LinuC

Linuxセキュリティ標準教科書

（Ver.1.0.1）

LPI-JAPAN

■まえがき

Linux は、多くのシステムにおいてサーバ OS として採用されるようになり、社会における重要な位置を任される OS へと成長しました。

それと同時に、Linux における高いセキュリティ面での安全性の確保は、サイバーテロなども想定される近年の社会において、IT のみならず行政や経済面などにおいても大きな課題となりつつあります。

このような課題の解決への貢献として、このたび特定非営利活動法人エルピーアイジャパンは、Linux 技術者や教育関係者向けに、「Linux セキュリティ標準教科書」を提供いたします。

本教科書は、Linux におけるセキュリティを一通り学習または再確認すために、最低限必要な知識についてまとめた内容となっています。

この教科書を企業や学校において活用していただき、Linux を採用したシステムを、より安全性の高いものへと強化する足がかりとしていただければ幸いです。

公開にあたっては、「Linux セキュリティ標準教科書」に添付されたライセンス（クリエイティブ・コモンズ・ライセンス）の下に公開されています。

本教材は、最新の技術動向に対応するため、随時アップデートを行っていきます。

また、テキスト作成やアップデートについては、意見交換の Wiki サイトで、誰でもオープンに参加できます。

Wiki サイトの URL： http://www.lpi.or.jp/linuxtext/security/

■本教科書の対象

本教材は、Linux を使ったサーバを構築する技術者が、セキュリティについての知識を学習および再確認する事を想定して作成しています。

本教科書には、LinuC レベル1、LinuC レベル2、および LinuC レベル3の303試験の範囲に含まれる内容を含めており、内容の難易度に関わらず、サーバの安全性確保に役立つ知識について記述しています。

LinuC レベル1相当の知識がある方を前提としています。

■使用している環境

本教科書では、CentOS6.3を使用しています。

ほとんどの場合、ディストリビューションに関わらず記述の内容をそのまま実行できますが、一部ディストリビューション（特に Debian 系のディストリビューション）などの場合、ファイルのパスが異なる場合があります。

また、執筆時点後において、ソフトウェアのアップデートに伴い、オプションなどが変更になる可能性があります。

■本書の記述

本書では、解説中に以下の記述をしています。

●書式

　コマンドの書式、オプションなどの解説をしています。

●実行例

　実際に、コマンドを実行している例を載せています。

●編集例

　設定ファイルなどを編集した例を載せています。

■実行例における表記

本書に掲載されている実行例において、プロンプトが’#’になっているものは、root 権限で実行しています。

```
例：[root@localhost]# cat /etc/sysconfig/iptables
```

また、プロンプトが’$’になっているものは、一般ユーザの権限で実行しています。

```
例：[lpij@cent64 ~]$ /usr/sbin/getenforce
```

本書では、セキュリティについて述べている部分が多いため、多くが root 権限で実行する事を想定する内容となっています。

■執筆者・制作者紹介

伊本 貴士　メディアスケッチ株式会社　代表取締役（執筆・企画進行担当）

これまで、Linux はオープンソースの OS として、世界中の有志が開発に携わってきました。

そのため、日々進化しており、セキュリティ面でも ACL や SELinux などの機能を備え優秀な OS へと成長しました。しかし、いかに優秀な OS を採用したとしても、それを使う側が、使いこなせなければ意味がありません。

本書は、事例や図を多くして、難しいセキュリティに関する知識をわかりやすく理解できるように心がけて作成しました。

まずは本書でセキュリティ対策の基本を確認していただき、実践に活用する足がかりにしていただければ幸いです。

面 和毅　日本ヒューレット・パッカード株式会社（執筆担当）

SIer としてシステムの構築・運用を手がける傍ら、LIDS/SELinux 等の OSS や、仮想技術のセキュリティに関しての調査を行ってきましたが、やはり実務の現場では「こうするべき」というセキュリティと、運用上での利便性とのギャップによく直面します。

実運用環境では、やはりセキュリティは優先度が低く、効率良く業務を回すことが優先されます。

しかし、その一方で、システムのセキュリティ脆弱性を利用した犯罪も増えてきています。

また、業務上でもコンプライアンスなどの対応で、セキュリティを意識したシステムづくりが多くなってきています。

このテキストのセキュリティ技術を実際に試しながら理解していただき、今後のシステムを構築していく際にも、ある程度セキュリティを意識した構築が出来るようになっていただければと思います。

それは最終的に、自分のスキルアップにもつながっていますので。

藤森 健　ＮＥＣソリューションイノベータ株式会社（執筆担当）

学生時代に Run Run Linux 片手に試行錯誤した PC-UINIX、Linux に惚れてしまって早17年。

そのまま IT エンジニアとして Linux、OSS を活用したミッションクリティカル≒止まらないシステムの設計を糧としています。

本テキストは”きっかけ、一度動かしてみること”をゴールとし作成しました。

続きはインターネットの世界にある情報で試行錯誤していただきたいと思っています。

試行錯誤を通して、Linux、OSS の世界へ本格的に興味をもっていただければ幸いです。

小石川 雅紀　ＮＥＣソリューションイノベータ株式会社（執筆担当）

本書は、セキュリティ対策の基本をわかりやすく理解してもらうことを目指して執筆しました。

セキュリティというと難しいイメージがあるかもしれませんが、本書を通してソフトウェアの導入から実行まで実際に手を動かしながら理解することで、興味を持っていただければ幸いです。

安田 恭行　ＮＥＣソリューションイノベータ株式会社（執筆担当）

本書は自身の手で技術に触れることを目的として、設定手順をベースに執筆しました。

システム全体としてのセキュリティは当然大事ですが、セキュリティの分野は幅広く、それぞれの技術を独立して学ぶことも可能です。

本書で取り上げられている技術のうち興味の湧いた技術があれば、それを足掛かりにスキルアップしていただければと思います。

嶋中 賢佑　ＮＥＣソリューションイノベータ株式会社（執筆担当）

本書を執筆するにあたり、操作の流れをわかりやすくし、読み進める際に手を動かして学ぶことで、セキュリティ対策の基本が身に着くことを目指して執筆しました。

本書の内容は、いずれも簡単に始めることができるものばかりですので、実際に簡単な環境をセットアップしながら学んでいただければと思います。

松田 神一（監修・査読担当）

セキュリティに完全はありませんが、基本的な対策をしっかりと行うだけで、安全性は非常に高まります。

この教科書でセキュリティの基礎について学び、読者の皆さんが管理するシステムのセキュリティレベルを向上させていただければと思います。

木村 真之介（校正・Wiki サイト制作担当）

本教科書の校正等を担当しました。ログを見ると不正なアクセスが毎日のように来ています。とても不安になりますが、Linux には様々なセキュリティツールがあるので心強いです。本教科書はセキュリティ設定を見直すよいきっかけになると思います。

高橋 征義（レイアウト制作・電子書籍化担当）

■著作権

■使用に関する権利

●表示

本教材は、特定非営利活動法人エルピーアイジャパンに著作権が帰属するものであることを表示してください。

●改変禁止

本教材は、改変せず使用してください。ただし、引用等、著作権法上で認められている利用を妨げるものではありません。本教材に対する改変は、特定非営利活動法人エルピーアイジャパンまたは特定非営利活動法人エルピーアイジャパンが認める団体により行われています。フィードバックは誰でも参加できる Wiki サイトとメーリングリストで行われていますので、積極的にご参加ください。

Wiki サイトの URL： **http://www.lpi.or.jp/linuxtext/security/**

メーリングリスト： **http://list.ospn.jp/mailman/listinfo/linux-text**

●非営利

本教材は、非営利目的で教材として自由に利用することができます。商業上の利得や金銭的報酬を主な目的とした営利目的での利用は、特定非営利活動法人エルピーアイジャパンによる許諾が必要です。ただし、本教材を利用した教育において、本教材自体の対価を請求しない場合は、営利目的の教育であっても基本的に利用できます。その場合も含め、LPI-Japan 事務局（TEL：03-3568-4482　e-mail：info@lpi.or.jp）までお気軽にお問い合わせください。

※営利目的の利用とは以下のとおり規定しております。

営利企業または非営利団体において、商業上の利得や金銭的報酬を目的に、印刷実費以上の対価を受講者に請求して当教材の複製を用いた研修や講義の実施や書籍販売等を行い利益を得ること。

本教材の使用に関するお問い合わせ先
特定非営利活動法人エルピーアイジャパン（LPI-Japan）事務局 〒106-0041　東京都港区麻布台1-11-9　BPR プレイス神谷町7F TEL：03-3568-4482 FAX：03-3568-4483 E-Mail：info@lpi.or.jp Linux セキュリティ標準教科書 URL：https://linuc.org/textbooks/security/

目次

1. はじめに　～セキュリティに関わる問題の原因と対策～

セキュリティ上の問題を引き起こす原因は、大きく以下の３つがあります。

１）ヒューマンエラー

パスワードを他人に教える、他人の見えるところにメモに残す、部外者が自由にサーバのキーボードを打つことができるなど、システムの管理・運用に関係なく、利用者のセキュリティポリシー違反が原因で起きる問題。

２）不適切なサーバ運用

ソフトウェアの脆弱性を放置する、不適切な設定、不要なアクセスの許可、不適切なアクセス権の付与など日々のサーバ運用を適切に行わないことが原因で起きる問題。

３）プログラムコードのミスによる脆弱性の問題

特に、CGI プログラムなど、外部からの入力を受け付けるプログラムにおいて、SQL インジェクション対策を怠るなどのプログラムの設計やコーディングのミスが原因で起きる問題。

セキュリティ対策には非常に幅広い範囲を意識する必要がありますが、本教科書では主に「サーバ運用」に関する部分について説明します。

Linux におけるセキュリティ対策には、大きく以下の５つがあります。

１）設定の適正化

利用するソフトウェアの設定を適切に行い、管理者がその意味を理解し定期的にチェックすることで、セキュリティリスクは大幅に下げることができます。

当たり前の事と思われるかもしれませんが、OS やソフトウェアの設定を適切に行う、これが行われていないために攻撃され乗っ取られてしまう事例は多くあります。

２）パケットフィルタリング

不要なパケットは受信しない。これはセキュリティ対策における非常に重要なポイントです。Linux ではパケットフィルタリングを利用するなどの対策で、外部からの不要なアクセスを防ぐことができます。

３）権限の制御

ユーザやプログラムに不要な権限を与えない。これはセキュリティ対策における重要な事項です。サービスを提供するために必要以上の権限をユーザやプログラムに与えた場合、悪意のある第三者に対して、その権限を利用した攻撃の機会を与えることになります。

４）暗号化

ネットワークパケットの盗聴は、昔からサイバー攻撃の基本的手法です。インターネット上を平文のパスワードが流れるということは絶対にあってはいけません。また、悪意のあるユーザは常にインターネットの向う側にいるとは限りません。社内や学校内のネットワークでも、多くの人が接続するネットワーク内に、重要な情報を流す場合は、すべて暗号化すべきです。

５）検知

IT 技術の発展に伴いサイバー攻撃の手法は多種多様な広がりをみせています。そのため、セキュリティという観点においては、これで完璧というシステムは存在しません。十分な対策を行ってもシステムに侵入される可能性はありますが、改ざん検知と侵入検知システム（IDS）を利用することでリスクを減らすことができます。システムファイルが不正に改ざんされた場合、すぐに対策を打ち被害を最小限に抑える必要があります。そのためのツールが改ざん検知ソフトウェアです。また、侵入検知システムは人間が見逃しかねない攻撃と思われる怪しい兆候をソフトウェアが自動で検知し管理者に通知します。

本教科書では主に 2 章においてセキュリティの観点からチェックすべき設定項目について事故に繋がりやすいものを中心に紹介します。3 章では iptables によるパケットフィルタリングについて、4 章では SELinux を利用したアクセス制御、5 章では ACL を用いたファイルへのアクセス権制御について、6 章では OpenSSH を使った暗号化の方法、7 章では改ざん検知と侵入検知について紹介します。また、8 章ではセキュリティスキャンツールについての紹介を行います。セキュリティスキャンツールを利用することでシステムに脆弱性がないかどうかをチェックすることが可能です。9 章では最も利用者が多く、そのため最も攻撃の対象となりやすい WEB サーバ（apache）についてのセキュリティ対策を紹介します。

2. Linux サーバにおけるセキュリティ基本チェック

本章では、Linux で行う基本的なセキュリティ対策をチェック項目として紹介します。チェック項目表を活用して、セキュリティ対策に不備がないかをチェックしてください。

個々の詳細な説明は 3 章以降で取り上げます。

2.1 セキュリティチェック項目表

番号　チェック	カテゴリ	行うべき対策
1	ソフトウェア管理	ソフトウェアのアップデート情報をチェックする
2	ソフトウェア管理	利用しているパッケージを定期的にアップデートする
3	ソフトウェア管理	不要なソフトウェアをインストールしない
4	ユーザ管理	不要なユーザは、ログインできなくするか、削除する
5	ユーザ管理	必要に応じてユーザのパスワードに有効期限をつける
6	ユーザ管理	root になれる、もしくは sudo を使えるユーザを制限する
7	運用管理	ファイルおよびディレクトリのアクセス権を適切に設定する
8	運用管理	ランレベルを適切に設定し、不要なデーモンを起動させない
9	運用管理	不要な SUID, SGID を削除する
10	運用管理	ログを正確に残す
11	ネットワーク管理	パケットフィルタリング（iptables など）を有効にし、余計なパケットを受信しないようにする
12	ネットワーク管理	不要なポートを閉じる
13	ネットワーク管理	外部からアクセスして管理する場合は、SSH を利用する

2.1　適切なソフトウェア管理

2.1.1 ソフトウェアのアップデート情報をチェックする

ソフトウェアの脆弱性とは、ソフトウェアの欠陥や仕様上の問題でセキュリティの弱点となるものです。脆弱性のあるソフトウェアをそのままにした場合、たとえ適切に設定されていても、その弱点を攻撃され外部からの侵入を許し、情報漏洩などの事故を引き起こす可能性があります。

そのため、ソフトウェアに脆弱性が発見された場合、ただちに脆弱性に関する詳細情報や関連するソフトウェアのアップデート情報を確認し、対策を実施する必要があります。
そして、関連するソフトウェアが、脆弱性に関する問題を修正した新しいバージョンをリリースした場合は、すぐにアップデートすべきです。
また、運用状況によってアップデートが困難な場合や脆弱性が発見された直後のように関連ソフトウェアの修正版がリリースされていない場合でもセキュリティ情報を参照し回避策を実施する必要があります。

どのソフトウェアに、どのような脆弱性が発見されたかについては、以下のサイトで確認できます。

- JPCERT コーディネーションセンター
  http://www.jpcert.or.jp/
- JVN（JPCERT/CC と情報処理推進機構（IPA）が共同で管理している脆弱性情報データベース）
  http://jvn.jp/
- JVN iPedia
  http://jvndb.jvn.jp/
- IPA 緊急対策情報・注意喚起 一覧
  http://www.ipa.go.jp/security/announce/alert.html

脆弱性の問題を含むソフトウェアのバグ情報は各 Linux ディストリビューションのサイトからも確認できます。

- Red Hat Errata（英語）
  https://rhn.redhat.com/errata/
- CentOS-announce（英語）
  （CentOS は、メーリングリストにて情報発信を行っています。）

http://lists.centos.org/mailman/listinfo/centos-announce（メーリングリストの登録）
http://lists.centos.org/pipermail/centos-announce/（過去のアーカイブ）

- Debian セキュリティ情報
  http://www.debian.org/security/
- Vine Linux セキュリティ・更新（エラッタ）情報
  http://vinelinux.org/errata.html

Red Hat Enterprise Linux を含む多くの商用 Linux では購入者向けのサポートページでソフトウェアに関する詳細なアップデート情報が提供されています。商用 Linux を購入した場合は必要なユーザ登録を行った上でサポートページの情報を確認すべきです。

その他、各オープンソースソフトウェアのコミュニティサイトでも、バグやセキュリティに関する情報が発信されています。特に外部からの不特定多数のアクセスがあるサービスに利用しているような、重要なソフトウェアに関しては、こちらの情報もチェックすべきです。

- Apache HTTP Server Security Reports（英語）
  http://httpd.apache.org/security_report.html
- Apache Tomcat Security Reports（英語）
  http://tomcat.apache.org/security.html
- PostgreSQL Security Information（英語）
  http://www.postgresql.org/support/security/
- MySQL bugs
  http://bugs.mysql.com/
- PHP report bugs（英語）
  https://bugs.php.net/

2.1.2　利用しているパッケージを定期的にアップデートする

ソフトウェアパッケージのアップデートにはいくつか理由があります。ソフトウェアの重大なバグの修正もあれば機能強化、そしてセキュリティ上の脆弱性の修正などがあります。脆弱性の修正であれば速やかに適用すべきです。新しいバージョンがリリースされた場合、システム管理者はリリースノートなどを参照して、それが単なるバグフィックスなのかセキュリティフィックスなのかを区別して考えるとよいでしょう。なお、新しいバージョンが開発版の場合は、十分なテストがされていないため、適用には慎重になるべきです。

しかしながら、利用しているすべてのパッケージのセキュリティ情報をチェックすることは簡単ではありません。そこで、定期的にアップデート可能なパッケージをすべてアップデートするという方法を検討してもよいでしょう。

RedHat や CentOS などの RedHat 系のディストリビューションではパッケージ管理ソフトウェア yum を使う方法が簡単で効率的です（yum でサポートされていないソフトウェアの場合は、各ソフトウェアの公式ページでアップデートの手順を確認してください）。

yum コマンドで、現在アップデートが可能なパッケージの一覧を表示します。

```
yum list updates
または
yum check-update
```

実行例

```
[root@localhost]# yum list updates
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: ftp.nara.wide.ad.jp
 * extras: ftp.nara.wide.ad.jp
 * updates: ftp.nara.wide.ad.jp
Updated Packages
NetworkManager.i686                  1:0.8.1-34.el6_3                  updates
NetworkManager-glib.i686             1:0.8.1-34.el6_3                  updates
NetworkManager-gnome.i686            1:0.8.1-34.el6_3                  updates
ORBit2.i686                          2.14.17-3.2.el6_3                 updates
apr.i686                             1.3.9-5.el6_2                     updates
```

```
bind-libs.i686                          32:9.8.2-0.10.rc1.el6_3.6          updates
…以下省略
```

アップデート可能なパッケージのリストが表示されますので内容を確認してください。問題がなければ yum update コマンドでアップデート可能なものを全てアップデートすることができます。

```
yum update
```

yum update コマンドを実行すると、アップデート対象となるリストが表示された後、

Is this ok [y/N]:と表示されます。内容を確認して問題がなければ y キーを入力することでアップデートが開始されます。

実行例

```
[root@localhost]# yum update
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: ftp.nara.wide.ad.jp
 * extras: ftp.nara.wide.ad.jp
 * updates: ftp.nara.wide.ad.jp

     ====================   省略   ====================

Transaction Summary
================================================================================
Install       2 Package(s)
Upgrade     100 Package(s)

Total download size: 159 M
Is this ok [y/N]: y

Downloading Packages:
(1/102): NetworkManager-0.8.1-34.el6_3.i686.rpm          | 1.1 MB     00:00
(2/102): NetworkManager-glib-0.8.1-34.el6_3.i686.rpm     | 218 kB     00:00

     ====================   省略   ====================

  util-linux-ng.i686 0:2.17.2-12.7.el6_3
  xulrunner.i686 0:10.0.11-1.el6.centos
```

```
Complete!
```

アップデートの際に一部のパッケージで依存関係などの問題が生じ、すべてのパッケージを一度にアップデートできない場合は、yum update コマンドにパッケージ名を指定することで個々のパッケージのみをアップデートすることができます。

```
yum update パッケージ名
```

実行例

```
[root@localhost]# yum update sudo.i686
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
 * base: ftp.nara.wide.ad.jp
 * extras: ftp.nara.wide.ad.jp
 * updates: ftp.nara.wide.ad.jp

     ====================   省略   ====================

Total download size: 419 k
Is this ok [y/N]: y
Downloading Packages:
sudo-1.7.4p5-13.el6_3.i686.rpm                           | 419 kB     00:00
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating   : sudo-1.7.4p5-13.el6_3.i686                                 1/2
  Cleanup    : sudo-1.7.4p5-11.el6.i686                                   2/2
  Verifying  : sudo-1.7.4p5-13.el6_3.i686                                 1/2
  Verifying  : sudo-1.7.4p5-11.el6.i686                                   2/2

Updated:
  sudo.i686 0:1.7.4p5-13.el6_3

Complete!
```

2.1.3　不要なソフトウェアをインストールしない

インストールしているソフトウェアが増えれば、その分だけソフトウェアの脆弱性を抱える可能性が高くなります。そのためセキュリティの高いサーバを構築するには必要最低限なソフトウェアのみをインストールするというのが基本です。新規に Linux をインストールする場合はパッケージグループの選択で「最小 (Minimal)」を選び、その後で使うパッケージを個別にインストールすると良いでしょう。

2.4　適切なユーザ管理

2.4.1　不要なユーザは、ログインできなくするか、削除する

利用者が退職したなどの理由でユーザアカウントを有効にしておく必要がない場合、ログインできないようにするかユーザを削除する必要があります。

ユーザを削除するには、userdel コマンドを使います。

```
userdel ユーザ名
```

また、ユーザを削除すると同時にホームディレクトリも削除したい場合は、-r オプションを付けて実行してください。

```
userdel -r ユーザ名
```

実行例

```
[root@localhost]# userdel -r linuc
```

しばらく利用しないが利用再開の可能性がある、もしくは何かの理由で削除できないユーザの場合は、ユーザ削除ではなくログインできないようにすることができます。/etc/passwd を直接編集することでもログインできなくすることは可能ですが、編集ミスを考慮して passwd コマンドでユーザをロックする方が確実で安全です。

passwd コマンドの-l オプションでユーザアカウントをロックすることができます。

```
passwd -l ユーザ名
```

実行例

```
[root@localhost]# passwd -l linuc
ユーザー linuc 用のパスワードをロック。
passwd: 成功
```

