキング

あるアルゴリズムにおけるシステムの演算速度をテストするには、以下のコマンドを実行します。

```
~]$ openssl speed algorithm
```

ここでの *algorithm* は、使用する予定の対応アルゴリズムのいずれかになります。利用可能なアルゴリズムを一覧表示するには、**openssl speed** と入力してタブを押します。

4.8.9. OpenSSL の設定

OpenSSL にはマスター設定ファイルと呼ばれる設定ファイル **/etc/pki/tls/openssl.cnf** があり、OpenSSL ライブラリーがこれを読み込みます。各アプリケーション用の個別の設定ファイルを用いることもできます。設定ファイルには、**[section_name]** のようにセクション名が付いた多くのセクションが含まれています。最初の **[section_name]** までの部分は、デフォルトセクションと

呼ばれることに注意してください。OpenSSL が設定ファイル内で名前を検索する際には、最初に名前の付いたセクションが検索されます。別の設定ファイルがコマンド内のオプションで指定されていないければ、OpenSSL コマンドはすべて、マスター OpenSSL 設定ファイルを使用します。設定ファイルの詳細は、**config(5)** man ページで説明されています。

以下の 2 つの RFC は、証明書ファイルのコンテンツについて説明しています。

- 『Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile』
- 『Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile』

4.9. STUNNEL の使用

stunnel プログラムは、クライアントとサーバー間の暗号化ラッパーです。設定ファイルで指定されたポートをリッスンし、クライアントとの通信を暗号化し、通常のポートでリッスンしているオリジナルのデーモンにデータを転送します。こうすることで、それ自体で暗号化をサポートしていないサービスをセキュアにすることができます。また、SSL バージョン 2 や 3 など、POODLE SSL 脆弱性 (CVE-2014-3566) の影響を受け、安全上の理由から使用を避けたい暗号化サービスのセキュリティを改善することもできます。詳細は、<https://access.redhat.com/solutions/1234773> を参照してください。**CUPS** は、自身の設定で SSL を無効にする方法がないコンポーネントの例です。

4.9.1. stunnel のインストール

stunnel パッケージをインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install stunnel
```

4.9.2. stunnel の TLS ラッパーとしての設定

stunnel の設定は、以下の手順にしたがいます。

1. **stunnel** とどのサービスを使うにしても、有効な証明書が必要になります。適切な証明書がない場合は、**認証機関**に申し込むか、自己署名の証明書を自身で作成することもできます。



警告

実稼働環境で稼働しているサーバーには、常に認証機関で署名された証明書を使用してください。自己署名証明書は、テスト目的またはプライベートネットワークのみで使用することをお勧めします

認証局が提供する証明書についての詳細情報は、「**証明書署名要求の作成**」を参照してください。**stunnel** 用に自己署名証明書を作成するには、**root** で **/etc/pki/tls/certs/** ディレクトリに移動し、以下のコマンドを実行します。

```
certs]# make stunnel.pem
```

すべての質問に回答して、プロセスを完了します。

2. 証明書ができたなら、**stunnel** 用の設定ファイルを作成します。これはテキストファイルで、各行でオプションまたはサービス定義の開始を指定します。また、コメントや空の行を使って、読みやすくすることもできます。コメントはセミコロンで開始します。

stunnel RPM パッケージには **/etc/stunnel/** ディレクトリーが含まれ、設定ファイルはここで保存します。**stunnel** ではファイル名で特定の形式や拡張子は必要ありませんが、**/etc/stunnel/stunnel.conf** としてください。以下のコンテンツでは、**stunnel** を TLS ラッパーとして設定します。

```
cert = /etc/pki/tls/certs/stunnel.pem
; Allow only TLS, thus avoiding SSL
sslVersion = TLSv1
chroot = /var/run/stunnel
setuid = nobody
setgid = nobody
pid = /stunnel.pid
socket = l:TCP_NODELAY=1
socket = r:TCP_NODELAY=1

[service_name]
accept = port
connect = port
TIMEOUTclose = 0
```

別の方法では、**sslVersion = TLSv1** の行を以下の行で置き換えると SSL を避けられます。

```
options = NO_SSLv2
options = NO_SSLv3
```

オプションの目的は、以下のとおりです。

- **cert** — 証明書へのパスです。
- **sslVersion** — SSL のバージョンです。SSL と TLS は別個の暗号化プロトコルですが、ここでは **TLS** が使えることに注意してください。
- **chroot** — 変更後の root ディレクトリーです。ここでは、**stunnel** プロセスがより安全に実行できます。
- **setuid, setgid** — **stunnel** プロセスを実行するユーザーおよびグループ。**nobody** は制限されたシステムアカウントになります。
- **pid** — **stunnel** がプロセス ID を保存するファイルで、**chroot** に相対的になります。
- **socket** — ローカルおよびリモートのソケットオプション。このケースでは、ネーグルのアルゴリズムを無効にして、ネットワーク遅延を改善します。
- **[service_name]** — サービス定義の開始点。この下に続く行で使用されるオプションは、該当サービスのみに適用されます。この上にあるオプションは、**stunnel** でグローバルに適用されます。
- **accept** — リッスンするポートです。
- **connect** — 接続先のポートです。このポートは、セキュアにしているサービスが使用するものである必要があります。

- **TIMEOUTclose** — クライアントから *close_notify* 警告が出されるまでの待ち時間です。これが **0** の場合は、**stunnel** はまったく待機しません。
- **options** — OpenSSL ライブラリーのオプション。

例4.10 CUPS のセキュア化

CUPS 用に **stunnel** を TLS ラッパーとして設定するには、以下の値を使用します。

```
[cups]
accept = 632
connect = 631
```

632 の代わりに使用されていないポートを使うこともできます。**631** は **CUPS** が通常使用するポートです。

3. **chroot** ディレクトリーを作成し、**setuid** オプションで指定されているユーザーにこのディレクトリーへの書き込みアクセスを与えます。これを行うには、**root** で以下のコマンドを実行します。

```
~]# mkdir /var/run/stunnel
~]# chown nobody:nobody /var/run/stunnel
```

これで、**stunnel** が PID ファイルを作成できるようになります。

4. 使用中のシステムでファイアウォールが新たなポートへのアクセスを許可しない設定となっている場合は、この設定を許可するように変更します。詳細は、「[ファイアウォールのポート開放](#)」を参照してください。
5. 設定ファイルと **chroot** ディレクトリーを作成し、指定されたポートがアクセス可能なことを確認したら、**stunnel** を開始することができます。

4.9.3. stunnel の起動、停止、再起動

stunnel を起動するには、**root** で以下のコマンドを実行します。

```
~]# stunnel /etc/stunnel/stunnel.conf
```

デフォルトでは、**stunnel** は出力のログに **/var/log/secure** を使用します。

stunnel を停止するには、**root** で以下のコマンドを実行してプロセスを強制終了させます。

```
~]# kill `cat /var/run/stunnel/stunnel.pid`
```

stunnel の実行中に設定ファイルを編集した場合は、**stunnel** を停止して再起動すると、変更が反映されます。

4.10. 暗号化

4.10.1. LUKS のディスク暗号化の使用

Linux Unified Key Setup-on-disk-format (または LUKS) を使うと、Linux コンピューター上のパー

ティションを暗号化できます。これは特に、モバイルコンピューターやリムーバブルメディアを使う際に重要です。LUKS を使うと、複数のユーザーキーを使って、パーティションのバルク暗号化に使用されるマスターキーの暗号化解除ができるようになります。

LUKS の概要

LUKS の機能

- LUKS はブロックデバイス全体を暗号化するため、脱着可能なストレージメディアやノート PC のディスクドライブといった、モバイルデバイスのコンテンツ保護に適しています。
- 暗号化されたブロックデバイスにあるのは任意のコンテンツです。これは、スワップデバイスの暗号化に役立ちます。また、とりわけデータストレージ用にフォーマットしたブロックデバイスを使用する特定のデータベースに関しても有用です。
- LUKS は既存のデバイスマッパーカーネルサブシステムを使用します。
- LUKS はパラフレーズの強化を提供し、辞書攻撃から保護します。
- LUKS デバイスには複数のキースロットが含まれ、ユーザーはこれを使ってバックアップキーやパスフレーズを追加できます。

LUKS でできないこと:

- LUKS は、多くのユーザー (9 人以上) が同一デバイスに対して別々のアクセスを持つことが必要となるアプリケーションには適していません。
- LUKS は、ファイルレベルの暗号化を必要とするアプリケーションには適していません。

4.10.1.1. Red Hat Enterprise Linux における LUKS の実装

Red Hat Enterprise Linux 7 は、LUKS を使ってファイルシステムを暗号化します。デフォルトではインストール時に、ファイルシステムを暗号化するオプションのチェックが外されています。ハードディスクを暗号化するオプションを選択すると、コンピューターを起動するたびにパスフレーズを尋ねられます。このパスフレーズは、パーティションの暗号化解読に用いられるバルク暗号化鍵を「ロック解除」します。デフォルトのパーティションテーブルの変更を選択すると、暗号化するパーティションを選択できます。この設定は、パーティションテーブル設定で行われます。

LUKS に使用されるデフォルトの暗号 (**cryptsetup --help** を参照) は **aes-cbc-essiv:sha256** (ESSIV - Encrypted Salt-Sector Initialization Vector) です。インストールプログラムの **Anaconda** は、デフォルトでは XTS モード (**aes-xts-plain64**) を使用することに注意してください。LUKS のデフォルトの鍵のサイズは 256 ビット です。**Anaconda** (XTS モード) と併用する場合の LUKS のデフォルトの鍵のサイズは 512 ビット です。利用可能な暗号は以下の通りです。

- AES - Advanced Encryption Standard (高度暗号化標準) - [FIPS PUB 197](#)
- Twofish (128 ビットブロック暗号)
- Serpent
- cast5 - [RFC 2144](#)
- cast6 - [RFC 2612](#)

4.10.1.2. 手動でのディレクトリーの暗号化

**警告**

以下の手順を実行すると、暗号化しているパーティションの既存データがすべて削除されます。すべての情報が失われてしまうので、この手順を開始する前に、外部ソースへのデータのバックアップを必ず行なってください。

1. **root** としてシェルプロンプトに以下を入力し、ランレベル 1 に入ります。

```
telinit 1
```

2. 既存の **/home** のマウントを解除します。

```
umount /home
```

3. 直前の手順のコマンドが失敗した場合は、**fuser** を使用し、**/home** を独占しているプロセスを見つけてこれらを止めます。

```
fuser -mvk /home
```

4. **/home** がもはやマウントされていないことを確認します。

```
grep home /proc/mounts
```

5. パーティションをランダムなデータで埋めます。

```
shred -v --iterations=1 /dev/VG00/LV_home
```

このコマンドは、デバイスの連続書き込み速度で実行され、完了までに時間がかかる場合があります。使用デバイスに暗号化されていないデータが残っていないことを確認した上で、デバイスの暗号化されたデータを含む部分を難読化するのは重要なステップです。

6. パーティションを初期化します。

```
cryptsetup --verbose --verify-passphrase luksFormat  
/dev/VG00/LV_home
```

7. 新たに暗号化したデバイスを開きます。

```
cryptsetup luksOpen /dev/VG00/LV_home home
```

8. デバイスがあることを確認します。

```
ls -l /dev/mapper | grep home
```

9. ファイルシステムを作成します。

```
mkfs.ext3 /dev/mapper/home
```

-
- 10. ファイルシステムをマウントします。

```
mount /dev/mapper/home /home
```

- 11. ファイルシステムが表示されていることを確認します。

```
df -h | grep home
```

- 12. 以下を **/etc/crypttab** ファイルに追加します。

```
home /dev/VG00/LV_home none
```

- 13. **/etc/fstab** ファイルを編集して、**/home** の古いエントリを削除し、以下の行を追加します。

```
/dev/mapper/home /home ext3 defaults 1 2
```

- 14. デフォルトの SELinux セキュリティーコンテンツを復元します。

```
/sbin/restorecon -v -R /home
```

- 15. マシンを再起動します。

```
shutdown -r now
```

- 16. **/etc/crypttab** にあるエントリにより、コンピューターのブート時に**luks** パスフレーズが尋ねられます。

- 17. root としてログインし、バックアップを復元します。

これですべてのデータ用に暗号化されたパーティションを設定できたので、コンピューターをオフにしている間もデータを安全に保管できます。

4.10.1.3. 既存デバイスへの新規パスフレーズの追加

既存デバイスに新規パスフレーズを追加するには、以下のコマンドを使用します。

```
cryptsetup luksAddKey device
```

認証のために既存のパスフレーズが尋ねられた後に、新規パスフレーズの入力を求めるプロンプトが表示されます。

4.10.1.4. 既存のデバイスからのパスフレーズ削除

既存のデバイスからパスフレーズを削除するには、以下のコマンドを使用します。

```
cryptsetup luksRemoveKey device
```

削除したいパスフレーズを求めるプロンプトの後に、認証に必要な残りのパスフレーズを求めるプロンプトが表示されます。

4.10.1.5. Anaconda での暗号化したブロックデバイスの作成

システムのインストール時に、暗号化されたデバイスを作成することができます。これにより、暗号化パーティションを含むシステムを簡単に設定することができます。

ブロックデバイスの暗号化を有効にするには、自動パーティション設定を選択している場合は **システムの暗号化** チェックボックスに、個別パーティション、ソフトウェア RAID アレイまたは論理ボリュームを作成している場合は **暗号化** チェックボックスにチェックを入れます。パーティション設定が終了したら、暗号化のパスフレーズが尋ねられます。このパスフレーズは暗号化したデバイスへのアクセスに必要となります。LUKS デバイスが事前に存在しており、インストールプロセスの当初にそれらの正しいパスフレーズを指定している場合には、チェックボックスのあるパスフレーズ入力ダイアログが表示されます。このチェックボックスにチェックを入れると、既存の暗号化ブロックデバイスの利用可能なスロットに新規パスフレーズを追加することになります。



注記

自動パーティション設定 画面の **システムの暗号化** チェックボックスにチェックを入れた後に **カスタムレイアウトの作成** を選択しても、ブロックデバイスは自動的に暗号化されません。



注記

kickstart を使用すると、新たに暗号化されたブロックデバイスのパスフレーズを個別に設定することができます。

4.10.1.6. その他のリソース

Red Hat Enterprise Linux 7 における LUKS や暗号化ハードディスクについての詳細は、以下のリンクにアクセスしてください。

- [LUKS home page](#)
- [LUKS/cryptsetup FAQ](#)
- [LUKS - Linux Unified Key Setup Wikipedia article](#)
- [HOWTO: Creating an encrypted Physical Volume \(PV\) using a second hard drive and pvmove](#)

4.10.2. GPG 鍵の作成

GnuPG (GPG) は、ユーザーを識別し、通信 (確認できない人々との通信も含む) を認証するために使われます。GPG は、GPG 署名のある電子メールを読む人がその真正性を検証できるようにします。つまり、GPG は、あなたが署名した通信が実際にあなたからのものであることをかなりの精度で確認できるようにします。GPG は、第三者がコードを変更したり、会話を傍受したり、メッセージを改ざんしたりすることを防ぐ点で役に立ちます。

4.10.2.1. GNOME での GPG 鍵の生成

GNOME で GPG 鍵を作成するには、以下の手順にしたがいます。

1. **Seahorse** ユーティリティをインストールします。これにより GPG 鍵の管理が容易になります。

```
~]# yum install seahorse
```

2. 鍵を作成するには、アプリケーション → アクセサリメニューから、パスワードと暗号鍵を選択します。これでアプリケーション **Seahorse** が起動します。
3. ファイルメニューから**新規**を選択し、**PGP 鍵**を選択した後に**続行**をクリックします。
4. 氏名、電子メールアドレスおよび自身についての説明のオプションのコメント (例: John C. Smith, jsmith@example.com, Software Engineer) を入力します。**生成**をクリックします。鍵のパスフレーズを問い合わせるダイアログが表示されます。パスフレーズは強固なだけでなく覚えやすいものを選択してください。**OK** をクリックすると鍵が作成されます。



警告

パスフレーズを忘れると、データの暗号解除ができなくなります。

GPG 鍵の ID をを見つけるには、新規に作成された鍵の横にある **鍵の ID** コラムを確認します。多くの場合、鍵の ID を求められたら、**0x6789ABCD** などのように鍵の ID の前に**0x** を付けます。秘密鍵のバックアップを取り、安全な場所に保管してください。

4.10.2.2. KDE での GPG 鍵の作成

KDE で GPG 鍵を作成するには、以下の手順にしたがいます。

1. メインメニューからアプリケーション → ユーティリティ → 暗号ツールを選択して **KGpg** プログラムを起動します。これまで **KGpg** を使用したことがなければ、プログラムが独自の GPG 鍵ペアを生成するプロセスを詳しく説明します。
2. ダイアログボックスで、新しい鍵ペアを生成するよう求められます。名前、電子メールアドレス、およびオプションのコメントを入力します。鍵の長さ (ビット数) とアルゴリズムに加え、鍵の有効期限も選択できます。
3. 次のダイアログでパスフレーズを入力します。この時点で、鍵が **KGpg** のメインウィンドウに表示されます。



警告

パスフレーズを忘れると、データの暗号解除ができなくなります。

GPG 鍵の ID をを見つけるには、新規に作成された鍵の横にある **鍵の ID** コラムを確認します。多くの場合、鍵の ID を求められたら、**0x6789ABCD** などのように鍵の ID の前に**0x** を付けます。秘密鍵のバックアップを取り、安全な場所に保管してください。

4.10.2.3. コマンドラインを用いた GPG 鍵の生成

1. 次のシェルコマンドを使用します。

```
~]$ gpg2 --gen-key
```

このコマンドは、公開鍵と秘密鍵で構成される鍵ペアを生成します。受信側では、ユーザーからの通信の認証や暗号化解除を行うためにこのユーザーの公開鍵を使用します。そのため、とりわけメーリングリストのように、送信者からの認証済みの通信を受け取ることを希望すると思われる人々に向けて、できる限り幅広く公開鍵を配布してください。

- 一連のプロンプトにしたがってプロセスを進めます。デフォルト値を割り当てる場合は **Enter** キーを押します。最初のプロンプトは、使用を希望する鍵の種類を選択するよう尋ねます。

```
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection?
```

ほとんど多くの場合、デフォルトが適切な選択になります。RSA/RSA 鍵を選択すると、通信に署名するだけでなく、ファイルを暗号化することができます。

- 鍵のサイズを選択します。

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

ここでも、デフォルトの 2048 はほとんどすべてのユーザーに適しており、これは極めて強いレベルのセキュリティになります。

- 鍵の有効期限を選択します。デフォルトの **none** を使用するのではなく、失効日を選択する方がよいでしょう。例えば、鍵にある電子メールアドレスが無効になると、失効日が設定されているために、他の人々はその公開鍵を使用するのを止めるように通知されます。

```
Please specify how long the key should be valid.
0 = key does not expire
d = key expires in n days
w = key expires in n weeks
m = key expires in n months
y = key expires in n years
key is valid for? (0)
```

例えば、**1y** の値を入力すると、鍵の有効期間は 1 年間になります (鍵の生成後にこの失効日を変更することができます)。

- gpg2** プログラムが署名情報を尋ねる前に、以下のプロンプトが表示されます。

```
Is this correct (y/N)?
```

y を入力してプロセスを終了します。

- GPG 鍵用に氏名と電子メールアドレスを入力します。このプロセスは、ユーザーを個人として認証するためのものです。このため、本当の名前を入力してください。偽の電子メールアドレスを選択すると、他の人があなたの公開鍵を見つけにくくなります。これにより、通信の認証が困難になります。例えば、メーリングリストの自己紹介にこの GPG 鍵を使用する場合、そのリストで使用している電子メールアドレスを入力します。

コメントフィールドには、エイリアスやその他の情報を記載します。(一部の人は、目的別に複数の鍵を使用しており、「オフィス」または「オープンソースプロジェクト」などのコメントを付けてそれぞれの鍵を識別しています。)

7. すべてのエントリーが正しいければ、確認プロンプトで **0** と入力して続行するか、または問題がある場合はそれを修正するために他のオプションを使用します。最後に、秘密鍵のパスフレーズを入力します。**gpg2** プログラムは、入力エラーがないことを確認するためにパスフレーズを 2 回入力するように指示します。
8. 最後に、**gpg2** はランダムなデータを生成して鍵を可能な限り一意なものにします。このプロセスを短縮するには、この手順の実行中にマウスを動かし、ランダムなキーを入力するか、またはシステム上で他のタスクを実行します。この手順が完了すると、鍵が完成して使用可能な状態になります。

```
pub 1024D/1B2AFA1C 2005-03-31 John Q. Doe <jqdoe@example.com>
Key fingerprint = 117C FE83 22EA B843 3E86 6486 4320 545E 1B2A FA1C
sub 1024g/CEA4B22E 2005-03-31 [expires: 2006-03-31]
```

9. 鍵のフィンガープリントは、鍵の「署名」の短縮版です。これを使って、他の人々が実際の公開鍵を(改ざんされない状態で)受け取ったことを確認することができます。このフィンガープリントを書き留めておく必要はありません。フィンガープリントを表示するには、以下のコマンドをあなたの電子メールアドレスに置き換えて使用します。

```
~]$ gpg2 --fingerprint jqdoe@example.com
```

「GPG 鍵の ID」は、公開鍵を識別する 16 進法の 8 文字で構成されます。上記の例では、GPG 鍵 ID は **1B2AFA1C** です。多くの場合、鍵 ID を求められる際には、**0x6789ABCD** のように、鍵 ID の前に **0x** を付けます。



警告

パスフレーズを忘れると、鍵を使うことができなくなり、その鍵で暗号化されたすべてのデータが失われます。

4.10.2.4. 公開鍵の暗号化について

1. [Wikipedia - Public Key Cryptography](#)
2. [HowStuffWorks - Encryption](#)

4.10.3. 公開鍵暗号化における openCryptoki の使用

openCryptoki は PKCS#11 の Linux 実装で、トークンと呼ばれる暗号化デバイスへのアプリケーションプログラミングインターフェース (API) を定義する **公開鍵暗号標準** です。トークンは、ハードウェアまたはソフトウェアに実装することが可能です。本章では、Red Hat Enterprise Linux 7 での **openCryptoki** システムのインストール、設定および使用についての概要を説明します。

4.10.3.1. openCryptoki のインストールとサービスの起動

テスト目的のトークンのソフトウェア実装を含む、基本の **openCryptoki** パッケージをシステムにイ

インストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install opencryptoki
```

使用するハードウェアトークンのタイプによっては、特別なユースケース用のサポートを提供する追加パッケージのインストールが必要になる場合もあります。たとえば、*Trusted Platform Module* (TPM) デバイス用のサポートを入手するには、`opencryptoki-tpmtok` パッケージをインストールする必要があります。

Yum パッケージマネジャーを使用してパッケージをインストールする情報全般に関しては、Red Hat Enterprise Linux 7 システム管理者のガイドの [パッケージのインストール](#) セクションを参照してください。

openCryptoki サービスを有効にするには、**pkcsslotd** デーモンを実行する必要があります。**root** で以下のコマンドを実行すると、現行セッションでこのデーモンが起動します。

```
~]# systemctl start pkcsslotd
```

ブート時にサービスが自動的に起動するには、以下のコマンドを実行します。

```
~]# systemctl enable pkcsslotd
```

`systemd` ターゲットを使用したサービスの管理方法に関する詳細情報は、Red Hat Enterprise Linux 7 システム管理者のガイドの [systemd によるサービス管理](#) の章を参照してください。

4.10.3.2. openCryptoki の設定と使用

pkcsslotd デーモンは起動後に `/etc/opencryptoki/opencryptoki.conf` 設定ファイルを読み込みます。このファイルは、システムと機能するように設定されたトークンとそのスロットについての情報を収集するために使用されます。

The file defines the individual slots using key-value pairs. Each slot definition can contain a description, a specification of the token library to be used, and an ID of the slot's manufacturer. Optionally, the version of the slot's hardware and firmware may be defined. See the `opencryptoki.conf(5)` manual page for a description of the file's format and for a more detailed description of the individual keys and the values that can be assigned to them.

ランタイム時の **pkcsslotd** デーモンの動作を修正するには、**pkcsconf** ユーティリティーを使用します。このツールを使うと、デーモンの状態の表示と設定に加え、現在設定されているスロットとトークンの一覧表示と修正がでいます。たとえば、トークンについての情報を表示するには、以下のコマンドを実行します。(pkcsslotd デーモンと通信する必要がある root 以外のユーザーは、**pkcs11** システムグループに属している必要があることに注意してください)

```
~]$ pkcsconf -t
```

See the `pkcsconf(1)` manual page for a list of arguments available with the **pkcsconf** tool.



警告

pkcs11 グループには、完全に信頼できるユーザーのみを割り当ててください。このグループのメンバーは、**openCryptoki** サービスの他のユーザーが設定済み PKCS#11 トークンへアクセスできなくすることができます。またこのグループのメンバーは、**openCryptoki** の他のユーザーの権限で任意のコードを実行することができます。

4.10.4. OpenSSH に認証情報を提供するスマートカードの使用

スマートカードは、USB スティック、MicroSD またはスマートカードのフォームファクター形式の軽量のハードウェアセキュリティモジュールです。スマートカードにより、セキュアなキーストアをリモートで管理できます。Red Hat Enterprise Linux 7 では OpenSSH はスマートカードを使用した認証をサポートします。

OpenSSH でスマートカードを使用するには、カードの公開鍵を `~/.ssh/authorized_keys` ファイルに保存して、クライアント上で `opensc` パッケージで提供される **PKCS#11** ライブラリーをインストールします。**PKCS#11** は Public-Key Cryptography Standard のことで、トークンと呼ばれる暗号化デバイスにアプリケーションプログラミングインターフェース (API) を定義します。**root** として以下のコマンドを実行してください。

```
~]# yum install opensc
```

`opensc` (CoolKey and CAC) でサポートされないスマートカードを使用するには、**root** として以下のコマンドを実行して `coolkey` パッケージをインストールします。

```
~]# yum install coolkey
```

4.10.4.1. カードからの公開鍵の取得

カードの鍵を表示するには、**ssh-keygen** コマンドを使用します。**-D** ディレクティブで共有ライブラリー (以下の例では `OpenSC`) を指定します。

```
~]$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so
ssh-rsa AAAAB3NzaC1yc[...]+g4Mb9
```

4.10.4.2. サーバーへの公開鍵の保存

リモートサーバーでスマートカードを使用した認証を有効化するには、公開鍵をリモートサーバーに移動します。これには、取得した文字列 (鍵) をコピーして、リモートのシェルにペーストするか、ファイル (以下の例では **smartcard.pub**) にキーを保存して、**ssh-copy-id** コマンドを使用してください。

```
~]$ SSH_COPY_ID_LEGACY=1 ssh-copy-id -i smartcard.pub user@hostname
user@hostname's password:
```

```
Number of key(s) added: 1
```

```
Now try logging into the machine, with:  "ssh user@hostname"
and check to make sure that only the key(s) you wanted were added.
```

秘密鍵ファイルなしに公開鍵を保存するには、**SSH_COPY_ID_LEGACY=1** の環境変数を使用する必要があります。

4.10.4.3. スマートカード上の鍵を使用したサーバーの認証

OpenSSH は、スマートカードから公開鍵を読み込み、鍵自体を公開せずに秘密鍵を使用して操作を行います。つまり、秘密鍵がカードから出ることはありません。認証のためにスマートカードを使用してリモートサーバーに接続するには、以下のコマンドを実行してカードを保護する PIN を入力します。

```
[localhost ~]$ ssh -I /usr/lib64/pkcs11/opensc-pkcs11.so hostname
Enter PIN for 'Test (UserPIN)':
[hostname ~]$
```

hostname は、接続先の実際のホスト名に置き換えます。

次回、必要のない入力をしなくて済むように、`~/.ssh/config` ファイルに **PKCS#11** ライブラリーへのパスを保存します。

```
Host hostname
    PKCS11Provider /usr/lib64/pkcs11/opensc-pkcs11.so
```

追加オプションなしに **ssh** コマンドを実行して接続します。

```
[localhost ~]$ ssh hostname
Enter PIN for 'Test (UserPIN)':
[hostname ~]$
```

4.10.4.4. PIN でのログインを自動化するためのssh-agent の使用

ssh-agent を使用するように、環境変数を設定します。通常のセッションでは**ssh-agent** はすでに実行されているので、多くの場合、この手順を省略できます。以下のコマンドを使用して、認証エージェントに接続できるかどうかを確認してください。

```
~]$ ssh-add -l
Could not open a connection to your authentication agent.
~]$ eval `ssh-agent`
```

この鍵で接続するたびに PIN の入力をしなくて済むように、以下のコマンドを実行してエージェントにカードを追加します。

```
~]$ ssh-add -s /usr/lib64/pkcs11/opensc-pkcs11.so
Enter PIN for 'Test (UserPIN)':
Card added: /usr/lib64/pkcs11/opensc-pkcs11.so
```

ssh-agent からカードを削除するには、以下のコマンドを実行します。

```
~]$ ssh-add -e /usr/lib64/pkcs11/opensc-pkcs11.so
Card removed: /usr/lib64/pkcs11/opensc-pkcs11.so
```

4.10.4.5. その他のリソース

ハードウェアまたはソフトウェアトークンの設定は「[Smart Card support in Red Hat Enterprise Linux 7](#)」の記事に記載されています。

4.10.5. 信頼できる鍵および暗号化された鍵

信頼できる鍵および暗号化された鍵は、カーネルキーリングサービスを使用するカーネルが生成する可変長のシンメトリックキーです。この鍵はユーザースペースでは暗号解除された形式で表示されないため、その整合性が検証可能になります。このため、たとえば拡張検証モジュール (EVM) がこの鍵を使用して稼働中のシステムの整合性を検証かつ確認することができるようになります。ユーザーレベルのプログラムがアクセス可能なのは、暗号化された `blob` の形式での鍵のみです。

信頼できる鍵は、*Trusted Platform Module* (TPM) チップというハードウェアが必要になります。これは、鍵の作成と暗号化 (保護) の両方に使用されます。TPM は、*storage root key* (SRK) と呼ばれる 2048 ビットの **RSA** 鍵を使ってこの鍵を保護します。

さらに、信頼できる鍵は TPM の *プラットフォーム構成レジスター* (PCR) の特定の値のセットを使って保護される場合があります。PCR には、BIOS を反映する整合性管理の値、ブートローダー、およびオペレーティングシステムのセットが含まれています。つまり、PCR で保護された鍵は、これが暗号化された全く同一システム上の TPM でしか暗号解除できないことになります。ただし、PCR で保護された信頼できる鍵が読み込まれると (キーリングに追加されると)、すなわちそれに関連付けられている PCR の値が確認されると、新たな (または将来の) PCR の値で更新可能となります。このため、たとえば、新規カーネルのブートが可能になります。また、ひとつの鍵は異なる PCR の値を持つ複数の `blob` として保存することもできます。

暗号化された鍵は、カーネル **AES** 暗号化を使用するので、TPM を必要とせず、信頼できる鍵よりもスピーディーになります。暗号化された鍵は、カーネル生成の任意の数を使って作成され、ユーザースペース `blob` にエクスポートされる際に マスターキーで暗号化されます。このマスターキーは、信頼できる鍵かユーザーキーとすることができます。後者の場合、暗号化された鍵の安全性は、ユーザーキーによる暗号化と同等のものでしかないという欠点があります。

4.10.5.1. 鍵を使った作業

鍵を使う操作の前には、関連するカーネルモジュールの読み込みが必要になります。信頼できる鍵の場合は **trusted** モジュール、暗号化された鍵の場合は **encrypted-keys** モジュールです。**root** ユーザーで以下のコマンドを実行して、これらモジュールの両方を同時に読み込みます。

```
~]# modprobe trusted encrypted-keys
```

Trusted and encrypted keys can be created, loaded, exported, and updated using the **keyctl** utility. For detailed information about using **keyctl**, see `keyctl(1)`.