ロックしたユーザを、アンロックし再び使えるようにするには、-u オプションを使います。

```
passwd -u ユーザ名
```

実行例

```
[root@localhost]# passwd -u linuc
ユーザー linuc 用のパスワードをロック解除。
passwd: 成功
```

なお、root ユーザは su コマンドでロックされたユーザになることができます。

2.4.2　必要に応じてユーザのパスワードに有効期限をつける

セキュリティ対策として「パスワードを定期的に変更しましょう」と言われることが多いですがなぜパスワードを変更する必要があるのでしょうか。またそれは有効な対策なのでしょうか。

パスワードに関する攻撃と言えばツールを使ったパスワード総当たり攻撃や辞書を使ったキーワードによる攻撃を想像されるかもしれません。これらのパスワード推測型の攻撃にはパスワードの変更は有効ではありません。複雑で長いパスワードを設定することが有効です。

しかし、パスワードが漏れる可能性は他にもあります。

例えば、パスワードを入力する時のキーボードを隣や後ろから見られていたという場合です。これは、いかに難しいパスワードでも防ぎにくい一例です。

また、「新入社員 100 人分のアカウントを作成し、全員に仮パスワードを発行した」というのも、よくある話です。この時、仮パスワードを伝えるために、メモなどに記録すれば、それがどこかに漏れる可能性があります。

ユーザ自身が気をつけるべきことですが、このような場合の対策として、管理者はパスワードに有効期限を設けることを検討すると良いでしょう。

Linux では chage コマンドを使うことで、パスワードの有効期限を設けることが可能です。

まずは、chage コマンドの-l オプションでパスワードの有効期限設定を確認できます。

```
chage -l ユーザ名
```

実行例

```
[root@localhost]# chage -l linuc
Last password change                                    : Jan 07, 2013
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

Password expires(パスワードの失効日付)が never になっているので、上記の例の場合、パスワードの期限はありません。

chage コマンドの-M オプションで、パスワードの期限を日数で設定できます。

```
chage -M 日数 ユーザ名
```

実行例

```
[root@localhost]# chage -M 30 linuc
[root@localhost]# chage -l linuc
Last password change                                    : Jan 07, 2013
Password expires                                        : Feb 06, 2013
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 30
Number of days of warning before password expires       : 7
```

chage -M を実行した後、chage -l で再度期限の設定を確認すると、Password expires にパスワードの有効期限が設定されています。また、Maximum number of days between password change（パスワード変更までの最大日数）に、-M オプションで指定した 30（日）が表示されています。

セキュリティ対策とユーザの利便性は相反するものです。パスワードが外部に漏れている可能性を警戒するあまりパスワードの有効期限を極端に短くしてしまうと、ユーザは新しいパスワードを覚えられないので常にメモ用紙に記録したり、毎回簡単なパスワードを設定するかもしれません。管理者はシステムの用途や組織毎に適切なセキュリティポリシーを設定して利便性と安全性のバランスを保つ必要があります。

2.4.3　root になれる、もしくは sudo を使えるユーザを制限する

一般ユーザでログインしているときに root ユーザの権限で業務を行う場合は su コマンドで root ユーザに切り替えるか sudo コマンドで root ユーザとしてコマンドを実行する必要があります。

su や sudo コマンドを実行できるユーザアカウントが不正に利用されるとシステムを乗っ取られる危険があります。従って、su や sudo を実行できるユーザはできる限り制限した方が良いでしょう。

・su コマンドに関する設定

su コマンドを利用できるユーザを制限するには、pam_wheel.so を使います。pam_wheel.so は、su コマンドを利用できるユーザを、wheel グループに所属するユーザに限定します。

pam_wheel.so を有効にするには、/etc/pam.d/su というファイルで、以下のようにコメントになっている部分があるので先頭の#を消して設定を有効化します。

```
# auth          required                pam_wheel.so use_uid
```

これで、wheel グループに所属するユーザのみが su コマンドを使えるようになります。

・sudo コマンドに関する設定

sudo コマンドを利用できるユーザを制限するには visudo コマンドを使います。visudo コマンドは /etc/sudoers を vi で開きます。vi コマンドで直接開いて編集しても問題ありません。

/etc/sudoers において、以下のように記述すればユーザ linuc に sudo コマンドの利用を許可します。

設定例

```
linuc    ALL=(ALL)       ALL
```

この場合、linuc ユーザは sudo を使ってすべてのコマンドを実行できるようになります。一部のコマンドのみを実行可能にする場合は、以下のように記述します。以下は、iptables コマンドのみを実行可能にする記述例です。

設定例

```
linuc    ALL=(ALL)       /sbin/iptables
```

このように管理者の権限で実行できるコマンドを各ユーザの役割に応じて必要最低限の範囲に制限することはセキュリティにおける非常に重要な要素です。

2.4.4　複数のユーザでアカウントを共有しない

Linux に限らず、利用者が多くなるとユーザ管理が複雑になります。そこで、簡単のためにひとつのユーザアカウントを複数の利用者で共有したいと思うかもしれませんが、このような運用は推奨されません。適切なパスワード管理が困難になることや操作を行った人の特定が難しくなるため、パスワードの漏洩や成りすましのリスクを高めてしまうためです。特に root などの管理者権限の共有には注意が必要です。

複数ユーザでの共同作業ではグループ権限の活用を検討すると良いでしょう。

2.5　ファイルおよびディレクトリのアクセス権を適切に設定する

ファイルおよびディレクトリには、パーミッションと呼ばれる「読み込み・書き込み・実行」の権限を設定することができます。ファイルおよびディレクトリのパーミッションは必要最低限の設定にする必要があります。パーミッションは chmod コマンドで変更することができます。

特にセキュリティ上重要なファイルには不要な権限を与えるべきではありません。セキュリティ上重要なファイルの例として、/etc/securetty、/etc/shadow、/etc/sudoers などがあります。これらのファイルは、第三者に内容を見られるだけでもセキュリティを脅かす可能性があります。

また、アクセス権と同時にファイルを所有するユーザや所有するグループについても、誰がどのような操作をするのかを検討した上で適切に設定すると良いでしょう。

Linux カーネル 2.6 以降では ACL によるきめ細かなアクセス制御を行うことができます。ACL については、5 章で詳しく説明します。

2.6 ランレベルを適切に設定し、不要なデーモンを起動させない

Linux で不必要なデーモンを起動させたままにすると、本来なら使わないはずのポートが開いた状態となるので攻撃の隙を与える可能性があります。また、CPU などのリソースを無駄に消費してしまいます。不要なデーモンが起動しないように適切に設定します。

外部へ公開する Web サーバなどで GUI を使ったサーバ管理ツールを使わないのであればランレベルを 3 に設定した上で不要なデーモンを起動しないようにすると良いでしょう。ランレベル 5 は GUI による作業を前提としており X Window System などサーバには不要とされているデーモンが多数起動します。

システム起動時におけるデフォルトランレベルは、/etc/inittab で設定します。

ランレベルを 3 に設定する場合は /etc/inittab ファイルの initdefault パラメータを次のように設定します。

```
id:3:initdefault:
```

デフォルトランレベルを適切に設定した上で、起動するデーモンを必要最低限に設定します。システム起動時に自動起動するデーモンを必要最低限にするには chkconfig や ntsysv などのコマンドを利用して設定すると良いでしょう。

2.7 不要な SUID、SGID を削除する

SUID とは、実行ファイルに設定される権限に関する属性です。

SUID が設定されている実行ファイルは、誰が実行しても、ファイル所有者の権限で処理が実行されます。例えば、passwd コマンドなど、一般ユーザが自分のパスワードを変更するために root が所有するファイルを書き換える必要のある実行ファイルに付与されています。

不要な SUID が付与されているファイルがないかを確認するには、find コマンドを使って一覧を表示することができます。

以下は、/usr/bin にあるファイルの中で所有者が root で SUID が付与されている実行ファイルを表示しています。

実行例

```
[root@localhost]# find /usr/bin -user root -perm -4000 -print
/usr/bin/sudoedit
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/staprun
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/Xorg
/usr/bin/chage
/usr/bin/at
/usr/bin/crontab
```

SUID を設定するには、chmod コマンドのパーミッション指定の先頭に 4 をつけて 4 桁の数字を指定して実行します。

実行例

```
[root@localhost]# chmod 4755 command
[root@localhost]# ls -l command
-rwsr-xr-x. 1 root root 0  1 月 27 20:22 2013 command
```

SUID が付いているファイルは所有者の権限が、rwx ではなく rws になっていることを確認してください。

SUID を取り消すには、4 を付けずに、3 桁の数字で chmod コマンドを実行します。

実行例

```
[root@localhost]# chmod 755 command
[root@localhost]# ls -l command
-rwxr-xr-x. 1 root root 0  1 月 27 20:22 2013 command
```

SGID は、実行ファイルをグループの権限で動作させる属性です。

以下の例では、/usr/bin にあるファイルの中で所有者が root で SGID が付与されている実行ファイルを表示しています。

実行例

```
[root@localhost]# find /usr/bin -user root -perm -2000 -print
/usr/bin/locate
/usr/bin/wall
/usr/bin/write
/usr/bin/ssh-agent
```

以下のコマンドでは、command に SGID を付与しています。

SUID では 4 を指定しましたが、SGID では 2 を指定します。

実行例

```
[root@localhost]# chmod 2755 command
[root@localhost]# ls -l command
-rwxr-sr-x. 1 root root 0  1 月 27 20:22 2013 command
```

グループのパーミッションが r-s となっていることを確認してください。

以上のように、SUID および SGID が付与されている実行ファイルを把握して、実行ファイルに対して余計に付与されていた場合は、権限を設定しなおすようにしましょう。

chmod コマンドで SUID や SGID の設定をするには 8 進数の数字で指定するだけでなく u+s や u-s のような文字で指定することも可能です。詳しくは man chmod などを参照してください。

2.8 ログを正確に残す

第三者から攻撃を受けた場合や不正な操作が行われた場合、サーバをネットワークから切り離すなどの被害拡大防止の措置を行い、原因や何が実行されたのかを可能な限り正確に調査把握します。調査の基

本は、ログファイルから何が行われたのかを把握することです。そのためログファイルはできるだけ残すことが必要です。

なお、ログファイルをハードディスクに保存し続けるとディスクの空き容量が減り、いずれは容量不足に陥る危険性があります。ログローテーションをかけることで容量の問題は解決しますが、古いログが消えてしまうため、ハードディスクの使用状況を考慮しながら古いログファイルは定期的に低価格の外部ディスクやテープドライブなどに移すと良いでしょう。

基本的にLinuxのシステムログはsyslogの設定で/var/log配下に保存されています。ただし各ソフトウェアの設定によっては他のディレクトリ配下に保存されている場合もあります。

また、ログファイルに残されている時刻は非常に重要な情報です。サーバの時刻がずれていると操作が行われた時刻や他のサーバとの連携における処理の前後関係などが不明瞭となります。そのため全容が把握できなくなり間違った認識をしてしまう可能性があります。サーバの時刻を正確に保つにはNTPを使って常に自動補正すると良いでしょう。

2.9 パケットフィルタリングを有効にして余計なパケットを受信しない

パケットフィルタリングは届いたパケットに対して必要なパケットは通過させ不要なパケットは破棄するためのしくみです。Linux カーネル 2.6 以降では iptables が広く使われています。

iptables を無効にするとサーバは、攻撃を意図したパケットを含めて、すべてのパケットを受信します。これはあらゆる攻撃に対して無防備と言えます。確実に信頼できるネットワークでない限り、ネットワークに接続するすべてのサーバは iptables を有効にし、適切なポリシーでパケットフィルタリングを実施すると良いでしょう。

iptables はポリシーに基づいてパケットの通過・破棄を決定します。単純なポリシーの例としては、許可する宛先ポート番号を指定する設定で、特定のポートにアクセスするパケットだけを通過させます。

iptables は、ポート番号による単純な制御だけではなく、様々なポリシーを設定することでセキュリティ対策を実施することができます。以下は、その一例です。

・特定のパケットが届いた場合、拒否または破棄した上でログに残す。
・特定の IP アドレスからのみアクセス可能とする。
・短時間に何回も送られてくるパケットは応答に制限を設けて破棄する。(DoS、DDoS 攻撃対策)

このように、単純に接続先ポート番号だけではなく、接続元や接続回数などにより制御する事で、外部からの攻撃対策を行う事も、重要なセキュリティ対策です。

iptables の使い方については 3 章で詳しく紹介します。

2.10 不要なポートを閉じる

外部からサーバにアクセスするためには、サーバ側で接続用のポートが開いている必要があります。逆に言えば、サーバ側のポートが開いていなければ外部から侵入することはできません。したがって、入り口となる開いているポートは必要最低限に制限する必要があります。

現在開いているポート番号を確認するには管理者権限で netstat コマンドを実行します。

```
netstat -ntl
```

実行例

```
[root@localhost]# netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN
```

上記の例では、22 番ポートがすべての IP アドレス(0.0.0.0)に対して開いており、25 番ポートは自分（127.0.0.1）に対してのみ開かれています。

もし不要なポートが開いている場合は、関連するデーモンを終了させるなどしてポートを閉じます。

2.11 外部からネットワーク経由で管理する場合は SSH を利用する

近年、レンタルサーバやクラウドサービスを利用する事例が多くなる中、自社内など身近にサーバを置く事が少なくなりつつあります。

例えばハウジングサービスを利用している場合、サーバにログインする度にサーバの設置場所まで移動していては非常に効率が悪いので、インターネットを経由したリモートでのログイン手段が必須となってきます。

リモートログインでシェルを実行する手段としは SSH が広く使われています。telnet などを利用する時代もありましたが、telnet は通信内容が暗号化されないため、パスワードを含む全ての通信内容が第三者に盗聴され得るという欠点があります。そのためリモートログインの目的では、認証を含むすべての通信内容が暗号化される SSH に置き換えられ、現在ではほとんど使われていません。

多くの Linux ディストリビューションでは SSH 接続を行うためのソフトウェアとして OpenSSH というパッケージが提供されています。

なお、SSH で通信が暗号化されていたとしても、パスワードが第三者に漏れていれば不正なログインを許す可能性があります。そこで SSH にはパスワードだけではなく証明書を利用した認証方法も利用することができます。

SSHにはリモートログインの機能だけでなくFTPに置き換えるためのファイル転送機能が備わっています。FTPはtelnet同様に通信内容が暗号化されていないため重要なデータをやりとりする用途には向いていません。telnetやFTP以外にも暗号化をサポートしていないサービスを利用したい場合にはSSHのポート転送機能を利用することで通信内容を暗号化することが可能です。

このように、SSHはサーバ管理上かかせないと同時に、重要なデータをインターネットなど外部のネットワーク経由で送受信する際にも利用できる非常に重要なセキュリティ対策ツールです。

SSHに関しては、第6章に詳しく説明していますので、そちらも参照してください。

3 章 iptables によるパケットフィルタリング

3.1 iptables 概要

iptables は、Linux に実装されたパケットフィルタリング機能です。

また、iptables はパケットフィルタリングだけではなくネットワークアドレス変換(NAT)や特定のルールに沿ってパケットの流れる先を変更するなどの高機能な処理を行うこともできますが、本教科書ではセキュリティ機能としてのパケットフィルタリングにターゲットを絞って説明します。

パケットフィルタリングはネットワーク上に流れているパケットをそのパケットのヘッダー情報によって受け取りを拒否したり、破棄したり、通過させるものです。このような機能は簡易な Firewall として最近のインターネット接続ルータなどでもセキュリティを確保する際に利用されています。

Linux システム上でこの機能を利用することにより、システム上に不正なパケットが流れてきた場合でも、システムに影響を与える前にパケット自体をシャットアウトすることができます。

例えば Linux システムで、あるサービスを社内の特定の範囲で提供する場合に、特定のネットワーク以外からの通信を受け付けないようにしておけば、他のネットワークからの接続開始のパケットが入ってくることを防ぐことができ、システムのセキュリティ強度を上げることが出来ます。

iptables の詳しい説明は、「Linux 2.4 Packet Filtering HOWTO」
http://archive.linux.or.jp/JF/JFdocs/packet-filtering-HOWTO.html
や、iptables のマニュアル（日本語）
http://linuxjm.sourceforge.jp/html/iptables/man8/iptables.8.html
を参考にしてください。

3.2 iptables の基本的説明とコマンド

3.2.1 iptables の概略

iptables では用途に応じて、それぞれルールを記載する「テーブル」が定義されています。iptables コマンドなどでテーブルを指定しなければテーブル「filter」が使用されます。

表 3.1　iptables におけるテーブルの種類

テーブル名	用途	チェイン
filter	パケットフィルタ(デフォルト)	INPUT/FOWARD/OUTPUT
nat	NAT 用	PREROUTING/OUTPUT/POSTROUTING
mangle	QoS/SECMARK など特別なパケット変換(iproute2 などを一緒に用いる)	PREROUTING/INPUT/OUTPUT/POSTROUTING

また、それぞれのテーブルごとに使用できるチェインが決まっており、どのタイミングで処理を施すかをこのチェインを使って指定します。

表 3.2　iptables におけるチェインの種類

チェイン	説明
INPUT	マシン自体に入ってくるパケットに対するチェイン
FORWARD	マシンを経由するパケットに対するチェイン
OUTPUT	ローカルマシンで生成されたパケットに対するチェイン
PREROUTING	パケットが入ってきた場合、すぐにそのパケットを変換するためのチェイン
POSTROUTING	パケットが出て行くときに変換するためのチェイン

外部から受信したパケット、またシステム上のプロセスから生成されたパケットに対して、どのようなチェインでパケットが処理されていくかを下の図に示します。

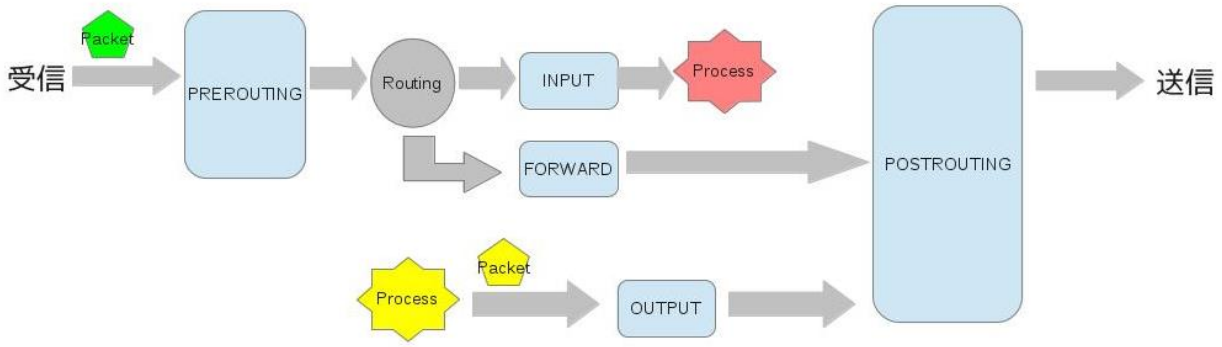


図 3.1　パケットが処理されていくチェインの順序

次に、各チェインにおいてルールを決めていきます。

パケットは各チェインで、設定された順番でルールを確認していき、条件にマッチした場合にルールで指定された処理が行われます。

すべてのルールにマッチしなかったパケットは、チェインごとのデフォルトのルールに従って処理されます。

3.2.2 iptables のコマンド

iptables を設定する際に使用するコマンドは、/sbin/iptables です。この iptables コマンドに様々なオプションをつけて、前述の各テーブル・チェインでのルールを作成・確認していきます。

iptables コマンドの基本的な使用方法を表に示します。

表 3.3　iptables コマンドの基本的な使用方法一覧

概要	記述例
ルールの追加	**iptables [-t table] -A チェイン ルールの詳細 [オプション]**
ルールの挿入	**iptables [-t table] -I チェイン [ルール番号] ルールの詳細 [オプション]**
ルールの置き換え	**iptables [-t table] -R チェイン ルール番号 ルールの詳細 [オプション]**
ルールの削除	**iptables [-t table] -D チェイン ルール番号 [オプション]**
ルールの表示	**iptables [-t table] -L [チェイン] [オプション]**
チェインのルール全削除	**iptables [-t table] -F [チェイン] [オプション]**
カウンタリセット	**iptables [-t table] -Z [チェイン] [オプション]**
新規ユーザ定義チェイン	**iptables [-t table] -N チェイン**
ユーザ定義チェイン削除	**iptables [-t table] -X [チェイン]**
ポリシー設定	**iptables [-t table] -P チェイン ターゲット [オプション]**
チェイン名前変更	**iptables [-t table] -E 旧チェイン名 新チェイン名 チ**

(*) -t でテーブル名を省略した場合には、デフォルトで filter テーブル（パケットフィルタリングに使用）が対象となる。

この中で特によく使用するのは、-A(ルール追加)/-D オプション(ルール削除)と、-L オプション(ルール確認)です。

3.3 CentOS6 上のパケットフィルタリング

3.3.1 CentOS6 上の iptables 設定

CentOS6 上の iptables は、デフォルトで有効になっています。最小構成で CentOS をインストールした場合には、下の図のようになっています。

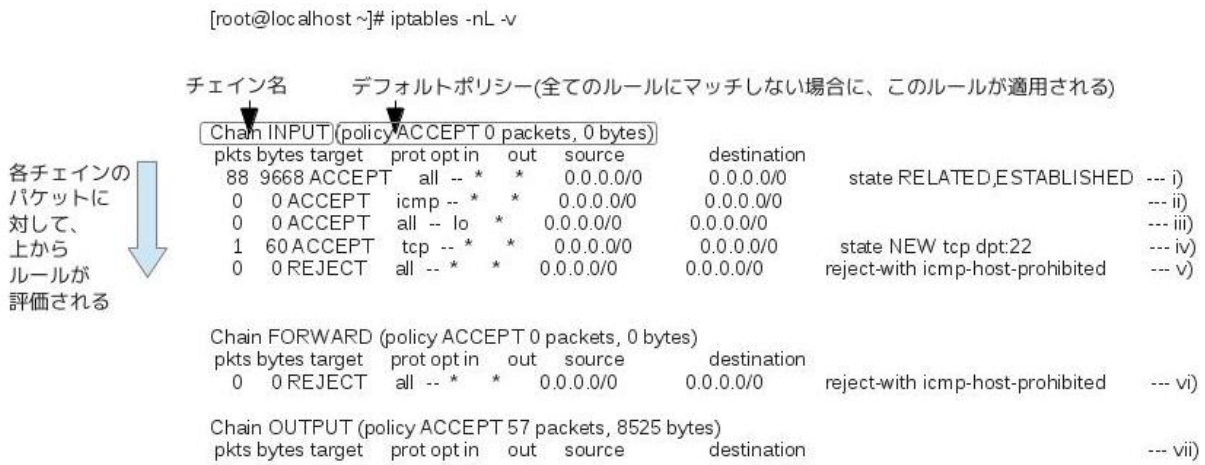


図 3.2　CentOS6 におけるデフォルトポリシー

■ INPUT チェイン(デフォルトポリシーは ACCEPT)

i) 一番上の行で、state が RELATED, ESTABLISHED のパケットは、ACCEPT(許可)になっています。RELATED は既存の接続に関係しているパケット、ESTABLISHED は過去双方向でパケットのやり取りがあった接続に属しているパケットです。ほとんどのパケットは、ここで許可されます。state の各説明は表の通りです。

表 3.4 iptables コマンドの基本的な使用方法一覧

INVALID	このパケットは既知の接続と関係していない。
ESTABLISHED	このパケットは、過去双方向にパケットがやり取りされた接続に属するパケットである。
NEW	このパケットが新しい接続を開始したか、双方向にはパケットがやり取りされていない接続に属するパケットである。
RELATED	このパケットが新しい接続を開始しているが、 FTP データ転送や ICMP エラーのように、既存の接続に関係している。

ii) icmp(ping など)に関してのパケットは、許可されます。

iii)送信元が lo(loopback)のパケットに関しては、全て許可されます。

iv) ssh(ポート番号 22 番)に対して新しい接続が発生した場合には許可されます。これにより SSH の通信が許可されます。

v) 上記以外の通信に関しては、パケットを REJECT(拒否)し、"icmp-host-prohibited"を返します。

■ FORWARD チェイン(デフォルトポリシーは ACCEPT)

vi) 転送のパケットは、全て REJECT(拒否)し、"icmp-host-prohibited"を返します。

■ OUTPUT チェイン(デフォルトポリシーは ACCEPT)

vii)OUTPUT チェインに関しては、ルールが存在しないため、デフォルトのポリシーが適用されます。デフォルトのポリシーは ACCEPT なので、OUTPUT のパケットに関しては、全て許可されます。

これら iptables の設定は、/etc/sysconfig/iptables で設定されています。

実行例