注記

TPM を使用するには (信頼できる鍵の作成および保護目的など)、これを有効かつアクティブにする必要があります。これは通常、マシンの BIOS で設定するか、ユーティリティの `tpm-tools` パッケージから **tpm_setactive** コマンドを使用するとできます。また、**TrouSers** アプリケーション (`trousers` パッケージ) のインストールと、**TrouSers** スイートの一部である **tcstd** デーモンが稼働して TPM と通信している必要もあります。

TPM を使用して信頼できる鍵を作成するには、以下の構文で **keyctl** コマンドを実行します。

```
keyctl add trusted name "new keylength [options]" keyring
```

上記の構文を使用したコマンド例は以下のようになります。

```
~]$ keyctl add trusted kmk "new 32" @u
642500861
```

上記の例では、**kmk** と呼ばれる 32 バイト (256 ビット) の長さの信頼できる鍵が作成され、ユーザーキーリング (@u) に置かれます。この鍵は、32 から 128 バイト (256 から 1024 ビット) の長さになります。**show** サブコマンドを使ってカーネルキーリングの現在の構成を一覧表示します。

```
~]$ keyctl show
Session Keyring
      -3 --alswrv      500    500  keyring: _ses
 97833714 --alswrv      500     -1   \_ keyring: _uid.1000
642500861 --alswrv      500    500   \_ trusted: kmk
```

print サブコマンドは、暗号化された鍵を標準出力に出します。この鍵をユーザースペースのプロブにエクスポートするには、以下のように **pipe** サブコマンドを使用します。

```
~]$ keyctl pipe 642500861 > kmk.blob
```

信頼できる鍵をユーザースペースのプロブから読み込むには、**add** コマンドでプロブを引数として使用します。

```
~]$ keyctl add trusted kmk "load `cat kmk.blob`" @u
268728824
```

これで TPM で保護された信頼できる鍵を用いて安全な暗号化された鍵を作成することができます。暗号化された鍵の生成には、以下のコマンド構文が使用されます。

```
~]$ keyctl add encrypted name "new [format] key-type:master-key-name  
keylength" keyring
```

上記の構文に基づき、既存の信頼できる鍵を使って暗号化された鍵を生成するコマンドは以下のようになります。

```
~]$ keyctl add encrypted encr-key "new trusted:kmk 32" @u
159771175
```

TPM が利用できないシステムで暗号化された鍵を作成するには、任意の数の列を使ってユーザーキーを生成し、それを使って実際の暗号化された鍵を保護します。

```
~]$ keyctl add user kmk-user "`dd if=/dev/urandom bs=1 count=32  
2>/dev/null`" @u
427069434
```

その後に、任意の数のユーザーキーを使って暗号化された鍵を生成します。

```
~]$ keyctl add encrypted encr-key "new user:kmk-user 32" @u
1012412758
```

list サブコマンドを使うと、指定されたカーネルキーリング内のすべての鍵を一覧表示できます。

```
~]$ keyctl list @u
2 keys in keyring:
427069434: --alswrv 1000 1000 user: kmk-user
1012412758: --alswrv 1000 1000 encrypted: encr-key
```



重要

マスターの信頼できる鍵で保護されていない暗号化された鍵の安全性は、その鍵の暗号化に使用されたユーザーマスターキー（任意の数の鍵）と同等にしかならないことに注意してください。このため、マスターユーザーキーはできるだけ安全に、可能であればブートプロセスの初期に、読み込むようにしてください。

4.10.5.2. その他のリソース

以下のオフラインおよびオンラインのリソースでは、信頼できる鍵および暗号化された鍵に関する追加情報が提供されています。

インストールされているドキュメント

- `keyctl(1)` — Describes the use of the **keyctl** utility and its subcommands.

オンラインのドキュメント

- [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) — Red Hat Enterprise Linux 7 の『SELinux User's and Administrator's Guide』では、**SELinux** の原則と、**SELinux** を設定して **Apache HTTP Server** などのさまざまなサービスで使用方法が詳細に説明されています。
- <https://www.kernel.org/doc/Documentation/security/keys-trusted-encrypted.txt> — Linux カーネルの信頼できる鍵および暗号化された鍵に関する機能についての公式ドキュメントです。

関連項目

- 「[高度暗号化標準 — AES](#)」では、**Advanced Encryption Standard** (高度暗号化標準) について簡単に説明しています。
- 「[公開鍵暗号](#)」では、公開鍵の暗号化のアプローチとそこで使用される様々な暗号化プロトコルについて説明しています。

4.10.6. 乱数ジェネレーターの使用

簡単に解読されない安全な暗号鍵を生成するには、乱数のソースが必要になります。一般的に、番号がよりランダムであればあるほど、一意の鍵を得られる可能性が高まります。乱数生成の**エントロピー**は通常、コンピューティング環境の「ノイズ」または **乱数ジェネレーター** のハードウェアを使用することで獲得されます。

`rng-tools` パッケージの一部である **rngd** デーモンは、エントロピーを引き出すために環境ノイズと乱数ジェネレーターの両方が使えます。このデーモンは、乱数度のソースが提供したデータがランダムかどうかをチェックし、その後にカーネルの乱数エントロピープールに保存します。これが生成した乱数は、`/dev/random` および `/dev/urandom` の各キャラクターデバイスから利用できます。

`/dev/random` と `/dev/urandom` の違いは、前者はブロックデバイスであり、適切な乱数出力の生成

にエントロピーが不十分だと判断すると、乱数の提供が停止される一方で、**/dev/urandom** は非ブロックソースであり、カーネルのエントロピープールを再利用して無制限の仮の乱数を提供できる、という点です。ただし、この場合のエントロピーは低くなります。このため、長期の暗号鍵の作成には、**/dev/urandom** は使用しないでください。

rng-tools パッケージをインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install rng-tools
```

rngd デーモンを起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start rngd
```

デーモンのステータスを確認するには、以下のコマンドを実行します。

```
~]# systemctl status rngd
```

オプションのパラメーターを使って **rngd** デーモンを起動するには、直接これを実行します。たとえば、乱数入力の代替ソース (**/dev/hwrng** 以外) を指定するには、以下のコマンドを実行します。

```
~]# rngd --rng-device=/dev/hwrng
```

The above command starts the **rngd** daemon with **/dev/hwrng** as the device from which random numbers are read. Similarly, you can use the **-o** (or **--random-device**) option to choose the kernel device for random-number output (other than the default **/dev/random**). See the **rngd(8)** manual page for a list of all available options.

任意のシステムでどのソースのエントロピーが利用可能であるかを確認するには、以下のコマンドを **root** として実行します。

```
~]# rngd -v
Unable to open file: /dev/tpm0
Available entropy sources:
DRNG
```

TPM デバイスがない場合には、エントロピーのソースとして Intel Digital Random Number Generator (DRNG) のみが表示されます。CPU が RDRAND プロセッサの命令をサポートするか確認するには、以下のコマンドを実行します。

```
~]$ cat /proc/cpuinfo | grep rdrand
```



注記

ソフトウェアコードの例および詳しい情報は「[Intel Digital Random Number Generator \(DRNG\) Software Implementation Guide](#)」を参照してください。

rng-tools パッケージには **rngtest** ユーティリティーも含まれており、これはデータの乱数度のチェックに使用できます。**/dev/random** の出力の乱数度をテストするには、以下のように **rngtest** ツールを使用します。

```
~]$ cat /dev/random | rngtest -c 1000
rngtest 5
```

Copyright (c) 2004 by Henrique de Moraes Holschuh

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```
rngtest: starting FIPS tests...
rngtest: bits received from input: 20000032
rngtest: FIPS 140-2 successes: 998
rngtest: FIPS 140-2 failures: 2
rngtest: FIPS 140-2(2001-10-10) Monobit: 0
rngtest: FIPS 140-2(2001-10-10) Poker: 0
rngtest: FIPS 140-2(2001-10-10) Runs: 0
rngtest: FIPS 140-2(2001-10-10) Long run: 2
rngtest: FIPS 140-2(2001-10-10) Continuous run: 0
rngtest: input channel speed: (min=1.171; avg=8.453; max=11.374)Mibits/s
rngtest: FIPS tests speed: (min=15.545; avg=143.126; max=157.632)Mibits/s
rngtest: Program run time: 2390520 microseconds
```

A high number of failures shown in the output of the **rngtest** tool indicates that the randomness of the tested data is insufficient and should not be relied upon. See the `rngtest(1)` manual page for a list of options available for the **rngtest** utility.

Red Hat Enterprise Linux 7 では **virtio RNG** (乱数ジェネレーター) デバイスが導入され、ホストマシンからエントロピーにアクセスできる **KVM** 仮想マシンが提供されています。推奨の設定では、`hwrng` はホストの Linux カーネルのエントロピープールに (`/dev/random` 経由で) フィードし、**QEMU** は、ゲストが要求したエントロピーのソースとして `/dev/random` を使用します。

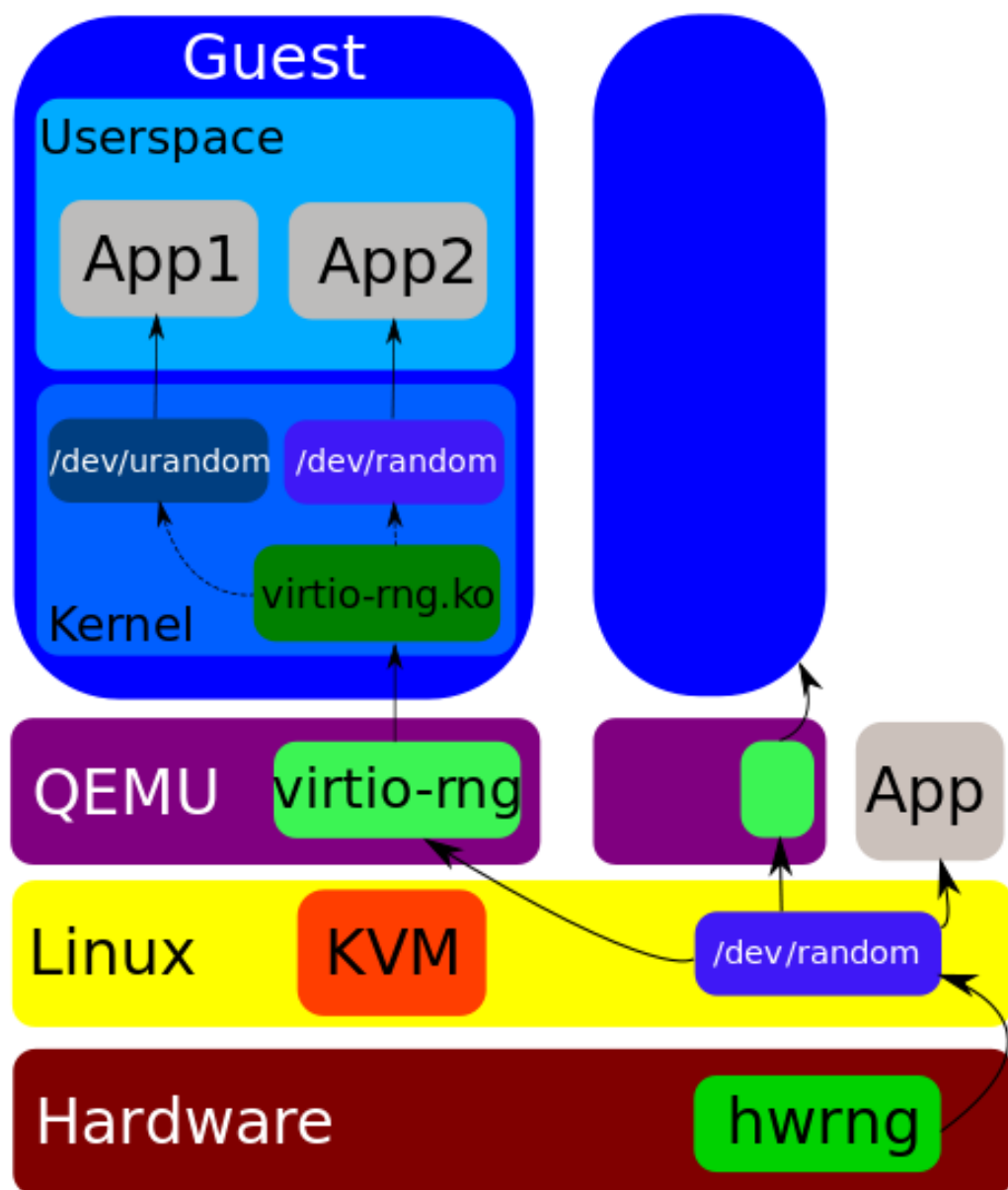


図4.3 virtio RNG デバイス

以前は Red Hat Enterprise Linux 7.0 および Red Hat Enterprise Linux 6 ゲストは、rngd ユーザー空間デーモン経由でホストからのエントロピーを使用することができました。Red Hat Enterprise Linux のインストールごとに、手動でデーモンを設定する必要があります。Red Hat Enterprise Linux 7.1 では、手動の手順がなくなり、プロセスをシームレスかつ自動化しています。rngd を使用する必要がなくなり、ご利用可能なエントロピーが特定のしきい値以下の場合には、ホストのカーネル自体がホストからエントロピーを取得します。ゲストのカーネルは、要求が入るとすぐにアプリケーションで利用できるように乱数を生成できるようになります。

Red Hat Enterprise Linux のインストーラー **Anaconda** には、インストーラーイメージに**virtio-rng** モジュールが含まれるようになり、Red Hat Enterprise Linux インストール時にホストのエントロピーが提供されます。

4.11. TLS 設定の強化

TLS (トランスポート層セキュリティ) は、ネットワーク通信をセキュアにするために使用する暗号化プロトコルです。優先する鍵交換プロトコル、認証方法、および暗号化アルゴリズムを設定することでシステムのセキュリティ設定を強化する際には、サポートするクライアントの範囲が広ければ広い

ほど、セキュリティーのレベルが低くなることを認識しておく必要があります。反対に、セキュリティー設定を厳密にすると、クライアントとの互換性が制限され、システムからロックアウトされるユーザーが出てくる可能性もあります。可能な限り厳密な設定を目指し、互換性のために必要な場合にのみ、これを緩めるようにしてください。

Red Hat Enterprise Linux 7 に含まれるライブラリーが提供するデフォルト設定は、ほとんどの導入において十分に安全なものです。**TLS** 実装は、可能な場合はセキュアなアルゴリズムを使用する一方で、レガシークライアントまたはサーバーとの接続を妨げません。セキュアなアルゴリズムやプロトコルをサポートしていないレガシーのクライアントやサーバーの接続が期待できない場合やそれらの接続が許可されない場合に、厳密なセキュリティー要件の環境で本セクションで説明されている強化設定を適用してください。

4.11.1. 有効にするアルゴリズムの選択

選択して設定する必要があるコンポーネントがいくつかあります。以下で説明するものはすべて、その設定結果（つまり、クライアントにおけるサポートレベル）やシステム上でソリューションが持つコンピューターのデマンドに直接影響します。

プロトコルのバージョン

最新バージョンの **TLS** は、すぐれたセキュリティーメカニズムを提供します。古いバージョンの **TLS**（さらに **SSL**）のサポートを含める切実な理由がなければ、最新バージョンの **TLS** のみを使ってシステムが接続するようにしてください。

SSL のバージョン 2 または 3 を使った処理を許可しないでください。これらのバージョンには、セキュリティーに関する重大な脆弱性があります。**TLS** のバージョン 1.0 以上を使った処理のみを許可してください。現行の **TLS** バージョン 1.2 を常に優先するようにしてください。



注記

TLS の全バージョンのセキュリティーは現在、**TLS** 拡張機能、特定の暗号（下記参照）、および他の回避策に依存していることに注意してください。**TLS** の接続ピアはすべてセキュアな再交渉記号 ([RFC 5746](#)) を実装する必要があり、圧縮をサポートしていない必要があります。また、**CBC** モードの暗号 (Lucky Thirteen 攻撃) に対するタイミング攻撃を緩和する方法を実装する必要があります。**TLS v1.0** クライアントはさらに、レコード分割 (BEAST 攻撃に対する回避策) を実装する必要があります。**TLS v1.2** は、**AES-GCM**、**AES-CCM**、または **Camellia-GCM** といった既知の問題のない **認証付き暗号** (Authenticated Encryption with Associated Data : AEAD) モードの暗号をサポートしています。ここに記載された緩和策はすべて、Red Hat Enterprise Linux に含まれる暗号ライブラリーに実装されています。

プロトコルのバージョンおよび推奨される使用方法についての概要は、[表4.6「プロトコルのバージョン」](#) を参照してください。

表4.6 プロトコルのバージョン

プロトコルのバージョン	推奨される使用方法
SSL v2	使用しないでください。重大なセキュリティー上の脆弱性があります。
SSL v3	使用しないでください。重大なセキュリティー上の脆弱性があります。

プロトコルのバージョン	推奨される使用方法
TLS v1.0	必要な場合は、相互運用性目的で使します。相互運用性を保証する方法では緩和できない既知の問題があります。このため、緩和策はデフォルトでは有効になっていません。最新の暗号化スイートには対応していません。
TLS v1.1	必要な場合は、相互運用性目的で使します。既知の問題はありませんが、Red Hat Enterprise Linux の TLS 実装すべてに含まれるプロトコル修正に依存します。最新の暗号化スイートには対応していません。
TLS v1.2	推奨されるバージョンです。最新の AEAD 暗号化スイートに対応しています。

Red Hat Enterprise Linux のコンポーネントのなかには、**TLS v1.1** や **v1.2** に対応しているのに **TLS v1.0** を使用する設定になっているものもあります。これは、最新バージョンの **TLS** に対応していない可能性のある外部サービスとの最高レベルの相互運用性を達成する目的でこのようになっています。相互運用性の要件に対応して、利用可能な最高レベルの **TLS** バージョンを有効にしてください。



重要

SSL v3 の使用は推奨されません。これは安全ではなく一般の使用には適していませんが、**SSL v3** をどうしても有効にする必要がある場合は、「[stunnel の使用](#)」を参照してください。暗号化に対応していない、または旧式で安全でない暗号化モードの使用しできないサービスを使用する場合でも、**stunnel** を使用して安全に通信を暗号化する方法が説明されています。

暗号化スイート

旧式の安全でない暗号化スイートではなく、最近のより安全なものを使ってください。eNULL および aNULL 暗号化スイートは暗号化や認証をまったく提供しないので、常に無効にしてください。**RC4** や **HMAC-MD5** をベースとした暗号化スイートには重大な欠陥があるので、可能な場合はこれらも無効にしてください。いわゆる エクスポート暗号化スイートも同様にしてください。これらは意図的に弱くなっているため、侵入が容易になっています。

128 ビット未満のセキュリティしか提供しない暗号化スイートは直ちに不安というわけではありませんが、これらは短期間の使用に考慮すべきではありません。128 ビット以上のセキュリティを使用するアルゴリズムは少なくとも数年間は解読不可能であることが期待されているので、強く推奨されます。**3DES** 暗号は 168 ビットの使用といわれていますが、実際に提供されているのは 112 ビットのセキュリティであることに注意してください。

(perfect) forward secrecy (PFS) をサポートしている暗号化スイートを常に優先させてください。PFS は、サーバー鍵が漏れたとしても、暗号化されたデータの秘密性は確保されます。これにより、すばやい **RSA** 鍵の交換がなくなる一方、**ECDHE** および **DHE** の使用が可能になります。これら 2 つのうちでは、**ECDHE** の方がより速いため、優先される選択肢となります。

AES-GCM などの **AEAD** 暗号はパディングオラクル攻撃 (padding oracle attacks) に対して脆弱ではないため、**CBC** モードの暗号よりも優先してください。さらに多くの場合、**AES-GCM** は特にハードウェアに **AES** 用の暗号化アクセラレーターがある場合、**CBC** モードの **AES** よりも高速です。

また、**ECDSA** 証明書を使って **ECDHE** 鍵交換を使用する際は、純粋な **RSA** 鍵交換よりも速くなります。レガシークライアント用のサポートを提供するには、サーバー上に証明書と鍵のペアを 2 つインストールします。ひとつは **ECDSA** 鍵 (新規クライアント用) で、もうひとつは **RSA** 鍵 (レガシークライアント用) です。

公開鍵の長さ

RSA 鍵を使用する際は常に、少なくとも **SHA-256** で署名された 最低 3072 ビットの鍵の長さを優先させます。これは、真の 128 ビットのセキュリティーでは十分な大きさです。



警告

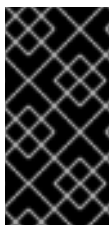
システムのセキュリティー強度は、チェーンの中の最も弱いリンクが示すものと同じであるということを念頭に置いてください。たとえば、強力な暗号化だけでは不十分なセキュリティーは保証されません。鍵と証明書も同様に重要で、**認証機関 (CA)** が鍵の署名に使用するハッシュ機能と鍵もまた重要になります。

4.11.2. TLS 実装の使用

Red Hat Enterprise Linux 7 には、完全機能の **TLS** 実装が同梱されています。本セクションでは、**OpenSSL** および **GnuTLS** の設定を説明します。個別アプリケーションでの **TLS** サポートの設定方法については、「[特定アプリケーションの設定](#)」を参照してください。

利用可能な **TLS** 実装は、各種の暗号化スイートをサポートします。これらのスイートは、**TLS** でセキュア化された通信の確立および使用時に一緒に送られる全要素を定義します。

「[有効にするアルゴリズムの選択](#)」で示された推奨事項を検討するとともに、各種の実装で含まれているツールを使って、ご自分のユースケースにとって最善のセキュリティーを提供する暗号化スイートを一覧表示、指定してください。そこでできた暗号化スイートを使って、各アプリケーションが接続を処理してセキュア化することができます。



重要

使用する **TLS** 実装またはその実装を利用するアプリケーションが更新またはアップグレードされた後は、必ず設定をチェックしてください。新しいバージョンは、ユーザーが有効化を希望せずかつ使用中の設定では無効にされない、新たな暗号化スイートを導入する場合があります。

4.11.2.1. OpenSSL での暗号化スイートの使用

OpenSSL is a toolkit and a cryptography library that support the **SSL** and **TLS** protocols. On Red Hat Enterprise Linux 7, a configuration file is provided at `/etc/pki/tls/openssl.cnf`. The format of this configuration file is described in `config(1)`. See also 「[OpenSSL の設定](#)」.

インストール済みの **OpenSSL** でサポートされている暗号化スイートすべてを一覧表示するには、以下のように **openssl** コマンドで **ciphers** サブコマンドを実行します。

```
~]$ openssl ciphers -v 'ALL:COMPLEMENTOFALL'
```

(**OpenSSL** ドキュメンテーションでは *cipher strings* および *keywords* と呼ばれる) 他のパラメーターを **ciphers** コマンドに渡して出力を絞ります。特別なキーワードを使うと、特定の条件を満たすスイートのみを一覧表示することができます。たとえば、**HIGH** グループに属するスイートのみを一覧表示するには、以下のコマンドを実行します。

```
~]$ openssl ciphers -v 'HIGH'
```

See the ciphers(1) manual page for a list of available keywords and cipher strings.

「有効にするアルゴリズムの選択」の推奨事項を満たす暗号化スイートを一覧表示するには、以下のようコマンドを実行します。

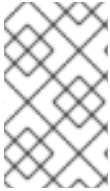
```
~]$ openssl ciphers -v 'kEECDH+aECDSA+AES:kEECDH+AES+aRSA:kEDH+aRSA+AES' |
column -t
ECDHE - ECDSA - AES256 - GCM - SHA384  TLSv1.2  Kx=ECDH  Au=ECDSA  Enc=AESGCM(256)
Mac=AEAD
ECDHE - ECDSA - AES256 - SHA384      TLSv1.2  Kx=ECDH  Au=ECDSA  Enc=AES(256)
Mac=SHA384
ECDHE - ECDSA - AES256 - SHA          SSLv3    Kx=ECDH  Au=ECDSA  Enc=AES(256)
Mac=SHA1
ECDHE - ECDSA - AES128 - GCM - SHA256 TLSv1.2  Kx=ECDH  Au=ECDSA  Enc=AESGCM(128)
Mac=AEAD
ECDHE - ECDSA - AES128 - SHA256      TLSv1.2  Kx=ECDH  Au=ECDSA  Enc=AES(128)
Mac=SHA256
ECDHE - ECDSA - AES128 - SHA          SSLv3    Kx=ECDH  Au=ECDSA  Enc=AES(128)
Mac=SHA1
ECDHE - RSA - AES256 - GCM - SHA384  TLSv1.2  Kx=ECDH  Au=RSA    Enc=AESGCM(256)
Mac=AEAD
ECDHE - RSA - AES256 - SHA384        TLSv1.2  Kx=ECDH  Au=RSA    Enc=AES(256)
Mac=SHA384
ECDHE - RSA - AES256 - SHA            SSLv3    Kx=ECDH  Au=RSA    Enc=AES(256)
Mac=SHA1
ECDHE - RSA - AES128 - GCM - SHA256  TLSv1.2  Kx=ECDH  Au=RSA    Enc=AESGCM(128)
Mac=AEAD
ECDHE - RSA - AES128 - SHA256        TLSv1.2  Kx=ECDH  Au=RSA    Enc=AES(128)
Mac=SHA256
ECDHE - RSA - AES128 - SHA            SSLv3    Kx=ECDH  Au=RSA    Enc=AES(128)
Mac=SHA1
DHE - RSA - AES256 - GCM - SHA384    TLSv1.2  Kx=DH    Au=RSA    Enc=AESGCM(256)
Mac=AEAD
DHE - RSA - AES256 - SHA256          TLSv1.2  Kx=DH    Au=RSA    Enc=AES(256)
Mac=SHA256
DHE - RSA - AES256 - SHA              SSLv3    Kx=DH    Au=RSA    Enc=AES(256)
Mac=SHA1
DHE - RSA - AES128 - GCM - SHA256    TLSv1.2  Kx=DH    Au=RSA    Enc=AESGCM(128)
Mac=AEAD
DHE - RSA - AES128 - SHA256          TLSv1.2  Kx=DH    Au=RSA    Enc=AES(128)
Mac=SHA256
DHE - RSA - AES128 - SHA              SSLv3    Kx=DH    Au=RSA    Enc=AES(128)
Mac=SHA1
```

上記のコマンドはセキュアでない暗号をすべて省略し、**ephemeral elliptic curve Diffie-Hellman** 鍵交換と **ECDSA** 暗号を優先します。また、**RSA** 鍵交換も省略します (*perfect forward secrecy* が保証されます)。

これはやや厳密な設定であることに注意してください。現実には条件を多少緩和して、より広い範囲のクライアントとの互換性を可能にする必要があるかもしれません。

4.11.2.2. GnuTLS での暗号化スイートの使用

GnuTLS は、**SSL** および **TLS** の各プロトコルとそれに関連する技術を実装する通信ライブラリーです。



注記

Red Hat Enterprise Linux 7 上の **GnuTLS** は、ほとんどのユースケースに十分なセキュリティをもたらす、最適なデフォルト設定値を提供します。特別なセキュリティ要件がない限り、提供されたデフォルト値の使用が推奨されます。

gnutls-cli コマンドを **-l** (または **--list**) オプションと実行すると、サポート対象の暗号化スイートすべてが一覧表示されます。