```
[root@localhost]# cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

3.3 CentOS6 での iptables 設定

CentOS6 では、iptables の設定を変更する際に iptables コマンドを直接使用する以外にも CUI/GUI のツールを使用することができます。これでより簡単に iptables の設定を変更することができます。以下、例として Web サーバを外部に公開する際(http/https)の設定を見て行きましょう。

3.3.1 iptables コマンドでの設定

最初に iptables の設定が保存されている/etc/sysconfig/iptables ファイルを直接編集する方法を紹介します。

http は Port 80、https は Port 443 を使用しますので SSH に対する既存の設定を参考に以下の例のようにそれらのポートに対しての新規接続を許可するように設定ファイルを書き換えます。その後、"/etc/init.d/iptables restart"を実行して設定ファイルを再読込させることで Port 80/443 への外部からの接続が新たに可能となります。

実行例

```
[root@www1 ~]# more /etc/sysconfig/iptables
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*nat
:PREROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT         --- ここを追加
-A INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT        --- ここを追加
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT

[root@www1 ~]# /etc/init.d/iptables restart
iptables: ファイアウォールルールを消去中:                    [  OK  ]
iptables: チェインをポリシー ACCEPT へ設定中 nat filter     [  OK  ]
iptables: モジュールを取り外し中:                            [  OK  ]
iptables: ファイアウォールルールを適用中:                    [  OK  ]
 [root@www1 ~]# iptables -nL -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   48  3582 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    1    58 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state
NEW tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp
dpt:80
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp
dpt:443
```

=== 次のページへ続く　===

```
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 31 packets, 4136 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

また、iptables コマンドを直接使用してルールやポリシーを変更する場合も考えてみましょう。

次の図のように iptables コマンドで Port 80、Port 443 に対しての新規の接続を許可するようなルールを追加します。

実行例

```
[root@localhost ~]# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -p tcp --dport 443 -j ACCEPT

[root@localhost ~]# iptables -nL -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  151 11578 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    1    58 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
    1    60 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with
icmp-host-prohibited  ---- (1)
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
---- (2)
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
---- (3)

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with
icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 23 packets, 2186 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

この状態では Port 80/443 宛のパケットは許可されません。

iptables ではパケットはルールの順番に沿って評価されます。

この場合は (2), (3) といった 80/tcp, 443/tcp 宛の許可が評価されるより先に (1) の「ここまでの条件にマッチしなかった場合には全て REJECT(拒否)」というルールが評価されます。その結果、すべてのパケットがその時点で REJECT(拒否)されてしまいます。

これを防ぐには、iptables の "-D"オプションを用いてあらかじめ (1) を削除し、その後で(2), (3)のルールを追加。そして最後に削除した (1)を追加します。

これにより、80/tcp, 443/tcp へのパケットはルールに従って許可され、それ以外のパケットが REJECT(拒否)されるようになります。

実行例

```
[root@localhost ~]# iptables -D INPUT -j REJECT --reject-with icmp-host-prohibited
[root@localhost ~]#
[root@localhost ~]# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
[root@localhost ~]# iptables -A INPUT -j REJECT --reject-with icmp-host-prohibited

[root@localhost ~]# iptables -nL -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 2014  152K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state
RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
    1    58 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp
dpt:22
    3   180 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
--- (1)
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
--- (2)
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with
icmp-host-prohibited --- (3)

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with
icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 272 packets, 29973 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

ただし、上記の設定だけではサーバを再起動させた場合に変更が引き継がれません。

下記のように /etc/init.d/iptables save を実行することで、現在の iptables の情報が /etc/sysconfig/iptables ファイルに出力され、次回サーバを再起動させた場合にも iptables で設定された情報が引き継がれるようになります。

実行例

```
[root@localhost ~]# /etc/init.d/iptables save
iptables: ファイアウォールのルールを /etc/sysconfig/iptable[  OK  ]中:
[root@localhost ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.7 on Fri Feb 15 23:21:58 2013
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [859:70467]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Fri Feb 15 23:21:58 2013
# Generated by iptables-save v1.4.7 on Fri Feb 15 23:21:58 2013
*nat
:PREROUTING ACCEPT [8:480]
:POSTROUTING ACCEPT [122:7374]
:OUTPUT ACCEPT [122:7374]
COMMIT
# Completed on Fri Feb 15 23:21:58 2013
```

3.3.2 CUI ツールでの設定

CentOS では、iptables コマンドを用いずに、もっと簡単にパケットフィルタリングの設定を変更することができるツール(system-config-firewall)が用意されています。まず、CUI ツールの使い方を説明します。

root アカウントで、yum コマンドを使い system-config-firewall-tui パッケージをインストールします。

実行例

```
[root@localhost ~]# yum -y install system-config-firewall-tui
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
c6-media                                                 | 4.0 kB     00:00 ...
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package system-config-firewall-tui.noarch 0:1.2.27-5.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                       Arch       Version           Repository    Size
================================================================================
Installing:
 system-config-firewall-tui    noarch     1.2.27-5.el6      c6-media      37 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 37 k
Installed size: 59 k
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : system-config-firewall-tui-1.2.27-5.el6.noarch                1/1
  Verifying  : system-config-firewall-tui-1.2.27-5.el6.noarch                1/1

Installed:
  system-config-firewall-tui.noarch 0:1.2.27-5.el6

Complete!
```

このツールでパケットフィルタリングの設定を行います。

root アカウントで"system-config-firewall-tui" を実行すると、図のような設定画面が表示されます。

図 3.3　CUI ツールによるファイアウォール設定

ここで「カスタマイズ」を選択すると、INPUT に対してのルールを編集することができます。

ここでは「どのポートへの新しい接続が、マシン上で許可されるか」を選択することになりますので、「WWW(HTTP)」と「安全な WWW(HTTPS)」を選択することにより、Port80/Port443 への外部からの接続が許可されるようになります。

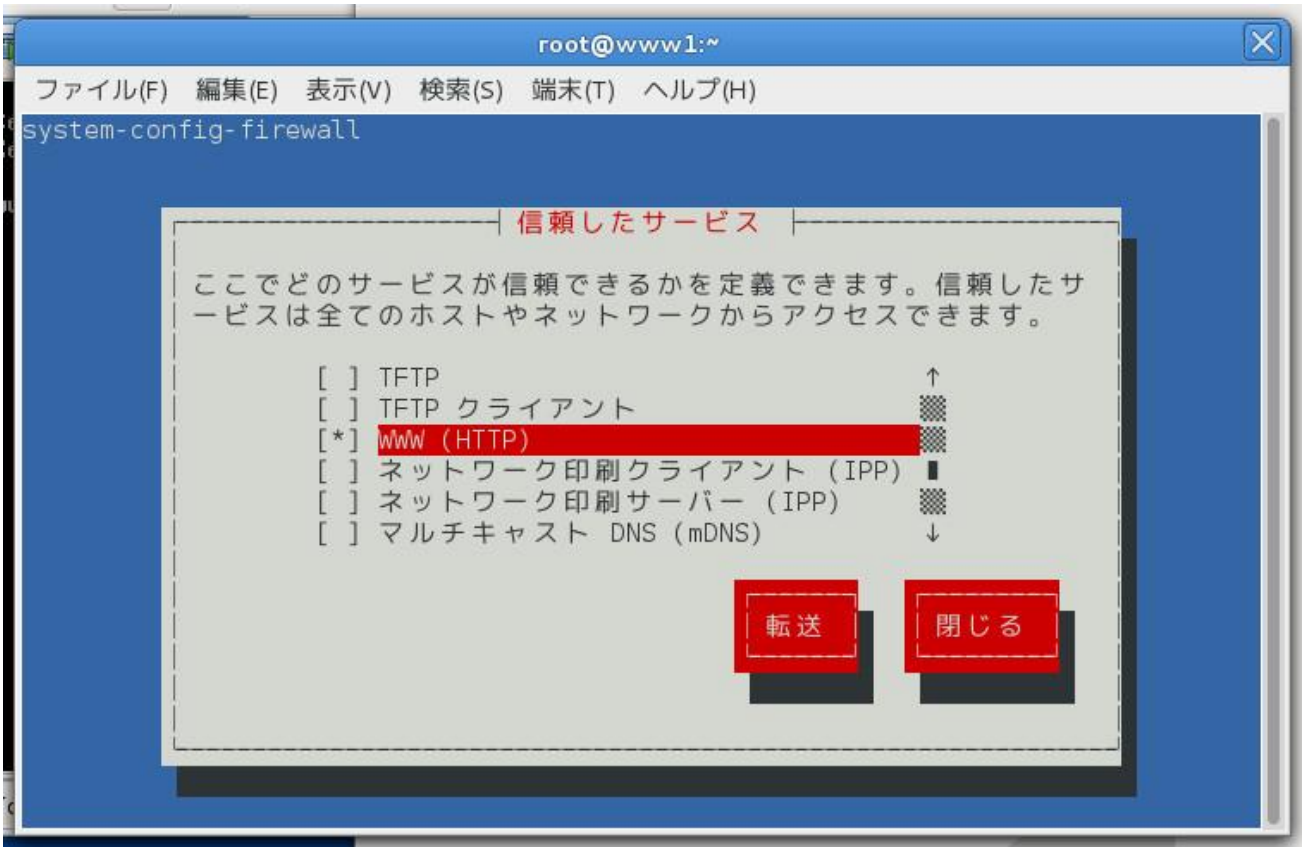


図 3.4　CUI ツールによる信頼したサービス選択（HTTP）

図 3.5　CUI ツールによる信頼したサービス選択（HTTPS）

3.3.3 GUI ツールでの設定

CentOS では、GUI ツールを使って CUI よりも簡単に iptables の設定をすることができます。

root アカウントで、yum コマンドで system-config-firewall パッケージをインストールします。

実行例

```
[root@www1 ~]# yum -y install system-config-firewall
Loaded plugins: fastestmirror, refresh-packagekit, security
Loading mirror speeds from cached hostfile
c6-media                                                 | 4.0 kB     00:00 ...
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package system-config-firewall.noarch 0:1.2.27-5.el6 will be installed
--> Finished Dependency Resolution
```

```
Dependencies Resolved

================================================================================
 Package                    Arch        Version             Repository     Size
================================================================================
Installing:
 system-config-firewall     noarch      1.2.27-5.el6        c6-media      119 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 119 k
Installed size: 577 k
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : system-config-firewall-1.2.27-5.el6.noarch                  1/1
  Verifying  : system-config-firewall-1.2.27-5.el6.noarch                  1/1

Installed:
  system-config-firewall.noarch 0:1.2.27-5.el6

Complete!
```

このツールでパケットフィルタリングの設定を行います。一般のアカウントで"system-config-firewall"を実行すると、図のように設定画面と root アカウントのパスワード入力の画面が表示されます。

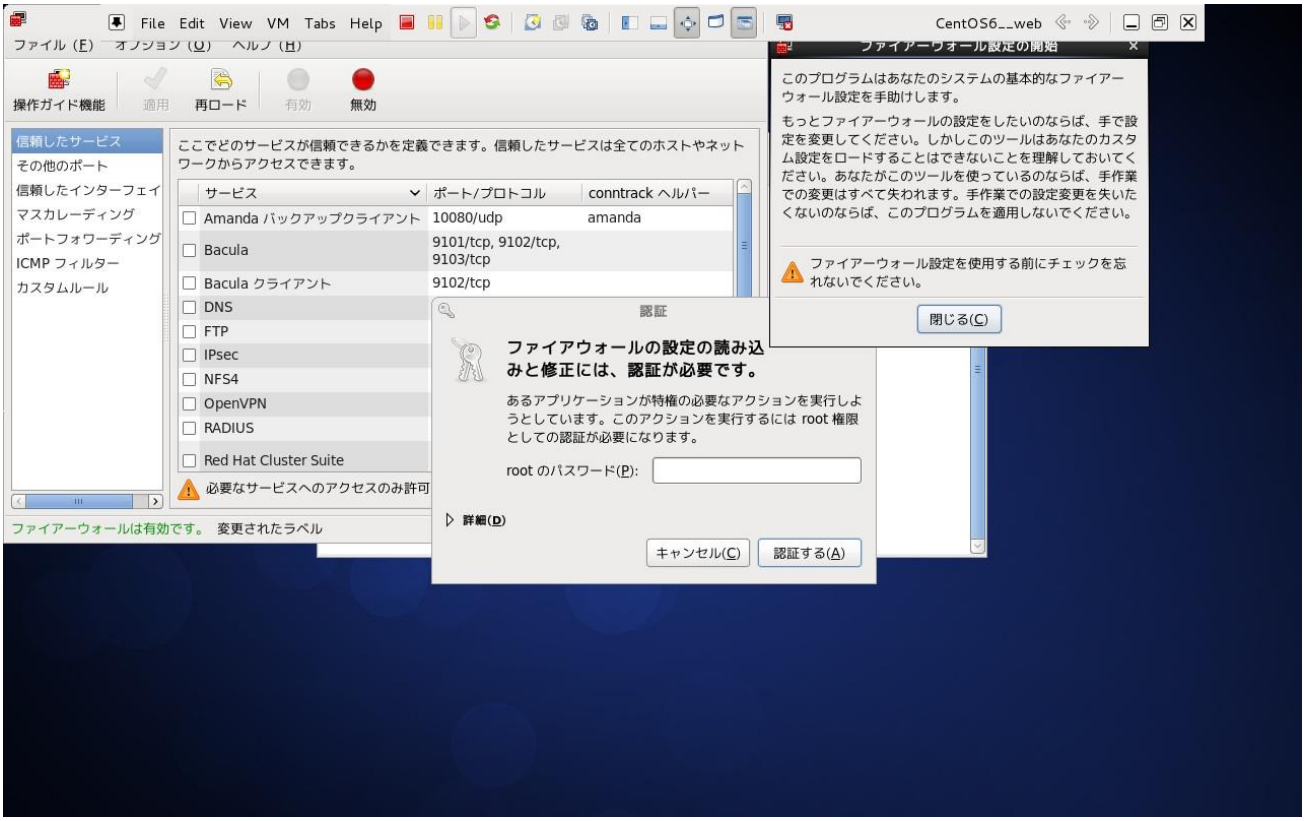


図 3.6　system-config-firewall におけるパスワード確認

このツールで、iptables 全体を有効、無効を切り替えたり、細かい設定をすることができます。なお、このツールで設定を変更したものを反映するには、ツール上部の「適用」ボタンを押す必要があります。「適用」ボタンを押すと設定を反映するかどうかの確認ダイアログが表示されます。

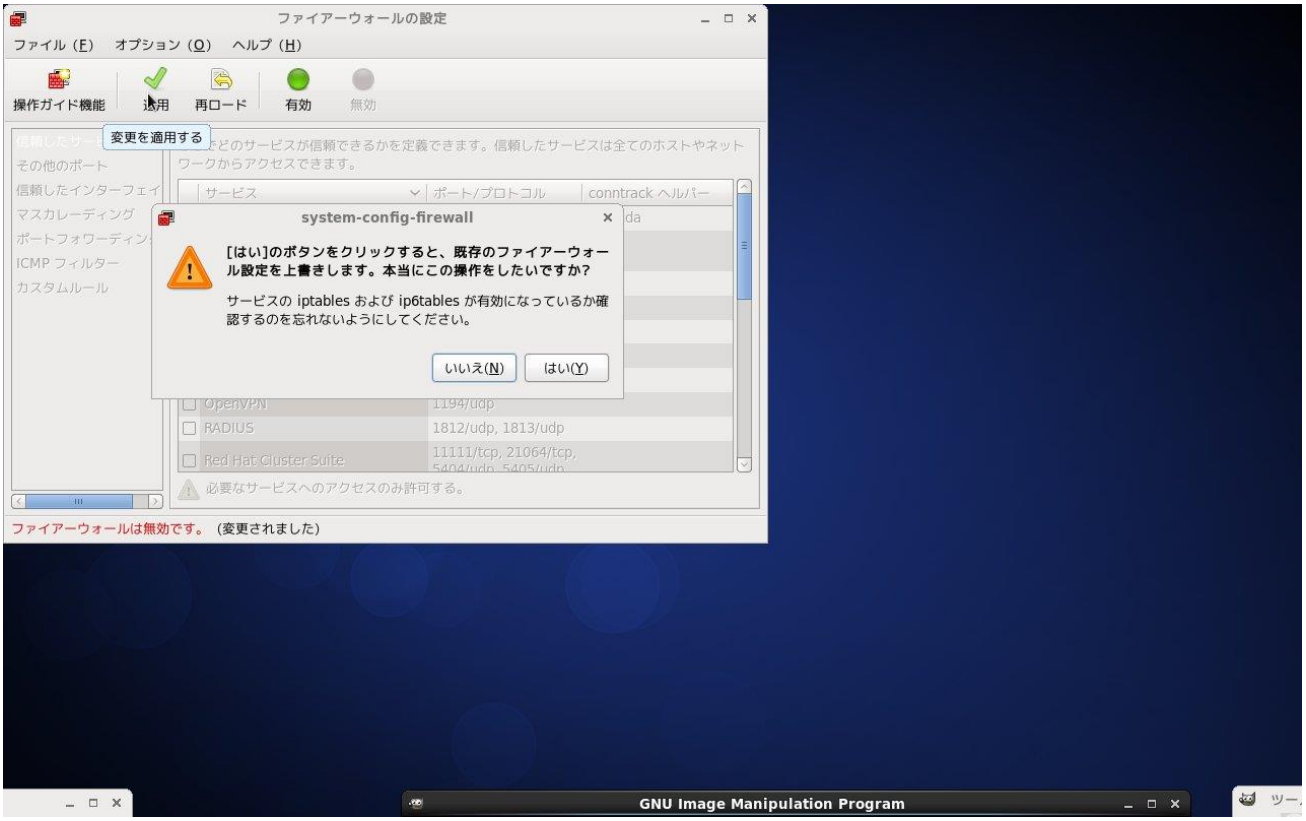


図 3.7　system-config-firewall における適用の確認

以下、設定を行うサブメニューを簡単に説明します。

1)　「信頼したサービス」

ここでどのサービスが信頼できるかを選択できます。選択したサービスに対しては、すべてのホストからアクセスすることができます。

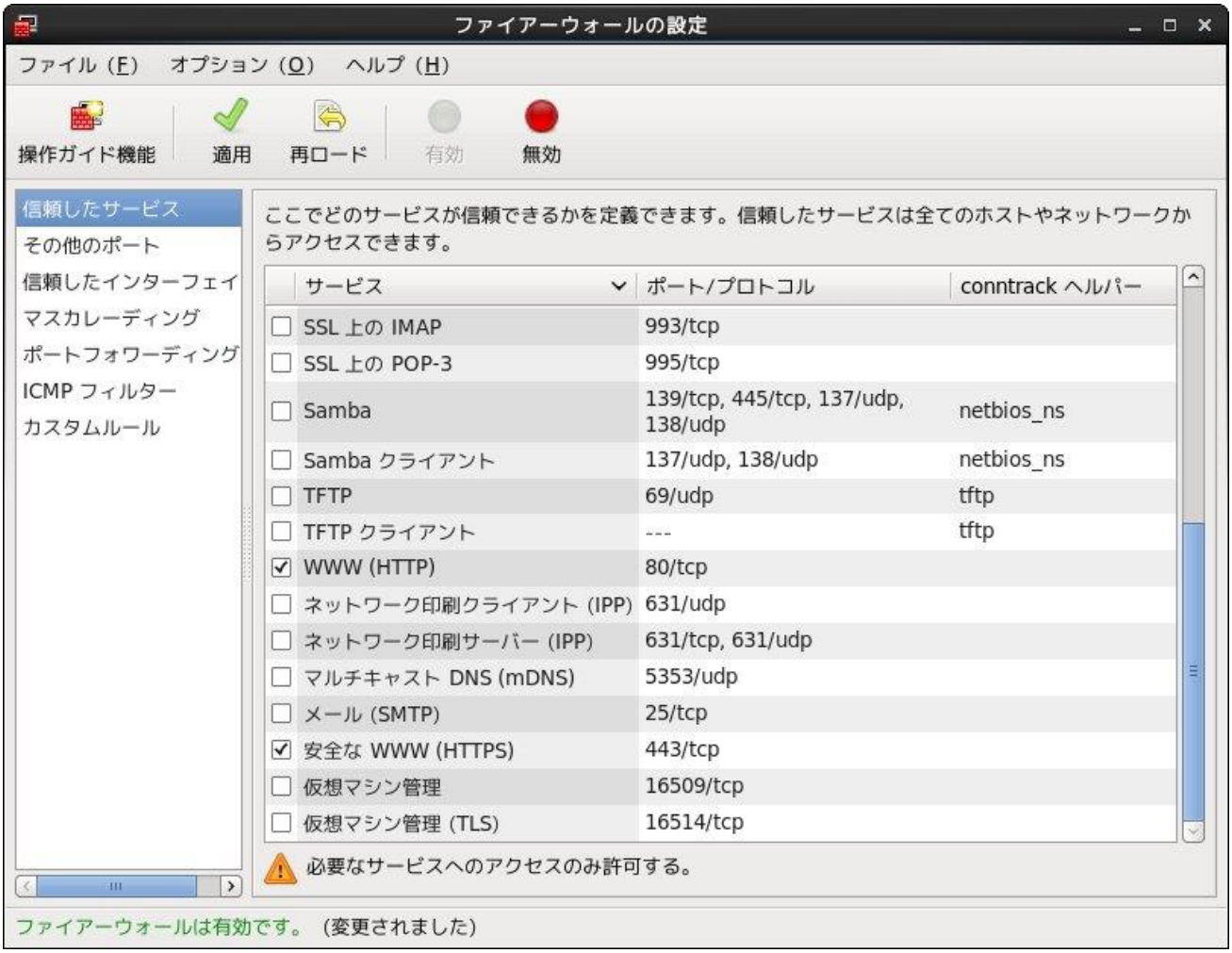


図 3.8　system-config-firewall の信頼したサービス一覧

2)　「その他のポート」

すべてのホストやネットワークからアクセスできることが必要な追加のポートか、ポートの範囲を追加できます。

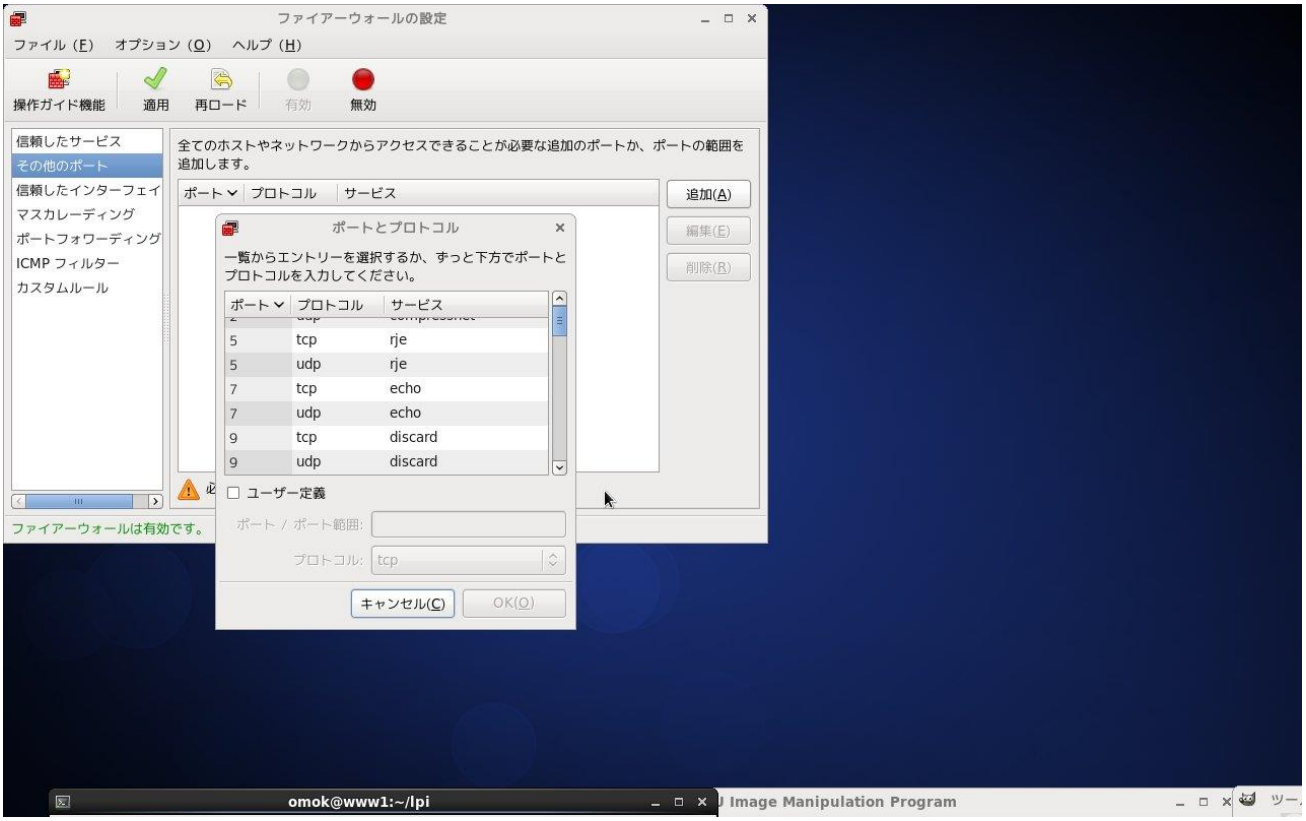


図 3.9　system-config-firewall におけるその他のポート

3)　信頼したインターフェイス

ここで選択したインターフェイスに対して、システムにフルアクセスできるという印をつけることができます。

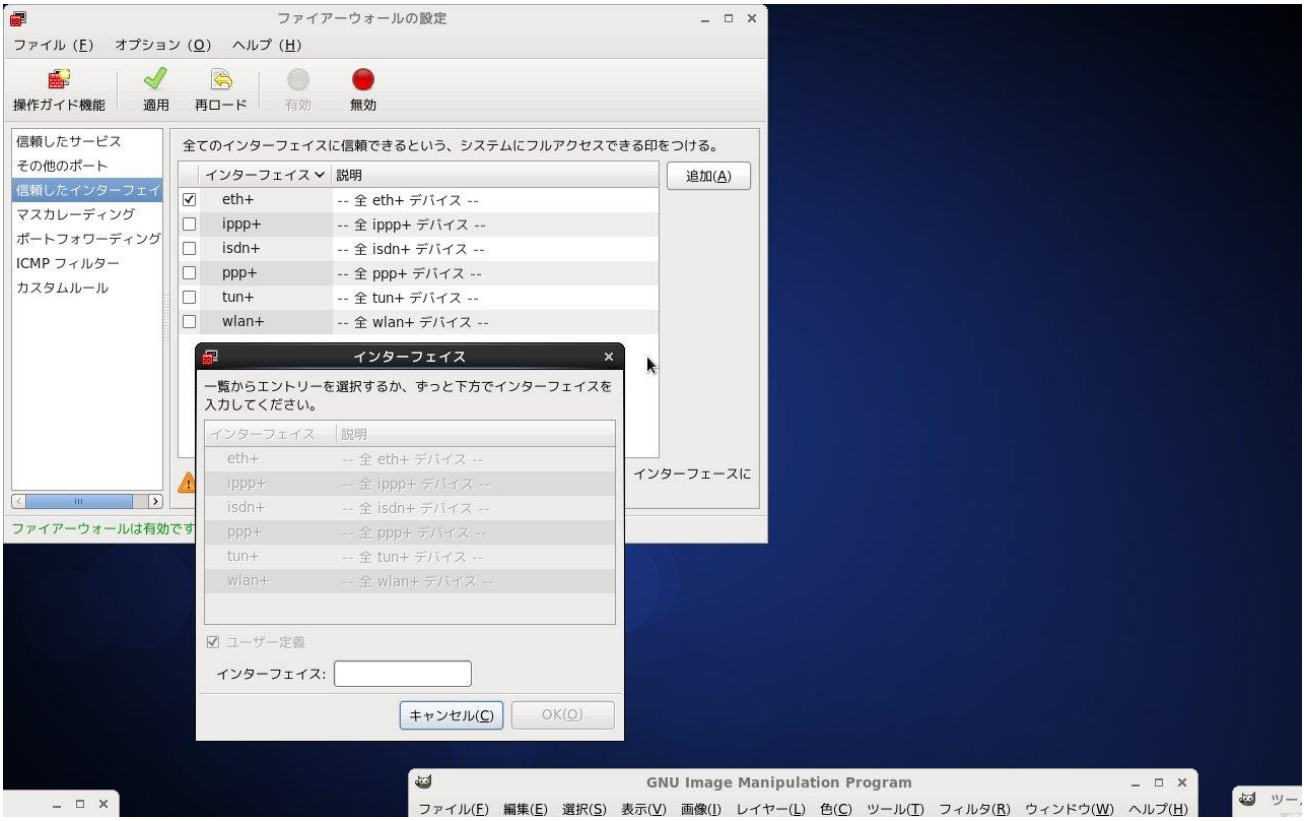


図 3.9　system-config-firewall の信頼したインターフェイス表示

4)　マスカレーディング

IP マスカレードを行う NIC を選択することができます。

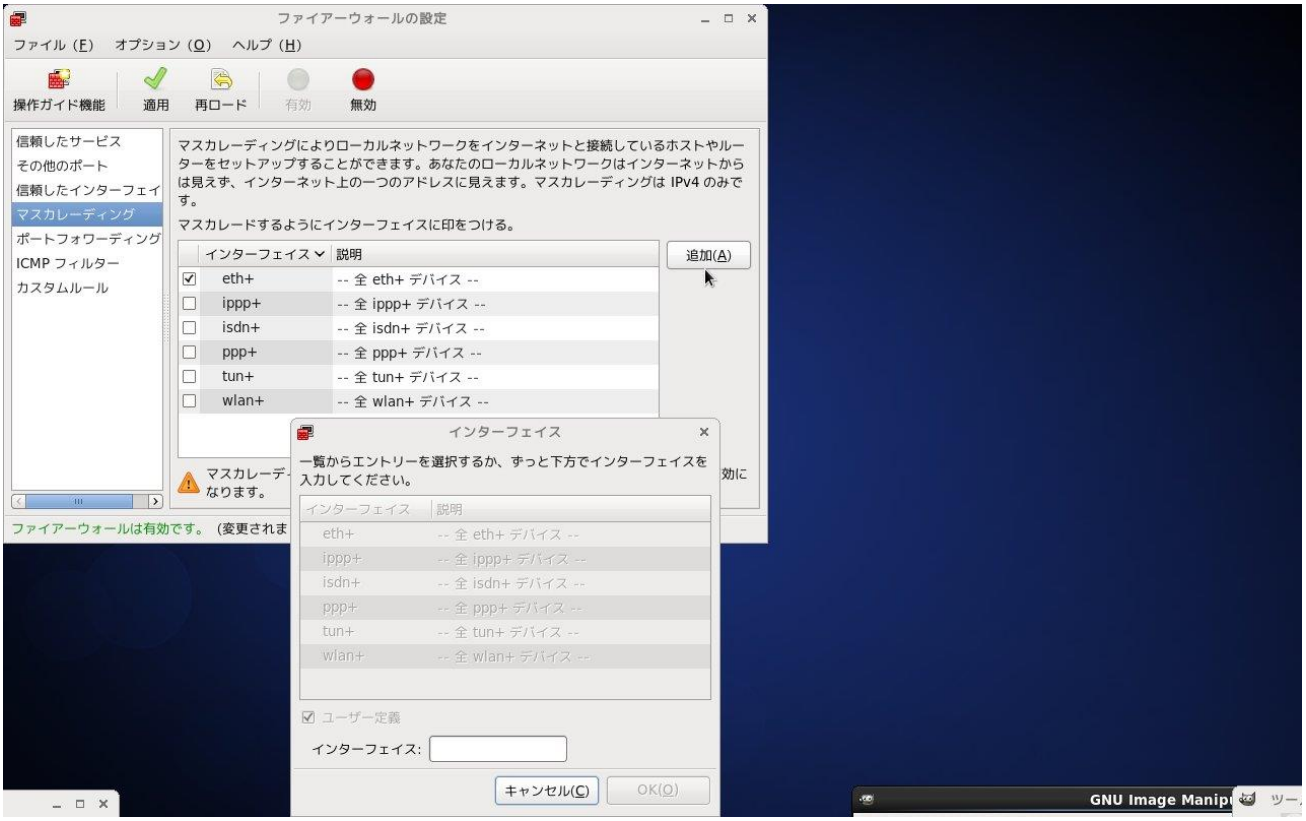


図 3.10　　system-config-firewall のマスカレーディング表示

5)　ポートフォワーディング

指定した NIC／ポートに入ってきたものを、別のホスト／ポートに転送することができます。

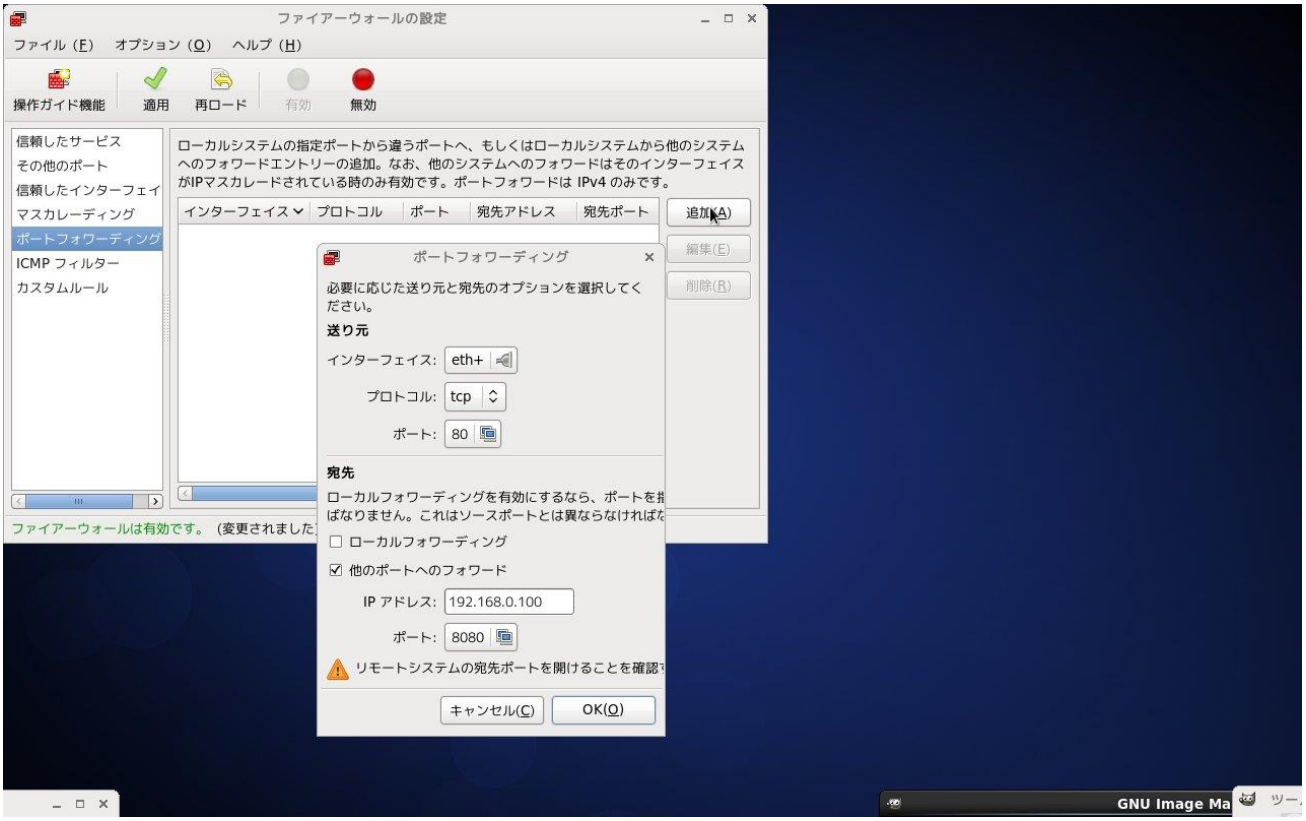


図 3.11　system-config-firewall のポートフォワーディング表示

6)　ICMP フィルタ

ICMP の応答を制限することができます。デフォルトでは無制限（すべての応答を返す）になっています。

図 3.12　system-config-firewall の ICMP フィルタ表示

7)　カスタムルール

プロトコル・テーブル名を指定して、あらかじめ記載してあるフィルタルールのファイルを読み込むことができます。

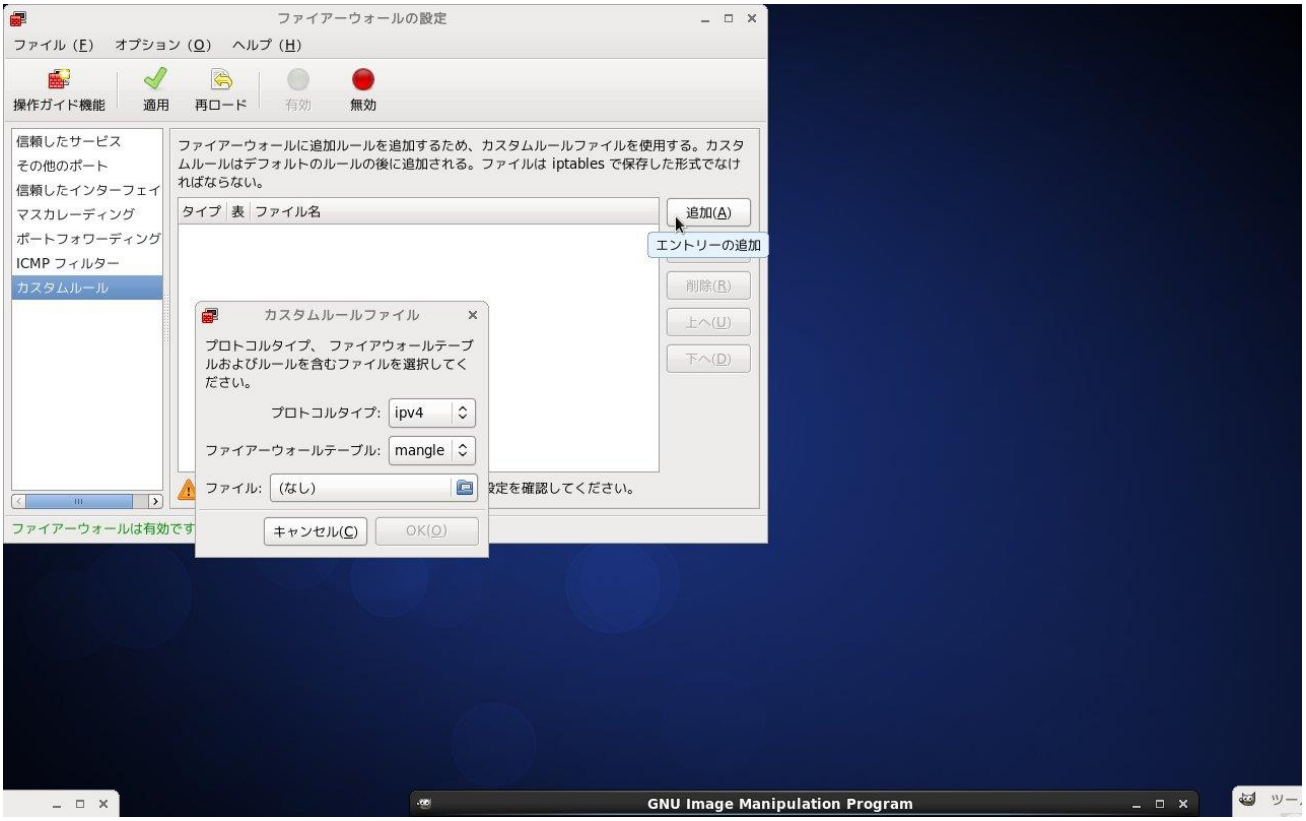


図 3.13　　system-config-firewall のカスタムルール表示

これらのメニューを使用して iptables のルールをカスタマイズできます。カスタマイズした設定を有効にするためには必ず「適用」ボタンを押してください。

3.3.4 最後に

サーバを公開する際に、インストールされているソフトを少なくするとともに、パケットフィルタリングを行うことはセキュリティを保つための基本的なテクニックです。特に iptables は簡単に設定できるツール類も用意されており技術的な文献や Web 記事などの情報源も多いため、ぜひ iptables の設定を有効にしてサーバの基本的な守りを固めてください。

４章 SELinux

4.1 SELinux 概要

SELinux は、2000 年にオープンソースとして公開され、現在ではほとんどのディストリビューションにデフォルトで組み込まれているセキュリティモジュールです。この機能を有効にすることにより通常の Linux のアクセス制御（rwx や uid/gid によるアクセス制御）に加えて SELinux でのアクセス制御が利用できます。この SELinux でのアクセス制御は root などの特権ユーザでも回避することができない強制アクセス制御（MAC: Mandatory Access Control）であるためセキュリティを強固にすることが可能です。

SELinux のコンセプトの背景や詳しい説明に関しては、様々な Web サイトに種々の解説記事が掲載されています。それらもぜひ参考にしてください。

4.1.1 SELinux を有効にするとどうなるか

CentOS をインストールすると、特に明示的に無効にしない限りデフォルトで SELinux が有効になっています。現在のシステムの全体で SELinux が有効になっているかは ″getenforce″ コマンドで確認できます。″getenforce″コマンドを実行した結果 ″Enforcing″ や ″Permissive″ と返ってくる場合はシステム上で SELinux が有効になっています。″Disabled″ が返ってくる場合はシステム上で SELinux は無効になっています。

実行例