```
~]$ gnutls-cli -l
```

-l オプションで表示された暗号化スイート一覧を絞り込むには、ひとつ以上のパラメーター (**GnuTLS** ドキュメンテーションでは *priority strings* および *keywords* と呼ばれる) を **--priority** オプションに渡します。利用可能な priority strings の全一覧は、<http://www.gnutls.org/manual/gnutls.html#Priority-Strings> にある **GnuTLS** ドキュメンテーションを参照してください。たとえば、少なくとも 128 ビットのセキュリティを提供する暗号化スイート一覧を表示するには、以下のコマンドを実行します。

```
~]$ gnutls-cli --priority SECURE128 -l
```

「[有効にするアルゴリズムの選択](#)」の推奨事項を満たす暗号化スイートを一覧表示するには、以下のようなコマンドを実行します。

```
~]$ gnutls-cli --priority SECURE256:+SECURE128:-VERS-TLS-ALL:+VERS-TLS1.2:-RSA:-DHE-DSS:-CAMELLIA-128-CBC:-CAMELLIA-256-CBC -l
Cipher suites for SECURE256:+SECURE128:-VERS-TLS-ALL:+VERS-TLS1.2:-RSA:-DHE-DSS:-CAMELLIA-128-CBC:-CAMELLIA-256-CBC
TLS_ECDHE_ECDSA_AES_256_GCM_SHA384                                0xc0, 0x2c
TLS1.2
TLS_ECDHE_ECDSA_AES_256_CBC_SHA384                                0xc0, 0x24
TLS1.2
TLS_ECDHE_ECDSA_AES_256_CBC_SHA1                                  0xc0, 0x0a
SSL3.0
TLS_ECDHE_ECDSA_AES_128_GCM_SHA256                                0xc0, 0x2b
TLS1.2
TLS_ECDHE_ECDSA_AES_128_CBC_SHA256                                0xc0, 0x23
TLS1.2
TLS_ECDHE_ECDSA_AES_128_CBC_SHA1                                  0xc0, 0x09
SSL3.0
TLS_ECDHE_RSA_AES_256_GCM_SHA384                                  0xc0, 0x30
TLS1.2
TLS_ECDHE_RSA_AES_256_CBC_SHA1                                    0xc0, 0x14
SSL3.0
TLS_ECDHE_RSA_AES_128_GCM_SHA256                                  0xc0, 0x2f
TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256                                  0xc0, 0x27
TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA1                                    0xc0, 0x13
SSL3.0
TLS_DHE_RSA_AES_256_CBC_SHA256                                    0x00, 0x6b
TLS1.2
TLS_DHE_RSA_AES_256_CBC_SHA1                                      0x00, 0x39
SSL3.0
TLS_DHE_RSA_AES_128_GCM_SHA256                                    0x00, 0x9e
TLS1.2
```



```

TLS_DHE_RSA_AES_128_CBC_SHA256                                0x00, 0x67
TLS1.2
TLS_DHE_RSA_AES_128_CBC_SHA1                                  0x00, 0x33
SSL3.0

Certificate types: CTYPE-X.509
Protocols: VERS-TLS1.2
Compression: COMP=NULL
Elliptic curves: CURVE-SECP384R1, CURVE-SECP521R1, CURVE-SECP256R1
PK-signatures: SIGN-RSA-SHA384, SIGN-ECDSA-SHA384, SIGN-RSA-SHA512, SIGN-
ECDSA-SHA512, SIGN-RSA-SHA256, SIGN-DSA-SHA256, SIGN-ECDSA-SHA256

```

上記のコマンドは、出力を 128 ビット以上のセキュリティがある暗号に絞り込み、より強力なものを優先しています。また、**RSA** 鍵交換と **DSS** 認証を禁止しています。

これはやや厳密な設定であることに注意してください。現実には条件を多少緩和して、より広い範囲のクライアントとの互換性を可能にする必要があるかもしれません。

4.11.3. 特定アプリケーションの設定

アプリケーションはそれぞれ、**TLS** 用に個別の設定メカニズムを提供します。本セクションでは、最も一般的に使用されているサーバーアプリケーションが使用する **TLS** 関連の設定ファイルについて説明し、よくある説明例を示します。

いずれの設定を選択しても、サーバーアプリケーションが *サーバー側の暗号命令* を強制し、使用される暗号化スイートが必ずユーザー設定の命令で決定されるようにしてください。

4.11.3.1. Apache HTTP サーバーの設定

Apache HTTP Server は、**TLS** に **OpenSSL** と **NSS** の両方のライブラリーを使用できます。選択した **TLS** ライブラリーによって、**mod_ssl** か **mod_nss** のモジュールをインストールする必要があります (その名前の付いたパッケージが提供)。たとえば、**OpenSSL mod_ssl** モジュールを提供するパッケージをインストールするには、root で以下のコマンドを実行します。

```
~]# yum install mod_ssl
```

mod_ssl パッケージは **/etc/httpd/conf.d/ssl.conf** 設定ファイルをインストールし、これを使うと **Apache HTTP Server** の **TLS** 関連の設定を修正できます。同様に、**mod_nss** パッケージは **/etc/httpd/conf.d/nss.conf** 設定ファイルをインストールします。

httpd-manual パッケージをインストールして **Apache HTTP Server** の完全なドキュメンテーションを取得します。これには、**TLS** 設定が含まれます。**/etc/httpd/conf.d/ssl.conf** 設定ファイルで利用可能なディレクティブの詳細は、/usr/share/httpd/manual/mod/mod_ssl.html で説明されています。各種設定の例は、/usr/share/httpd/manual/ssl/ssl_howto.html で確認できます。

/etc/httpd/conf.d/ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の 3 つのディレクティブを確認してください。

SSLProtocol

許可する **TLS** (または **SSL**) のバージョンを指定するディレクティブです。

SSLCipherSuite

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

SSLSslHonorCipherOrder

コメントを解除して、このディレクティブを **on** に設定すると、接続先のクライアントが指定された暗号化の命令に従います。

例を示します。

```
SSLProtocol all -SSLv2 -SSLv3
SSLCipherSuite HIGH:!aNULL:!MD5
SSLSslHonorCipherOrder on
```

上記の設定は最小限のものであり、「[有効にするアルゴリズムの選択](#)」にある推奨事項にしたがうことでさらに強化できることに注意してください。

mod_nss モジュールを設定、使用するには、**/etc/httpd/conf.d/nss.conf** 設定ファイルを修正します。**mod_nss** モジュールは **mod_ssl** から派生しているため、後者と多くの機能を共有しています。設定ファイルの構成だけでなく、利用可能なディレクティブも同様です。**mod_nss** ディレクティブには、**SSL** ではなく **NSS** の接頭辞が付くことに注意してください。**mod_nss** に適用できない **mod_ssl** 設定ディレクティブの一覧を含む **mod_nss** についての概要は、https://git.fedorahosted.org/cgi/mod_nss.git/plain/docs/mod_nss.html を参照してください。

4.11.3.2. Dovecot メールサーバーの設定

Dovecot メールサーバーが **TLS** を使用するように設定するには、**/etc/dovecot/conf.d/10-ssl.conf** 設定ファイルを修正します。このファイルで利用可能な基本的な設定ディレクティブの一部は、[/usr/share/doc/dovecot-2.2.10/wiki/SSL.DovecotConfiguration.txt](#) (このヘルプファイルは標準 **Dovecot** インストールに含まれています) で説明しています。

/etc/dovecot/conf.d/10-ssl.conf 設定ファイルの設定を修正する場合は、少なくとも下記の3つのディレクティブを確認してください。

ssl_protocols

許可する **TLS** (または **SSL**) のバージョンを指定するディレクティブです。

ssl_cipher_list

優先する暗号化スイートを指定する、もしくは許可しないスイートを無効にするディレクティブです。

ssl_prefer_server_ciphers

コメントを解除して、このディレクティブを **yes** に設定すると、接続先のクライアントが指定された暗号化の命令に従います。

例を示します。

```
ssl_protocols = !SSLv2 !SSLv3
ssl_cipher_list = HIGH:!aNULL:!MD5
ssl_prefer_server_ciphers = yes
```

上記の設定は最小限のものであり、「[有効にするアルゴリズムの選択](#)」にある推奨事項にしたがうことでさらに強化できることに注意してください。

4.11.4. その他の情報

TLS の設定および関連トピックについての詳細情報は、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- `config(1)` — Describes the format of the `/etc/ssl/openssl.conf` configuration file.
- `ciphers(1)` — Includes a list of available **OpenSSL** keywords and cipher strings.
- [/usr/share/httpd/manual/mod/mod_ssl.html](#) — **Apache HTTP Server** 用に `mod_ssl` が使用する `/etc/httpd/conf.d/ssl.conf` 設定ファイルで利用可能なディレクティブを詳細に説明しています。
- [/usr/share/httpd/manual/ssl/ssl_howto.html](#) — **Apache HTTP Server** 用に `mod_ssl` が使用する `/etc/httpd/conf.d/ssl.conf` 設定ファイルでの現実的な設定例が含まれています。
- [/usr/share/doc/dovecot-2.2.10/wiki/SSL.DovecotConfiguration.txt](#) — **Dovecot** メールサーバーが使用する `/etc/dovecot/conf.d/10-ssl.conf` 設定ファイルで使用可能な基本的設定ディレクティブについて説明しています。

オンラインのドキュメント

- [Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide](#) — Red Hat Enterprise Linux 7 の『SELinux User's and Administrator's Guide』では、**SELinux** の原則と、**SELinux** を設定して **Apache HTTP Server** などのさまざまなサービスで使用方法が詳細に説明されています。
- <http://tools.ietf.org/html/draft-ietf-uta-tls-bcp-00> — **TLS** および **DTLS** のセキュアな使用における推奨事項です。

関連項目

- 「[SSL/TLS](#)」では、**SSL** および **TLS** プロトコルを簡潔に説明しています。
- 「[OpenSSL の使用](#)」では、**OpenSSL** を使用して鍵を作成、管理し、証明書を生成し、ファイルを暗号化、暗号化解除する方法を説明しています。

4.12. MACSEC (IEEE 802.1AE) の使用

Media Access Control Security (MACsec) は、LAN にある全トラフィックを GCM-AES-128 アルゴリズムで暗号化および認証します。MACsec は、**IP** トラフィックだけでなく、**ARP** や **Neighbour Discovery** または **DHCP** も保護することができます。**MACsec** は、**IPsec** または **SSL** や **TLS** とは異なる階層でも機能します。他のネットワーク層のセキュリティプロトコルと **MACsec** を組み合わせて、これらの標準が提供する異なるセキュリティ保証を活用してください。

MACsec ネットワークのアーキテクチャー、ユースケースのシナリオ、設定例に関する詳しい情報は「[MACsec: a different solution to encrypt network traffic](#)」の記事を参照してください。

第5章 システム監査

Linux Audit システムは、システム上のセキュリティ関連情報を追跡する方法を提供します。事前設定ルールに基づき、Audit はシステム上で発生しているイベントについての情報をできるだけ多く記録するためのログエントリを生成します。この情報は、セキュリティポリシーの違反者と違反者によるアクションを判断する上でミッションクリティカルな環境で必須のものです。Audit は新たなセキュリティをシステムに追加するわけではありません。システム上で使われているセキュリティポリシーの侵害を発見するために使用されます。これらの侵害は、SELinux などの追加のセキュリティ対策でさらに防ぐことができます。

Audit がログファイルに記録できる情報のいくつかを、以下のリストで要約しています。

- イベントの日付と時間、タイプ、結果。
- サブジェクトとオブジェクトの機密性のラベル。
- イベントを開始したユーザーの ID とイベントの関連性。
- Audit 設定の全修正および Audit ログファイルへのアクセス試行。
- SSH、Kerberos、およびその他の認証メカニズムのすべての使用。
- `/etc/passwd` のような、信頼できるデータベースへの変更。
- システムからの情報のインポートおよびシステムへの情報のエクスポートの試行。
- ユーザー ID、サブジェクトおよびオブジェクトラベル、その他の属性に基づく include または exclude イベント。

Audit システムの使用は、多くのセキュリティ関連の証明書における要件でもあります。Audit は、以下の証明書またはコンプライアンスガイドの要件に合致するかそれらを超えるように設計されています。

- Controlled Access Protection Profile (CAPP)
- Labeled Security Protection Profile (LSPP)
- Rule Set Base Access Control (RSBAC)
- NISPOM (National Industrial Security Program Operating Manual)
- Federal Information Security Management Act (FISMA)
- Payment Card Industry — Data Security Standard (PCI-DSS)
- セキュリティ技術実装ガイド (STIG: Security Technical Implementation Guide)

Audit は以下でも認定されています。

- National Information Assurance Partnership (NIAP) および Best Security Industries (BSI) による評価。
- Red Hat Enterprise Linux 5 上での LSPP/CAPP/RSBAC/EAL4+ の認定。
- Red Hat Enterprise Linux 6 上での Operating System Protection Profile / Evaluation Assurance Level 4+ (OSPP/EAL4+) の認定。

使用例

ファイルアクセスの監視

Audit は、ファイルやディレクトリーがアクセス、修正、実行されたか、またはファイル属性が変更されたかを追跡することができます。これはたとえば、重要なファイルへのアクセスを検出し、これらのファイルが破損した場合に監査証跡を入手可能とする際に便利なものです。

システムコールの監視

Audit は、特定のシステムコールが使用されるたびにログエントリーを生成するように設定できます。これを使用すると、**settimeofday** や **clock_adjtime**、その他の時間関連のシステムコールを監視することで、システム時間への変更を追跡できます。

ユーザーが実行したコマンドの記録

Audit はファイルが実行されたかどうかを追跡できるので、特定のコマンドの実行を毎回記録するようにルールを定義することができます。たとえば、**/bin** ディレクトリー内の実行可能ファイルすべてについてルールを定義することができます。その結果できるログエントリーをユーザー ID で検索すると、ユーザーごとに実行されたコマンドの監査証跡を生成することができます。

システムのパス名実行の記録

ルールの呼び出し時にパスを inode に変換するファイルアクセスをウォッチする以外に、Audit はルール呼び出し時に存在しない場合やルール呼び出し後にファイルが置き換えられた場合でもパスの実行をウォッチできるようになりました。これにより、ルールは、プログラム実行ファイルをアップグレードした後、またはインストールされる前にも機能を継続できます。

セキュリティイベントの記録

pam_faillock 認証モジュールは、失敗したログイン試行を記録することができます。Audit で失敗したログイン試行も記録するように設定すると、ログインを試みたユーザーについての追加情報が提供されます。

イベントの検索

Audit は **ausearch** ユーティリティを提供します。これを使うと、ログエントリーをフィルターにかけ、いくつもの条件に基づく完全な監査証跡を提供することができます。

サマリーレポートの実行

aureport ユーティリティを使うと、記録されたイベントのデیلیーレポートを生成することができます。システム管理者は、このレポートを分析し、疑わしいアクティビティをさらに調べることができます。

ネットワークアクセスの監視

iptables と **ebtables** ユーティリティは Audit イベントを開始するように設定でき、これでシステム管理者はネットワークアクセスを監視できるようになります。



注記

システムのパフォーマンスは、Audit が収集する情報量によって影響される可能性があります。

5.1. AUDIT システムのアーキテクチャー

Audit システムは、ユーザースペースアプリケーションおよびユーティリティと、カーネル側のシステムコール処理という 2 つの主要パートで構成されます。カーネルコンポーネントは、ユーザースペースアプリケーションからシステムコールを受け、これを **user**、**task**、または **exit** のいずれかのフィルターで振り分けます。システムが **exclude** フィルターを通過すると、前述のフィルターの 1 つに掛けられます。このフィルターは Audit ルール設定に基づいて、システムコールを Audit デーモンに送信してさらに処理します。図5.1「Audit システムのアーキテクチャー」では、このプロセスを示しています。

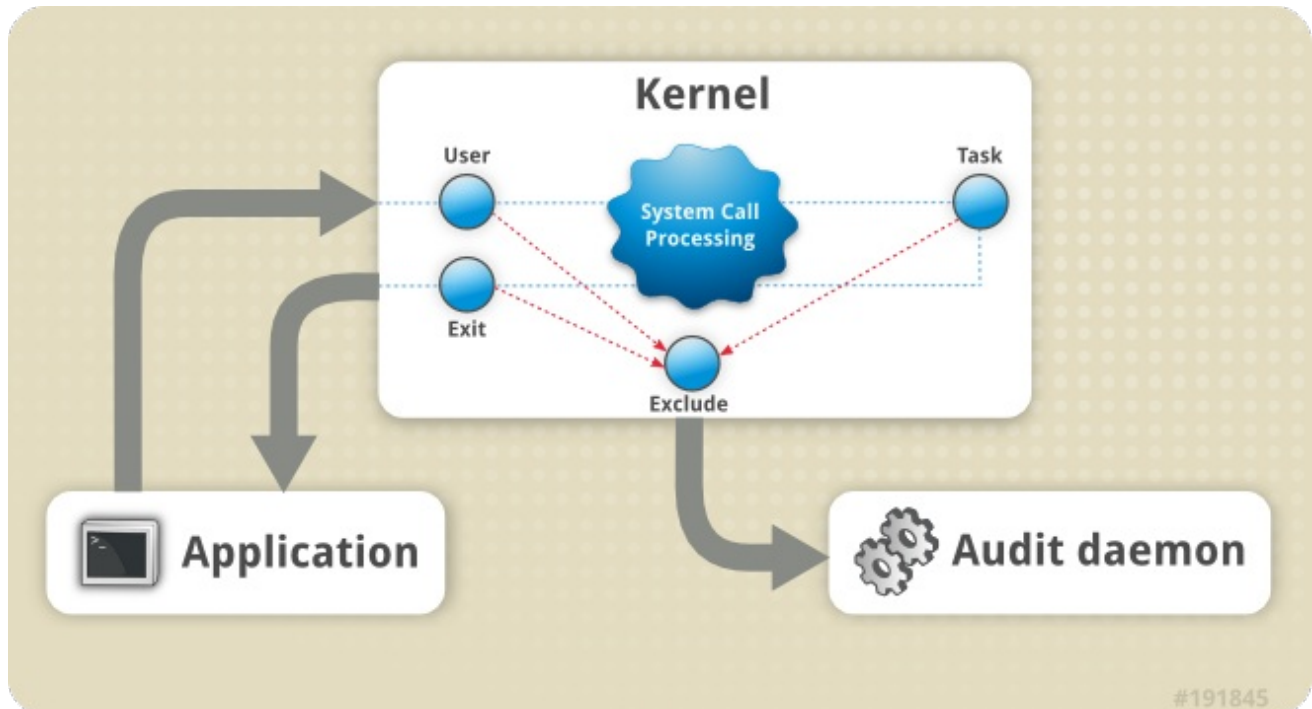


図5.1 Audit システムのアーキテクチャー

ユーザースペースの Audit デーモンはカーネルから情報を収集し、ログファイルエントリを作成します。他のユーザースペースユーティリティは、Audit デーモン、カーネル Audit コンポーネント、または Audit ログファイルと対話します。

- **audisp** — Audit ディスパッチャーデーモンは Audit デーモンと対話し、イベントを他のアプリケーションに送信してさらに処理します。このデーモンの目的は、プラグインメカニズムを提供して、リアルタイムの分析プログラムが Audit イベントと対話できるようにすることです。
- **auditctl** — Audit 制御ユーティリティはカーネル Audit コンポーネントと対話し、ルールを管理するだけでなくイベント生成プロセスの多くの設定やパラメーターも制御します。
- 残りの Audit ユーティリティは Audit ログファイルのコンテンツを入力として受け取り、ユーザーの要件に基づいて出力を生成します。たとえば、**aureport** ユーティリティは記録された全イベントのレポートを生成します。

5.2. AUDIT パッケージのインストール

Audit システムを使用するには、audit パッケージがシステムにインストールされている必要があります。audit パッケージ (audit および audit-libs) は、デフォルトで Red Hat Enterprise Linux 7 にインストールされています。これらのパッケージがインストールされていない場合は、root で以下のコマンド実行してインストールします。

```
~]# yum install audit
```

5.3. AUDIT サービスの設定

The Audit daemon can be configured in the `/etc/audit/auditd.conf` configuration file. This file consists of configuration parameters that modify the behavior of the Audit daemon. Empty lines and text following a hash sign (`#`) are ignored. A complete listing of all configuration parameters and their explanation can be found in the `audit.conf(5)` man page.

5.3.1. セキュアな環境用の `auditd` 設定

デフォルトの `auditd` 設定はほとんどの環境に適しているはずですが、使用環境が厳密なセキュリティポリシーに対応する必要がある場合には、Audit デーモンの設定には、以下の設定が推奨されます。

- Audit ログファイルを格納するディレクトリー (通常 `/var/log/audit/`) を、別個のパーティションに置きます。これにより、他のプロセスがこのディレクトリーのスペースを使うことを防ぎます。また、Audit デーモン用に残りのスペースがどれだけあるか、正確に分かるようになります。
- 単一 Audit ログファイルの最大サイズを指定する `max_log_file` パラメーターは、Audit ログファイルを格納するパーティション上で利用可能なスペースを最大活用するように設定する必要があります。
- `max_log_file_action` パラメーターは、`max_log_file` で設定された上限に達した際に取られるアクションを決定します。これは、`keep_logs` に設定して、Audit ログファイルが上書きされないようにすべきです。
- `space_left` パラメーターは、ここで指定した値にディスク上の残りの空きスペースが達すると、`space_left_action` パラメーターで定義したアクションが起動します。`space_left` で指定するディスクの残り空きスペースの値は大きなものに設定し、これがなくなる前に管理者が対応してスペースを利用可能にできるようにすべきです。`space_left` の値は、Audit ログファイルが生成されるレートによって異なります。
- `space_left_action` パラメーターは、`email` または `exec` に設定して適切な通知方法にすることが推奨されます。
- `admin_space_left` パラメーターは、ここで指定した値にディスク上の残りの空きスペースが達すると、`admin_space_left_action` パラメーターで定義したアクションが起動します。`admin_space_left` の値は大きなものに設定し、管理者が実行するアクションをログ記録できるようにする必要があります。
- `admin_space_left_action` は `single` に設定して、システムをシングルユーザーモードにし、管理者がディスクスペースを解放できるようにする必要があります。
- `disk_full_action` パラメーターは、Audit ログファイルを格納するパーティションに空きスペースが無くなった場合に起動するアクションを指定します。このパラメーターは、`halt` または `single` のどちらかに設定する必要があります。これにより、Audit がイベントをログ記録できなくなった際に、システムがシャットダウンするか、シングルユーザーモードで稼働するようになります。
- `disk_error_action` は、Audit ログファイルがあるパーティションでエラーが検出された場合に起動するアクションを指定します。このパラメーターは、ハードウェアの機能不全処理に関するローカルのセキュリティポリシーによって、`syslog`、`single`、`halt` のいずれかに設定する必要があります。
- `flush` 設定パラメーターは、`incremental_async` に設定してください。ハードドライブと強制同期する前にディスクに送信できる記録数を決める `freq` パラメーターと合わせて機能し

ます。**freq** パラメーターは、**100** に設定してください。これらのパラメーターにより、アクティビティーが集中した場合に高いパフォーマンスを保ちつつ、Audit イベントデータがディスク上のログファイルと確実に同期されるようにします。

残りの設定オプションは、ローカルのセキュリティーポリシーにあわせて設定します。

5.4. AUDIT サービスの起動

auditd を適切に設定したら、サービスを起動して Audit 情報を収集し、ログファイルに保存します。root で以下のコマンドを実行して **auditd** を起動します。

```
~]# service auditd start
```



注記

service コマンドは、**auditd** デーモンと正しく対話する唯一の方法です。**audit** の値が正しく記録されるように **service** コマンドを使用する必要があります。**systemctl** コマンドは、**enable** と **status** の 2 つのアクションにのみ使用できます。

オプションでは、root で以下のコマンドを実行すると、**auditd** がブート時に起動するように設定できます。

```
~]# systemctl enable auditd
```

auditd では、**service auditd action** コマンドを使用して、他のアクションが実行できます。ここでの **action** は、以下のいずれかになります。

- **stop** — **auditd** を停止します。
- **restart** — **auditd** を再起動します。
- **reload** または **force-reload** — **auditd** の設定を **/etc/audit/auditd.conf** ファイルから再読み込みします。
- **rotate** — **/var/log/audit/** ディレクトリー内のログファイルを交代させます。
- **resume** — Audit イベントのログが一旦停止された後、再開します。たとえば、Audit ログファイルがあるディスクパーティションの未使用スペースが十分でない場合などです。
- **condrestart** または **try-restart** — **auditd** がすでに実行中の場合にのみ、これを再起動します。
- **status** — 実行中の **auditd** のステータスを表示します。

5.5. AUDIT ルールの定義

Audit システムは、ログファイルにキャプチャーされるものを定義するルールセットで作動します。指定可能な Audit ルールには、以下の 3 つのタイプがあります。

- 制御ルール — Audit システムの動作と設定のいくつかの修正が可能になります。
- ファイルシステムルール — ファイルウォッチとも呼ばれ、特定のファイルまたはディレクトリーへのアクセスの監査が可能になります。

- システムコールルール — 特定のプログラムが実行するシステムコールのログ記録が可能になります。

Audit ルールは、**auditctl** ユーティリティのコマンドラインで指定するか（この方法で指定したルールは再起動後は維持されません）、**/etc/audit/audit.rules** ファイルに書き込んで指定することができます。以下の 2 つのセクションでは、Audit ルール定義における両方のアプローチをまとめています。

5.5.1. auditctl ユーティリティを使った Audit ルールの定義



注記

Audit サービスおよび Audit ログファイルと対話するすべてのコマンドは、root 権限が必要になります。これらのコマンドは、必ず root ユーザーとして実行してください。さらに、Audit サービスの設定には **CAP_AUDIT_CONTROL** が、ユーザーメッセージのロギングには **CAP_AUDIT_WRITE** が必要です。

auditctl コマンドを使うと、Audit システムの基本的な機能を制御し、どの Audit イベントをログ記録するかを決定するルールが定義できます。

制御ルールの定義

以下の制御ルールを使うと、Audit システムの動作が修正できます。

-b

カーネルにおける 既存の Audit バッファの最大値を設定します。例を示します。

```
~]# auditctl -b 8192
```

-f

重大なエラーが検出された際に実行されるアクションを設定します。例を示します。

```
~]# auditctl -f 2
```

上記の設定では、重大なエラーが発生した際にカーネルパニックが起動されます。

-e

Audit システムを有効または無効にする、もしくは設定をロックします。例を示します。

```
~]# auditctl -e 2
```

上記のコマンドは、Audit 設定をロックします。

-r

1 秒あたりに生成されるメッセージ数を設定します。例を示します。

```
~]# auditctl -r 0
```

上記の設定では、生成されるメッセージ数は制限されません。

-s

Audit システムのステータスをレポートします。例を示します。

```
~]# auditctl -s
AUDIT_STATUS: enabled=1 flag=2 pid=0 rate_limit=0 backlog_limit=8192
lost=259 backlog=0
```

-l

読み込まれている Audit ルールすべてを一覧表示します。例を示します。

```
~]# auditctl -l
-w /etc/passwd -p wa -k passwd_changes
-w /etc/selinux -p wa -k selinux_changes
-w /sbin/insmod -p x -k module_insertion
⋮
```

-D

読み込まれている Audit ルールすべてを削除します。例を示します。

```
~]# auditctl -D
No rules
```

ファイルシステムルールの定義

ファイルシステムのルールを定義するには、以下の構文を使用します。

```
auditctl -w path_to_file -p permissions -k key_name
```

ここでは、

- *path_to_file* は、監査対象のファイルもしくはディレクトリーになります。
- *permissions* は、ログ記録されるパーミッションになります。
 - **r** — ファイルまたはディレクトリーの読み取りアクセスです。
 - **w** — ファイルまたはディレクトリーの書き込みアクセスです。
 - **x** — ファイルまたはディレクトリーの実行アクセスです。
 - **a** — ファイルまたはディレクトリーの属性を変更します。
- *key_name* は、どのルールまたはルールセットが特定のログエントリーを生成したかを特定する際に役立つオプションの文字列です。

例5.1 ファイルシステムルール

/etc/passwd ファイルへのすべての書き込みアクセスとこのファイルのすべての属性変更をログ記録するルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -w /etc/passwd -p wa -k passwd_changes
```

-k オプションに続く文字列は任意のものであることに注意してください。

/etc/selinux/ ディレクトリー内の全ファイルへのすべての書き込みアクセスとこれらファイルのすべての属性変更をログ記録するルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -w /etc/selinux/ -p wa -k selinux_changes
```

Linux カーネルにモジュールを挿入する **/sbin/insmod** コマンドの実行をログ記録するルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -w /sbin/insmod -p x -k module_insertion
```

システムコールルールの定義

システムコールのルールを定義するには、以下の構文を使用します。

```
auditctl -a action,filter -S system_call -F field=value -k key_name
```