```
[omok@cent64 ~]$ /usr/sbin/getenforce
Enforcing
```

SELinux を有効にすると、すべてのファイル／ソケットなどのリソース（object）やプロセス（subject）に通常の rwx や UID/GID とは別に「コンテキスト（contexts)」と呼ばれるものが付与されます。このコンテキストには以下の識別子が存在します。

- ユーザ(user)
- ロール(role)
- タイプ(type) -- プロセスの場合には特に「ドメイン」とも言う
- MLS(MLS) -- 高度な Multi Level Security を提供できるが、通常のシステムではあまり使われない

コンテキストはこれらの識別子によって〝ユーザ:ロール:タイプ:MLS レベル〟のように構成されています。この中でもロールとユーザはオブジェクトを限定する際などに使われる物で SELinux では大部分はタイプ／ドメインに関して記載されたポリシー（アクセス制御の設定ファイル）に基づいてアクセス制御が行われています。

例として、CentOS 6 での識別子を見てみましょう。現在、ほとんどのコマンドが SELinux 対応になっているためコマンドを実行する際に〝-Z〟オプションをつけることで SELinux の識別子が見られるようになっています。ファイルやディレクトリに関しては〝ls -lZ〟とすることで SELinux のラベルを見ることが可能です。

実行例

```
[omok@cent64 ~]$ ls -lZ /home
drwx------. omok omok unconfined_u:object_r:user_home_dir_t:s0 omok
[omok@cent64 ~]$ ls -lZ /tmp
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 keyring-GvT9he
drwx------. gdm  gdm  system_u:object_r:xdm_tmp_t:s0   orbit-gdm
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 orbit-omok
drwx------. gdm  gdm  system_u:object_r:xdm_tmp_t:s0   pulse-Nzl1ikYbhOYz
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 pulse-zkowinrYCWfA
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 virtual-omok.NNlWxt
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 virtual-omok.fhsxDC
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 virtual-omok.fqPCvR
drwx------. omok omok unconfined_u:object_r:user_tmp_t:s0 virtual-omok.zpvN3k
-rw-------. root root system_u:object_r:root_t:s0      yum.log
```

プロセスに関しては、〝ps -axZ〟とすることで、各プロセスのドメイン情報を見ることができます。

実行例

```
[omok@cent64 ~]$ ps axZ
LABEL                               PID TTY      STAT   TIME COMMAND
system_u:system_r:init_t:s0           1 ?        Ss     0:02 /sbin/init
system_u:system_r:kernel_t:s0         2 ?        S      0:00 [kthreadd]
system_u:system_r:kernel_t:s0         3 ?        S      0:00 [migration/0]
system_u:system_r:kernel_t:s0         4 ?        S      0:00 [ksoftirqd/0]
system_u:system_r:kernel_t:s0         5 ?        S      0:00 [migration/0]
system_u:system_r:kernel_t:s0         6 ?        S      0:00 [watchdog/0]
system_u:system_r:kernel_t:s0         7 ?        S      0:00 [migration/1]
```

```
system_u:system_r:kernel_t:s0       8 ?        S      0:00 [migration/1]
system_u:system_r:kernel_t:s0       9 ?        S      0:00 [ksoftirqd/1]
system_u:system_r:kernel_t:s0      10 ?        S      0:00 [watchdog/1]
system_u:system_r:kernel_t:s0      11 ?        S      0:01 [events/0]
system_u:system_r:kernel_t:s0      12 ?        S      0:00 [events/1]
system_u:system_r:kernel_t:s0      13 ?        S      0:00 [cgroup]
system_u:system_r:kernel_t:s0      14 ?        S      0:00 [khelper]
system_u:system_r:kernel_t:s0      15 ?        S      0:00 [netns]
system_u:system_r:kernel_t:s0      16 ?        S      0:00 [async/mgr]
system_u:system_r:kernel_t:s0      17 ?        S      0:00 [pm]
system_u:system_r:kernel_t:s0      18 ?        S      0:00 [sync_supers]

以下略
```

4.2 CentOS での SELinux とツール

先に紹介したように SELinux のアクセス制御は個々のファイル／プロセスに付加されている識別子によって行われています。このアクセス制御を行うルールの部分は「ポリシー」と呼ばれます。ポリシーは iptables のルールのように自作することもできますが CentOS のデフォルトでは "Targeted ポリシー" という安全面と運用面でバランスが取れたものが採用されています。そこでこのポリシーを必要に応じて調整することが利用者にとっては最も確実な SELinux のカスタマイズ方法です。

利用者はこのポリシーを直接カスタマイズすることもできますが、CentOS5 以降は CUI/GUI のツールで簡単に設定することが可能です。

4.2.1 CUI ツール（semanage）

RHEL5/CentOS5 から追加されたコマンドです。man ページに "SELinux Policy Management tool" と書いてあることからわかる通り、SELinux に関する様々な設定を行うことができます。

このコマンドは、デフォルトではインストールされておらず、"policycoreutils-python" パッケージに含まれています。policycoreutils-python パッケージは root で yum コマンドを用いてインストールすることができます。

実行例

```
[root@cent64 ~]# yum -y install policycoreutils-python
```

semanage コマンドはシステム上の次の項目を制御できます。

・SELinux の ON/OFF
・ファイル／ディレクトリに対してのセキュリティコンテキスト（ユーザ:ロール:タイプ:MLS）の変更
・ユーザに対するセキュリティコンテキストの割り当て
・インターフェース、ポートなどネットワーク周りに関してのセキュリティコンテキストの割り当て
・boolean（後述）の ON/OFF

semanage コマンドで制御できる項目は、man コマンドで表示されるマニュアルや "--help" オプションで表示される使用方法などにも記載されています。

実行例

```
[root@cent64 ~]# semanage --help
/usr/sbin/semanage:
semanage [ -S store ] -i [ input_file | - ]
semanage [ -S store ] -o [ output_file | - ]

semanage login -{a|d|m|l|D|E} [-nrs] login_name | %groupname
semanage user -{a|d|m|l|D|E} [-LnrRP] selinux_name
semanage port -{a|d|m|l|D|E} [-nrt] [ -p proto ] port | port_range
semanage interface -{a|d|m|l|D|E} [-nrt] interface_spec
semanage module -{a|d|m} [--enable|--disable] module
semanage node -{a|d|m|l|D|E} [-nrt] [ -p protocol ] [-M netmask] addr
semanage fcontext -{a|d|m|l|D|E} [-efnrst] file_spec
semanage boolean -{d|m} [--on|--off|-1|-0] -F boolean | boolean_file
semanage permissive -{d|a|l} [-n] type
semanage dontaudit [ on | off ]

以下略
```

4.3 GUI ツール ("SELinux Management"/system-config-selinux)

RHEL5/CentOS5 から追加された、SELinux の設定を行える GUI ツールです。このツールもデフォルトではインストールされておらず、"policycoreutils-gui" パッケージに含まれています。

policycoreutils-gui パッケージは root で yum コマンドを用いてインストールすることができます。

実行例