ここでは、

- *action* および *filter* は、特定のイベントがいつログ記録されるかを指定します。*action* は、**always** または **never** になります。*filter* は、イベントにどのカーネルルール適合のフィルターを適用するかを指定します。ルール適合フィルターは、**task**、**exit**、**user**、**exclude** のいずれかになります。これらフィルターの詳細については、「[Audit システムのアーキテクチャー](#)」の最初の部分を参照してください。
- *system_call* は、名前でシステムコールを指定します。システムコールの全一覧は、**/usr/include/asm/unistd_64.h** ファイルで確認できます。複数のシステムコールをひとつのグループにまとめて、**-S** オプションの後にそれらを指定することも可能です。
- *field=value* specifies additional options that further modify the rule to match events based on a specified architecture, group ID, process ID, and others. For a full listing of all available field types and their values, refer to the auditctl(8) man page.
- *key_name* は、どのルールまたはルールセットが特定のログエントリーを生成したかを特定する際に役立つオプションの文字列です。

例5.2 システムコールのルール

システムが 64 ビットアーキテクチャーを使用していて、プログラムが **adjtimex** または **settimeofday** システムコールを使用するたびにログエントリーを作成するルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
```

ユーザー ID が 1000 以上 (**-F auid!=4294967295** オプションを使用すると、ログイン UID が設定されていないユーザーが除外されます) のシステムユーザーがファイルを削除またはファイルの名前を変更するたびにログエントリーを作成するルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000 -F auid!=4294967295 -k delete
```


システムコールルールの構文を使って、ファイルシステムのルールを定義することもできます。以下のコマンドでは、**-w /etc/shadow -p wa** ファイルシステムルールに似たシステムコールのルールが作成されます。

```
~]# auditctl -a always,exit -F path=/etc/shadow -F perm=wa
```

5.5.2. 実行可能なファイルルールの定義

実行可能なファイルルールを定義するには、以下の構文を使用します。

```
auditctl -a action,filter [ -F arch=cpu -S system_call] -F  
exe=path_to_executable_file -k key_name
```

ここでは、

- *action* および *filter* は、特定のイベントがいつログ記録されるかを指定します。*action* は、**always** または **never** になります。*filter* は、イベントにどのカーネルルール適合のフィルターを適用するかを指定します。ルール適合フィルターは、**task**、**exit**、**user**、**exclude** のいずれかになります。これらフィルターの詳細については、「[Audit システムのアーキテクチャー](#)」の最初の部分を参照してください。
- *system_call* は、名前でシステムコールを指定します。システムコールの全一覧は、**/usr/include/asm/unistd_64.h** ファイルで確認できます。複数のシステムコールをひとつのグループにまとめて、**-S** オプションの後にそれらを指定することも可能です。
- *path_to_executable_file* は、監査する実行可能ファイルへの絶対パスに置き換えてください。
- *key_name* は、どのルールまたはルールセットが特定のログエントリを生成したかを特定する際に役立つオプションの文字列です。

例5.3 実行可能なファイルルール

/bin/id プログラムのすべ手の実行をロギングするルールを定義するには、以下のコマンドを実行します。

```
~]# auditctl -F exe=/bin/id -S execve -k execution_bin_id
```

5.5.3. 永続的な **Audit** ルールの定義と **/etc/audit/audit.rules** ファイルでの制御

再起動後も維持される Audit ルールを定義するには、そのルールを **/etc/audit/audit.rules** ファイルに含める必要があります。このファイルは、同じ **auditctl** コマンドライン構文を使ってルールを指定します。空の行やハッシュ記号 (**#**) に続くテキストは無視されます。

auditctl コマンドで **-R** オプションを使うと、指定されたファイルからルールを読み取ることもできます。例を示します。

```
~]# auditctl -R /usr/share/doc/audit/rules/30-stig.rules
```

制御ルールの定義

ファイルに含めることができるのは、**-b**、**-D**、**-e**、**-f**、**-r**、および **--loginuid-immutable** の制御ルールのみで、これらは Audit の動作を修正します。これらのオプションに関する詳細は、「[制御ルールの定義](#)」を参照してください。

例5.4 audit.rules で制御ルール

```
# Delete all previous rules
-D

# Set buffer size
-b 8192

# Make the configuration immutable -- reboot is required to change audit rules
-e 2

# Panic when a failure occurs
-f 2

# Generate at most 100 audit messages per second
-r 100

# Make login UID immutable once it is set (may break containers)
--loginuid-immutable 1
```

ファイルシステムおよびシステムコールのルールの定義

ファイルシステムおよびシステムコールのルールは、**auditctl** 構文を使って定義します。「[auditctl ユーティリティーを使った Audit ルールの定義](#)」の例は、以下のルールファイルのようになります。

例5.5 audit.rules でファイルシステムおよびシステムコールのルール

```
-w /etc/passwd -p wa -k passwd_changes
-w /etc/selinux/ -p wa -k selinux_changes
-w /sbin/insmod -p x -k module_insertion

-a always,exit -F arch=b64 -S adjtimex -S settimeofday -k time_change
-a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=1000
-F auid!=4294967295 -k delete
```

事前設定ルールのファイル

/usr/share/doc/audit/rules/ ディレクトリーでは、audit パッケージが各種の証明書基準にしたがって事前設定ルールのファイル一式を提供しています。

- **30-nispom.rules** — NISPOM (National Industrial Security Program Operating Manual) の第 8 章で指定されている要件を満たす Audit ルール設定です。
- **30-pci-dss-v31.rules** — Payment Card Industry Data Security Standard (PCI DSS) v3.1 で指定されている要件を満たす Audit RUEHL 設定です。
- **30-stig.rules** — セキュリティー技術実装ガイド (STIG: Security Technical Implementation Guide) で設定された要件を満たす Audit ルール設定です。

これらの設定ファイルを使用するには、オリジナルの **/etc/audit/audit.rules** ファイルのバックアップを作成し、選択する設定ファイルをこの **/etc/audit/audit.rules** にコピーします。

```
~]# cp /etc/audit/audit.rules /etc/audit/audit.rules_backup
~]# cp /usr/share/doc/audit/rules/30-stig.rules /etc/audit/audit.rules
```



注記

Audit ルールには、順序付けができるように番号指定のスキームがあります。名前付けスキームに関する詳しい情報は **/usr/share/doc/audit/rules/README-rules** ファイルを参照してください。

5.6. AUDIT ログファイルについて

デフォルトでは、Audit システムはログエントリを **/var/log/audit/audit.log** ファイルに保存します。ログローテーションが有効になっていれば、ローテーションされた **audit.log** ファイルは同じディレクトリに保存されます。

下記の Audit ルールは、**/etc/ssh/sshd_config** ファイルの読み取りまたは修正の試行をすべてログ記録します。

```
-w /etc/ssh/sshd_config -p warx -k sshd_config
```

auditd デーモンが実行中の場合、以下のコマンドが Audit ログファイルに新規イベントを作成します。

```
~]# cat /etc/ssh/sshd_config
```

このイベントは **audit.log** ファイル内で以下のように見えます。

```
type=SYSCALL msg=audit(1364481363.243:24287): arch=c000003e syscall=2
success=no exit=-13 a0=7ffffd19c5592 a1=0 a2=7ffffd19c4b50 a3=a items=1
ppid=2686 pid=3538 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000
fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=1 comm="cat"
exe="/bin/cat" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
key="sshd_config"
type=CWD msg=audit(1364481363.243:24287): cwd="/home/shadowman"
type=PATH msg=audit(1364481363.243:24287): item=0
name="/etc/ssh/sshd_config" inode=409248 dev=fd:00 mode=0100600 ouid=0
ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0
```

上記のイベントは、3つの記録(それぞれ **type=** キーワードで開始)で構成されており、これらは同じタイムスタンプとシリアル番号を共有しています。各記録は、いくつかの **name=value** のペアで構成されており、これらは空白またはコンマ区切りとなっています。以下で上記のイベントを詳しく分析します。

最初の記録

type=SYSCALL

type フィールドには、記録のタイプが記載されます。この例では、この記録がカーネルへのシステムコールで開始されたことを **SYSCALL** の値が示しています。

タイプの値およびそれらの説明に関する全一覧は、「[Audit 記録のタイプ](#)」を参照してください。

msg=audit(1364481363.243:24287):

msg フィールドでは以下を記録しています。

- **audit(time_stamp:ID)** の形式で、記録のタイムスタンプと一意の ID。複数の記録は、それらが同一の Audit イベントの一部として生成されている場合は、同じタイムスタンプと ID を共有できます。
- 各種のイベント固有の **name=value** ペア。これは、カーネルもしくはユーザースペースアプリケーションが提供します。

arch=c000003e

arch フィールドには、システムの CPU アーキテクチャーについての情報が含まれます。**c000003e** という値は、十六進法でエンコードされています。**ausearch** コマンドで Audit 記録を検索する際は、**-i** または **--interpret** オプションを使うと自動的に十六進法の値がヒューマンリーダブルなものに変換されます。**c000003e** の値は、**x86_64** として解釈されます。

syscall=2

The **syscall** field records the type of the system call that was sent to the kernel. The value, **2**, can be matched with its human-readable equivalent in the **/usr/include/asm/unistd_64.h** file. In this case, **2** is the **open** system call. Note that the **ausyscall** utility allows you to convert system call numbers to their human-readable equivalents. Use the **ausyscall --dump** command to display a listing of all system calls along with their numbers. For more information, refer to the **ausyscall(8)** man page.

success=no

success フィールドは、その特定のイベントで記録されたシステムコールが成功したかどうかを記録します。このケースでは、コールは成功しませんでした。

exit=-13

exit フィールドには、システムコールが返した終了コードを指定する値が含まれています。この値は、システムコールによって異なります。以下のコマンドを使うと、この値をヒューマンリーダブルなものに変換できます。**ausearch --interpret --exit -13** (Audit ログに終了コード **-13** で失敗したイベントが含まれていることが前提)。

a0=7fffd19c5592, a1=0, a2=7fffd19c5592, a3=a

a0 から **a3** までのフィールドは、このイベントにおけるシステムコールの最初の 4 つの引数を十六進法で記録します。これらの引数は、使用されているシステムコールによって異なります。**ausearch** ユーティリティーでこれらを変換できます。

items=1

items フィールドには、イベントのパス記録の数が含まれています。

ppid=2686

ppid フィールドは、親プロセス ID (PPID) を記録します。このケースでは、**2686** は **bash** プロセスの PPID でした。

pid=3538

pid フィールドは、プロセス ID (PID) を記録します。このケースでは、**3538** は、**cat** プロセスの PID でした。

auid=1000

audit は、loginuid である Audit ユーザー ID を記録します。この ID は、ログイン時にユーザーに割り当てられ、ユーザーの ID が変更された後でもすべてのプロセスに引き継がれます (たとえば、**su - john** コマンドでユーザーアカウントを切り替えた場合)。

uid=1000

uid フィールドは、分析されているプロセスを開始したユーザーのユーザー ID を記録します。ユーザー ID は、以下のコマンドでユーザー名に変換できます。**ausearch -i --uid UID**。このケースでは、**1000** は **shadowman** のユーザー ID です。

gid=1000

gid フィールドは、分析されているプロセスを開始したユーザーのグループ ID を記録します。

euid=1000

euid フィールドは、分析されているプロセスを開始したユーザーの実効ユーザー ID を記録します。

suid=1000

suid フィールドは、分析されているプロセスを開始したユーザーのセットユーザー ID を記録します。

fsuid=1000

fsuid フィールドは、分析されているプロセスを開始したユーザーのファイルシステムユーザー ID を記録します。

egid=1000

egid フィールドは、分析されているプロセスを開始したユーザーの実効グループ ID を記録します。

sgid=1000

sgid フィールドは、分析されているプロセスを開始したユーザーのセットグループ ID を記録します。

fsgid=1000

fsgid フィールドは、分析されているプロセスを開始したユーザーのファイルシステムグループ ID を記録します。

tty=pts0

tty フィールドは、分析されているプロセスが開始されたターミナルを記録します。

ses=1

ses フィールドは、分析されているプロセスが開始されたセッションのセッション ID を記録します。

comm="cat"

comm フィールドは、分析されているプロセスを開始するために使用されたコマンドのコマンドライン名を記録します。このケースでは、**cat** コマンドを使ってこの Audit イベントが開始されました。

exe="/bin/cat"

exe フィールドは、分析されているプロセスを開始するために使用された実行可能ファイルへのパスを記録します。

subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

subj フィールドは、分析されているプロセスの実行時にラベル付けされた SELinux コンテンツを記録します。

key="sshd_config"

key フィールドは、Audit ログでこのイベントを生成したルールに関連付けられている管理者による定義の文字列を記録します。

2 番目の記録

type=CWD

2 番目の記録では、**type** フィールドの値は、現在の作業ディレクトリーである **CWD** になります。このタイプは、最初の記録で指定されているシステムコールを開始したプロセスが実行されている作業ディレクトリーの記録に使われます。

この記録の目的は、相対パスが関連する **PATH** 記録に保存された場合に、現行プロセスの位置を記録することにあります。この方法により、このプロセスの絶対パスが再構築可能になります。

msg=audit(1364481363.243:24287)

msg フィールドは、最初の記録と同じタイムスタンプと ID の値になります。

cwd="/home/shadowman"

cwd フィールドは、システムコールが開始されたディレクトリーへのパスになります。

3 番目の記録

type=PATH

3 番目の記録では、**type** フィールドの値は **PATH** になります。Audit イベントには、システムコールに引数として渡されたすべてのパスに **PATH** タイプの記録が含まれます。この Audit イベントでは、1 つのパス (**/etc/ssh/sshd_config**) のみが引数として使われています。

msg=audit(1364481363.243:24287):

msg フィールドは、最初と 2 番目の記録と同じタイムスタンプと ID の値になります。

item=0

item フィールドは、**SYSCALL** タイプの記録で参照されたアイテム総数のうち、どのアイテムが現行の記録かを示します。この数字はゼロベースで、**0** は最初のアイテムであることを示します。

name="/etc/ssh/sshd_config"

name フィールドは、システムコールに引数として渡されたファイルまたはディレクトリーへの完全パスを記録します。このケースでは、**/etc/ssh/sshd_config** ファイルとなります。

inode=409248

inode フィールドには、このイベントで記録されたファイルまたはディレクトリーに関連する inode 番号が含まれます。以下のコマンドを実行すると、**409248** inode 番号に関連するファイルまたはディレクトリーが表示されます。

```
~]# find / -inum 409248 -print
/etc/ssh/sshd_config
```

dev=fd:00

dev フィールドは、このイベントで記録されたファイルまたはディレクトリーを含むデバイスのマイナーおよびメジャー ID を示します。このケースでは、値は **/dev/fd/0** デバイスを示しています。

mode=0100600

mode フィールドは、ファイルまたはディレクトリーパーミッションを数字表記でエンコードされた形で記録します。このケースでは、**0100600** は **-rw-----** に変換可能です。つまり、root ユーザーにのみ、**/etc/ssh/sshd_config** ファイルへの読み取りおよび書き込みパーミッションがあることを示します。

ouid=0

ouid フィールドは、オブジェクトの所有者のユーザー ID を記録します。

ogid=0

ogid フィールドは、オブジェクトの所有者のグループ ID を記録します。

rdev=00:00

rdev フィールドには、特定のファイルにのみ使われる、記録されたデバイス識別子が含まれます。このケースでは、記録されたファイルは通常のファイルなので、使われていません。

obj=system_u:object_r:etc_t:s0

obj フィールドは、実行時に記録されたファイルまたはディレクトリーにラベル付けされる SELinux コンテキストを記録します。

上記で分析された Audit イベントには、イベントに含まれる可能性のあるフィールドのサブセットのみが含まれています。イベントフィールドおよびそれらの説明の全一覧は、「[Audit イベントフィールド](#)」を参照してください。イベントのタイプおよびそれらの説明の全一覧は、「[Audit 記録のタイプ](#)」を参照してください。

例5.6 追加の audit.log イベント

以下の Audit イベントでは、**auditd** デーモンが正常にスタートしたことを記録しています。**ver** フィールドでは、開始された Audit デーモンのバージョンを示しています。

```
type=DAEMON_START msg=audit(1363713609.192:5426): auditd start, ver=2.2
format=raw kernel=2.6.32-358.2.1.el6.x86_64 auid=1000 pid=4979
subj=unconfined_u:system_r:auditd_t:s0 res=success
```

以下の Audit イベントでは、UID が 1000 のユーザーが root ユーザーとしてログインを試み、失敗したことを記録しています。

```
type=USER_AUTH msg=audit(1364475353.159:24270): user pid=3280 uid=1000
auid=1000 ses=1 subj=unconfined_u:unconfined_r:unconfined_t:s0-
s0:c0.c1023 msg='op=PAM:authentication acct="root" exe="/bin/su"
hostname=? addr=? terminal=pts/0 res=failed'
```

5.7. AUDIT ログファイルの検索

ausearch ユーティリティーを使うと、Audit ログファイル内で特定のイベントを検索できます。デフォルトでは、**ausearch** は **/var/log/audit/audit.log** ファイルを検索します。**ausearch options -if file_name** コマンドを使うと、別のファイルを指定できます。**ausearch** コマンドで複数のオプションを渡すと、フィールドタイプの間で **AND** 演算子を使用したり、同じフィールドタイプの複数のインスタンス間で **OR** を使用したりするのに相当します。

例5.7 ausearch を使った Audit ログファイルの検索

失敗したログイン試行を **/var/log/audit/audit.log** ファイル内で検索するには、以下のコマンドを実行します。

```
~]# ausearch --message USER_LOGIN --success no --interpret
```

すべてのアカウント、グループ、役割変更を検索するには、以下のコマンドを実行します。

```
~]# ausearch -m ADD_USER -m DEL_USER -m ADD_GROUP -m USER_CHAUTHOK -m  
DEL_GROUP -m CHGRP_ID -m ROLE_ASSIGN -m ROLE_REMOVE -i
```

特定のユーザーがそのユーザーのログイン ID (**audit**) を使って実行したログインのアクションをすべて検索するには、以下のコマンドを実行します。

```
~]# ausearch -ua 1000 -i
```

昨日から現在までに失敗したすべてのシステムコールを検索するには、以下のコマンドを実行します。

```
~]# ausearch --start yesterday --end now -m SYSCALL -sv no -i
```

For a full listing of all **ausearch** options, refer to the **ausearch(8)** man page.

5.8. AUDIT レポートの作成

aureport ユーティリティーを使うと、Audit ログファイルに記録されたイベントについてのサマリーとレポートが生成できます。デフォルトでは、レポート作成のために **/var/log/audit/** 内のすべての **audit.log** ファイルがクエリーされます。**aureport options -if file_name** コマンドを使うと、レポート作成に別のファイルをクエリーするよう指定できます。

例5.8 aureport を使った Audit レポートの生成

今日を除いた過去 3 日間にログ記録されたイベントのレポートを生成するには、以下のコマンドを実行します。

```
~]# aureport --start 04/08/2013 00:00:00 --end 04/11/2013 00:00:00
```

実行可能ファイルのイベントすべてについてのレポートを生成するには、以下のコマンドを実行します。

```
~]# aureport -x
```


上記の実行可能ファイルのレポートのサマリーを生成するには、以下のコマンドを実行します。

```
~]# aureport -x --summary
```

全ユーザーの失敗したイベントのレポートサマリーを生成するには、以下のコマンドを実行します。

```
~]# aureport -u --failed --summary -i
```

システムユーザーごとのすべてのログイン試行失敗のレポートサマリーを生成するには、以下のコマンドを実行します。

```
~]# aureport --login --summary -i
```

ユーザー **1000** のすべてのファイルアクセスイベントを検索する **ausearch** クエリーからレポートを生成するには、以下のコマンドを実行します。

```
~]# ausearch --start today --loginuid 1000 --raw | aureport -f --summary
```

クエリーされたすべての Audit ファイルおよびそれらに含まれるイベントの時間帯についてのレポートを生成するには、以下のコマンドを実行します。

```
~]# aureport -t
```

For a full listing of all **aureport** options, refer to the **aureport(8)** man page.

5.9. その他のリソース

Audit システムについての詳細情報は、以下のリソースを参照してください。

オンラインのリソース

- Linux Audit ドキュメントのプロジェクトページ: <https://github.com/linux-audit/audit-documentation/wiki>
- Hack In the Box 誌の記事『Investigating kernel Return Codes with the Linux Audit System』: <http://magazine.hackinthebox.org/issues/HITB-Ezine-Issue-005.pdf>

インストールされているドキュメント

audit パッケージが提供するドキュメンテーションは、**/usr/share/doc/audit/** ディレクトリーにあります。

man ページ

- **audispd.conf(5)**
- **auditd.conf(5)**
- **ausearch-expression(5)**
- **audit.rules(7)**
- **audispd(8)**

- auditctl(8)
- auditd(8)
- aulast(8)
- aulastlog(8)
- aureport(8)
- ausearch(8)
- ausyscall(8)
- autrace(8)
- auvirt(8)

第6章 コンプライアンスおよび OPENS CAP を使った脆弱性のスキャン

6.1. RED HAT ENTERPRISE LINUX におけるセキュリティーコンプライアンス

コンプライアンス監査は、あるオブジェクトがコンプライアンスポリシーのすべてのルールにしたがっているかどうかを判断するプロセスです。コンプライアンスポリシーは、コンピューティング環境で使用される望ましい設定を指定するセキュリティー専門家が定義します。これは多くの場合、チェックリストの形式を取ります。

コンプライアンスポリシーは組織によって大幅に異なることがよくあり、同一組織内でもシステムが異なると様々なものになります。システムの目的や組織におけるシステム重要性に基づいて、これらのポリシーは異なってきます。カスタム化したソフトウェア設定や導入の特徴によっても、カスタム化したポリシーのチェックリストが必要になってきます。

Red Hat Enterprise Linux は、完全に自動化されたコンプライアンス監査を可能にするツールを提供します。これらのツールは SCAP (Security Content Automation Protocol) 標準に基づいており、コンプライアンスポリシーの自動化に合わせるように設計されています。

Red Hat Enterprise Linux 7 でサポートされているセキュリティーコンプライアンスツール

- **SCAP Workbench** — `scap-workbench` グラフィカルユーティリティーは、単一のローカルまたはリモートシステム上で構成スキャンと脆弱性スキャンを実行するように設計されています。これらのスキャンと評価に基づくセキュリティーレポートの生成にも使用できます。
- **OpenSCAP** — `oscap` コマンドラインユーティリティーは、ローカルシステムの上で構成スキャンと脆弱性スキャンを実行するように設計されています。これにより、セキュリティーコンプライアンスのコンテンツを検証し、スキャンと評価に基づいてレポートとガイドを生成します。
- **Script Check Engine (SCE)** — **SCE** は **SCAP** プロトコルの拡張機能であり、この機能を使用すると管理者が Bash、Python、Ruby などのスクリプト言語を使ってセキュリティーコンテンツを記述できるようになります。**SCE** 拡張機能は、`openscap-engine-sce` パッケージで提供されます。
- **SCAP Security Guide (SSG)** — `The scap-security-guide` パッケージは、Linux システム向けの最新のセキュリティーポリシーコレクションを提供します。このガイダンスは、セキュリティー強化に関する実践的なアドバイスのカタログから構成されます (該当する場合は、法規制要件に関連付けられています)。このプロジェクトは、一般的なポリシー要件と特定の実装ガイドラインの間にある開きを埋めることを目的としています。

複数のリモートのシステムで自動コンプライアンス監査を実行する必要がある場合は、Red Hat Satellite 用の OpenSCAP ソリューションを利用することができます。詳細は、「[Red Hat Satellite での OpenSCAP の使用](#)」および「[その他のリソース](#)」を参照してください。

6.2. コンプライアンスポリシーの定義

セキュリティーまたはコンプライアンスポリシーがまったくのゼロから書かれることはめったにありません。**ISO 27000** 標準シリーズやその派生物、他のソースなどがセキュリティーポリシーのテンプレートや実践上の推奨事項を提供するので、最初はこれらが便利です。しかし、情報セキュリティープログラムを構築している組織では、個別のニーズに合わせるためにポリシーテンプレートを修正する必

要があります。テンプレートは、企業環境への関連性に基づいて選択すべきです。テンプレートには組織に適用できない埋め込みの前提が含まれていたり、テンプレートで特定の判断が明らかに必要であることなどから、テンプレート選択後にこれを調整する必要があります。

Red Hat Enterprise Linux の監査機能は、Security Content Automation Protocol (SCAP) 標準をベースとしています。SCAP は相互運用可能な使用を統合したもので、形式と命名を標準化します。これらの形式と命名により、ソフトウェアの弱点およびセキュリティ設定の情報がマシンとユーザーの両方に通信されます。SCAP は、自動設定、脆弱性とパッチのチェック、技術的制御コンプライアンスアクティビティ、およびセキュリティ措置などをサポートする仕様の複数目的のフレームワークです。

言い換えると、SCAP は、ベンダーにとらわれないセキュリティポリシーの表示方法です。このため、現代の企業では幅広く使われています。SCAP 仕様は、セキュリティコンテンツの形式が周知かつ標準化されたエコシステムを作り出す一方で、スキャナーやポリシーエディターの導入は義務化されません。このような状態では、企業がいくつものセキュリティベンダーを用いていても、組織がセキュリティポリシー (SCAP コンテンツ) を構築するのは一度で済みます。

SCAP の最新バージョンには、いくつかの基礎的標準が含まれています。これらのコンポーネントは、SCAP 内での機能により、以下のようにグループ化されています。

SCAP コンポーネント

- **言語** — このグループは、コンプライアンスポリシーを表現する標準的な語彙および慣習を定義する SCAP 言語で構成されます。
 - *eXtensible Configuration Checklist Description Format (XCCDF)* — セキュリティガイダンスを表示、組織、および管理するための言語です。
 - *Open Vulnerability and Assessment Language (OVAL)* — スキャンされたシステムの状態について論理的アサーションを実行するための言語です。
 - *Open Checklist Interactive Language (OCIL)* — ユーザーにクエリーを行い、その質問に対するユーザーの返答を解釈するための標準的な方法を提供する言語です。
 - *Asset Identification (AI)* — セキュリティアセットを特定するためのデータモデル、方法、およびガイダンスを提供する言語です。
 - *Asset Reporting Format (ARF)* — 収集されたセキュリティアセットおよびアセットとセキュリティレポートとの関係についての情報の送信形式を表示する言語です。
- **列挙** — このグループには、命名形式および興味のある特定のセキュリティ関連の分野からのアイテムの公式一覧または辞書を定義する、SCAP 基準が含まれています。
 - *Common Configuration Enumeration (CCE)* — アプリケーションおよびオペレーティングシステム用のセキュリティ関連設定要素の列挙です。
 - *Common Platform Enumeration (CPE)* — 情報技術 (IT) システム、プラットフォーム、およびソフトウェアパッケージの特定に使用される構造化命名スキームです。
 - *Common Vulnerabilities and Exposures (CVE)* — 周知のソフトウェア脆弱性および露出のコレクションへの参照方法です。
- **メトリクス** — このグループは、セキュリティリスクを特定、評価するフレームワークで構成されています。

- *Common Configuration Scoring System (CCSS)* — セキュリティー関連の設定要素を評価し、それらをスコアに割り当ててユーザーが適切な応答ステップの優先度を決定できるようにする、メトリクスシステムです。
- *Common Vulnerability Scoring System (CVSS)* — ソフトウェアの脆弱性を評価し、それらをスコアに割り当ててユーザーがセキュリティーリスクの優先度を決定できるようにするメトリクスシステムです。
- **整合性** — SCAP コンポーネントとスキャンの結果の整合性を維持するための SCAP 仕様です。
- *Trust Model for Security Automation Data (TMSAD)* — セキュリティー自動化ドメイン内の XML ファイルのコンテンツにおいて、署名、ハッシュ、鍵情報、および ID 情報を表示するための既存の仕様の使い方を説明した推奨事項です。

各 SCAP コンポーネントには、XML ベースのドキュメント形式とその XML ネームスペースがあります。SCAP で表示されているコンプライアンスポリシーは、単一の OVAL 定義 XML ファイル、データストリームファイル、単一 zip アーカイブ、またはポリシーチェックリストを表示する XCCDF ファイルを含む別個の XML ファイルセットのいずれかになります。

6.2.1. XCCDF ファイル形式

XCCDF 言語は、情報交換、ドキュメント生成、組織および状況に応じた調整、自動コンプライアンステスト、およびコンプライアンススコアリングをサポートするように設計されています。この言語は主に記述的で、セキュリティースキャンを実行するコマンドは含まれていません。しかし XCCDF ドキュメントは他の SCAP コンポーネントを参照できるため、関連する評価ドキュメント (OVAL, OCIL) を除くすべてのターゲットプラットフォームのなかで移植可能なコンプライアンスポリシーの作成に使用することができます。

コンプライアンスポリシーを表示する一般的な方法は XML ファイルセットで、このうちのひとつが XCCDF チェックリストになります。この XCCDF は通常、評価リソース、複数の OVAL、OCIL および Script Check Engine (SCE) ファイルを参照しています。さらに、このファイルセットには、CPE 辞書ファイルとこの辞書用のオブジェクトを定義する OVAL ファイルを含めることができます。

XCCDF は XML ベースの言語であることから、非常に多くの XML 要素と属性を定義、使用します。以下では、主要な XCCDF 要素を簡単に説明しています。XCCDF の詳細については、[NIST Interagency Report 7275 Revision 4](#) を参照してください。