```
[root@cent64 ~]# yum -y install policycoreutils-gui
```

policycoreutils-gui は、GNOME メニューの「システム」→「管理」→「SELinux Management」で開くことができます。

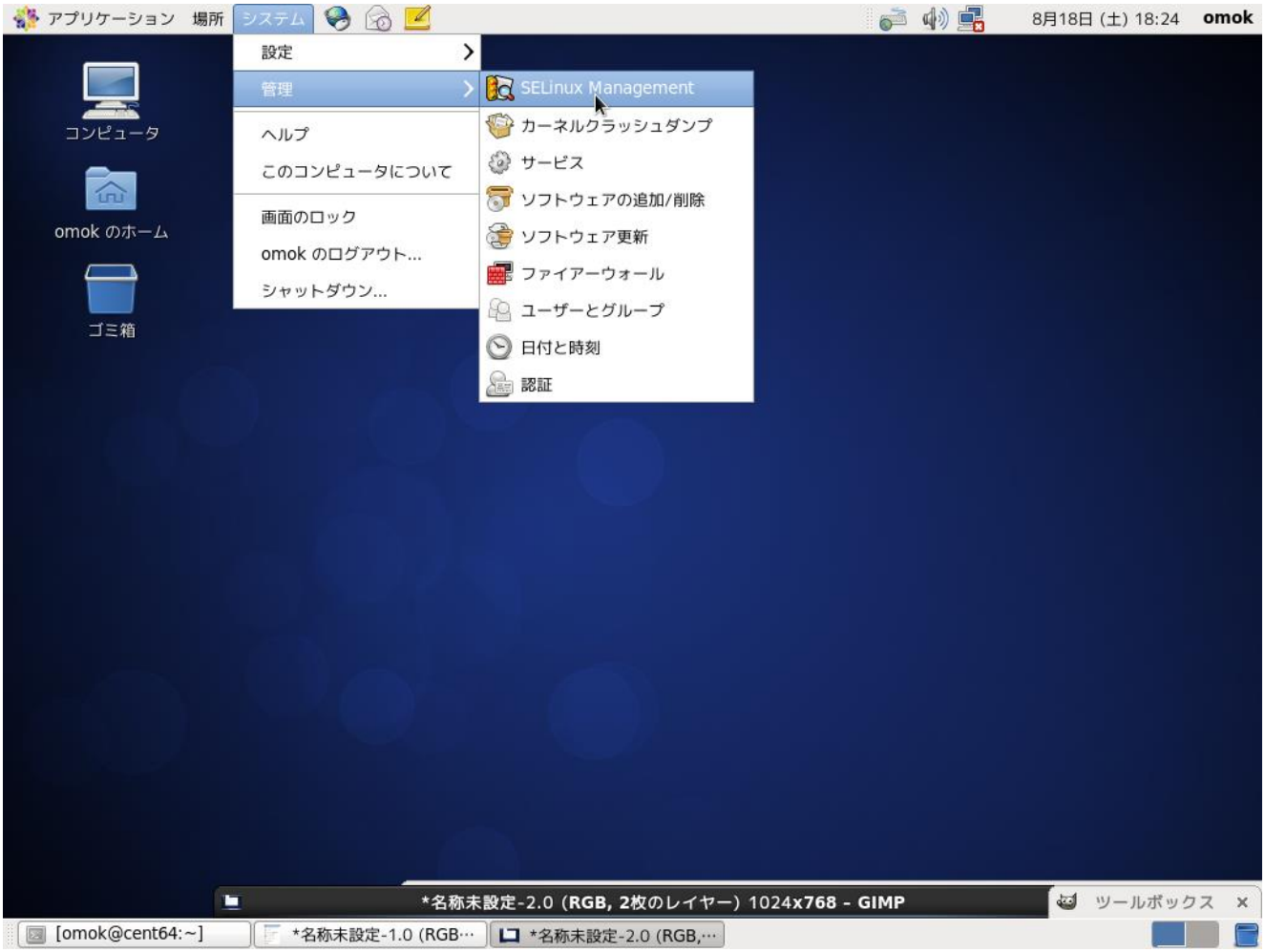


図 4.1　SELinux Management の起動

このツールでは CUI ツールの semanage と同じく次の項目を制御できます。

- SELinux の ON/OFF
- ファイル／ディレクトリに対してのセキュリティコンテキスト（ユーザ:ロール:タイプ:MLS）の変更
- ユーザに対するセキュリティコンテキストの割り当て
- インターフェース、ポートなどネットワーク周りに関してのセキュリティコンテキストの割り当て
- boolean（後述）の ON/OFF
- ポリシーモジュール単位の SELinux の有効／無効（Permissive）

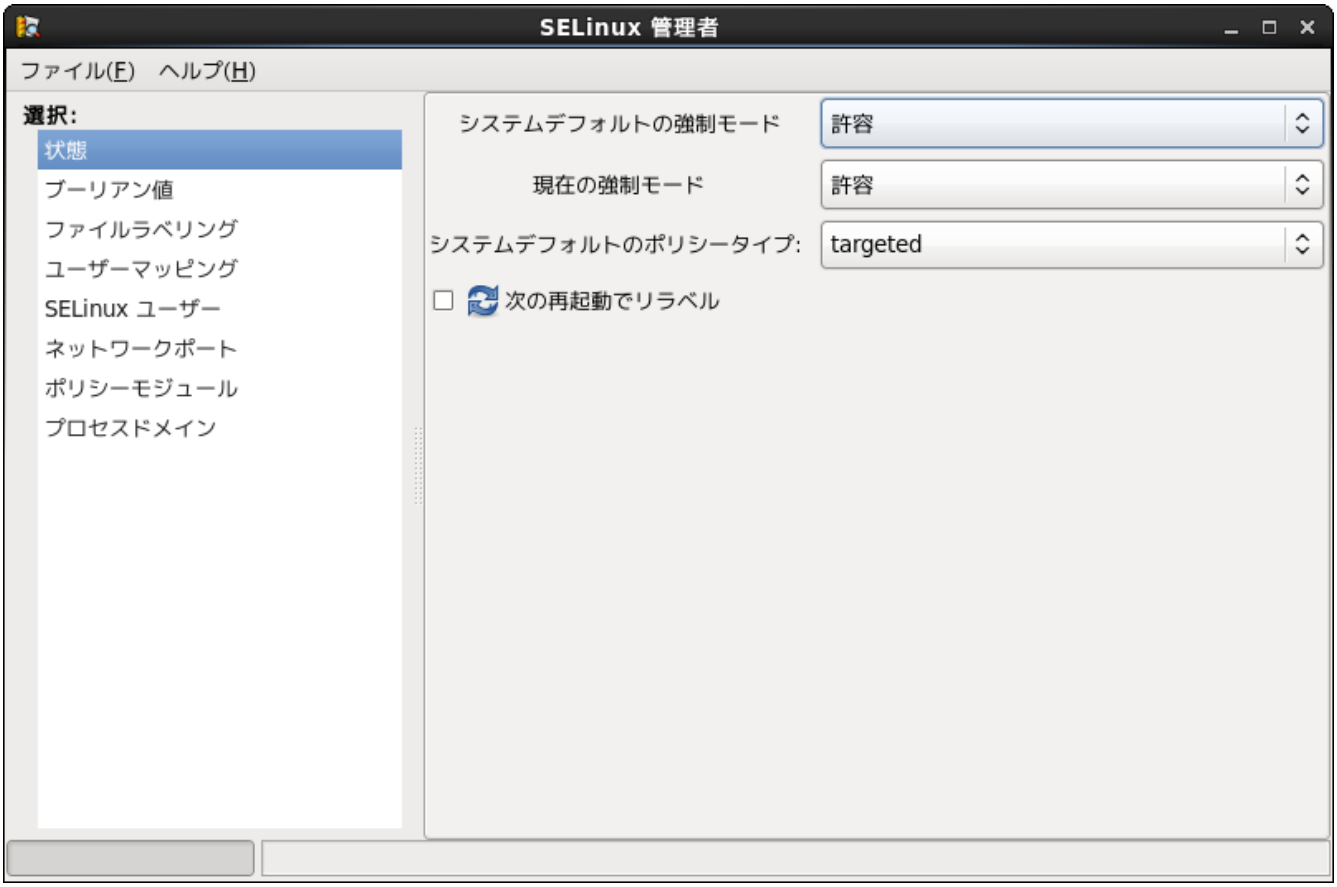


図 4.2　SELinux Management における起動後の画面

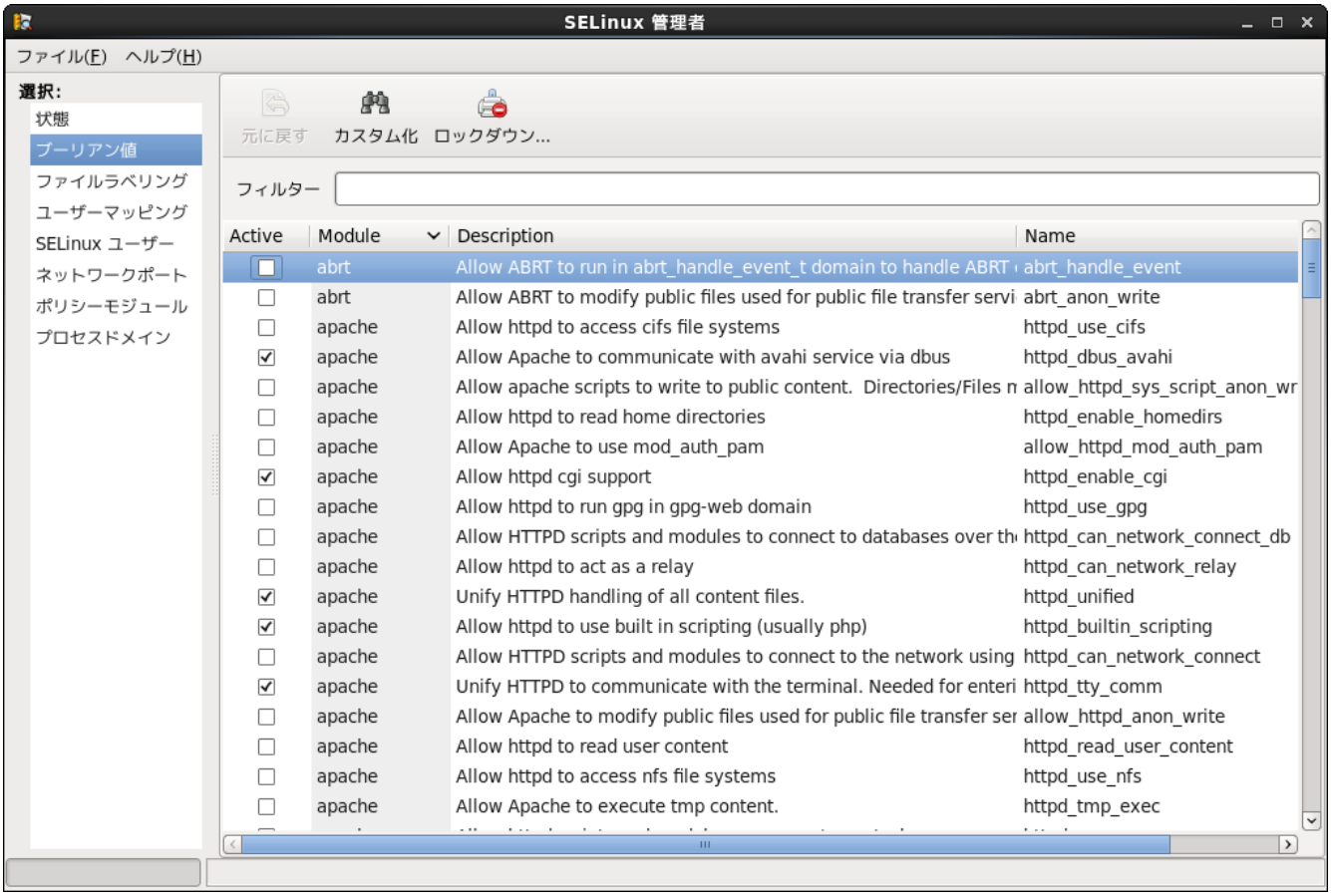


図 4.3　SELinux Management におけるブーリアン値の設定一覧

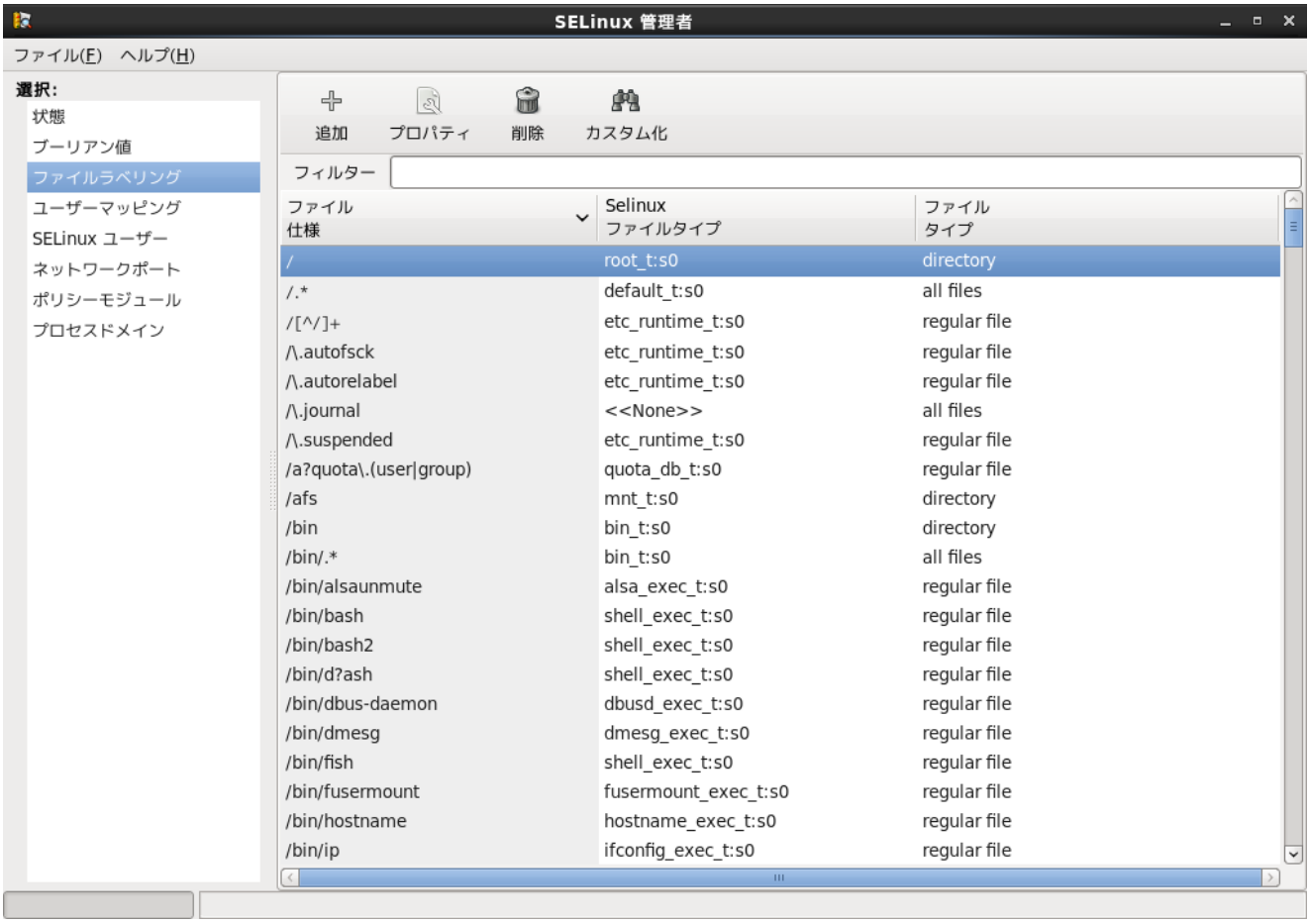


図 4.4　SELinux Management におけるファイルラベリング一覧

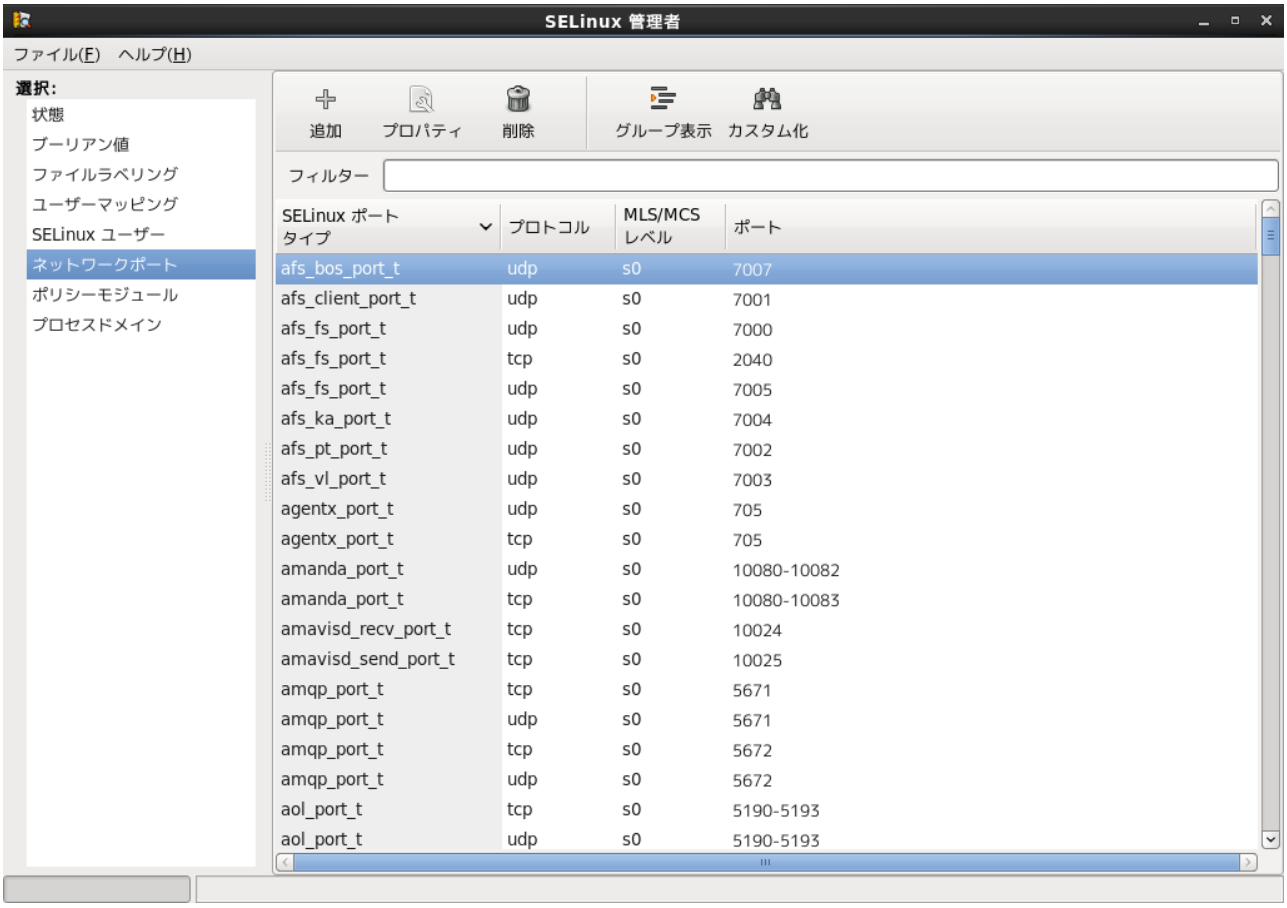


図 4.5　SELinux Management におけるネットワークポート一覧

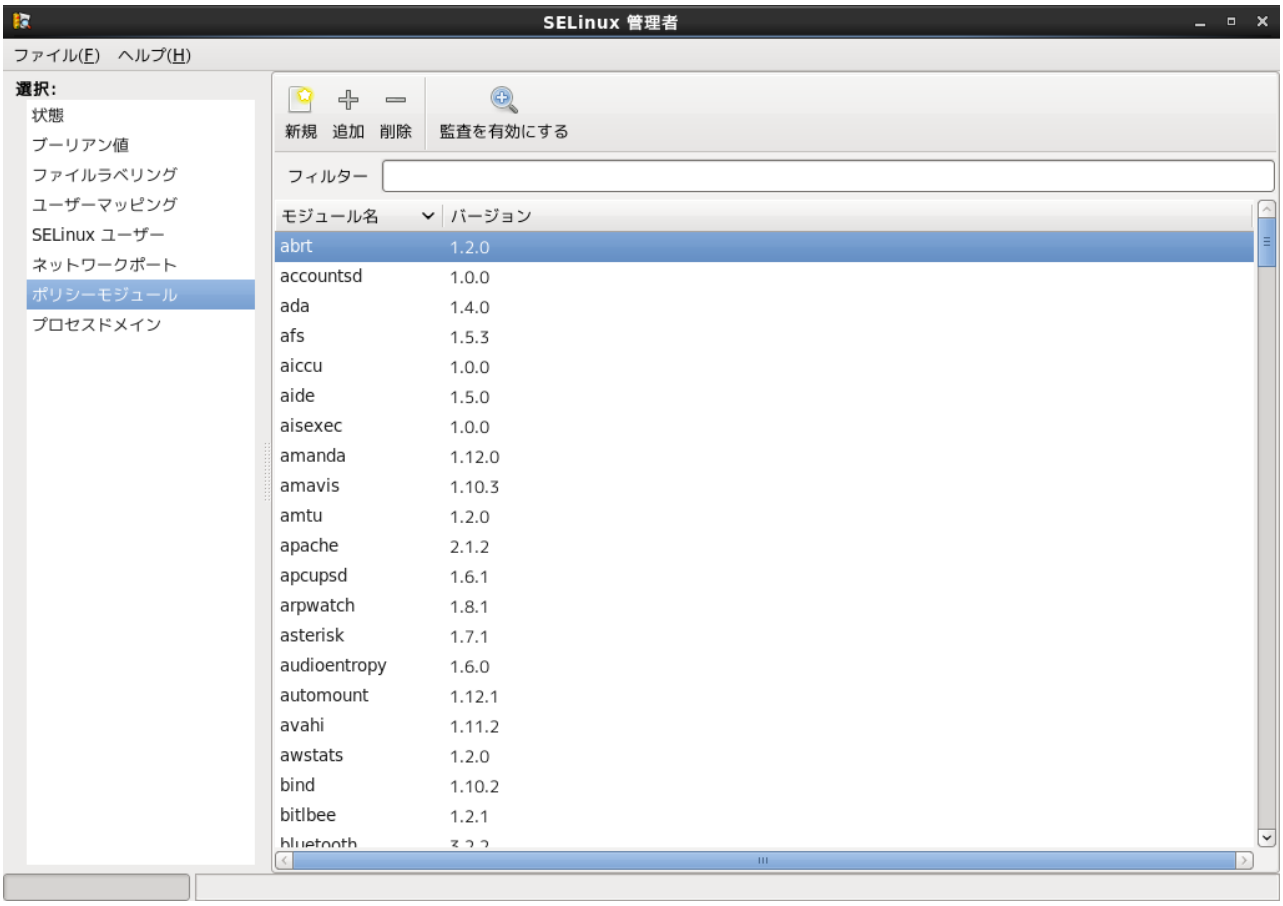


図 4.6　SELinux Management におけるポリシーモジュール一覧

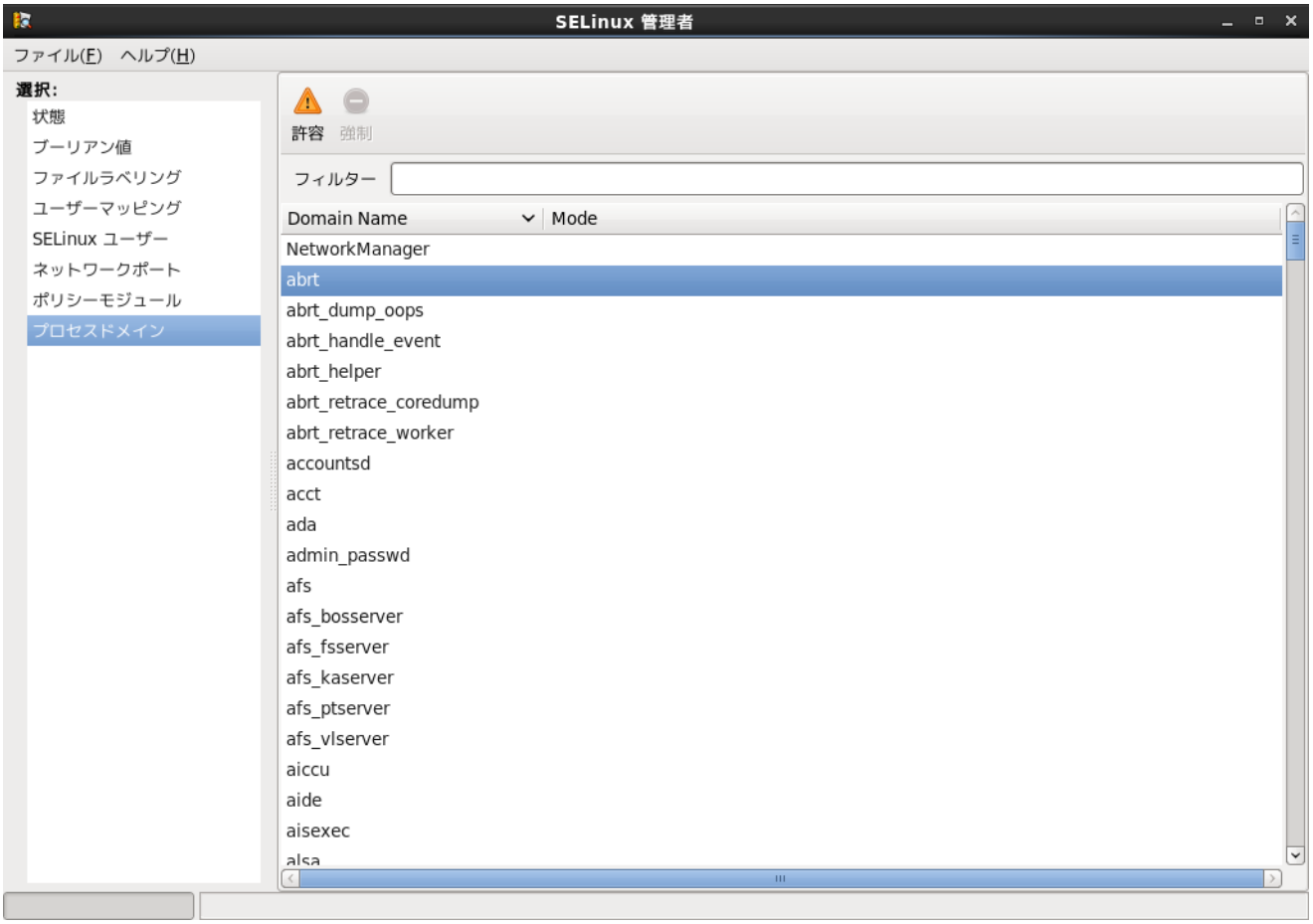


図 4.7　SELinux Management におけるプロセスドメイン一覧

4.3 CentOS 6 での SELinux の調整

4.3.1 Boolean（ブーリアン値）

SELinux ではポリシー自体を変更することは繁雑なため、ポリシーそのものは変更せずに一部のセキュリティ機能の ON/OFF を切り替えることで必要なセキュリティ強度を調整できるようになっています。この機能を "Boolean" と呼びます。

この "Boolean" で ON/OFF の調整ができるセキュリティ項目及び説明は "semanage boolean -l" というコマンドで取得できます。CentOS 6 では、約 150 種の項目が ON/OFF で調整可能になっています。

実行例

```
[omok@cen[root@cent64 ~]# semanage boolean -l
SELinux boolean                 State  Default 説明

ftp_home_dir                    (オフ   ,   オフ)  Allow ftp to read and write files in the user
home directories
smartmon_3ware                  (オフ   ,   オフ)  Enable additional permissions needed to
support devices on 3ware controllers.
xdm_sysadm_login                (オフ   ,   オフ)  Allow xdm logins as sysadm
xen_use_nfs                     (オフ   ,   オフ)  Allow xen to manage nfs files
mozilla_read_content            (オフ   ,   オフ)  Control mozilla content access
ssh_chroot_rw_homedirs          (オフ   ,   オフ)  Allow ssh with chroot env to read and write
files in the user home directories
tftp_anon_write                 (オフ   ,   オフ)  Allow tftp to modify public files used for
public file transfer services.
allow_console_login             (オン   ,   オン)  Allow direct login to the console device.
Required for System 390
spamassassin_can_network        (オフ   ,   オフ)  Allow user spamassassin clients to use the
network.
httpd_can_network_relay         (オフ   ,   オフ)  Allow httpd to act as a relay
openvpn_enable_homedirs         (オン   ,   オン)  Allow openvpn to read home directories
allow_execheap                  (オフ   ,   オフ)  Allow unconfined executables to make their
heap memory executable.  Doing this is a
iring text relocation that are not labeled textrel_shlib_t)
allow_httpd_sys_script_anon_write (オフ   ,   オフ)  Allow apache scripts to write to public
content.  Directories/Files must be labeled public_rw_content_t.
mysql_connect_any               (オフ   ,   オフ)  Allow mysqld to connect to all ports

以下略
```

これらの boolean 値は、CUI ツールでは setsebool コマンドや semanage コマンド、GUI ツールでは上述の SELinux Management ツールでリアルタイムに変更ができます。

例えば、CentOS 6 上で SELinux を有効にしている場合、デフォルトではローカルユーザでの ftp 接続ができません。

実行例

```
testuser@lucretia:~$ ftp 172.16.148.174
Connected to 172.16.148.174.
220 (vsFTPd 2.2.2)
Name (172.16.148.174:testuser): testuser
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/testuser
Login failed.
ftp> bye
421 Service not available, remote server has closed connection
```

この場合に、"allow_ftpd_full_access" という boolean 値を有効 “1” に設定すればローカルユーザで ftp 接続が可能になります。

実行例

```
[root@cent64 ~]# semanage boolean --on allow_ftpd_full_access
[root@cent64 ~]# semanage boolean -l
SELinux boolean                State  Default 説明

httpd_enable_cgi               (オン   ,   オン)  Allow httpd cgi support
virt_use_nfs                   (オフ   ,   オフ)  Allow virt to manage nfs files
-- snip ---
allow_ftpd_full_access         (オン   ,   オン)  Allow ftp servers to login to local users and
read/write all files on the system, governed by DAC.
-- snip ---
cron_can_relabel               (オフ   ,   オフ)  Allow system cron jobs to relabel filesystem
for restoring file contexts.
git_system_use_cifs            (オフ   ,   オフ)  Allow Git daemon system to access cifs file
systems.
```