XCCDF ドキュメントの主要 XML 要素

- **<xccdf:Benchmark>** — XCCDF ドキュメント全体を囲むルート要素です。これには、タイトル、説明、作者一覧、最新の修正日、チェックリスト受け付けのステータスなどのチェックリストのメタデータが含まれる場合があります。
- **<xccdf:Rule>** — チェックリスト要件を表示し、その説明を保持する重要な要素です。これには、特定のルールコンプライアンスを確認または強制したり、ルール自体を修正するアクションを定義する、子要素が含まれる場合があります。
- **<xccdf:Value>** — ベンチマーク内の他の XCCDF 要素の属性を表示するために使用される重要な要素です。
- **<xccdf:Group>** — この要素は、XCCDF ドキュメントを組織化して、要件と同じコンテキストまたはタイプのグループや構造体を含みます。これは、**<xccdf:Rule>**、**<xccdf:Value>**、および **<xccdf:Group>** 要素を収集することでなされます。

- **<xccdf:Profile>** — この要素は、XCCDF ベンチマークの名前付き調整に使われます。ベンチマークがいくつかの異なるテーラリングを持つことが可能になります。**<xccdf:Profile>** は、**<xccdf:select>** や **<xccdf:refine-rule>** などのいくつかのセクター要素を使って、どの要素が実行中に修正されたり処理されたりするかを決定します。
- **<xccdf:Tailoring>** — この要素では、ベンチマーク外のベンチマークプロファイルの定義が可能になります。これは、コンプライアンスポリシーの手動のテーラリングでは、妥当なものになる場合があります。
- **<xccdf:TestResult>** — この要素は、ターゲットシステム上の特定のベンチマークのスキャン結果を保持する役割を果たします。各 **<xccdf:TestResult>** は、特定のスキャンのコンプライアンスポリシーを定義する際に使用されたプロファイルを参照するようにしてください。また、スキャンに関連するターゲットシステムについての重要情報も含めるようにしてください。
- **<xccdf:rule-result>** — **<xccdf:TestResult>** の子要素で、ベンチマークからターゲットシステムに特定のルールを適用した結果を保持するために使用されます。
- **<xccdf:fix>** — **<xccdf:Rule>** の子要素で、特定のルールに従っていないターゲットシステムの改善に使われます。これにターゲットシステム上で実行するコマンドまたはスクリプトを含めると、システムをルールにしたがわせることができます。
- **<xccdf:check>** — **<xccdf:Rule>** の子要素で、特定ルールの評価方法を定義する外部ソースを参照します。
- **<xccdf:select>** — ポリシーから特定のルールまたはルールのグループを除外もしくは含めるために使われるセクター要素です。
- **<xccdf:set-value>** — **<xccdf:Value>** 要素の属性を変更せずに、この要素で指定された現在の値の上書きに使われるセクター要素です。
- **<xccdf:refine-value>** — ポリシーテーラリング中に特定の**<xccdf:Value>** 要素の制限を指定するために使われるセクター要素です。
- **<xccdf:refine-rule>** — このセクター要素は、選択されたルールの属性の上書きを可能にします。

例6.1 XCCDF ドキュメントの例

```
<?xml version="1.0" encoding="UTF-8"?>
<Benchmark xmlns="http://checklists.nist.gov/xccdf/1.2"
  id="xccdf_com.example.www_benchmark_test">
  <status>incomplete</status>
  <version>0.1</version>
  <Profile id="xccdf_com.example.www_profile_1">
    <title>Profile title is compulsory</title>
    <select idref="xccdf_com.example.www_group_1"
      selected="true"/>
    <select idref="xccdf_com.example.www_rule_1"
      selected="true"/>
    <refine-value idref="xccdf_com.example.www_value_1"
      selector="telnet service"/>
  </Profile>
  <Group id="xccdf_com.example.www_group_1">
    <Value id="xccdf_com.example.www_value_1">
      <value selector="telnet_service">telnet-server</value>
```



```

    <value selector="dhcp_servide">dhcpd</value>
    <value selector="ftp_service">tftpd</value>
  </Value>
  <Rule id="xccdf_com.example.www_rule_1">
    <title>The telnet-server Package Shall Not Be Installed </title>
    <rationale>
      Removing the telnet-server package decreases the risk
      of the telnet service's accidental (or intentional) activation
    </rationale>
    <fix platform="cpe:/o:redhat:enterprise_linux:6"
      reboot="false"
      disruption="low"
      system="urn:xccdf:fix:script:sh">
      yum -y remove
      <sub idref="xccdf_com.example.www_value_1"/>
    </fix>
    <check system="http://oval.mitre.org/XMLSchema/oval-definitions-
5">
      <check-export value-id="xccdf_com.example.www_value_1"
        export-name="oval:com.example.www:var:1"/>
      <check-content-ref href="exemplary.oval.xml"
        name="oval:com.example.www:def:1"/>
    </check>
    <check system="http://open-scap.org/page/SCE">
      <check-import import-name="stdout"/>
      <check-content-ref href="telnet_server.sh"/>
    </check>
  </Rule>
</Group>
</Benchmark>

```

6.2.2. OVAL ファイル形式

Open Vulnerability Assessment Language (OVAL) は、SCAP の必須かつ最も古いコンポーネントです。OVAL 標準の主なゴールは、セキュリティ製品の間で相互運用性を確保することです。これは、以下の 3 つのドメインの標準化でなされます。

1. ターゲットシステム設定の表示。
2. ターゲットシステムが特定のマシン状態にあるかを分析。
3. 指定されたマシン状態と観察されたマシン状態の比較結果のレポート。

他のツールやカスタム化されたスクリプトとは異なり、OVAL 言語は宣言型でリソースの望ましい状態を記述します。OVAL 言語コードはスキャナーと呼ばれる OVAL インタープリターツールを用いて実行されますが、直接実行されることは決してありません。OVAL が宣言型であるために、評価されるシステムの状態が偶然に修正されることはありません。セキュリティスキャナーは可能な限り高い権限で実行されることが多いので、これは重要な点になります。

OVAL 仕様はコメントと貢献を公開で受け付けています。また、多くの IT 企業が米連邦政府の援助を受けた NPO である MITRE Corporation と協力しています。OVAL 仕様は継続的に発展しており、バージョン番号で改訂版を区別しています。最新バージョンは 2015 年 4 月にリリースされた 5.11.1 になります。

他のすべての SCAP コンポーネントと同様に、OVAL は XML に基づいています。OVAL 標準は、いくつかのドキュメント形式を定義します。これらはそれぞれ異なる種類の情報を含み、異なる目的に使われます。

OVAL ドキュメント形式

- *OVAL Definitions* 形式は、最も一般的な OVAL ファイル形式で、システムスキャンに直接使用されます。OVAL Definitions のドキュメントは、ターゲットシステムの望ましい状態を記述します。
- *OVAL Variables* 形式は、OVAL Definitions ドキュメントの修正に使用される変数を定義します。OVAL Variables ドキュメントは通常、OVAL Definitions ドキュメントと一緒に使われ、ランタイム時にターゲットシステムのセキュリティコンテンツを調整します。
- *OVAL System Characteristics* 形式は、評価対象のシステムについての情報を保持します。OVAL System Characteristics ドキュメントは通常、実際のシステムの状態と、OVAL Definitions ドキュメントが定義する期待された状態を比較するために使用されます。
- *OVAL Results* は、最も包括的な OVAL 形式で、システム評価の結果を報告するために使われます。OVAL Results ドキュメントには通常、評価された OVAL 定義のコピー、バインドされた OVAL 変数、OVAL システムの特徴、システムの特徴と定義の比較に基づいて計算されたテスト結果が含まれます。
- *OVAL Directives* 形式は、特定の詳細を除外または含めることで、OVAL Result ドキュメントの長さを調整するために使われます。
- *OVAL Common Model* 形式には、いくつかの他の OVAL スキームに使用される構造体と列挙の定義が含まれます。これは、OVAL 定義を再利用して、複数のドキュメントでの重複を防ぐために使われます。

OVAL Definitions ドキュメントは、設定要件一式で構成されます。各要件は、*definitions*、*tests*、*objects*、*states*、および *variables* の 5 つの基本的なセクションで構成されます。*definitions* セクション内における要素は、特定の定義を満たすためにどのテストが実行されるかを記述します。*test* 要素は、*objects* と *states* をリンクさせます。システム評価中に、ある *object* 要素が示す評価システムのリリースが特定の *state* 要素に対応すると、テストが成功したとみなされます。*variables* セクションでは、*states* セクションからの要素の調整に使用される可能性のある外部の変数を定義します。これらのセクションに加えて、OVAL Definitions ドキュメントには通常、*generator* および *signature* の各セクションも含まれています。*generator* セクションには、ドキュメントの出所についての情報とそのコンテンツに関連する様々な追加情報が含まれます。

OVAL ドキュメントの基本的セクションからの各要素は、以下の形式の識別子で明確に特定されます。

`oval:namespace:type:ID`

ここでの *namespace* は、識別子を定義するネームスペースです。*type* は、*definitions* 要素の場合は *def*、*tests* 要素の場合は *tst*、*objects* 要素の場合は *obj*、*states* 要素の場合は *ste*、*variables* 要素の場合は *var* になります。*ID* は、識別子の整数値になります。

例6.2 OVAL Definitions ドキュメントの例

```
<?xml version="1.0" encoding="utf-8"?>
<oval_definitions
  xmlns:lin-def="http://oval.mitre.org/XMLSchema/oval-definitions-5#linux"
  xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
```



```

xmlns="http://oval.mitre.org/XMLSchema/oval-definitions-5"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<generator>
  <oval:product_name>vim</oval:product_name>
  <oval:schema_version>5.10.1</oval:schema_version>
  <oval:timestamp>2012-11-22T15:00:00+01:00</oval:timestamp>
</generator>
<definitions>
  <definition class="inventory"
    id="oval:org.open-scap.cpe.rhel:def:7"
    version="1">
    <metadata>
      <title>Red Hat Enterprise Linux 7</title>
      <affected family="unix">
        <platform>Red Hat Enterprise Linux 7</platform>
      </affected>
      <reference ref_id="cpe:/o:redhat:enterprise_linux:7"
        source="CPE"/>
      <description>
        The operating system installed on the system is Red Hat
        Enterprise Linux 7
      </description>
    </metadata>
    <criteria>
      <criterion comment="Red Hat Enterprise Linux 7 is installed"
        test_ref="oval:org.open-scap.cpe.rhel:tst:7"/>
    </criteria>
  </definition>
</definitions>
<tests>
  <lin-def:rpminfo_test check_existence="at_least_one_exists"
    id="oval:org.open-scap.cpe.rhel:tst:7"
    version="1"
    check="at least one"
    comment="redhat-release is version 7">
    <lin-def:object object_ref="oval:org.open-scap.cpe.redhat-
      release:obj:1"/>
    <lin-def:state state_ref="oval:org.open-scap.cpe.rhel:ste:7"/>
  </lin-def:rpminfo_test>
</tests>
<objects>
  <lin-def:rpmverifyfile_object id="oval:org.open-scap.cpe.redhat-
    release:obj:1"
    version="1">
    <!-- This object represents rpm package which owns /etc/redhat-
      release file -->
    <lin-def:behaviors nolinkto='true'
      nomd5='true'
      nosize='true'
      nouser='true'
      nogroup='true'
      nomtime='true'
      nomode='true'
      nordev='true'
      noconfigfiles='true'
      noghostfiles='true' />
  </lin-def:rpmverifyfile_object>
</objects>

```

```

<lin-def:name operation="pattern match"/>
<lin-def:epoch operation="pattern match"/>
<lin-def:version operation="pattern match"/>
<lin-def:release operation="pattern match"/>
<lin-def:arch operation="pattern match"/>
<lin-def:filepath>/etc/redhat-release</lin-def:filepath>
</lin-def:rpmverifyfile_object>
</objects>
<states>
  <lin-def:rpminfo_state id="oval:org.open-scap.cpe.rhel:ste:7"
    version="1">
    <lin-def:name operation="pattern match">^redhat-release</lin-
def:name>
    <lin-def:version operation="pattern match">^7[^\d]</lin-
def:version>
    </lin-def:rpminfo_state>
  </states>
</oval_definitions>

```

6.2.3. データストリームの形式

SCAP データストリームは、SCAP バージョン 1.2 から使用されているファイル形式で、XCCDF や OVAL、その他のコンポーネントファイルなど、XCCDF チェックリストで表示されているコンプライアンスポリシーの定義に使用可能なもののバンドルを表します。また、特定のデータストリームを SCAP コンポーネントにしたがってファイルに分割することを可能にするインデックスとカタログも含まれています。

データストリームは XML 形式を使用し、これはコンテンツのテーブルで形成されたヘッダーと **<ds:component>** 要素の一覧で構成されています。これらの要素にはそれぞれ、XCCDF や OVAL、CPE などの SCAP コンポーネントが含まれます。データストリームファイルには同じタイプの複数のコンポーネントが含まれている場合があり、組織で必要とされるセキュリティポリシーすべてがカバーされます。

例6.3 データストリームヘッダーの例

```

<ds:data-stream-collection
xmlns:ds="http://scap.nist.gov/schema/scap/source/1.2"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:cat="urn:oasis:names:tc:entity:xmlns:xml:catalog"
id="scap_org.open-scap_collection_from_xccdf_ssg-rhel7-xccdf-
1.2.xml"
schematron-version="1.0">
  <ds:data-stream id="scap_org.open-scap_datastream_from_xccdf_ssg-
rhel7-xccdf-1.2.xml"
    scap-version="1.2" use-case="OTHER">
    <ds:dictionaries>
      <ds:component-ref id="scap_org.open-scap_cref_output--ssg-rhel7-
cpe-dictionary.xml"
        xlink:href="#scap_org.open-scap_comp_output--ssg-rhel7-cpe-
dictionary.xml">
        <cat:catalog>
          <cat:uri name="ssg-rhel7-cpe-oval.xml"

```

```

        uri="#scap_org.open-scap_cref_output--ssg-rhel7-cpe-
oval.xml"/>
    </cat:catalog>
</ds:component-ref>
</ds:dictionaries>
<ds:checklists>
    <ds:component-ref id="scap_org.open-scap_cref_ssg-rhel7-xccdf-
1.2.xml"
        xlink:href="#scap_org.open-scap_comp_ssg-rhel7-xccdf-1.2.xml">
        <cat:catalog>
            <cat:uri name="ssg-rhel7-oval.xml"
                uri="#scap_org.open-scap_cref_ssg-rhel7-oval.xml"/>
        </cat:catalog>
    </ds:component-ref>
</ds:checklists>
<ds:checks>
    <ds:component-ref id="scap_org.open-scap_cref_ssg-rhel7-oval.xml"
        xlink:href="#scap_org.open-scap_comp_ssg-rhel7-oval.xml"/>
    <ds:component-ref id="scap_org.open-scap_cref_output--ssg-rhel7-
cpe-oval.xml"
        xlink:href="#scap_org.open-scap_comp_output--ssg-rhel7-cpe-
oval.xml"/>
    <ds:component-ref id="scap_org.open-scap_cref_output--ssg-rhel7-
oval.xml"
        xlink:href="#scap_org.open-scap_comp_output--ssg-rhel7-
oval.xml"/>
</ds:checks>
</ds:data-stream>
<ds:component id="scap_org.open-scap_comp_ssg-rhel7-oval.xml"
    timestamp="2014-03-14T16:21:59">
    <oval_definitions xmlns="http://oval.mitre.org/XMLSchema/oval-
definitions-5"
        xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
        xmlns:ind="http://oval.mitre.org/XMLSchema/oval-definitions-
5#independent"
        xmlns:unix="http://oval.mitre.org/XMLSchema/oval-definitions-
5#unix"
        xmlns:linux="http://oval.mitre.org/XMLSchema/oval-definitions-
5#linux"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://oval.mitre.org/XMLSchema/oval-common-5
oval-common-schema.xsd
        http://oval.mitre.org/XMLSchema/oval-definitions-5
oval-definitions-schema.xsd
        http://oval.mitre.org/XMLSchema/oval-definitions-
5#independent
independent-definitions-schema.xsd
        http://oval.mitre.org/XMLSchema/oval-definitions-5#unix
unix-definitions-schema.xsd
        http://oval.mitre.org/XMLSchema/oval-definitions-5#linux
linux-definitions-schema.xsd">

```

6.3. SCAP WORKBENCH の使用

SCAP Workbench (scap-workbench) はグラフィカルユーティリティーで、これを使うと単一のローカルまたはリモートシステム上で構成スキャンと脆弱性スキャンが実行できます。また、システム修復や、スキャン評価に基づくレポート生成もできます。**oscap** コマンドラインユーティリティーと比べると、**SCAP Workbench** には限定的な機能しかないことに注意してください。また、**SCAP Workbench** は XCCDF およびデータストリームファイルの形式でしかセキュリティコンテンツを処理できません。

以下のセクションでは、SCAP Workbench をインストール、起動、利用して、システムスキャンや修復、スキャンのカスタマイズを実行する方法と、これらタスクに関連する例を説明します。

6.3.1. SCAP Workbench のインストール

システムに **SCAP Workbench** をインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install scap-workbench
```

このコマンドにより、SCAP Workbench の正常な機能に必要なすべてのパッケージがインストールされます。これには、それ自体がユーティリティーを提供する scap-workbench パッケージも含まれています。システムに qt および openssh パッケージなどの必要な依存関係がインストールされていれば、それらのパッケージは自動的に最新バージョンに更新されることに注意してください。

SCAP Workbench を効果的に使用する前に、システムでセキュリティコンテンツをインストールまたはインポートする必要があります。たとえば、現時点で最も新しくかつ詳細な Linux 向けセキュリティポリシーを含む SCAP Security Guide (SSG) パッケージ (scap-security-guide) をインストールできます。SCAP Security Guide パッケージをシステムにインストールするには、root で次のコマンドを実行します。