```
testuser@lucretia:~/lpi/selinux$ ftp 172.16.148.174
Connected to 172.16.148.174.
220 (vsFTPd 2.2.2)
Name (172.16.148.174:testuser): testuser
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> by
221 Goodbye.
```

4.3.2　ファイルラベリング

前章で説明した CUI/GUI ツールを用いて、ファイルシステムのコンテキストの確認／変更ができます。

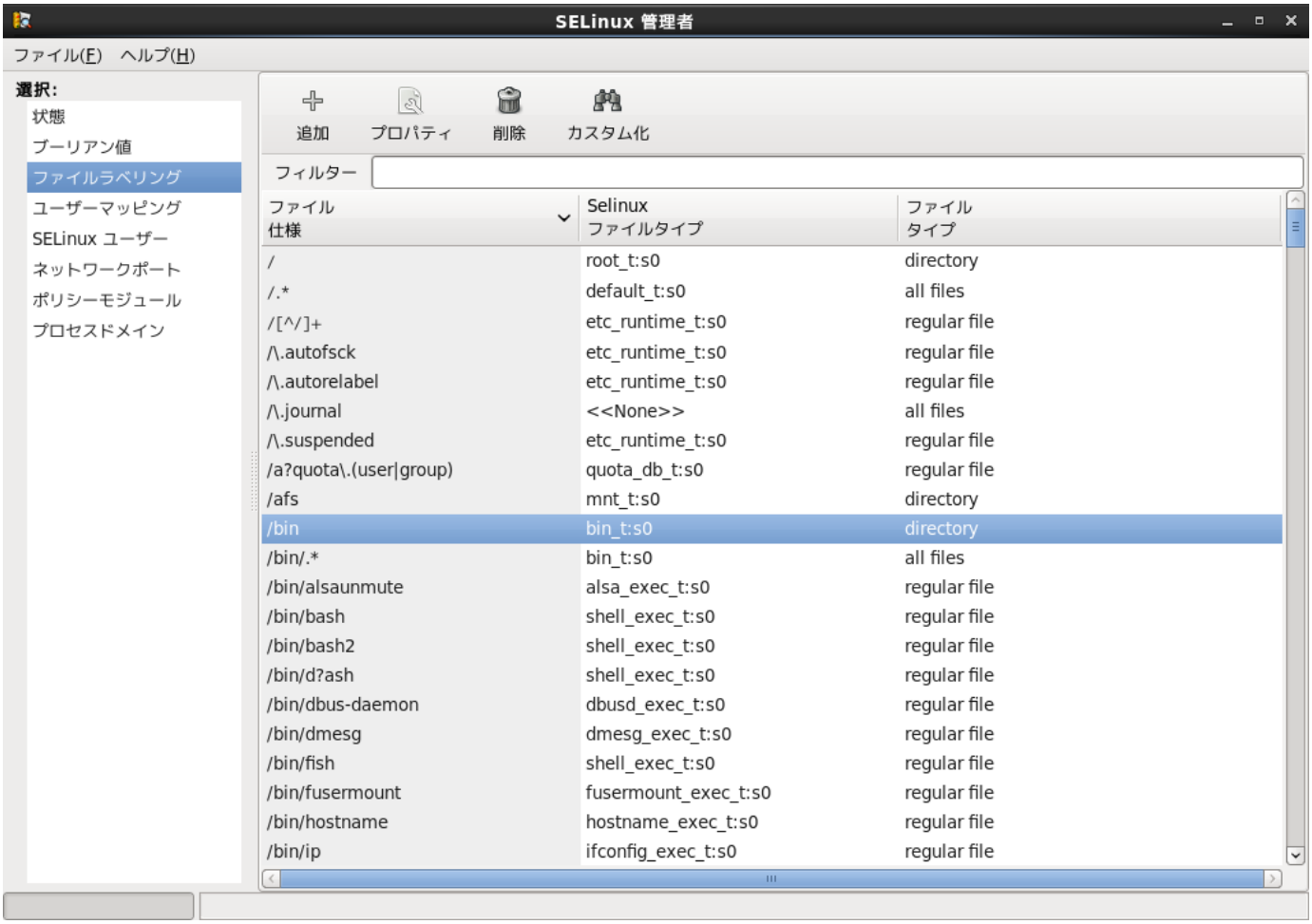


図 4.8　SELinux Management におけるファイルラベリング一覧

SELinux を有効にした際にプログラムがきちんと動作しない場合は、ファイルに対して適切なコンテキストが指定されていない可能性があります。その場合はこのツールを用いてコンテキストを修正する必要があります。

4.3.3　プロセスドメイン

CentOS 6 からは、各プロセスドメインに SELinux の有効・無効（SELinux のアクセス制御下に置くか否か）を指定することができます。これにより SELinux を有効にした際にシステムがきちんと動作しなくなった場合には、システム全体で SELinux を無効にするのではなく、システムが動作しなくなったことに関連する特定のドメインのみを SELinux の制御下から外すことで、全体的なセキュリティをなるべく高く保ちつつシステムを正常に動かすことができます。

例えば、Apache の CGI がきちんと動作しなくなった場合には、httpd_user_script や httpd_sys_script を〝許容(Permissive モード)〟にします。これにより、CGI の script 関連が SELinux の制御下から外れるので、この状態でシステムがきちんと動作するかを確認し、それでも動作しない場合には更にいくつかのドメインを〝許容〟にします。

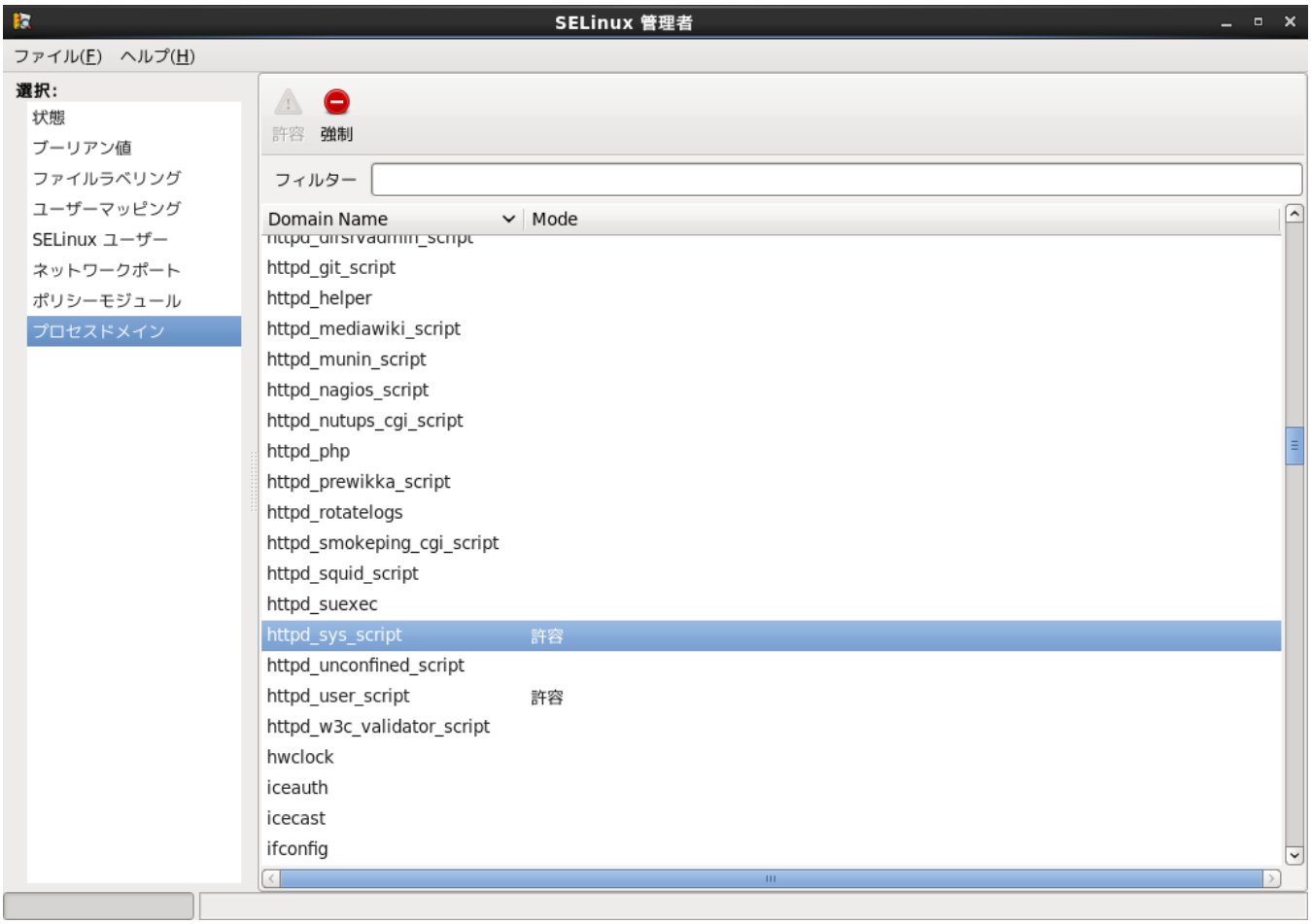


図 4.9　SELinux Management におけるプロセスドメイン一覧

もちろん許容したことでセキュリティの強度は低下します。しかし「SELinux を利用して、とりあえず、一定のレベルのセキュリティを維持した状態で運用したい」というニーズを満たすことができますので、そのようなニーズがある場合にはこの方法を検討しても良いでしょう。

セキュリティの強度を高めるには、すべてのドメインで「許容(Permissive)」ではなく「強制(Enforcing)」で動作するように CGI のスクリプトを変更するか、可能であればポリシーの修正を行うことが望ましいです。

4.4　SELinux による問題のトラブルシューティング

以前に比べて SELinux ポリシーも洗練されたため可能性は格段に低くなりましたが SELinux が有効になっていることによりアプリケーションが正常に動かなくなってしまう場合があります。この節では、そのような場合の対応方法を説明します。

4.4.1　問題の切り分け

最初に、「SELinux が原因で動作しなくなっているのか」を見分ける必要があります。

一番簡単な方法は、root アカウントで “setenforce 0”として、システム全体の SELinux を許容（Permissive）モードにし、アプリケーションが動作するかを確認することです。ほとんどの場合、この方法で切り分けができます。

ただし、この場合にはやはり SELinux は厳密な意味では動作していますので、より確実に確認したい場合には、/etc/selinux/config を次のように変更します。

SELINUX=enforcing
の行を
SELINUX=disabled
に変更。

システムを再起動し、SELinux を確実に無効な状態にしてアプリケーションが動作するかを確認します。これにより、アプリケーションが正常に動作するようになった場合には、SELinux のポリシーがアプリケーションで必要とする動作を拒否したために問題を起こしていますので次の項に進みます。

SELinux を無効にしてもアプリケーションが正常に動作しない場合には、SELinux とは無関係の箇所で問題が発生していますので、個別にトラブルシューティングを行なってください。

4.4.2　ログの確認

SELinux が問題になっている場合には、/var/log/audit/audit.log に SELinux により動作が拒否された際のログが出力されていますので、これを確認します。

例として、CentOS6.1 で vsftpd が動作しなくなる問題を使ってログの見方とトラブルシューティング方法を説明しましょう（この問題は、既に CentOS では修正されていますので、最新の CentOS を使用した場合には発生しません)。

vsftp で一般ユーザのログインを許可していても、一般ユーザで ftp にログインするとエラーが出力されてログインできません。

実行例

```
testuser@lucretia:~/lpi/selinux.20121110$ ftp 172.16.148.174
Connected to 172.16.148.174.
220 (vsFTPd 2.2.2)
Name (172.16.148.174:omok): omok
331 Please specify the password.
Password:
500 OOPS: cannot change directory:/home/omok
Login failed.
ftp>
```

この際に、/var/log/audit/audit.log には、次のようなログが出力されています。

出力例

```
type=USER_AUTH msg=audit(1352814625.935:17238): user pid=3095 uid=0 auid=0 ses=1
subj=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:authentication acct="omok"
exe="/usr/sbin/vsftpd" hostname=172.16.148.1 addr=172.16.148.1 terminal=ftp res=success'
type=USER_ACCT msg=audit(1352814625.936:17239): user pid=3095 uid=0 auid=0 ses=1
subj=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:accounting acct="omok"
exe="/usr/sbin/vsftpd" hostname=172.16.148.1 addr=172.16.148.1 terminal=ftp res=success'
type=CRED_ACQ msg=audit(1352814625.938:17240): user pid=3095 uid=0 auid=0 ses=1
subj=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:setcred acct="omok"
exe="/usr/sbin/vsftpd" hostname=172.16.148.1 addr=172.16.148.1 terminal=ftp res=success'
type=AVC msg=audit(1352814625.940:17241): avc:  denied  { search } for  pid=3100 comm="vsftpd"
name="home" dev=dm-0 ino=130563 scontext=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=system_u:object_r:home_root_t:s0 tclass=dir
type=SYSCALL msg=audit(1352814625.940:17241): arch=c000003e syscall=80 success=no exit=-13
a0=7ffd4a3c6030 a1=1f4 a2=0 a3=7fffbce10210 items=0 ppid=3095 pid=3100 auid=0 uid=0 gid=0
euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500 tty=(none) ses=1 comm="vsftpd"
exe="/usr/sbin/vsftpd" subj=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023 key=(null)
```

ログの中で「denied（拒否）」と書かれている箇所を探します。この例では、最後から二番目の行に図のように denied があります。

```
                                                        拒否された動作
type=AVC msg=audit(1352814625.940:17241): avc:  denied  { search } for  pid=3100
      アクセス元       アクセス先
comm="vsftpd" name="home" dev=dm-0 ino=130563

scontext=unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023      アクセス元のコンテキスト

tcontext=system_u:object_r:home_root_t:s0 tclass=dir
              アクセス先のコンテキスト          アクセス先の種類
```

このログは、vsftpd（コンテキスト：unconfined_u:system_r:ftpd_t:s0-s0:c0.c1023）が、search というアクセスを home（コンテキスト：system_u:object_r:home_root_t:s0）というディレクトリ（dir）に行い、拒否されたという意味です。

4.4.3 audit（監査）の有効化とモジュールの作成

1）SELinux のポリシーで必要とされるログを確実に取得するために、SELinux を Permissive モードにして audit（監査）を有効にします。今回は ftp ドメインを対象とします。GUI ツール（SELinux Management ツール）を使用して図のように問題が発生しているサービスに対して audit を有効にします。

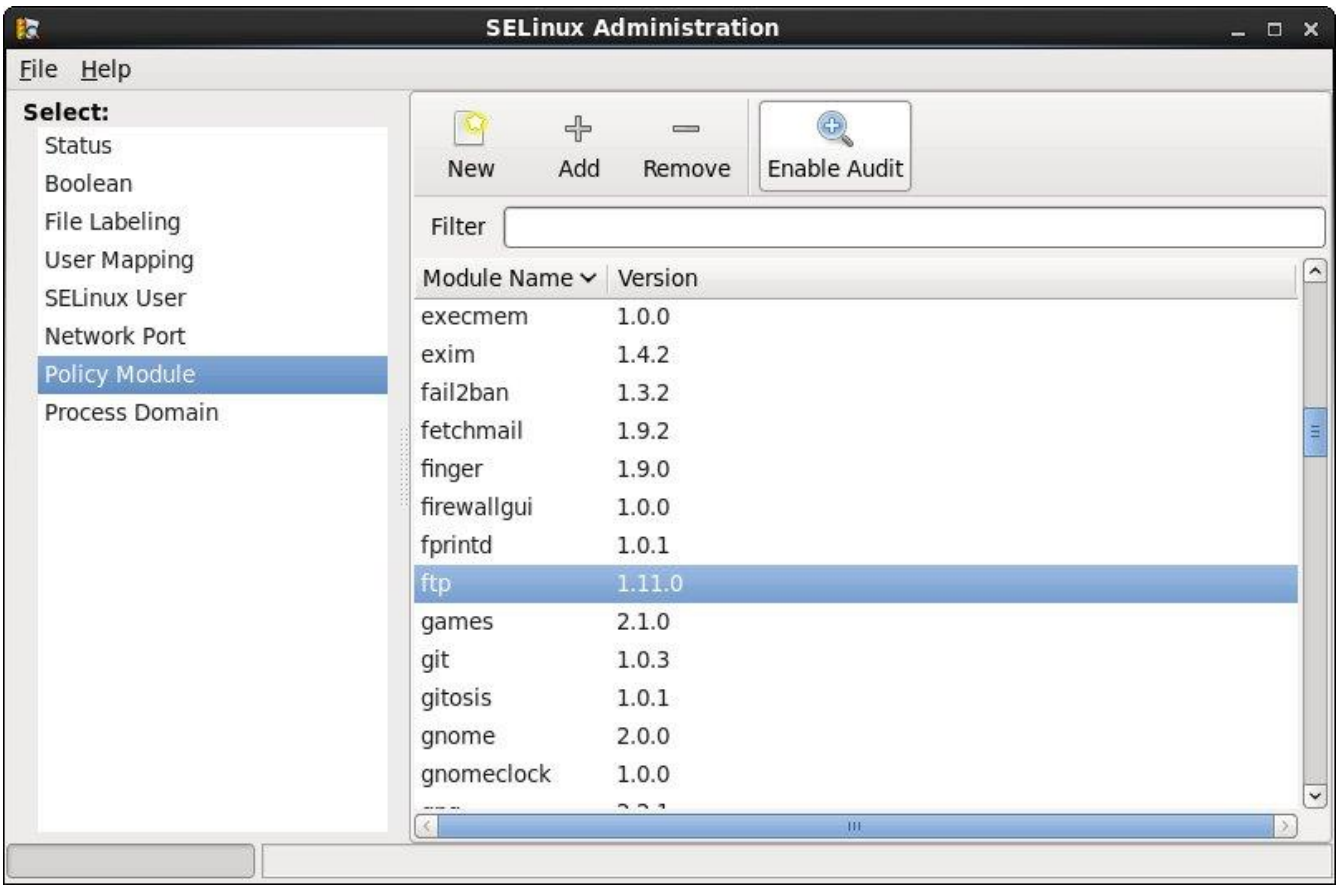


図 4.10　SELinux Management におけるポリシーモジュール管理

2) この状態において、一般ユーザでシステムに ftp ログインすると、下記のようなエラーが発生します。

出力例

```
type=AVC msg=audit(1352909794.156:232): avc:  denied  { rlimitinh } for  pid=2459
comm="unix_chkpwd" scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=system_u:system_r:chkpwd_t:s0-s0:c0.c1023 tclass=process
type=AVC msg=audit(1352909794.156:232): avc:  denied  { siginh } for  pid=2459
comm="unix_chkpwd" scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=system_u:system_r:chkpwd_t:s0-s0:c0.c1023 tclass=process
type=AVC msg=audit(1352909794.156:232): avc:  denied  { noatsecure } for  pid=2459
comm="unix_chkpwd" scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=system_u:system_r:chkpwd_t:s0-s0:c0.c1023 tclass=process
type=SYSCALL msg=audit(1352909794.156:232): arch=c000003e syscall=59 success=yes exit=0
a0=7fd5f325dc78 a1=7fffb2e120d0 a2=7fd5f3463e88 a3=7 items=0 ppid=2457 pid=2459 auid=4294967295
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295
comm="unix_chkpwd" exe="/sbin/unix_chkpwd" subj=system_u:system_r:chkpwd_t:s0-s0:c0.c1023
key=(null)
type=USER_AUTH msg=audit(1352909794.189:233): user pid=2457 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:authentication
acct="omok" exe="/usr/sbin/vsftpd" hostname=router addr=172.16.148.1 terminal=ftp res=success'
type=USER_ACCT msg=audit(1352909794.190:234): user pid=2457 uid=0 auid=4294967295
ses=4294967295 subj=system_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:accounting acct="omok"
exe="/usr/sbin/vsftpd" hostname=router addr=172.16.148.1 terminal=ftp res=success'
type=CRED_ACQ msg=audit(1352909794.190:235): user pid=2457 uid=0 auid=4294967295 ses=4294967295
subj=system_u:system_r:ftpd_t:s0-s0:c0.c1023 msg='op=PAM:setcred acct="omok"
exe="/usr/sbin/vsftpd" hostname=router addr=172.16.148.1 terminal=ftp res=success'

type=AVC msg=audit(1352909794.193:236): avc:  denied  { search } for  pid=2462 comm="vsftpd"
name="home" dev=dm-0 ino=130563 scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=system_u:object_r:home_root_t:s0 tclass=dir
type=AVC msg=audit(1352909794.193:236): avc:  denied  { search } for  pid=2462 comm="vsftpd"
name="omok" dev=dm-0 ino=130825 scontext=system_u:system_r:ftpd_t:s0-s0:c0.c1023
tcontext=unconfined_u:object_r:user_home_dir_t:s0 tclass=dir
type=SYSCALL msg=audit(1352909794.193:236): arch=c000003e syscall=80 success=yes exit=0
a0=7fd5f9079c40 a1=1f4 a2=0 a3=7fffb2e11f90 items=0 ppid=2457 pid=2462 auid=4294967295 uid=0
gid=0 euid=500 suid=500 fsuid=500 egid=500 sgid=500 fsgid=500 tty=(none) ses=4294967295
comm="vsftpd" exe="/usr/sbin/vsftpd" subj=system_u:system_r:ftpd_t:s0-s0:c0.c1023 key=(null)
```

このログを ftp.log として、適当なディレクトリに保存しておきます。

3) 保存した ftp.log を元に、ログから SELinux モジュールを作成するツール audit2allow コマンドを実行して、ftp が動作するのに必要な SELinux のポリシーを作成します。

```
# audit2allow -M ftp_local -i ftp.log
```

4)　コマンドを実行したディレクトリ内に次の 2 つのファイルができます。

ftp_local.te　　----------　　ftp が動作するのに必要な SELinux のポリシー

ftp_local.pp　　----------　　上記の ftp_local.te をシステムが読み込める形（バイナリ）にしたもの

まずは ftp_local.te を確認してみましょう。

設定例

```
module ftp_local 1.0;

require {
        type chkpwd_t;
        type ftpd_t;
        type home_root_t;
        type user_home_dir_t;
        class process { siginh noatsecure rlimitinh };
        class dir search;
}

#============= ftpd_t ==============
allow ftpd_t chkpwd_t:process { siginh rlimitinh noatsecure };
#!!!! This avc can be allowed using one of the these booleans:
#     allow_ftpd_full_access, ftp_home_dir

allow ftpd_t home_root_t:dir search;
#!!!! This avc can be allowed using one of the these booleans:
#     allow_ftpd_full_access, ftp_home_dir

allow ftpd_t user_home_dir_t:dir search;
```

ftp_local.te ファイルは大きく上の「require」の項と下の「allow」の項にわかれます。

require 部分には、このポリシーモジュールが必要とするドメインや定義が記載されます。

allow 部分には、ftp が正しく動作するために許可するべきアクセスが記載されています。例えば、

```
allow ftpd_t home_root_t:dir search;
```

の箇所では、ftpd プロセスに付加される ftpd_t ドメインが、home_root_t タイプが付加されたディレクトリに対して、search を行うことを許可するという意味になります。

これらの詳しい内容に関しても、Web サイトに種々の解説記事が載っていますので、そちらを参考にしてください。

この ftp_local.pp を

```
# semodule -i ftp_local.pp
```

としてシステムに読み込ませます。

5) システムの SELinux を Enforcing モードにします。この状態で、ftp が実行できることが確認できます。

実行例