```
~]# yum install scap-security-guide
```

システムに scap-security-guide をインストールした後は、特に指定されていない場合は、SSG セキュリティコンテンツは **/usr/share/xml/scap/ssg/content/** ディレクトリーに格納され、他のセキュリティコンプライアンス作業を進めることができます。

個別のニーズに合致する可能性のある既存の SCAP コンテンツの他のソースが必要な場合は、[「その他のリソース」](#) を参照してください。

6.3.2. SCAP Workbench の稼働

SCAP Workbench ユーティリティーと SCAP コンテンツを正常にインストールできたら、システム上で **SCAP Workbench** の使用が可能になります。**SCAP Workbench** を **GNOME Classic** デスクトップ環境から稼働するには、**Super** キーを押して **アクティビティー** に入り、**scap-workbench** と入力してから **Enter** を押します。**Super** キーはキーボードと他のハードウェアによって外観が異なりますが、普通は **Windows** または **Command** キーで、通常 **スペースバー** キーの左隣にあります。

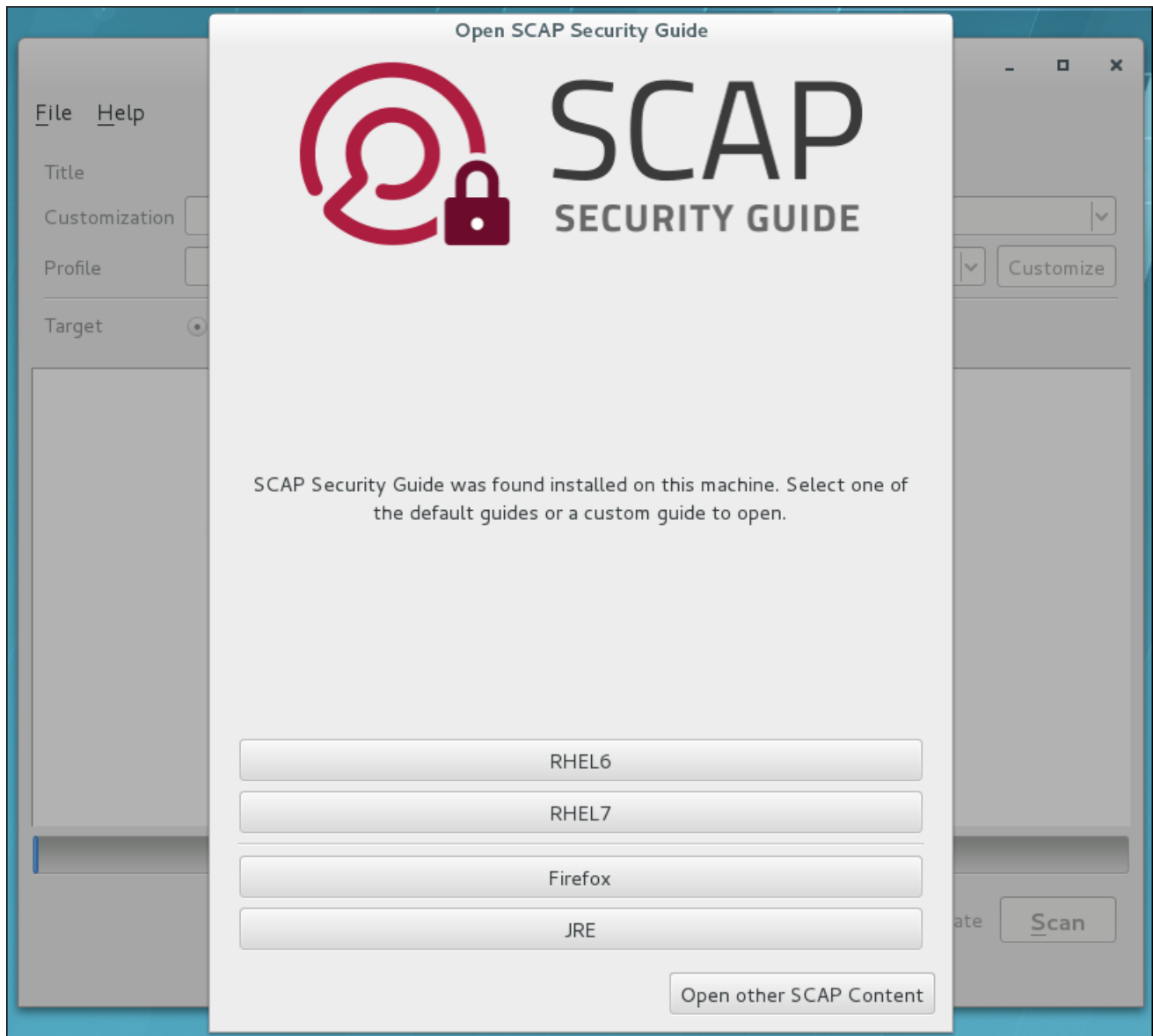


図6.1 SCAP セキュリティーガイドの表示ウィンドウ

ユーティリティーを起動すると、**Open SCAP Security Guide** ウィンドウが開きます。ガイドの 1 つを選択すると、**SCAP Workbench** ウィンドウが開きます。このウィンドウは、7 つのインタラクティブな要素で構成されており、システムのスキャン実行前にこれらに慣れておくといよいでしょう。

File

このメニュー一覧は、SCAP 関連のコンテンツの読み込みや保存オプションを提供します。最初の **Open SCAP Security Guide** ウィンドウを表示するには、同じ名前のメニューアイテムをクリックします。または、**Open Other Content** をクリックして XCCDF 形式の別のカスタマイズファイルを読み込みます。カスタマイズを XCCDF XML ファイルで保存するには、**Save Customization Only** の項目を使用します。**Save All** では、選択したディレクトリーまたは RPM パッケージとして、SCAP ファイルを保存できます。

カスタマイズ

このコンボボックスは、特定のセキュリティーポリシーに使用されるカスタマイズを表示します。このコンボボックスをクリックして、システム評価に適用されるカスタムルールを選択できます。デフォルト値は **(no customization)** で、これは使用されているセキュリティーポリシーに変更がないことを意味します。選択されているセキュリティープロファイルに変更を加えた場合、ファイルメニューの **Save Customization Only** をクリックするとこれらの変更を XML ファイルで保存できます。

Profile

このコンボボックスには、選択されたセキュリティープロファイル名が表示されます。コンボボックスをクリックすると、XCCDF またはデータストリームファイルからセキュリティープロファイルを選択できます。選択したセキュリティープロファイルのプロパティーを継承する新規プロファイルを作成するには、**Customize** ボタンをクリックします。

Target

評価対象のシステムがローカルかリモートかを 2 つのラジオボタンから選択します。

Selected Rules

このフィールドには、セキュリティーポリシーの対象となるセキュリティルールが一覧表示されます。特定のセキュリティルールを展開すると、そのルールについての詳細情報が表示されます。

ステータスバー

このグラフィカルバーは、実行中の操作のステータスを表示します。

リモートリソースの取得

このチェックボックスは、スキャナーに対して、XML ファイルで定義されているリモートの OVAL コンテンツをダウンロードするように指示を出すことができます。

Dry run

スキャンを実行するのではなく、診断ウィンドウにコマンドラインの引数を取得するには、このチェックボックスを使用します。

Remediate

このチェックボックスにチェックを入れると、システム評価中に修復機能が有効になります。チェックが入っていると、SCAP Workbench はポリシーで定義された状態にマッチしないシステム設定の修正を試行します。

Scan

このボタンを押すと、指定されたシステムの評価が開始されます。

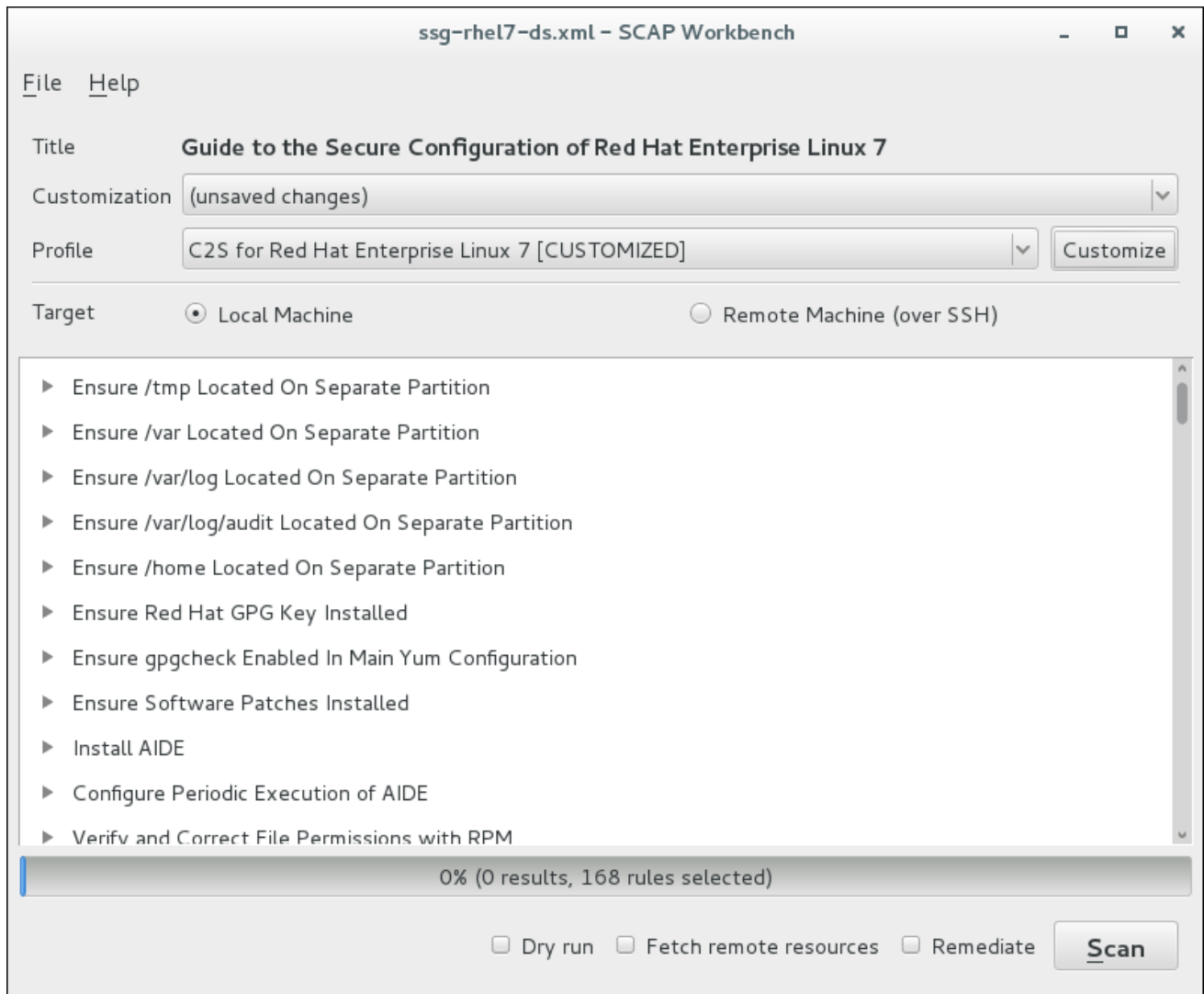


図6.2 SCAP Workbench のウィンドウ

6.3.3. システムのスキャン

SCAP Workbench の主な機能は、特定の XCCDF またはデータストリームファイルにしたがって、選択されたシステム上でセキュリティースキャンを実行することです。選択されたセキュリティーポリシーに対してシステムを評価するには、以下のステップにしたがいます。

1. **Open SCAP Security Guide** ウィンドウまたは、**File** メニューの **Open Other Content** を使用してセキュリティーポリシーを選択し、該当の XCCDF、SCAP RPM またはデータストリームファイルを検索します。



警告

セキュリティーポリシーを選択すると、保存されていない以前のカスタマイズの変更は失われます。失われたオプションを再度適用するには、利用可能なプロファイルとカスタマイズコンテンツを再度選択する必要があります。新規のセキュリティーポリシーでは、以前のカスタマイズが適用できない場合があることに注意してください。

2. 自分のニーズにカスタマイズされたセキュリティコンテンツのある事前に用意されたファイルを使用するには、**Customization** コンボボックスでそのファイルをクリックして読み込みます。また、利用可能なセキュリティプロファイルを変更してカスタマイズした tailoring ファイルを作成することもできます。詳細は、「[セキュリティプロファイルのカスタマイズ](#)」を参照してください。
 - a. 現行のシステム評価にカスタマイズを使用しない場合は、**(no customization)** オプションを選択します。以前にカスタマイズが選択されていなければ、これがデフォルトオプションになります。
 - b. 現行のシステム評価に使用する特定の tailoring ファイルを検索するには、**(open customization file...)** オプションを選択します。
 - c. これまでにカスタマイズファイルを使用したことがある場合は、**SCAP Workbench** はこのファイルを保存していてリストに追加します。これにより、同スキャンの反復適用が簡素化されます。
3. **Profile** コンボボックスをクリックして適切な選択セキュリティプロファイルを選択します。
 - a. 選択したプロファイルを修正するには、**Customize** ボタンをクリックします。プロファイルのカスタマイズに関する詳細は、「[セキュリティプロファイルのカスタマイズ](#)」を参照してください。
4. **Target** ラジオボタンのいずれかを選択して、スキャンするマシンをローカルリモートから選択します。
 - a. リモートシステムを選択した場合は、以下の例のようにユーザー名、ホスト名、ポート情報を入力して指定します。以前のリモートスキャンを使用した場合には、最近スキャンしたマシンの一覧からリモートマシンを選択することもできます。

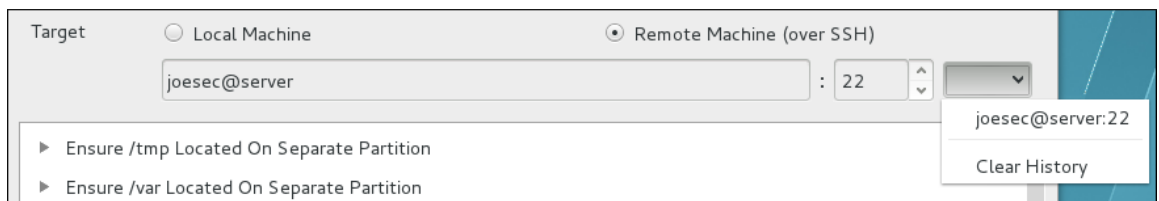


図6.3 リモートシステムの指定

5. **Remediate** チェックボックスを選択すると、システム設定の自動修正が有効になります。このオプションが有効になっていると、システムスキャン中に関連チェックが失敗した場合には、**SCAP Workbench** はポリシーが適用するセキュリティルールにしたがってシステム設定を変更するよう試みます。



警告

修正オプションが有効な状態でのシステム評価は、慎重に行わないとシステムが機能不全に陥る場合があります。

6. **Scan** ボタンをクリックしてシステムスキャンを開始します。

6.3.4. セキュリティープロファイルのカスタマイズ

ご使用のセキュリティポリシーに適応したセキュリティプロファイルを選択した後で **Customize** ボタンをクリックすると、これをさらに調節することができます。新しい Customization ウィンドウが開いて、現在選択されている XCCDF プロファイルをその XCCDF ファイルを実際に変更することなく修正できます。

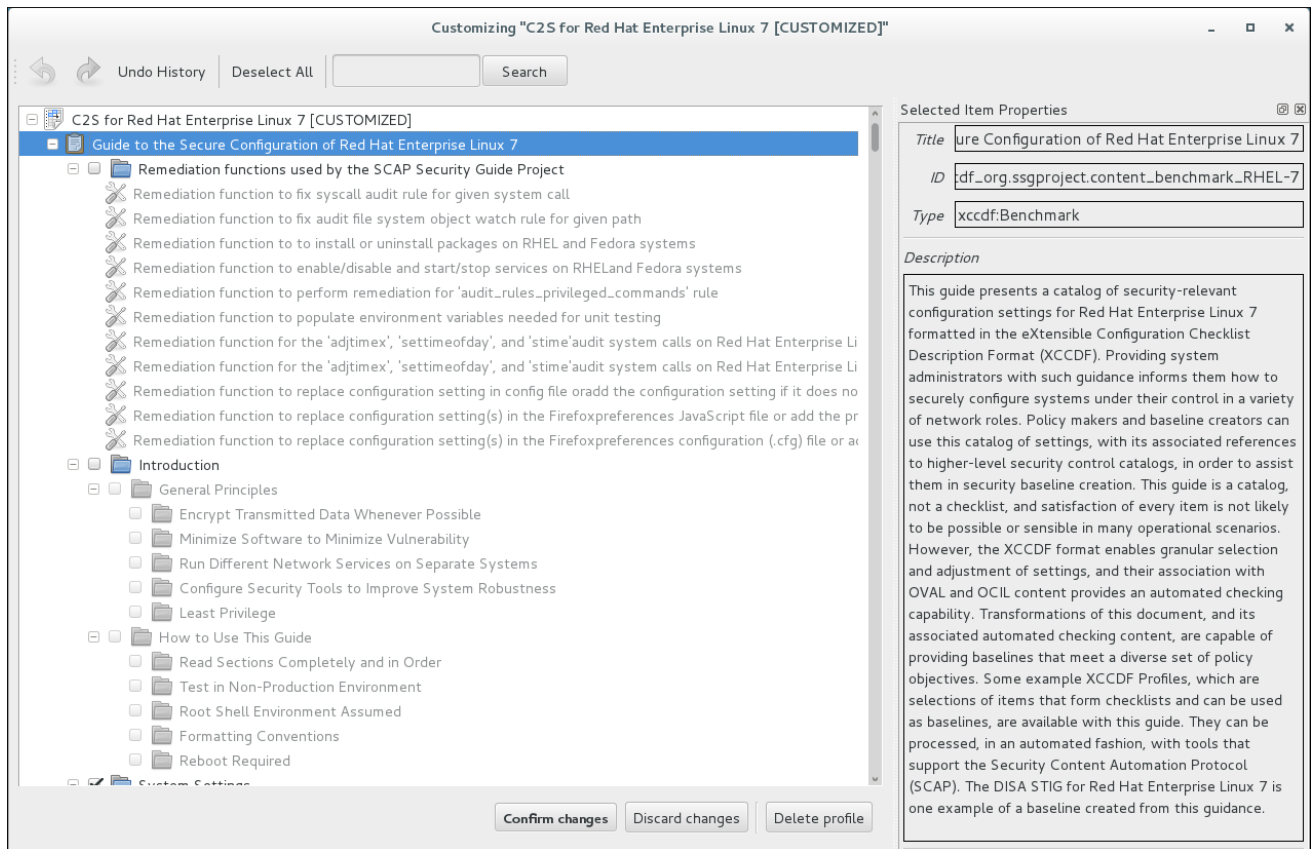


図6.4 セキュリティープロファイルのカスタマイズ

Customization ウィンドウには、選択されたセキュリティプロファイルに関連する XCCDF 要素がすべて含まれており、各要素の詳細情報と機能も表示されます。このウィンドウのメインフィールドで各要素のチェックボックスにチェックを入れたり外したりすることで、これらの要素を有効または無効にすることができます。**Customization** は、元に戻す と やり直す 機能もサポートしており、ウィンドウ左上の矢印アイコンをクリックすることでこれらの機能を実行できます。

後ほど評価に使用する変数を変更することもできます。**Customization** ウィンドウの任意のアイテムを検索し、右側の部分に移動して、**Modify value** フィールドを使用します。

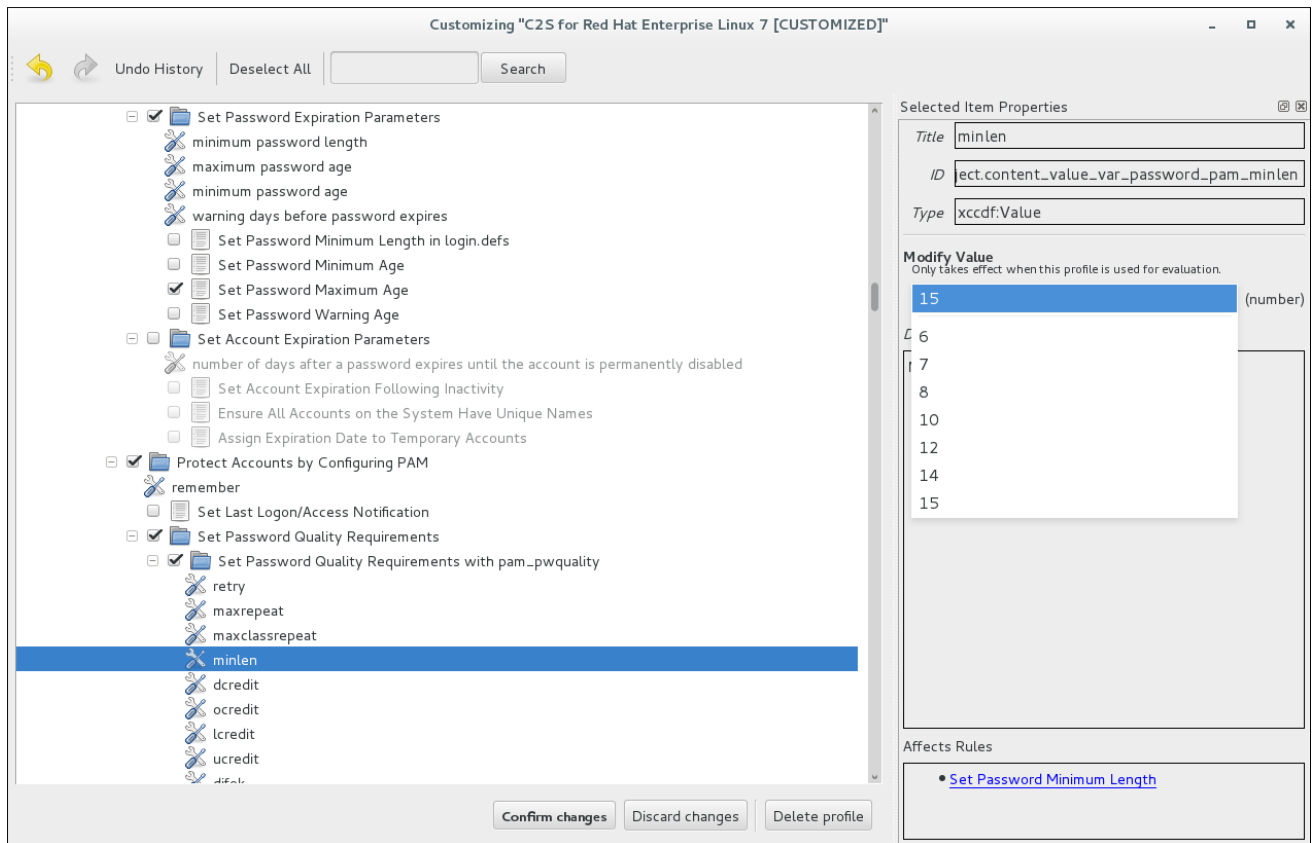


図6.5 Customization ウィンドウの選択アイテムの値設定

プロファイルのカスタマイズを終えたら、**Confirm Customization** ボタンをクリックして変更を確認します。これで変更がメモリーに記憶されますが、**SCAP Workbench** を終了したり、新規 SCAP コンテンツの選択や別の Customization オプションを選択するなどの特定の変更が行われると、変更は保持されません。変更を保存するには、**SCAP Workbench** ウィンドウの **Save Customization** ボタンをクリックします。こうすることで、セキュリティプロファイルの変更が選択されたディレクトリに XCCDF Customization ファイルとして保存できます。この Customization ファイルは他のプロファイルでも選択可能であることに注意してください。

6.3.5. SCAP コンテンツの保存

SCAP Workbench では、システム評価に使用された SCAP コンテンツの保存も可能になっています。Customization ファイルを別個に保存するか（「[セキュリティプロファイルのカスタマイズ](#)」を参照）または **Save content** コンボボックスをクリックして、**Save into a directory** または **Save as RPM** オプションを選択するとすべてのセキュリティコンテンツを一度に保存できます。

Save into a directory オプションを選択すると、**SCAP Workbench** は XCCDF もしくはデータストリームファイルと Customization ファイルの両方を指定された場所に保存します。これはバックアップとして便利な方法です。

Save as RPM オプションを選択すると、**SCAP Workbench** に XCCDF もしくはデータストリームファイルと Customization ファイルを含む RPM パッケージを作成するよう指示できます。これは、リモートでスキャンができないシステムに希望するセキュリティコンテンツを配布する場合やコンテンツをさらに処理するために配布する際に便利です。

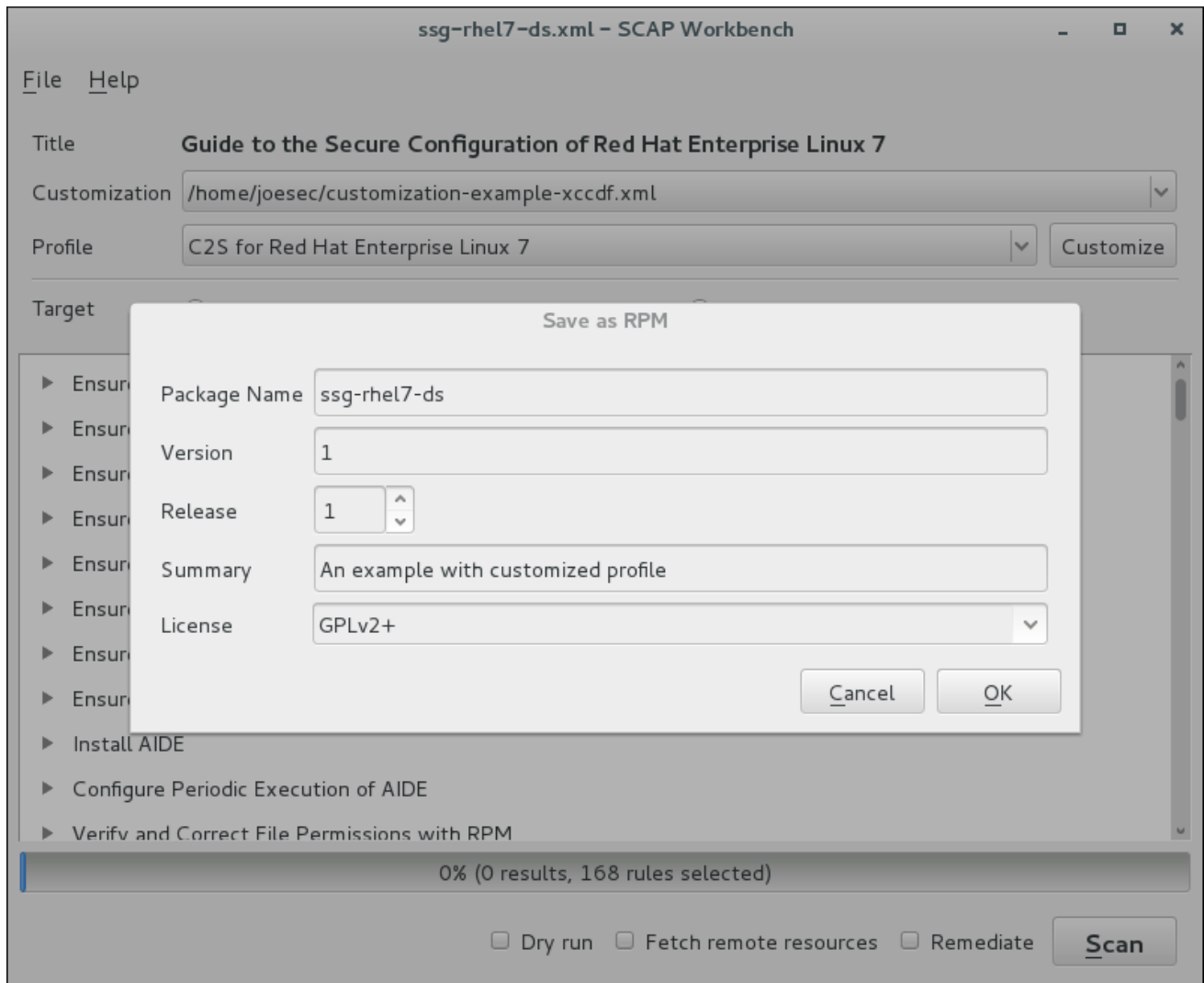


図6.6 現行 SCAP コンテンツを RPM パッケージとして保存

6.3.6. スキャン結果の表示とスキャンレポートの生成

システムスキャンが終了すると、**Scan** ボタンの代わりに **Clear**、**Save Results**、**Show Report** の 3 つのボタンが表示されます。



警告

Clear ボタンをクリックすると、スキャン結果が恒久的に削除されます。

スキャン結果を XCCDF、ARF、または HTML ファイルの形式で保存するには、**Save Results** コンボボックスをクリックします。人間が解読できる形式でスキャン結果を生成するには、**HTML Report** オプションを選択します。さらに自動処理を施したい場合は、XCCDF および ARF (データストリーム) 形式が適しています。これら 3 つのオプションは、何度でも選択できます。

スキャン結果を保存せずにすぐに確認する場合は、**Show Report** ボタンをクリックします。デフォルトのウェブブラウザで HTML の一時ファイル形式でスキャン結果が表示されます。

6.4. OSCAP の使用

oscap コマンドラインユーティリティーを使うと、ローカルシステムのスキャン、セキュリティコンプライアンスコンテンツの確認、これらのスキャンおよび評価を基にしたレポートとガイドの生成ができます。このユーティリティーは OpenSCAP ライブラリーへのフロントエンドとしてのサービスを提供し、その機能を処理する SCAP コンテンツのタイプに基づいてモジュール (サブコマンド) にグループ化します。

以下のセクションでは **oscap** のインストール方法と最も一般的な操作の実行方法を説明します。これらのタスクの説明に例を用いています。特定のサブコマンドの詳細を確認するには、**oscap** コマンドの **--help** オプションを使用してください。

```
oscap [options] module module_operation  
[module_operation_options_and_arguments] --help
```

ここでの *module* は、処理されている SCAP コンテンツのタイプになります。また、*module_operation* は、SCAP コンテンツにおける特定の操作のサブコマンドになります。

例6.4 特定の **oscap** 操作に関するヘルプ

```
~]$ oscap ds sds-split --help  
oscap -> ds -> sds-split  
  
Split given SourceDataStream into separate files  
  
Usage: oscap [options] ds sds-split [options] SDS TARGET_DIRECTORY  
  
SDS - Source data stream that will be split into multiple files.  
TARGET_DIRECTORY - Directory of the resulting files.  
  
Options:  
  --datastream-id <id>          - ID of the datastream in the  
collection to use.  
  --xccdf-id <id>              - ID of XCCDF in the datastream that  
should be evaluated.
```

oscap の全機能を学習し、オプションの完全一覧を確認するには、**oscap(8)** man ページを参照してください。

6.4.1. **oscap** のインストール

システムに **oscap** をインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install openscap-scanner
```

このコマンドを使用すると、**oscap** が正常に動作するために必要なパッケージ (openscap パッケージを含む) をすべてインストールできます。独自のセキュリティコンテンツを作成するには、Script Check Engine (SCE) を提供する openscap-engine-sce パッケージもインストールする必要があります。この SCE は SCAP プロトコルの拡張機能であり、この機能を使うとコンテンツ作成者は Bash、Python、Ruby などのスクリプト言語を使ってセキュリティコンテンツを作成できるようになります。openscap-engine-sce パッケージは Optional チャンネルからのみ入手可能であることに注意してください。[Enabling Supplementary and Optional Repositories](#) を参照してください。

オプションとして **oscap** のインストール後に、**oscap** のバージョンの機能やこのバージョンがサポートする仕様、特定の **oscap** ファイルの保存場所、使用可能な SCAP オブジェクトの種類、他の役立つ情報をチェックできます。これらの情報を表示するには、以下のコマンドを実行します。

```
~]$ oscap -V
OpenSCAP command line tool (oscap) 1.0.4
Copyright 2009--2014 Red Hat Inc., Durham, North Carolina.

==== Supported specifications ====
XCCDF Version: 1.2
OVAL Version: 5.10.1
CPE Version: 2.3
CVSS Version: 2.0
CVE Version: 2.0
Asset Identification Version: 1.1
Asset Reporting Format Version: 1.1

==== Capabilities added by auto-loaded plugins ====
SCE Version: 1.0 (from libopenscap_sce.so.8)

==== Paths ====
Schema files: /usr/share/openscap/schemas
Schematron files: /usr/share/openscap/xsl
Default CPE files: /usr/share/openscap/cpe
Probes: /usr/libexec/openscap

==== Inbuilt CPE names ====
Red Hat Enterprise Linux - cpe:/o:redhat:enterprise_linux
Red Hat Enterprise Linux 5 - cpe:/o:redhat:enterprise_linux:5
Red Hat Enterprise Linux 6 - cpe:/o:redhat:enterprise_linux:6
Red Hat Enterprise Linux 7 - cpe:/o:redhat:enterprise_linux:7
Fedora 16 - cpe:/o:fedoraproject:fedora:16
Fedora 17 - cpe:/o:fedoraproject:fedora:17
Fedora 18 - cpe:/o:fedoraproject:fedora:18
Fedora 19 - cpe:/o:fedoraproject:fedora:19
Fedora 20 - cpe:/o:fedoraproject:fedora:20
Fedora 21 - cpe:/o:fedoraproject:fedora:21
Red Hat Enterprise Linux Optional Productivity Applications -
cpe:/a:redhat:rhel_productivity
Red Hat Enterprise Linux Optional Productivity Applications 5 -
cpe:/a:redhat:rhel_productivity:5

==== Supported OVAL objects and associated OpenSCAP probes ====
system_info                probe_system_info
family                     probe_family
filehash                   probe_filehash
environmentvariable        probe_environmentvariable
textfilecontent54          probe_textfilecontent54
textfilecontent            probe_textfilecontent
variable                   probe_variable
xmlfilecontent             probe_xmlfilecontent
environmentvariable58      probe_environmentvariable58
filehash58                 probe_filehash58
inetlisteningserver        probe_inetlisteningserver
rpminfo                    probe_rpminfo
partition                  probe_partition
```

iflisteners	probe_iflisteners
rpmverify	probe_rpmverify
rpmverifyfile	probe_rpmverifyfile
rpmverifypackage	probe_rpmverifypackage
selinuxboolean	probe_selinuxboolean
selinuxsecuritycontext	probe_selinuxsecuritycontext
file	probe_file
interface	probe_interface
password	probe_password
process	probe_process
runlevel	probe_runlevel
shadow	probe_shadow
uname	probe_uname
xinetd	probe_xinetd
sysctl	probe_sysctl
process58	probe_process58
fileextendedattribute	probe_fileextendedattribute
routingtable	probe_routingtable

oscap を効果的に使用する前に、システムでセキュリティーコンテンツをインストールまたはインポートする必要があります。たとえば、現時点で最も新しくかつ詳細な Linux 向けセキュリティーポリシーを含む SCAP Security Guide (SSG) パッケージ (scap-security-guide) をインストールできます。SCAP Security Guide パッケージをシステムにインストールするには、`root` で次のコマンドを実行します。

```
~]# yum install scap-security-guide
```

システムに `scap-security-guide` をインストールした後は、特に指定されていなければ、SSG セキュリティーコンテンツは `/usr/share/xml/scap/ssg/content/` ディレクトリーに格納され、他のセキュリティーコンプライアンス作業を進めることができます。

個別のニーズに合致する可能性のある既存の SCAP コンテンツの他のソースが必要な場合は、[「その他のリソース」](#) を参照してください。

システムに SCAP コンテンツをインストールした後に、コンテンツへのファイルパスを提供すると **oscap** はコンテンツを処理できます。**oscap** ユーティリティーは SCAP バージョン 1.2 をサポートしており、SCAP バージョン 1.1 および 1.0 とも後方互換性があるので、特別な要件を必要とせずに以前のバージョンの SCAP コンテンツを処理できます。

6.4.2. SCAP コンテンツの表示

SCAP 標準は、数多くのファイル形式を定義します。**oscap** ユーティリティーは、多くの形式に適合するファイルを処理したり、作成したりすることができます。SCAP コンテンツの特定ファイルをさらに処理するには、そのファイルタイプでの **oscap** の使い方を理解する必要があります。特定のファイルの使い方が分からない場合は、ファイルを開いて読んでみるか、**oscap** の **info** モジュールを使うことができます。これはファイルを解析し、ヒューマンリーダブルな形式で関連情報を抽出します。

以下のコマンドを実行すると、SCAP ドキュメントの内部構造を調べることができます。また、ドキュメントタイプ、仕様バージョン、ドキュメントのステータス、ドキュメントの公開日、ドキュメントがファイルシステムにコピーされた日付などの便利な情報が表示されます。

```
oscap info file
```

ここでの *file* は、調べているセキュリティーコンテンツファイルへの完全パスになります。以下の例で **oscap info** コマンドの使用方法を示します。

例6.5 SCAP コンテンツの情報表示

```

~]$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
Document type: Source Data Stream
Imported: 2014-03-14T12:22:01

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-rhel7-xccdf-1.2.xml
Generated: (null)
Version: 1.2
Checklists:
    Ref-Id: scap_org.open-scap_cref_ssg-rhel7-xccdf-1.2.xml
    Profiles:
        xccdf_org.ssgproject.content_profile_test
        xccdf_org.ssgproject.content_profile_rht-ccp
        xccdf_org.ssgproject.content_profile_common
        xccdf_org.ssgproject.content_profile_stig-
rhel7-server-upstream
    Referenced check files:
        ssg-rhel7-oval.xml
        system:
http://oval.mitre.org/XMLSchema/oval-definitions-5
Checks:
    Ref-Id: scap_org.open-scap_cref_ssg-rhel7-oval.xml
    Ref-Id: scap_org.open-scap_cref_output--ssg-rhel7-cpe-oval.xml
    Ref-Id: scap_org.open-scap_cref_output--ssg-rhel7-oval.xml
Dictionaries:
    Ref-Id: scap_org.open-scap_cref_output--ssg-rhel7-cpe-
dictionary.xml

```

6.4.3. システムのスキャン

oscap の最も重要な機能は、ローカルシステムの設定および脆弱性スキャンを実行することです。以下はコマンドの一般的な構文になります。

```
oscap [options] module eval [module_operation_options_and_arguments]
```

oscap ユーティリティーでは、**XCCDF** (eXtensible Configuration Checklist Description Format) ベンチマークと **OVAL** (Open Vulnerability and Assessment Language) 定義の両方で表示される SCAP コンテンツに対してシステムのスキャンを行うことができます。セキュリティポリシーは単一の **OVAL** または **XCCDF** ファイルの場合もあれば、複数の別個の XML ファイルの場合もあります。後者の場合は、各ファイルが異なるコンポーネントを表します (**XCCDF**、**OVAL**、**CPE**、**CVE**、その他)。スキャン結果は、標準出力と XML ファイルの両方にプリントすることができます。この結果のファイルは **oscap** でさらに処理され、人間が判読可能な形式のレポートを生成することができます。以下は、このコマンドの最も一般的な使用例になります。

例6.6 SSG OVAL 定義を使用したシステムのスキャン

すべての定義を評価しながら SSG OVAL 定義ファイルに対してシステムのスキャンを実行するには、以下のコマンドを実行します。

```

~]$ oscap oval eval --results scan-oval-results.xml
/usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml

```


スキャン結果は、**scan-oval-results.xml** ファイルとして現在のディレクトリーに保存されます。

例6.7 SSG OVAL 定義を使用したシステムのスキャン

SSG データストリームファイルで表されるセキュリティポリシーから特定の OVAL 定義を評価するには、以下のコマンドを実行します。

```
~]$ oscap oval eval --id oval:ssg:def:100 --results scan-oval-
results.xml /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

スキャン結果は、**scan-oval-results.xml** ファイルとして現在のディレクトリーに保存されます。

例6.8 SSG XCCDF ベンチマークを使用したシステムのスキャン

システム上の **xccdf_org.ssgproject.content_profile_rht-ccp** プロファイルの SSG XCCDF ベンチマークを実行するには、以下のコマンドを実行します。

```
~]$ oscap xccdf eval --profile
xccdf_org.ssgproject.content_profile_rht-ccp --results scan-xccdf-
results.xml /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

スキャン結果は、**scan-xccdf-results.xml** ファイルとして現在のディレクトリーに保存されます。



注記

--profile コマンドラインの引数は、特定の XCCDF またはデータストリームファイルからセキュリティプロファイルを選択します。利用可能なプロファイルの一覧は、**oscap info** コマンドを実行すると確認できます。**--profile** コマンドライン引数が省略されると、SCAP 標準で必要とされるデフォルトの XCCDF プロファイルが使用されます。デフォルトの XCCDF プロファイルは適切なセキュリティポリシーである場合もありますが、そうでない場合もあることに注意してください。

6.4.4. レポートおよびガイドの生成

oscap のもうひとつの便利な機能は、ヒューマンリーダブルな形式で SCAP コンテンツを生成することです。**oscap** ユーティリティーを使うと、XML ファイルを HTML またはプレーンテキスト形式に変換できます。この機能は、セキュリティガイドやチェックリストの生成に使用できます。これらは、セキュアなシステム設定のガイドになるとともに、情報ソースの役割も果たします。システムスキャンの結果も、読みやすい結果レポートに変換することができます。一般的なコマンド構文は以下のようになります。

```
oscap module generate sub-module [specific_module/sub-
module_options_and_arguments] file
```

ここでの *module* は **xccdf** または **oval** に、*sub-module* は生成されたドキュメントのタイプに、*file* は XCCDF または OVAL ファイルになります。

以下でこのコマンドの最も一般的な使用例を示します。

例6.9 チェックリスト付きガイドの生成

チェックリスト付きの SSG ガイドを **xccdf_org.ssgproject.content_profile_rht-ccp** プロファイル用に作成するには、以下のコマンドを実行します。

```
~]$ oscap xccdf generate guide --profile
xccdf_org.ssgproject.content_profile_rht-ccp
/usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml > ssg-guide-
checklist.html
```

ガイドは、**ssg-guide-checklist.html** ファイルとして現在のディレクトリーに保存されます。

例6.10 SSG OVAL スキャン結果をレポートに変換する

SSG OVAL スキャンの結果を HTML ファイルに変換するには、以下のコマンドを実行します。

```
~]$ oscap oval generate report scan-oval-results.xml > ssg-scan-oval-
report.html
```

結果のレポートは **ssg-scan-oval-report.html** ファイルとして現在のディレクトリーに保存されます。この例では、**scan-oval-results.xml** ファイルが保存されている場所からこのコマンドが実行されることを想定しています。それ以外の場合は、スキャン結果を含んでいるファイルの完全修飾パスを指定する必要があります。

例6.11 SSG XCCDF スキャン結果をレポートに変換する

SSG XCCDF スキャンの結果を HTML ファイルに変換するには、以下のコマンドを実行します。

```
~]$ oscap xccdf generate report scan-xccdf-results.xml > scan-xccdf-
report.html
```

結果レポートは **ssg-scan-xccdf-report.html** ファイルとして現在のディレクトリーに保存されます。別の方法では、**--report** コマンドライン引数を使用すると、スキャン時にこのレポートが生成できます。

```
~]$ oscap xccdf eval --profile
xccdf_org.ssgproject.content_profile_rht-ccp --results scan-xccdf-
results.xml --report scan-xccdf-report.html
/usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

6.4.5. SCAP コンテンツの検証

システム上でセキュリティーポリシーを使い始める前に、ポリシーを検証してポリシー内の構文エラーやセマンティックエラーを避けるようにしてください。**oscap** ユーティリティーを使うと、SCAP XML スキーマに対してセキュリティーコンテンツを検証することができます。検証結果は、標準エラー streams (stderr) にプリントされます。このような検証コマンドの一般的な構文は以下のようになります。

oscap module validate [module_options_and_arguments] file

ここでの *file* は、検証されているファイルへの完全パスになります。唯一の例外はデータストリームモジュール (ds) で、これは **validate** ではなく **sds-validate** 演算を使用します。以下の例で示されているように、特定のデータストリーム内の SCAP コンポーネントは自動的に検証され、各コンポーネントは別個に指定されないことに注意してください。

```
~]$ oscap ds sds-validate /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

OVAL 仕様のような特定の SCAP コンテンツの場合、スキマトロン検証も実行できます。スキマトロン検証は標準の検証よりも遅いものの、より深い分析を提供するのでより多くのエラーを検出できます。以下の SSG 例では、このコマンドの通常の使用方法を示しています。

```
~]$ oscap oval validate --schematron /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

6.4.6. OpenSCAP を使用してシステムを修正

OpenSCAP を使用すると、コンプライアンスに準拠しない状態のシステムを自動的に修正できます。システムの修正には、XCCDF ファイルと手順が必要です。scap-security-guide package には、特定の修正手順が含まれます。

システムの修正は以下の 2 つの手順で構成されます。

1. **OpenSCAP** は XCCDF 評価を実行します。
2. 結果の評価は、OVAL 定義を評価することによって実行されます。失敗した各ルールは修正の候補としてマークされます。
3. **OpenSCAP** は適切な修正要素を検索および解決し、環境を準備して、修正スクリプトを実行します。
4. 修正スクリプトの出力は **OpenSCAP** によって取得され、**rule-result** 要素内に格納されます。修正スクリプトの戻り値も格納されます。
5. **OpenSCAP** が修正スクリプトを実行すると、すぐに OVAL 定義が再び評価されます (修正スクリプトが正しく適用されたことを検証します)。この 2 回目の実行時に、OVAL 評価が成功を返す場合、ルールの結果は **fixed** になり、それ以外の場合は **error** になります。
6. 修正の詳細な結果は出力 XCCDF ファイルに格納されます。このファイルには 2 つの **TestResult** 要素が含まれます。最初の **TestResult** 要素は修正前のスキャンを表します。2 つ目の **TestResult** は最初の要素から派生し、修正結果を含みます。

修正に関して、**OpenSCAP** には、オンライン、オフライン、およびレビューの 3 つの操作モードがあります。

6.4.6.1. OpenSCAP オンライン修正

オンライン修正は、スキャン時に修正要素を実行します。評価と修正は単一のコマンドの一部として実行されます。

オンライン修正を有効にするには、**--remediate** コマンドラインオプションを使用します。たとえば、scap-security-guide パッケージを使用してオンライン修正を実行するには、以下のコマンドを実行します。

■

```
~]$ oscap xccdf eval --remediate --profile
xccdf_org.ssgproject.content_profile_rht-ccp --results scan-xccdf-
results.xml /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

このコマンドの出力は 2 つのセクションから構成されます。最初のセクションには、修正前のスキャンの結果が示され、2 目目のセクションには修正の適用後のスキャンの結果が示されます。2 目目の部分には **fixed** と **error** の結果のみを含めることができます。**fixed** 結果は、スキャンが修正が合格した後に実行されたことを示します。**error** 結果は修正の適用後であっても評価が合格していないことを示します。

6.4.6.2. OpenSCAP オフライン修正

オフライン修正では、修正実行を延期できます。最初のステップでは、システムの評価のみが実行され、結果は XCCDF ファイルの **TestResult** 要素に格納されます。

2 目目のステップでは、**oscap** が修正スクリプトを実行し、結果を検証します。データが失われないよう結果を入力ファイルに格納することが推奨されます。オフライン修正中に、**OpenSCAP** は入力ファイルに基づいて新しい **TestResult** 要素を作成し、すべてのデータを継承します。新しく作成された **TestResult** で異なる唯一の点は失敗した **rule-result** 要素です。これらの要素に対して修正が実行されます。

scap-security-guide パッケージを使用してオフライン修正を実行するには、以下のコマンドを実行します。

```
~]$ oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_rht-
ccp --results scan-xccdf-results.xml /usr/share/xml/scap/ssg/content/ssg-
rhel7-ds.xml
```

```
~]$ oscap xccdf remediate --results scan-xccdf-results.xml scan-xccdf-
results.xml
```

6.4.6.3. OpenSCAP 修正レビュー

レビューモードでは、ユーザーがレビューのために修正手順をファイルに保存できます。この操作中に修正コンテンツは実行されません。

修正手順をシェルスクリプトの形式で生成するには、以下のコマンドを実行します。

```
~]$ oscap xccdf generate fix --template urn:xccdf:fix:script:sh --profile
xccdf_org.ssgproject.content_profile_rht-ccp --output my-remediation-
script.sh /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

6.5. DOCKER での OPENS CAP の使用

oscap-docker コマンドラインユーティリティでは、**oscap** プログラムを使用して docker 形式のコンテナイメージやコンテナをローカルのシステムとほぼ同じようにスキャンすることができます。

以下のセクションでは、**oscap-docker** のインストールについて説明し、使用方法の基本例を紹介します。サブコマンドに関する情報は、**oscap-docker** または **oscap** に **--help** オプションを指定して使用します。

イメージおよびコンテナのスキャンを有効にするには、docker パッケージもインストールする必要があります。**Docker** のインストールに関する説明は、『Getting Started with Containers』の「[Getting Docker in Red Hat Enterprise Linux 7](#)」の章を参照してください。

以下のコマンドを実行して **oscap-docker** をインストールします。

```
# yum install openscap-utils
```

例6.12 oscap-docker の使用

```
oscap-docker scan_target[-cve] target_identifier [oscap-arguments]
```

scan_target はスキャンするイメージまたはコンテナ、*target_identifier* はターゲットの名前または ID に置き換えてください。

以下のコマンドの 2 番目は、コンテナイメージをアタッチし、オペレーティングシステムの系統およびバージョンを判断して、指定のシステムに適した CVE ストリームをダウンロードしてから、最終的に脆弱性スキャンを実行します。

```
# docker images
REPOSITORY                                TAG                IMAGE ID
registry.access.redhat.com/rhel7          latest            c453594215e4
```

```
# oscap-docker image-cve registry.access.redhat.com/rhel7
```

以下のコマンドの 2 番目は、実行中のコンテナの chroot 環境で **OpenSCAP** スキャンを実行します。結果は、定義したマウントポイントによって、コンテナイメージのスキャンとは異なる可能性があります。今回の例では、OVAL パッチ定義 **com.redhat.rhsa-all.xml** を使用しました。

```
# docker ps
CONTAINER ID    IMAGE                COMMAND             NAMES
5ef05eef4a01   registry.access.redhat.com/rhel7  "/bin/bash"        sleepy_kirch
```

```
# oscap-docker container 5ef05eef4a01 oval eval com.redhat.rhsa-all.xml
```

6.6. ATOMIC での OPENSAP の使用

システム上にあるすべてのコンテナイメージおよびコンテナが既知の CVE 脆弱性や一般的な設定ミスがないかを検証するには、**atomic scan** コマンドを使用して **OpenSCAP** スキャン機能を使用します。

Atomic スキャン

お使いのシステム上に、コンテナ管理用に **atomic** ツールをインストールするには、root で以下のコマンドを実行します。

```
# yum install atomic
```

atomic ツールをインストールした後はスキャナーも必要です。Red Hat は**OpenSCAP** ベースの *rhel7/openscap* docker イメージを選択することを推奨します。root として以下のコマンドを実行して、インストールします。

```
# atomic install rhel7/openscap
```

OpenSCAP docker イメージが設定されたら、**atomic scan** コマンドを実行します。以下のコマンドを root で実行すると、コンテナおよびコンテナイメージをスキャンします。

```
# atomic scan $ID
```

\$ID は、コンテナの ID に置き換えます。全コンテナイメージまたはコンテナをスキャンする場合には、それぞれ **--images** または **--containers** ディレクティブを使用します。どちらのタイプもスキャンするには、**--all** ディレクティブを使用します。

OpenSCAP スキャナー

atomic scan のデフォルトスキャナーである *rhel7/openscap* コンテナイメージは、Red Hat Enterprise Linux システムのみを対象にするスキャンタイプ 2 種をサポートします。サポートされるスキャンタイプは、以下のコマンドを root で実行すると表示できます。

```
# atomic scan --scanner openscap --list
```

デフォルトのスキャンタイプは *CVE scan* です。Red Hat が発表した CVE OVAL 定義に指定されている既知のセキュリティー脆弱性の対象であるかどうかを確認する際に使用します。



警告

CVE scan タイプで使用する OVAL の定義は、ビルドプロセス時におくコンテナイメージにバンドルされるので、常に最新なわけではありません。

サポートされているスキャンタイプの 2 つ目は *standards_compliance* ですが、バンドルされた SCAP Security Guide の Standard System Security Profile が評価に使用されます。これは、Red Hat Enterprise Linux のセキュリティーベースラインのプロファイルです。

例6.13 Atomic スキャンでのコンテナイメージのスキャン

以下に示した **atomic scan** の使用例では、Red Hat Enterprise Linux イメージをスキャンして、**--verbose** ディレクティブで検出された脆弱性を表示する方法を紹介しています。

```
#docker pull rhel7
Using default tag: latest
98a88a8b722a: Download complete
# atomic scan 98a88a8b722a
Container/Image      Cri      Imp      Med      Low
-----
98a88a8b722a         0        0        0        0
# atomic scan --verbose 98a88a8b722a
docker run -t --rm -v /etc/localtime:/etc/localtime -v /run/atomic/2016-10-14-06-42-55-991951:/scanin -v /var/lib/atomic/openscap/2016-10-14-
```

```
06-42-55-991951:/scanout:rw,Z -v /etc/oscaped:/etc/oscaped:ro
rhel7/openscap oscaped-evaluate scan --no-standard-compliance --targets
chroots-in-dir:///scanin --output /scanout
INFO:OpenSCAP Daemon one-off evaluator 0.1.6
WARNING:Can't import the 'docker' package. Container scanning
functionality will be disabled.
INFO:Creating tasks directory at '/var/lib/oscaped/tasks' because it
didn't exist.
INFO:Creating results directory at '/var/lib/oscaped/results' because it
didn't exist.
INFO:Creating results work in progress directory at
'/var/lib/oscaped/work_in_progress' because it didn't exist.
INFO:Evaluated EvaluationSpec, exit_code=0.
INFO:Evaluated EvaluationSpec, exit_code=0.
INFO:[100.00%] Scanned target
'chroot:///scanin/98a88a8b722a71835dd761c88451c681a8f1bc6e577f90d4dc8b23
4100bd4861'

98a88a8b722a (registry.access.redhat.com/rhel7:latest)

98a88a8b722a passed the scan

Files associated with this scan are in /var/lib/atomic/openscap/2016-10-
14-06-42-55-991951.
```



注記

[Red Hat Enterprise Linux Atomic Host の製品ドキュメント](#)で、**atomic** コマンドの使用方法やコンテナの詳細にわたる説明を入手できます。Red Hat カスタマーポータルでも「[Atomic command line interface \(CLI\)](#)」のガイドが提供されています。

6.7. RED HAT SATELLITE での OPENSAP の使用

複数の Red Hat Enterprise Linux システムを稼働している際には、すべてのシステムがセキュリティポリシーにしたがっており、リモートの 1 か所からスキャンと評価を実行することが重要になります。これは、Satellite クライアントに spacewalk-oscaped パッケージがインストールされている Red Hat Satellite 5.5 以降を使用することで達成できます。このパッケージは、**Red Hat Network Tools** チャンネルから入手できます。[How to enable/disable a repository using Red Hat Subscription-Manager?](#) を参照してください。

このソリューションは、2 つの方法でのセキュリティコンプライアンススキャンの実行、表示、およびスキャン結果のさらなる処理をサポートしています。**OpenSCAP Satellite Web Interface** を使うか **Satellite API** からコマンドとスクリプトを実行することができます。このソリューションのセキュリティコンプライアンス、要件、機能に関する詳細情報は、[Red Hat Satellite ドキュメント](#)を参照してください。

6.8. 実用的な使用例

本セクションでは、Red Hat 製品用に提供されている特定のセキュリティコンテンツの実用的な使用例を紹介します。

6.8.1. Red Hat 製品のセキュリティ脆弱性の監査

Red Hat は自社製品用に継続的に OVAL 定義を提供しています。これらの定義により、インストール済みソフトウェアにおける脆弱性の完全自動化された監査が可能になります。このプロジェクトに関する詳細情報は、<http://www.redhat.com/security/data/metrics/> を参照してください。これらの定義をダウンロードするには、以下のコマンドを実行します。

```
~]$ wget http://www.redhat.com/security/data/oval/com.redhat.rhsa-all.xml
```

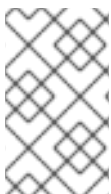
Red Hat Satellite 5 のユーザーには、パッチ定義の XCCDF の部分が便利かもしれません。これらの定義をダウンロードするには、以下のコマンドを実行します。

```
~]$ wget http://www.redhat.com/security/data/metrics/com.redhat.rhsa-all.xccdf.xml
```

システムにインストール済みのソフトウェアに関するセキュリティー脆弱性を監査するには、以下のコマンドを実行します。

```
~]$ oscap oval eval --results rhsa-results-oval.xml --report oval-report.html com.redhat.rhsa-all.xml
```

oscap ユーティリティーは、National Vulnerability Database にリンクしている CVE に Red Hat セキュリティーアドバイザリーをマッピングします。そして、どのセキュリティーアドバイザリーが適用されていないかを報告します。



注記

これらの OVAL 定義は、Red Hat がリリースしているソフトウェアや更新のみをカバーしていることに注意してください。サードパーティーのソフトウェアのパッチステータスを検出するには、追加の定義を提供する必要があります。

6.8.2. SCAP セキュリティーガイドを使ったシステム設定の監査

SCAP Security Guide (SSG) プロジェクトのパッケージ (scap-security-guide) には、Linux システム向けの最新のセキュリティーポリシーセットが含まれます。システムに SCAP Security Guide パッケージをインストールするには、root で以下のコマンドを実行します。

```
~]# yum install scap-security-guide
```

scap-security-guide の一部は Red Hat Enterprise Linux 7 の設定のガイダンスでもあります。scap-security-guide で利用可能なセキュリティーコンテンツを調べるには、以下のように **oscap info** モジュールを使用します。

```
~]$ oscap info /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```

このコマンドの出力は SSG ドキュメントの概要で、利用可能な設定プロファイルが含まれています。ご使用のシステム設定を監査するには、適切なプロファイルを選択して適切な評価コマンドを実行します。たとえば、以下のコマンドは、Red Hat Certified Cloud Providers 用のドラフト SCAP プロファイルに対し特定のシステムを評価するために使われます。

```
~]$ oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_rht-ccp --results ssg-rhel7-xccdf-result.xml --report ssg-rhel7-report.html /usr/share/xml/scap/ssg/content/ssg-rhel7-ds.xml
```


6.9. その他のリソース

様々なセキュリティコンプライアンスの分野に関する詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- `oscap(8)` — The manual page for the **oscap** command-line utility provides a complete list of available options and their usage explanation.
- `scap-workbench(8)` — The manual page for the **SCAP Workbench** application provides a basic information about the application as well as some links to potential sources of SCAP content.
- `scap-security-guide(8)` — The manual page for **scap-security-guide** provides further documentation about the various available SCAP security profiles. Examples how to utilize the provided benchmarks using the **OpenSCAP** utility are provided as well.

オンラインのドキュメント

- [The OpenSCAP project page](#) — OpenSCAP プロジェクトのホームページでは、**oscap** ユーティリティと SCAP に関連する他のコンポーネントおよびプロジェクトの詳細情報が提供されています。
- [SCAP Workbench プロジェクトページ](#) — SCAP Workbench プロジェクトのホームページでは、**scap-workbench** アプリケーションの詳細情報が提供されています。
- [SCAP Security Guide \(SSG\) プロジェクトページ](#) — SSG プロジェクトのホームページでは、Red Hat Enterprise Linux 向けの最新セキュリティコンテンツが提供されています。
- [National Institute of Standards and Technology \(NIST\) SCAP page](#) — このページには SCAP 関連の大量の資料があります。SCAP の出版物や仕様、SCAP 検出プログラムもあります。
- [National Vulnerability Database \(NVD\)](#) — このページは、SCAP コンテンツおよび他の SCAP 標準ベースの脆弱性管理データの最大のリポジトリです。
- [Red Hat OVAL content repository](#) — Red Hat Enterprise Linux システム向けの OVAL 定義を含むリポジトリです。
- [MITRE CVE](#) — これは、MITRE corporation が提供する既知のセキュリティ脆弱性に関するデータベースです。
- [MITRE OVAL](#) — このページは、MITRE corporation が提供する OVAL 関連のプロジェクトを紹介しています。OVAL 関連の他の情報に加え、OVAL 言語の最新バージョンや 2 万 2 千を超える OVAL 定義を数える巨大な OVAL コンテンツのリポジトリがあります。
- [Red Hat Satellite ドキュメント](#) — このガイドセットでは、OpenSCAP を使って複数のシステム上でシステムセキュリティを維持する方法が説明されています。

第7章 米連邦政府の標準および規制

どの組織も、米連邦政府または業界のセキュリティー仕様、標準および規制の遵守に取り組むことで一定のセキュリティーレベルを維持することができます。本章では、これらの標準および規制のいくつかについて説明します。

7.1. 連邦情報処理標準 (FIPS: FEDERAL INFORMATION PROCESSING STANDARD)

連邦情報処理標準 (FIPS) のパブリケーションである 140-2 は、暗号モジュールの品質を検証するために連邦政府と業界の作業部会によって開発されたコンピューターセキュリティー標準です。FIPS のパブリケーション (140-2 を含む) は以下の URL にあります。<http://csrc.nist.gov/publications/PubsFIPS.html>。Red Hat Enterprise Linux 7.3 のリリース時点で 140-3 のパブリケーションは草稿なので、これが完成版の標準ではない可能性があることに注意してください。

FIPS 140-2 標準は、暗号化ツールが正しく実装されていることを確認します。FIPS 標準の上記の各レベルや他の仕様についての詳細は、正規の FIPS 140-2 標準 (<http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>) を参照してください。

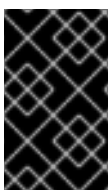
すべての FIPS 140-2 証明書の完全一覧については <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm> を参照してください。コンプライアンス要件については [Red Hat Government: Standards page](#) を参照してください。

7.1.1. FIPS モードの有効化

Red Hat Enterprise Linux を連邦情報処理標準 (FIPS) パブリケーション 140-2 に準拠させるには、いくつかの変更を加えて認定済み暗号モジュールを使用することが必要です。システムのインストール中または後に、FIPS モードを有効にしてください。

システムのインストール時

厳格な **FIPS 140-2** へのコンプライアンスを満たすには、システムのインストール中にカーネルコマンドラインに **fips=1** カーネルオプションを追加します。このオプションでは、FIPS で承認されているアルゴリズムを使って鍵を生成し、継続的なモニタリングテストを実施することができます。インストール後には、システムは自動的に FIPS モードで起動します。



重要

マウスを移動させるか、キーを何度も押して、インストールプロセス中にシステムに十分なエントロピーを確保します。推奨されるキー入力の回数は 256 回以上です。255 回以下の場合は、一意でない鍵が生成される可能性があります。

システムインストールの後

システムのインストール後にシステム、カーネル、ユーザー領域を FIPS モードに切り替えるには、以下のステップに従います。

1. プレリンクが無効になっていることを確認します。

モジュール内の整合性検証の適切な実行にあたり、ライブラリーやバイナリーのプレリンクは無効にする必要があります。プレリンクは `prelink` パッケージにより行われますが、デフォルトではインストールされません。プレリンクを無効にするには、`/etc/sysconfig/prelink` 設定ファイルで **PRELINKING=no** オプションを設定します。全システムファイルの既存のプレリンクを無効にするには、**`prelink -u -a`** コマンドを使用します。

2. dracut-fips パッケージをインストールします。

```
~]# yum install dracut-fips
```

AES New Instructions (AES-NI) サポートのある CPU は、dracut-fips-aesni パッケージもインストールします。

```
~]# yum install dracut-fips-aesni
```

3. **initramfs** ファイルを再生成します。

モジュール内の整合性検証を有効化して、カーネルの起動中に必要なモジュールがすべて揃っているようにするには、**initramfs** ファイルを再生成する必要があります。

```
~]# dracut -v -f
```

**警告**

この操作により、既存の **initramfs** ファイルが上書きされます。

4. ブートローダーの設定を変更します。

FIPS モードで起動するには、ブートローダーのカーネルコマンドラインに **fips=1** オプションを追加します。**/boot** または **/boot/EFI** パーティションが個別のパーティションにある場合には、**boot=<partition>** パラメーター (<partition> は /boot または /boot/EFI を指す) もカーネルのコマンドラインに追加してください。

ブートパーティションを特定するには、以下のコマンドを実行します。

```
~]$ df /boot
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda1              495844       53780   416464  12% /boot
```

ブート間でデバイス名が変更されても **boot=** 設定オプションが機能するようにするには、以下のコマンドを実行してパーティションの UUID (Universally Unique Identifier) を特定します。

```
~]$ blkid /dev/sda1
/dev/sda1: UUID="05c000f1-f899-467b-a4d9-d5ca4424c797" TYPE="ext4"
```

UUID をカーネルコマンドラインに追加します。

```
boot=UUID=05c000f1-f899-467b-a4d9-d5ca4424c797
```

お使いのブートローダーによっては、以下の変更を加えてください。

- grub2

`/etc/default/grub` ファイルの `GRUB_CMDLINE_LINUX` キーに、`fips=1` と `boot=<partition of /boot or /boot/EFI>` のオプションを追加します。`/etc/default/grub` への変更を適用するには、以下のように `grub.cfg` ファイルを再構築します。

- BIOS ベースのマシンでは、**root** で以下のコマンドを実行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンドを実行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

- `zipl` (IBM z Systems アーキテクチャーのみ)

カーネルコマンドラインで `/etc/zipl.conf` に `fips=1` および `boot=<partition of /boot>` オプションを追加し、**root** として以下のコマンドを実行して変更を適用します。

```
~]# zipl
```

5. システムを再起動します。

7.2. NISPOM (NATIONAL INDUSTRIAL SECURITY PROGRAM OPERATING MANUAL)

National Industrial Security Program (NISP) の 1 つの構成要素である NISPOM (DoD 5220.22-M と呼ばれる) は、すべての政府指定業者向けに、秘密情報 (classified information) に関する一連の手順と要件を規定しています。現行の NISPOM は、2006 年 2 月 28 日付けのもので、2013 年 3 月 28 日からのメジャーな変更が組み込まれています。NISPOM 文書は以下の URL からダウンロードできます。<http://www.nispom.org/NISPOM-download.html>

7.3. PCI DSS (PAYMENT CARD INDUSTRY DATA SECURITY STANDARD)

(ソース: <https://www.pcisecuritystandards.org/about/index.shtml>) **PCI Security Standards Council** は、2006 年に発足したグローバル規模の開かれた協議会であり、データセキュリティスタンダード (**DSS: Data Security Standard**) を含む **PCI** セキュリティー基準の開発、管理、教育および普及に関する討議の場を提供しています。

PCI DSS 標準は、https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml からダウンロードできます。

7.4. セキュリティー技術導入ガイド (SECURITY TECHNICAL IMPLEMENTATION GUIDE)

セキュリティ技術導入ガイド (STIG: Security Technical Implementation Guide) は、ソフトウェアとハードウェアの標準化された安全なインストールと保守についての方法論です。

STIG についての詳細は、以下の URL を参照してください。<http://iase.disa.mil/stigs/Pages/index.aspx>

付録A 暗号の標準

A.1. 同期式の暗号

A.1.1. 高度暗号化標準 — AES

In cryptography, the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. Government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor, the Data Encryption Standard (DES).^[2]

A.1.1.1. AES の歴史

AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process. Fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable. It became effective as a standard May 26, 2002. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information (see the Security section in the Wikipedia article on AES).^[3]

The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process. Rijndael is a portmanteau of the names of the two inventors.^[4]

A.1.2. データ暗号化標準 — DES

The Data Encryption Standard (DES) is a block cipher (a form of shared secret encryption) that was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 and which has subsequently enjoyed widespread use internationally. It is based on a symmetric-key algorithm that uses a 56-bit key. The algorithm was initially controversial with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny which motivated the modern understanding of block ciphers and their cryptanalysis.^[5]

A.1.2.1. DES の歴史

DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes. There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are unfeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES).^[6]

In some documentation, a distinction is made between DES as a standard and DES the algorithm which is referred to as the DEA (the Data Encryption Algorithm).^[7]

A.2. 公開鍵暗号

Public-key cryptography is a cryptographic approach, employed by many cryptographic algorithms and cryptosystems, whose distinguishing characteristic is the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Using the techniques of public key-private key cryptography, many methods of protecting communications or authenticating messages formerly unknown have become practical. They do not require a secure initial exchange of one or more secret keys as is required when using symmetric key algorithms. It can also be used to create digital signatures.^[8]

Public key cryptography is a fundamental and widely used technology around the world, and is the approach which underlies such Internet standards as Transport Layer Security (TLS) (successor to SSL), PGP and GPG.^[9]

The distinguishing technique used in public key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same as the key used to decrypt it. Each user has a pair of cryptographic keys — a public key and a private key. The private key is kept secret, whilst the public key may be widely distributed. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot be feasibly (ie, in actual or projected practice) derived from the public key. It was the discovery of such algorithms which revolutionized the practice of cryptography beginning in the middle 1970s.^[10]

In contrast, Symmetric-key algorithms, variations of which have been used for some thousands of years, use a single secret key shared by sender and receiver (which must also be kept private, thus accounting for the ambiguity of the common terminology) for both encryption and decryption. To use a symmetric encryption scheme, the sender and receiver must securely share a key in advance.^[11]

Because symmetric key algorithms are nearly always much less computationally intensive, it is common to exchange a key using a key-exchange algorithm and transmit data using that key and a symmetric key algorithm. PGP, and the SSL/TLS family of schemes do this, for instance, and are called hybrid cryptosystems in consequence.^[12]

A.2.1. Diffie-Hellman

Diffie-Hellman key exchange (D-H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.^[13]

A.2.1.1. Diffie-Hellman の歴史

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976, although it later emerged that it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, by Malcolm J. Williamson but was kept classified. In 2002, Hellman suggested the algorithm be called Diffie-Hellman-Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public-key cryptography (Hellman, 2002).^[14]

Although Diffie-Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).^[15]

U.S. Patent 4,200,770, now expired, describes the algorithm and credits Hellman, Diffie, and Merkle as inventors.^[16]

A.2.2. RSA

暗号学において、RSA (初めて公に説明した Rivest、Shamir および Adleman の頭文字を表す) は公開鍵暗号のアルゴリズムです。これは、暗号と署名のどちらにも適しているとされる最初のアルゴリズムであり、当初の公開鍵暗号における主要な優位性の 1 つとなってきました。RSA は、電子商取引のプロトコルで広く使用されており、十分な長さの鍵と最新の実装が使用されていることからセキュアであると考えられています。

A.2.3. DSA

DSA (Digital Signature Algorithm) is a standard for digital signatures, a United States federal government standard for digital signatures. DSA is for signatures only and is not an encryption algorithm. ^[17]

A.2.4. SSL/TLS

トランスポート層セキュリティ (TLS: Transport Layer Security) とその前身である Secure Socket Layer (SSL) は、インターネットなどのネットワーク上の通信に対してセキュリティを提供する暗号プロトコルです。TLS と SSL は、エンドツーエンドでトランスポート層におけるネットワーク接続のセグメントを暗号化します。

Several versions of the protocols are in widespread use in applications like web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP).^[18]

A.2.5. Cramer-Shoup 暗号システム

The Cramer-Shoup system is an asymmetric key encryption algorithm, and was the first efficient scheme proven to be secure against adaptive chosen ciphertext attack using standard cryptographic assumptions. Its security is based on the computational intractability (widely assumed, but not proved) of the decisional Diffie-Hellman assumption. Developed by Ronald Cramer and Victor Shoup in 1998, it is an extension of the ElGamal cryptosystem. In contrast to ElGamal, which is extremely malleable, Cramer-Shoup adds additional elements to ensure non-malleability even against a resourceful attacker. This non-malleability is achieved through the use of a collision-resistant hash function and additional computations, resulting in a ciphertext which is twice as large as in ElGamal.^[19]

A.2.6. ElGamal 暗号

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key agreement. It was described by Taher ElGamal in 1985. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems.^[20]

[2] "Advanced Encryption Standard." **Wikipedia**. 14 November 2009
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[3] "Advanced Encryption Standard." **Wikipedia**. 14 November 2009
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[4] "Advanced Encryption Standard." **Wikipedia**. 14 November 2009

http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[5] "Data Encryption Standard." **Wikipedia**. 14 November 2009
http://en.wikipedia.org/wiki/Data_Encryption_Standard

[6] "Data Encryption Standard." **Wikipedia**. 14 November 2009
http://en.wikipedia.org/wiki/Data_Encryption_Standard

[7] "Data Encryption Standard." **Wikipedia**. 14 November 2009
http://en.wikipedia.org/wiki/Data_Encryption_Standard

[8] "Public-key Encryption." **Wikipedia**. 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[9] "Public-key Encryption." **Wikipedia**. 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[10] "Public-key Encryption." **Wikipedia**. 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[11] "Public-key Encryption." **Wikipedia**. 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[12] "Public-key Encryption." **Wikipedia**. 14 November 2009 http://en.wikipedia.org/wiki/Public-key_cryptography

[13] "Diffie-Hellman." **Wikipedia**. 14 November 2009 <http://en.wikipedia.org/wiki/Diffie-Hellman>

[14] "Diffie-Hellman." **Wikipedia**. 14 November 2009 <http://en.wikipedia.org/wiki/Diffie-Hellman>

[15] "Diffie-Hellman." **Wikipedia**. 14 November 2009 <http://en.wikipedia.org/wiki/Diffie-Hellman>

[16] "Diffie-Hellman." **Wikipedia**. 14 November 2009 <http://en.wikipedia.org/wiki/Diffie-Hellman>

[17] "DSA." **Wikipedia**. 24 February 2010 http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

[18] "TLS/SSL." **Wikipedia**. 24 February 2010 http://en.wikipedia.org/wiki/Transport_Layer_Security

[19] "Cramer-Shoup cryptosystem." **Wikipedia**. 24 February 2010
http://en.wikipedia.org/wiki/Cramer-Shoup_cryptosystem

[20] "ElGamal encryption" **Wikipedia**. 24 February 2010
http://en.wikipedia.org/wiki/ElGamal_encryption

付録B AUDIT システムのリファレンス

B.1. AUDIT イベントフィールド

表B.1「イベントフィールド」では、現在サポートされている Audit イベントフィールドを一覧表示しています。イベントフィールドとは、Audit ログファイルで等号記号の前にある値です。

表B.1 イベントフィールド

イベントフィールド	説明
a0、a1、a2、a3	システムコールの最初の 4 つの引数を十六進法で記録します。
acct	ユーザーのアカウント名を記録します。
addr	IPv4 または IPv6 アドレスを記録します。このフィールドは通常、 hostname フィールドの後に来て、ホスト名が解決するアドレスを含みます。
arch	システムの CPU アーキテクチャーについての情報を十六進法にエンコードして記録します。
audit	Audit ユーザー ID を記録します。この ID は、ログイン時にユーザーに割り当てられ、ユーザーの ID が変更された後でもすべてのプロセスに引き継がれます (たとえば、 su - john コマンドでユーザーアカウントを切り替えた場合)。
capability	Records the number of bits that were used to set a particular Linux capability. For more information on Linux capabilities, refer to the capabilities(7) man page.
cap_fi	引き継いだファイルのシステムベースの機能設定に関連したデータを記録します。
cap_fp	許可されたファイルのシステムベースの機能設定に関連したデータを記録します。
cap_pe	効果的なプロセスベースの機能設定に関連したデータを記録します。
cap_pi	継承されたプロセスベースの機能設定に関連したデータを記録します。
cap_pp	許可されたプロセスベースの機能設定に関連したデータを記録します。
cgroup	Audit イベント生成時のプロセスが含まれる cgroup へのパスを記録します。

イベントフィールド	説明
cmd	実行されたコマンドライン全体を記録します。これは、 exe フィールドがたとえば /bin/bash を記録するようなシェルインタプリターの場合に便利です。シェルインタプリターと cmd フィールドは、 helloworld.sh --help のような実行されたコマンドラインの残りの部分を記録するからです。
comm	実行されたコマンドを記録します。これは、 exe フィールドがたとえば /bin/bash を記録するようなシェルインタプリターの場合に便利です。シェルインタプリターと comm フィールドは、 helloworld.sh のような実行されたスクリプト名を記録するからです。
cwd	システムコールが開始されたディレクトリーへのパスを記録します。
data	TTY 記録に関連するデータを記録します。
dev	イベントで記録されたファイルまたはディレクトリーを含むデバイスのマイナーおよびメジャー ID を記録します。
devmajor	メジャーデバイス ID を記録します。
devminor	マイナーデバイス ID を記録します。
egid	分析されているプロセスを開始したユーザーの実効グループ ID を記録します。
euid	分析されているプロセスを開始したユーザーの実効ユーザー ID を記録します。
exe	分析されているプロセスを開始するために使用された実行可能ファイルへのパスを記録します。
exit	システムコールが返した終了コードを記録します。この値は、システムコールによって異なります。以下のコマンドを使うと、この値をヒューマンリーダブルなものに変換できます。 ausearch --interpret --exit exit_code
family	IPv4 または IPv6 の使用されたアドレスプロトコルのタイプを記録します。
filetype	ファイルのタイプを記録します。
flags	ファイルシステム名のフラグを記録します。
fsgid	分析されているプロセスを開始したユーザーのファイルシステムグループ ID を記録します。
fsuid	分析されているプロセスを開始したユーザーのファイルシステムユーザー ID を記録します。

イベントフィールド	説明
gid	グループ ID を記録します。
hostname	ホスト名を記録します。
icmptype	受信した Internet Control Message Protocol (ICMP) パッケージのタイプを記録します。このフィールドを含む Audit メッセージは通常、 iptables が生成します。
id	変更されたアカウントのユーザー ID を記録します。
inode	Audit イベントで記録されたファイルまたはディレクトリーに関連する inode 番号を記録します。
inode_gid	inode の所有者のグループ ID を記録します。
inode_uid	inode の所有者のユーザー ID を記録します。
items	この記録にアタッチされたパス記録の数を記録します。
key	Audit ログで特定のイベントを生成したルールに関連付けられているユーザー定義の文字列を記録します。
list	Audit ルールリストの ID を記録します。以下が既知の ID リストになります。 <ul style="list-style-type: none"> • 0 — user • 1 — task • 4 — exit • 5 — exclude
mode	ファイルまたはディレクトリーのパーミッションを数字表記にエンコードして記録します。
msg	記録のタイムスタンプと一意の ID、または各種のイベント固有の <name>=<value> ペアを記録します。これらは、カーネルもしくはユーザースペースアプリケーションが提供します。
msgtype	ユーザーベースの AVC 拒否の場合に返されたメッセージのタイプを記録します。このメッセージタイプは、D-Bus で決定されます。
name	システムコールに引数として渡されたファイルまたはディレクトリーへの完全パスを記録します。
new-disk	仮想マシンに割り当てられた新規ディスクリソースの名前を記録します。

イベントフィールド	説明
new-mem	仮想マシンに割り当てられた新規メモリーリソースの容量を記録します。
new-vcpu	仮想マシンに割り当てられた新規の仮想 CPU リソースの数を記録します。
new-net	仮想マシンに割り当てられた新規ネットワークインターフェースのリソースの MAC アドレスを記録します。
new_gid	ユーザーに割り当てられたグループ ID を記録します。
oauuid	システムにアクセスするためにログインし \$u を使用してのログインなどではなく、ターゲットプロセスを開始したユーザーのユーザー ID を記録します。このフィールドは、タイプ OBJ_PID の記録専用になります。
ocomm	ターゲットプロセスの開始に使用されたコマンドを記録します。このフィールドは、タイプ OBJ_PID の記録専用になります。
opid	ターゲットプロセスのプロセス ID を記録します。このフィールドは、タイプ OBJ_PID の記録専用になります。
oses	ターゲットプロセスのセッション ID を記録します。このフィールドは、タイプ OBJ_PID の記録専用になります。
ouid	ターゲットプロセスの実際のユーザー ID を記録します。
obj	オブジェクトの SELinux コンテキストを記録します。オブジェクトは、ファイルやディレクトリー、またはサブジェクトのアクションを受信するものになります。
obj_gid	オブジェクトのグループ ID を記録します。
obj_lev_high	オブジェクトの高い SELinux レベルを記録します。
obj_lev_low	オブジェクトの低い SELinux レベルを記録します。
obj_role	オブジェクトの SELinux の役割を記録します。
obj_uid	オブジェクトの UID を記録します。
obj_user	オブジェクトに関連付けられたユーザーを記録します。
ogid	オブジェクトの所有者のグループ ID を記録します。
old-disk	新規ディスクリソースが仮想マシンに割り当てられた際の古いディスクリソース名を記録します。

イベントフィールド	説明
old-mem	新しいメモリー容量が仮想マシンに割り当てられた際の古いメモリーリソースの容量を記録します。
old-vcpu	新規の仮想 CPU が仮想マシンに割り当てられた際の古い仮想 CPU リソースの数を記録します。
old-net	新規ネットワークインターフェースが仮想マシンに割り当てられた際の古いネットワークインターフェースリソースの MAC アドレスを記録します。
old_prom	ネットワークの無作為フラグの以前の値を記録します。
ouid	ターゲットプロセスを開始したユーザーの実際のユーザー ID を記録します。
path	AVC 関連の Audit イベントでシステムコールに引数として渡されたファイルまたはディレクトリへの完全パスを記録します。
perm	イベント生成に使用されたファイルパーミッション (つまり、読み取り、書き込み、実行、または属性変更) を記録します。
pid	<p>pid フィールドのセマンティックは、このフィールドの値の発生元によって異なります。</p> <p>ユーザースペースから生成されたフィールドの場合、このフィールドはプロセス ID になります。</p> <p>カーネルが生成したフィールドの場合、このフィールドはスレッド ID になります。スレッド ID は単一スレッドプロセスの場合、プロセス ID と同じになります。このスレッド ID の値は、ユーザースペースで使用する <code>pthread_t</code> ID の値とは異なることに注意してください。詳細は、<code>gettid(2)</code> man ページを参照してください。</p>
ppid	親プロセス ID (PID) を記録します。
prom	ネットワークの無作為フラグを記録します。
proto	使用されたネットワークプロトコルを記録します。このフィールドは、 iptables で生成された Audit イベントに固有のものになります。
res	Audit イベントを開始した操作結果を記録します。
result	Audit イベントを開始した操作結果を記録します。
saddr	ソケットアドレスを記録します。
sauid	送信者の Audit ログインユーザー ID を記録します。この ID は、元の auid を送信しているユーザーをカーネルが判別できないため、D-Bus が提供します。

イベントフィールド	説明
ses	分析されているプロセスが開始されたセッションのセッション ID を記録します。
sgid	分析されているプロセスを開始したユーザーのセットグループ ID を記録します。
sig	プログラムを異常終了させたシグナル数を記録します。これは通常、システム侵入を知らせるものです。
subj	サブジェクトの SELinux コンテキストを記録します。サブジェクトは、プロセスやユーザー、またはオブジェクトに対して動作を行なっているものになります。
subj_clr	サブジェクトの SELinux クリアランスを記録します。
subj_role	サブジェクトの SELinux の役割を記録します。
subj_sen	サブジェクトの SELinux の秘密度を記録します。
subj_user	サブジェクトに関連付けられたユーザーを記録します。
success	システムコールが成功したか、失敗したかを記録します。
suid	分析されているプロセスを開始したユーザーのセットユーザー ID を記録します。
syscall	カーネルに送信されたシステムコールのタイプを記録します。
terminal	ターミナル名 (/dev/ なしで) を記録します。
tty	制御ターミナルの名前を記録します。プロセスに制御ターミナルがない場合は、 (none) の値を使います。
uid	分析されているプロセスを開始したユーザーの実際のユーザー ID を記録します。
vm	Audit イベントが始まった仮想マシン名を記録します。

B.2. AUDIT 記録のタイプ

表B.2「記録のタイプ」では、現在サポートされている Audit 記録のタイプを一覧表示しています。イベントのタイプは、すべての Audit 記録の最初にある **type=** フィールドで指定されています。

表B.2 記録のタイプ

イベントタイプ	説明
ADD_GROUP	ユーザースペースグループが追加されると開始します。
ADD_USER	ユーザースペースユーザーのアカウントが追加されると開始します。
ANOM_ABEND ^[a]	プロセスが異常終了すると開始します (コアダンプを引き起こすシグナルが有効になっていれば、このシグナルを伴います)。
ANOM_ACCESS_FS ^[a]	ファイルまたはディレクトリーアクセスが異常終了すると開始します。
ANOM_ADD_ACCT ^[a]	ユーザースペースアカウントの追加が異常終了すると開始します。
ANOM_AMTU_FAIL ^[a]	Abstract Machine Test Utility (AMTU) の失敗が検出されると開始します。
ANOM_CRYPTO_FAIL ^[a]	暗号化システムの失敗が検出されると開始します。
ANOM_DEL_ACCT ^[a]	ユーザースペースアカウントの削除が異常終了すると開始します。
ANOM_EXEC ^[a]	ファイル実行が異常終了すると開始します。
ANOM_LOGIN_ACCT ^[a]	アカウントのログイン試行が異常終了すると開始します。
ANOM_LOGIN_FAILURES ^[a]	ログイン試行が失敗の制限数に達すると開始します。
ANOM_LOGIN_LOCATION ^[a]	許可されていない場所からログイン試行が行われると開始します。
ANOM_LOGIN_SESSIONS ^[a]	ログイン試行が同時セッションの最大数に達すると開始します。
ANOM_LOGIN_TIME ^[a]	ログイン試行が pam_time などによって妨げられる場合に開始します。
ANOM_MAX_DAC ^[a]	Discretionary Access Control (DAC) 失敗の最大数に達すると開始します。
ANOM_MAX_MAC ^[a]	Mandatory Access Control (MAC) 失敗の最大数に達すると開始します。
ANOM_MK_EXEC ^[a]	ファイルを実行可能にすると開始します。
ANOM_MOD_ACCT ^[a]	ユーザースペースアカウントの修正が異常終了すると開始します。

イベントタイプ	説明
ANOM_PROMISCUOUS ^[a]	デバイスが無作為モードを有効または無効にすると開始します。
ANOM_RBAC_FAIL ^[a]	Role-Based Access Control (RBAC) セルフテストの失敗が検出されると開始します。
ANOM_RBAC_INTEGRITY_FAIL ^[a]	Role-Based Access Control (RBAC) ファイル整合性テストの失敗が検出されると開始します。
ANOM_ROOT_TRANS ^[a]	ユーザーが root になると開始します。
AVC	SELinux パーミッションチェックを記録するために開始します。
AVC_PATH	SELinux パーミッションチェックが発生すると、 dentry および vfsmount のペアを記録するために開始します。
BPRM_FCAPS	ユーザーがファイルシステム機能でプログラムを実行すると開始します。
CAPSET	root で機能を外すなど、プロセススペースの機能に設定されている機能を記録するために開始します。
CHGRP_ID	ユーザースペースのグループ ID が変更されると開始します。
CHUSER_ID	ユーザースペースのユーザー ID が変更されると開始します。
CONFIG_CHANGE	Audit システムの設定が変更されると開始します。
CRED_ACQ	ユーザーがユーザースペースの認証情報を必要とする際に開始します。
CRED_DISP	ユーザーがユーザースペースの認証情報を廃棄する際に開始します。
CRED_REFR	ユーザーがユーザースペースの認証情報を更新する際に開始します。
CRYPTO_FAILURE_USER	暗号解読、暗号化、ランダム化のいずれかの暗号化演算が失敗すると、開始します。
CRYPTO_KEY_USER	暗号化の目的で使用される暗号鍵識別子を記録するために開始します。
CRYPTO_LOGIN	暗号化責任者のログイン試行が検出されると開始します。
CRYPTO_LOGOUT	暗号化責任者のログアウト試行が検出されると開始します。
CRYPTO_PARAM_CHANGE_USER	暗号化パラメーターで変更が検出されると開始します。
CRYPTO_REPLAY_USER	再生攻撃が検出されると開始します。

イベントタイプ	説明
CRYPTO_SESSION	TLS セッション確立中にパラメーター一式を記録するために開始します。
CRYPTO_TEST_USER	FIPS-140 規格で必要とされている暗号化テスト結果を記録するために開始します。
CWD	現在の作業ディレクトリーを記録するために開始します。
DAC_CHECK	DAC チェックの結果を記録するために開始します。
DAEMON_ABORT	エラーのためにデーモンが停止する際に開始します。
DAEMON_ACCEPT	auditd デーモンがリモート接続を受け入れる際に開始します。
DAEMON_CLOSE	auditd デーモンがリモート接続を閉じる際に開始します。
DAEMON_CONFIG	デーモンの設定変更が検出されると開始します。
DAEMON_END	デーモンが正常に停止されると開始します。
DAEMON_RESUME	auditd デーモンがロギングを再開する際に開始します。
DAEMON_ROTATE	auditd デーモンが Audit ログファイルをローテーションする際に開始します。
DAEMON_START	auditd デーモンが起動されると開始します。
DEL_GROUP	ユーザースペースグループが削除されると開始します。
DEL_USER	ユーザースペースのユーザーが削除されると開始します。
DEV_ALLOC	デバイスが割り当てられると開始します。
DEV_DEALLOC	デバイスの割り当てが解除されると開始します。
EOE	複数記録イベントの終了を記録するために開始します。
EXECVE	execve(2) システムコールの引数を記録するために開始します。
FD_PAIR	pipe および socketpair のシステムコールの使用を記録するために開始します。
FS_RELABEL	ファイルシステムのラベル張り替え操作が検出されると開始します。
GRP_AUTH	ユーザースペースのグループに対してグループパスワードを使って認証が行われると開始します。

イベントタイプ	説明
INTEGRITY_DATA ^[b]	カーネルが実行するデータ整合性検証イベントを記録するために開始します。
INTEGRITY_HASH ^[b]	カーネルが実行するハッシュタイプ整合性検証イベントを記録するために開始します。
INTEGRITY_METADATA ^[b]	カーネルが実行するメタデータ整合性検証イベントを記録するために開始します。
INTEGRITY_PCR ^[b]	Platform Configuration Register (PCR) 無効化メッセージを記録するために開始します。
INTEGRITY_RULE ^[b]	ポリシールールを記録するために開始します。
INTEGRITY_STATUS ^[b]	整合性検証のステータスを記録するために開始します。
IPC	システムコールが参照する Inter-Process Communication オブジェクトについての情報を記録するために開始します。
IPC_SET_PERM	IPCオブジェクト上の IPC_SET 制御操作が設定する新しい値についての情報を記録するために開始します。
KERNEL	Audit システムの初期化を記録するために開始します。
KERNEL_OTHER	サードパーティーのカーネルモジュールからの情報を記録するために開始します。
LABEL_LEVEL_CHANGE	オブジェクトのレベルラベルが修正されると開始します。
LABEL_OVERRIDE	管理者がオブジェクトのレベルラベルを上書きすると開始します。
LOGIN	ユーザーがシステムにアクセスするためにログインする際に関連するログイン情報を記録するために開始します。
MAC_CIPSOV4_ADD	Commercial Internet Protocol Security Option (CIPSO) ユーザーが新たな Domain of Interpretation (DOI) を追加すると開始します。DOI の追加は、NetLabel が提供するカーネルのパケットラベル付け機能の一部です。
MAC_CIPSOV4_DEL	CIPSO ユーザーが既存の DOI を削除すると開始します。DOI の追加は、NetLabel が提供するカーネルのパケットラベル付け機能の一部です。
MAC_CONFIG_CHANGE	SELinux ブール値が変更されると開始します。
MAC_IPSEC_EVENT	IPSec イベントが検出されるか IPSec 設定が変更されると、IPSec イベントについての情報を記録するために開始します。

イベントタイプ	説明
MAC_MAP_ADD	新たな Linux Security Module (LSM) ドメインマッピングが追加されると開始します。LSM ドメインマッピングは、NetLabel が提供するカーネルのパケットラベル付け機能の一部です。
MAC_MAP_DEL	既存の LSM ドメインマッピングが追加されると開始します。LSM ドメインマッピングは、NetLabel が提供するカーネルのパケットラベル付け機能の一部です。
MAC_POLICY_LOAD	SELinux ポリシーファイルが読み込まれると開始します。
MAC_STATUS	SELinux モード (enforcing、permissive、off) が変更されると開始します。
MAC_UNLBL_ALLOW	NetLabel が提供するカーネルのパケットラベル付け機能を使用する際にラベルのないトラフィックが許可されると開始します。
MAC_UNLBL_STCADD	NetLabel が提供するカーネルのパケットラベル付け機能を使用する際に静的ラベルが追加されると開始します。
MAC_UNLBL_STCDEL	NetLabel が提供するカーネルのパケットラベル付け機能を使用する際に静的ラベルが削除されると開始します。
MMAP	mmap(2) システムコールのファイル記述子およびフラグを記録するために開始します。
MQ_GETSETATTR	mq_getattr(3) および mq_setattr(3) のメッセージキュー属性を記録するために開始します。
MQ_NOTIFY	mq_notify(3) システムコールの引数を記録するために開始します。
MQ_OPEN	mq_open(3) システムコールの引数を記録するために開始します。
MQ_SENDRECV	mq_send(3) および mq_receive(3) のシステムコールの引数を記録するために開始します。
NETFILTER_CFG	Netfilter チェーンの修正が検出されると開始します。
NETFILTER_PKT	Netfilter チェーンをトラバースするパケットを記録するために開始します。
OBJ_PID	信号の送信先のプロセスについての情報を記録するために開始します。
PATH	ファイル名パスの情報を記録するために開始します。
RESP_ACCT_LOCK^[c]	ユーザーのアカウントがロックされると開始します。

イベントタイプ	説明
RESP_ACCT_LOCK_TIME ^[c]	ユーザーのアカウントが一定期間ロックされると開始します。
RESP_ACCT_REMOTE ^[c]	ユーザーのアカウントがリモートセッションからロックされると開始します。
RESP_ACCT_UNLOCK_TIME ^[c]	ユーザーのアカウントが設定された期間の後にロック解除されると開始します。
RESP_ALERT ^[c]	警告 E メールが送信されると開始します。
RESP_ANOMALY ^[c]	異常に対してアクションが取られないと開始します。
RESP_EXEC ^[c]	プログラムの実行を元とする脅威に侵入検出プログラムが反応すると開始します。
RESP_HALT ^[c]	システムがシャットダウンすると開始します。
RESP_KILL_PROC ^[c]	プロセスが終了すると開始します。
RESP_SEB00L ^[c]	SELinux ブール値が設定されると開始します。
RESP_SINGLE ^[c]	システムがシングルユーザーモードになると開始します。
RESP_TERM_ACCESS ^[c]	セッションが終了すると開始します。
RESP_TERM_LOCK ^[c]	ターミナルがロックされると開始します。
ROLE_ASSIGN	管理者がユーザーに SELinux の役割を割り当てると開始します。
ROLE_MODIFY	管理者が SELinux の役割を編集すると開始します。
ROLE_REMOVE	管理者がユーザーを SELinux の役割から削除すると開始します。
SELINUX_ERR	内部 SELinux エラーが検出されると開始します。
SERVICE_START	サービスが起動すると開始します。
SERVICE_STOP	サービスが停止すると開始します。
SOCKADDR	ソケットアドレスを記録するために開始します。
SOCKETCALL	sys_socketcall システムコール (多くのソケット関連のシステムコールを多重化するために使用) の引数を記録するために開始します。

イベントタイプ	説明
SYSCALL	カーネルへのシステムコールを記録するために開始します。
SYSTEM_BOOT	システムが起動すると開始します。
SYSTEM_RUNLEVEL	システムのランレベルが変更されると開始します。
SYSTEM_SHUTDOWN	システムがシャットダウンすると開始します。
TEST	テストメッセージの成功の値を記録するために開始します。
TRUSTED_APP	このタイプの記録は、監査を必要とするサードパーティーのアプリケーションで使用可能です。
TTY	TTY 入力及管理プロセスに送信されると開始します。
USER_ACCT	ユーザースペースユーザーのアカウントが編集されると開始します。
USER_AUTH	ユーザースペースの認証試行が検出されると開始します。
USER_AVC	ユーザースペースの AVC メッセージが生成されると開始します。
USER_CHAUTHOK	ユーザーアカウントの属性が編集されると開始します。
USER_CMD	ユーザースペースのシェルコマンドが実行されると開始します。
USER_END	ユーザースペースのセッションが終了すると開始します。
USER_ERR	ユーザーアカウントの状態のエラーが検出されると開始します。
USER_LABELED_EXPORT	SELinux ラベルの付いたオブジェクトがエクスポートされると開始します。
USER_LOGIN	ユーザーがログインすると開始します。
USER_LOGOUT	ユーザーがログアウトすると開始します。
USER_MAC_POLICY_LOAD	ユーザースペースのデーモンが SELinux ポリシーを読み込むと開始します。
USER_MGMT	ユーザースペースの管理データを記録するために開始します。
USER_ROLE_CHANGE	ユーザーの SELinux の役割が変更されると開始します。
USER_SELINUX_ERR	ユーザースペースの SELinux エラーが検出されると開始します。
USER_START	ユーザースペースのセッションが開始すると開始します。

イベントタイプ	説明
USER_TTY	管理プロセスへの TTY 入力についての説明メッセージがユーザースペースから送信されると開始します。
USER_UNLABELED_EXPORT	SELinux ラベルなしでオブジェクトがエクスポートされると開始します。
USYS_CONFIG	ユーザースペースのシステム設定変更が検出されると開始します。
VIRT_CONTROL	仮想マシンが起動、一時停止、停止すると開始します。
VIRT_MACHINE_ID	仮想マシンへのラベルのバインディングを記録するために開始します。
VIRT_RESOURCE	仮想マシンのリソース割り当てを記録するために開始します。
<p>[a] All Audit event types prepended with ANOM are intended to be processed by an intrusion detection program.</p> <p>[b] This event type is related to the Integrity Measurement Architecture (IMA), which functions best with a Trusted Platform Module (TPM) chip.</p> <p>[c] All Audit event types prepended with RESP are intended responses of an intrusion detection system in case it detects malicious activity on the system.</p>	

付録C 改訂履歴

改訂 1-23.3 翻訳完了	Mon Jul 3 2017	Mie Yamamoto
改訂 1-23.2 翻訳ファイルを XML ソースバージョン 1-23 と同期	Tue Feb 28 2017	Terry Chuang
改訂 1-23.1 翻訳ファイルを XML ソースバージョン 1-23 と同期	Mon Nov 28 2016	Terry Chuang
改訂 1-23 7.3 GA 公開用バージョン	Tue Nov 1 2016	Mirek Jahoda
改訂 1-19 スマートカードのセクションの追加	Mon Jul 18 2016	Mirek Jahoda
改訂 1-18 OpenSCAP デモンおよび Atomic Scan セクションの追加	Mon Jun 27 2016	Mirek Jahoda
改訂 1-17 さまざまなアップデートがある非同期リリース	Fri Jun 3 2016	Mirek Jahoda
改訂 1-16 7.2 GA 後の修正	Tue Jan 5 2016	Robert Krátký
改訂 1-15 7.2 GA リリース向けのバージョン	Tue Nov 10 2015	Robert Krátký
改訂 1-14.18 さまざまなアップデートがある非同期リリース	Mon Nov 09 2015	Robert Krátký
改訂 1-14.17 7.1 GA リリース向けのバージョン	Wed Feb 18 2015	Robert Krátký
改訂 1-14.15 Red Hat カスタマーポータルでの並び替え順序の更新	Fri Dec 06 2014	Robert Krátký
改訂 1-14.13 POODLE vuln を反映する更新	Thu Nov 27 2014	Robert Krátký
改訂 1-14.12 7.0 GA リリース向けバージョン	Tue Jun 03 2014	Tomáš Čapek