```
omok@localhost:~$ ftp 172.16.148.188
Connected to 172.16.148.188.
220 (vsFTPd 2.2.2)
Name (172.16.148.188:omok): omok
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> by
221 Goodbye.
```

また、このままでは ftp の audit ログが/var/log/audit/audit.log を圧迫してしまいますので、GUI ツールで ftp に対しての audit を無効にしておきます。

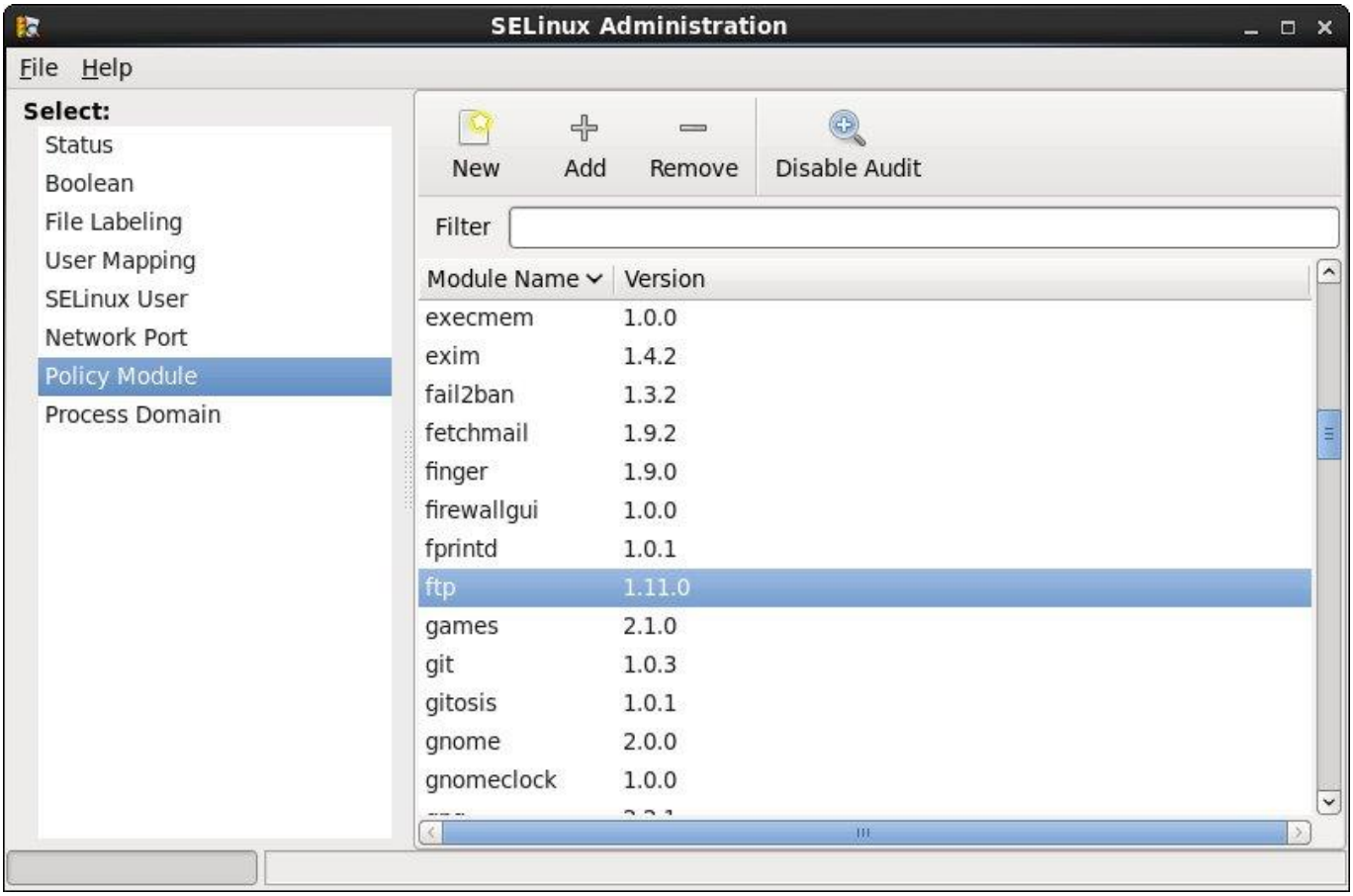


図 4.11　SELinux Management における監査設定

4.5 SELinux を用いた保護の実例

デフォルトで SELinux を有効にすると、脆弱性が比較的よく見つかる Apache などのソフトウェアには固有のドメインが付加された状態となります。これにより該当プロセスはサンドボックス化されるため、万が一これらのソフトウェアの脆弱性を利用した不正なアクセスに遭遇したとしても、システムの重要なリソースに対して深刻な影響を受ける可能性が小さくなります。

例として、バッファオーバーフローの場合を見てみましょう。バッファオーバーフローを防ぐ方法としては libsafe の導入がありますが、ここでは libsafe パッケージ内に含まれているバッファオーバーフローテストプログラムを使います。

SELinux は決められた「アクセス制御」を強制するためのものなので、バッファオーバーフローそのものを防ぐことはできません。しかし、以下のように万が一バッファオーバーフローを悪用された場合でも被害を最小限にすることができます。

1. wget で下記のように libsafe パッケージをダウンロードします。

```
$ wget http://pubs.research.avayalabs.com/src/libsafe-2.0-16.i386.rpm
```

2. ダウンロードした rpm パッケージを

```
rpm2cpio libsafe-2.0-16.i386.rpm | cpio -id
```

として適当なディレクトリ内に展開し、バッファオーバーフローを利用した exploit のテストプログラムを使用します。exploit 内の t1 というテストプログラムを/tmp 以下にコピーします。

実行例

```
[omok@cent64 exploits]$ pwd
/tmp/libsafe
[omok@cent64 libsafe]$ ls
libsafe-2.0-16.i386.rpm
[omok@cent64 libsafe]$ rpm2cpio libsafe-2.0-16.i386.rpm |cpio -id
1936 blocks
[omok@cent64 libsafe]$
```

```
[omok@cent64 libsafe]$ ls
lib  libsafe-2.0-16.i386.rpm  usr
[omok@cent64 libsafe]$ cd usr/doc/libsafe-2.0/exploits/
[omok@cent64 exploits]$ ls
Makefile          exploit-non-exec-stack    t1.c   t3     t4     t5
README            exploit-non-exec-stack.c  t10.c  t3.c   t4.c   t5.c
canary-exploit    int.sh                    t1w    t3w    t4w    t6
canary-exploit.c  t1                        t1w.c  t3w.c  t4w.c  t6.c
[omok@cent64 exploits]$ cp ./t1 /tmp
```

3. ローカルユーザで/tmp/t1 を実行します。SELinux が有効な状態でも、バッファオーバーフローは防げませんので、シェルが実行されてしまいます。これは/tmp/t1 を実行したシェル(bash)が、unconfined_t（無制限）ドメインで実行されているため、シェルの実行もできてしまうためです。

実行例

```
[omok@cent64 exploits]$ getenforce
Enforcing
[omok@cent64 exploits]$ id
uid=500(omok) gid=501(omok) 所属グループ=501(omok)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[omok@cent64 exploits]$ /tmp/t1
This program tries to use strcpy() to overflow the buffer.
If you get a /bin/sh prompt, then the exploit has worked.
Press any key to continue...
sh-4.1$ id
uid=500(omok) gid=501(omok) groups=501(omok)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

4. 今度は、この/tmp/t1 を Apache と同じ httpd_t ドメインで実行します。ドメインを変えてプログラムを実行するには、runcon というプログラムを使います。

```
$ runcon -u system_u -r system_r -t httpd_t /tmp/t1
```

runcon コマンドを使い httpd_t ドメインとして/tmp/t1 を実行した際には、SELinux により httpd_t ドメインが他のドメインに遷移する権限が与えられていないため、アクセスが拒否されます。

実行例

```
[omok@cent64 ~]$ runcon ^C
[omok@cent64 ~]$ ps axZ|grep http
system_u:system_r:httpd_t:s0     3875 ?        Ss     0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4119 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4120 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4121 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4122 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4123 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4124 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4125 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4126 ?        S      0:00 /usr/sbin/httpd
system_u:system_r:httpd_t:s0     4127 ?        S      0:00 /usr/sbin/httpd
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 4645 pts/0 S+   0:00 grep http
[omok@cent64 ~]$ runcon -u system_u -r system_r -t httpd_t /bin/ls
runcon: /bin/ls: 許可がありません
[omok@cent64 ~]$ runcon -u system_u -r system_r -t httpd_t /tmp/t1
runcon: /tmp/t1: 許可がありません
```

この場合に SELinux のログファイルは、次のようになります。

出力例

```
[root@cent64 ~]# more /tmp/audit.log
type=AVC msg=audit(1353280648.820:253): avc:  denied  { transition } for  pid=4
680 comm="runcon" path="/tmp/t1" dev=dm-0 ino=1586382 scontext=unconfined_u:unc
onfined_r:unconfined_t:s0-s0:c0.c1023 tcontext=system_u:system_r:httpd_t:s0-s0:
c0.c1023 tclass=process
type=SYSCALL msg=audit(1353280648.820:253): arch=c000003e syscall=59 success=no
 exit=-13 a0=7fffafd2a8ea a1=7fffafd295f0 a2=7fffafd29600 a3=1f items=0 ppid=46
57 pid=4680 auid=500 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tt
y=pts0 ses=5 comm="runcon" exe="/usr/bin/runcon" subj=unconfined_u:unconfined_r
:unconfined_t:s0-s0:c0.c1023 key=(null)
```

上記の例からもわかる通り、万が一 Apache に不具合があってバッファオーバーフローなどの脆弱性を利用されたとしても、SELinux を有効にしてアクセス制御をきっちり行なっていれば、被害を最小限に食い止めることができます。

5 章 ACL

5.1 ACL 概要

ACL は、Linux のカーネル 2.6 から標準採用された機能です。POSIX 準拠の ACL のため POSIX ACL とも言われます。ACL を用いると従来の Linux での権限よりも細やかにアクセス権限を設定することができます。

Linux 以外の OS、特に商用の Solaris や Windows などでは、ユーザごとのファイルやディレクトリに対してのアクセス権限を簡単に設定できます。また、付加できる権限の種類もより細やかなものになっています。そのため、Linux をエンタープライズ環境で使用する際のセキュリティを担保するために、ACL は必須のものと言えます。

5.2 ACL を試す

5.2.1 ACL を使用するための環境

ACL はファイルシステムの拡張属性を利用しているため、ACL を利用するためのツール類をインストールする他に、拡張属性がサポートされているファイルシステムを用いる必要があります。現在使用されている ext、ext3、XFS などほとんどのファイルシステムでは、拡張属性がサポートされています。

また、ファイルシステムをマウントする際にも、"acl"オプションを使用する必要があります。しかし、CentOS 6 を含む最近のディストリビューションではシステムをインストールするとマウントオプションはデフォルトで ACL をサポートするようになっていますので、まず問題になることはないでしょう。

万が一、使用しているシステムでファイルシステムが"acl"オプション付きでマウントされていない場合には、図のように/etc/fstab に"acl"オプションを付加する必要があります。

設定例

```
# /etc/fstab
# Created by anaconda on Mon Nov 19 02:30:58 2012
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=d5f1ebb8-9a01-4ff2-be44-f7f21c83fdf2 /     ext4    defaults        1 1
UUID=cfffd7e3-066a-4dcc-9367-65a1927cd1da /disk ext4    defaults,acl    1 2
<------------ acl オプションを付加
UUID=30569fd1-2be4-4dbc-88c9-fbdd41432f73 swap  swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

ACL は各ファイル／ディレクトリに対して設定します。ACL を変更できるのは、そのファイル／ディレクトリの所有者と root だけです。

5.2.2 ACL を使用するためのツール

ACL を使用するためのツールとしては、以下の 2 つのコマンドがあります。

1)　getfacl

各ファイル・ディレクトリに対して、ファイル名・所有者・グループ・設定されている ACL を表示します。

出力は図のようになります。

出力例

```
# file: sample.txt
# owner: user1
# group: user1
user::rw-
user:user1:rw-
group::r--
```

```
other::r--
```

getfacl の出力の詳しい説明に関しては、man getfacl で確認できます。

2) setfacl

各ファイル・ディレクトリに対して、ACL を設定します。オプションは、図の通りです。

```
setfacl 2.2.51 -- set file access control lists
Usage: setfacl [-bkndRLP] { -m|-M|-x|-X ... } file ...
  -m, --modify=acl        modify the current ACL(s) of file(s)
  -M, --modify-file=file  read ACL entries to modify from file
  -x, --remove=acl        remove entries from the ACL(s) of file(s)
  -X, --remove-file=file  read ACL entries to remove from file
  -b, --remove-all        remove all extended ACL entries
  -k, --remove-default    remove the default ACL
      --set=acl           set the ACL of file(s), replacing the current ACL
      --set-file=file     read ACL entries to set from file
      --mask              do recalculate the effective rights mask
  -n, --no-mask           don't recalculate the effective rights mask
  -d, --default           operations apply to the default ACL
  -R, --recursive         recurse into subdirectories
  -L, --logical           logical walk, follow symbolic links
  -P, --physical          physical walk, do not follow symbolic links
      --restore=file      restore ACLs (inverse of `getfacl -R')
      --test              test mode (ACLs are not modified)
  -v, --version           print version and exit
  -h, --help              this help text
```

setfacl の詳しい使用方法に関しては、man setfacl で確認できます。

5.2.3 ACL のテスト

それでは、ACL を実際にテストしてみましょう。以下、CentOS6 を使用して ACL のテストをしていきます。その他のディストリビューションでも、新しいバージョンでは ACL をサポートしているため、同様のテストができます。

1) /disk 以下に、testfile_01 という空のファイルを作成します。このファイルのパーミッションは、rw-r--r--.となっており、root 以外のユーザには読み込みのみが許可されている状態です。パーミッションの最後の"."は、ACL が設定されていないことを意味しています。

実行例

```
[root@localhost disk]# ls -l
合計 16
drwx------. 2 root root 16384 11 月 19 02:28 2012 lost+found
-rw-r--r--. 1 root root     0 11 月 27 15:49 2012 testfile_1
```

2) 次に"setfacl -m"オプションを用いて、このファイルに対して、ユーザ test01 が read/write できるように、ACL の権限を与えます。ls -l で確認すると、rw-rw-r--+となっていることがわかります。パーミッションの最後が"+"に変わっています。これは、このファイルに対して ACL が設定されていることを意味しています。

実行例

```
[root@localhost disk]# setfacl -m u:test01:rw testfile_1
[root@localhost disk]# ls -l
合計 20
drwx------. 2 root root 16384 11 月 19 02:28 2012 lost+found
-rw-rw-r--+ 1 root root     0 11 月 27 15:49 2012 testfile_1
```

3) "getfacl"コマンドで、このファイルに対しての ACL を確認します。test01 に対して rw の権限が与えられているのがわかります。

実行例

```
[root@localhost disk]# getfacl testfile_1
# file: testfile_1
# owner: root
# group: root
user::rw-
user:test01:rw-
group::r--
mask::rw-
other::r--
```

4) test01 ユーザで、このファイルに対しての権限を確認します。書き込みと読み込みが正常に行えることがわかります。

実行例

```
[root@localhost disk]# su - test01
[test01@localhost ~]$ echo "This is a test from test01.">> /disk/testfile_1
[test01@localhost ~]$ cat /disk/testfile_1
This is a test from test01.
```

5) test02 ユーザで、このファイルに対しての権限を確認します。読み取りはできますが、書き込みが「許可がありません」とエラーになります。

実行例

```
[test01@localhost ~]$ su - test02
パスワード:
[test02@localhost ~]$ cat /disk/testfile_1
This is a test from test01.
[test02@localhost ~]$ echo "This is a test from test02." >> /disk/testfile_1
-bash: /disk/testfile_1: 許可がありません
```

このように、ファイルに対する各ユーザの権限を ACL で指定できるようになっていることがわかります。

また、ファイルの ACL を削除するにも同じく setfacl コマンドを使用します。

6) 再び root アカウントになって、"setfacl -x"オプションで、/disk/testfile_1 に対して test01 が所有している ACL の権限を消去します。

実行例

```
[root@localhost ~]# getfacl /disk/testfile_1
getfacl: Removing leading '/' from absolute path names
# file: disk/testfile_1
# owner: root
# group: root
user::rw-
user:test01:rw-
group::r--
mask::rw-
other::r--

[root@localhost ~]# setfacl -x u:test01 /disk/testfile_1
[root@localhost ~]# getfacl /disk/testfile_1
getfacl: Removing leading '/' from absolute path names
# file: disk/testfile_1
# owner: root
# group: root
user::rw-
group::r--
mask::r--
other::r--
```

7) test01 は、/disk/testfile_1 に対して、読み込みのみが許可され、書き込みはできなくなっていることがわかります。

実行例

```
[root@localhost ~]# su - test01
[test01@localhost ~]$ cat /disk/testfile_1
This is a test by test01.
[test01@localhost ~]$ echo "This is a test 2 by test01." >> /disk/testfile_1
-bash: /disk/testfile_1: 許可がありません
```

5.3 ACL の応用

5.3.1 ACL の応用(Web)

ACL を利用すると、サーバのセキュリティを簡単に高めることができます。具体的な例として、Web サーバの場合を見てみましょう。

Web サーバ(Apache)で、各ユーザの Web ディレクトリを公開している状態を想定します。各ユーザのホームディレクトリに public_html ディレクトリを作成して、httpd.conf で UseDir ディレクティブを使って公開をしている状態です。

1) テストとして、test01, test02, test03 ユーザでそれぞれ~[username]/public_html 以下を公開する状態に設定します。

設定例

```
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    #UserDir disabled
    UserDir enabled test01 test02 test03

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disabled" line above, and uncomment
    # the following line instead:
    #
    #UserDir public_html
    UserDir public_html

</IfModule>

<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

（細かい設定方法は、http://httpd.apache.org/docs/2.2/ja/howto/public_html.html#userdir を参考にしてください。）

なお、SELinux を有効にしている状態で、各ユーザの/home 以下のディレクトリを apache を用いて公開するには、SELinux の Boolean の「httpd_enable_homedirs」を下記のように ON にする必要があります。デフォルトでは OFF になっています。

実行例

```
[root@localhost ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
[root@localhost ~]# setsebool -P httpd_enable_homedirs on
[root@localhost ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

（setsebool で"-P"オプションを指定すると、システムの再起動後も boolean「httpd_enable_homedirs」は常に on になります。）

2）この設定で http://[システムのアドレス]/~test01/public_html/index.html を表示させるには、/home/test01 のパーミッションをデフォルトの 700 から 711 に変更する必要があります。

実行例

```
[root@localhost conf]# ls -l /home
合計 20
drwx--x--x.  5 test01 test01 4096 11 月 29 11:02 2012 test01
drwx------.  5 test02 test02 4096 11 月 29 11:01 2012 test02
drwx------.  4 test03 test03 4096 11 月 27 15:48 2012 test03
```

これで、http://[システムのアドレス]/~test01/public_html/index.html が表示されます。

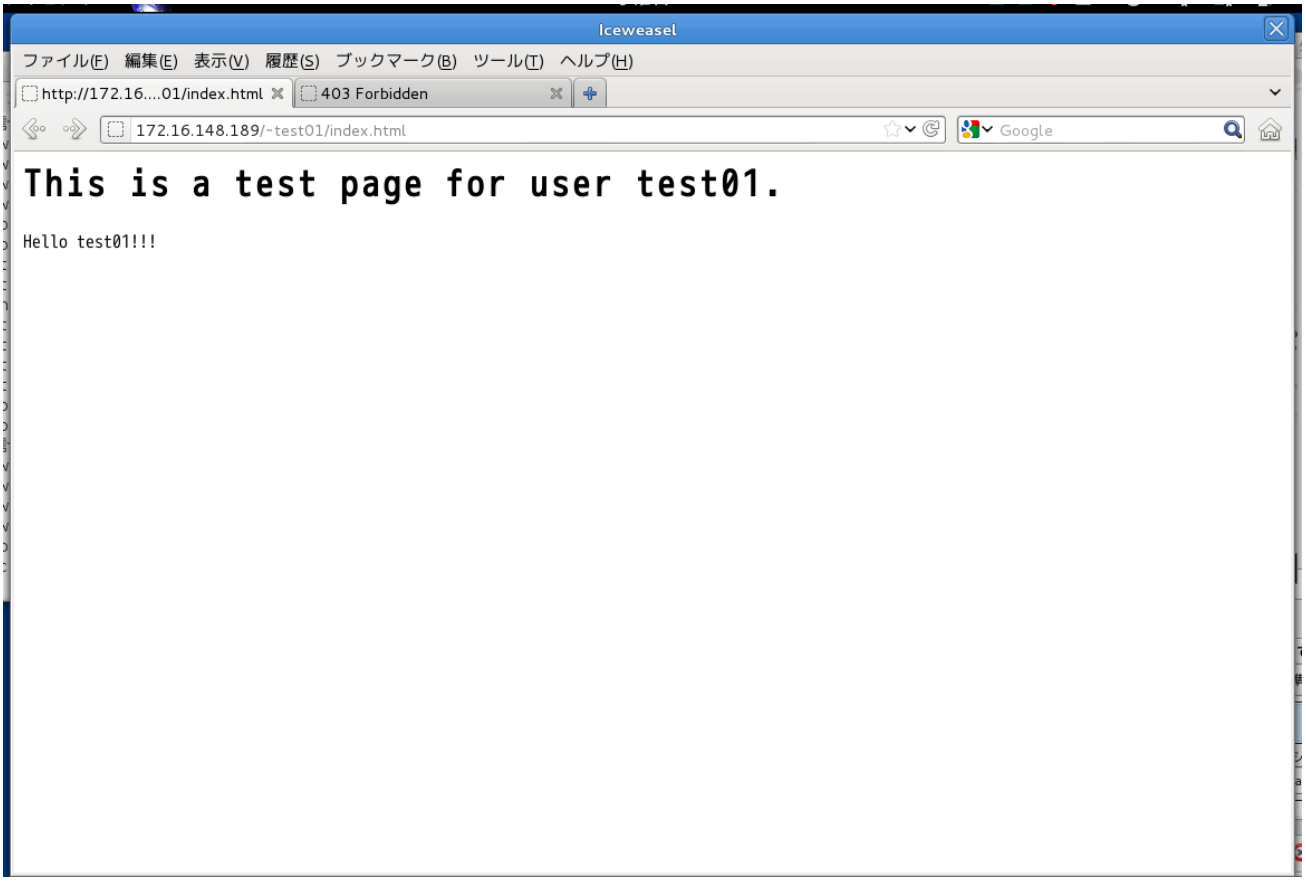


図 5.1　index.html の表示画面例

一方、700 のままの、http://[システムのアドレス]/~test02/public_html/index.html は表示されません。

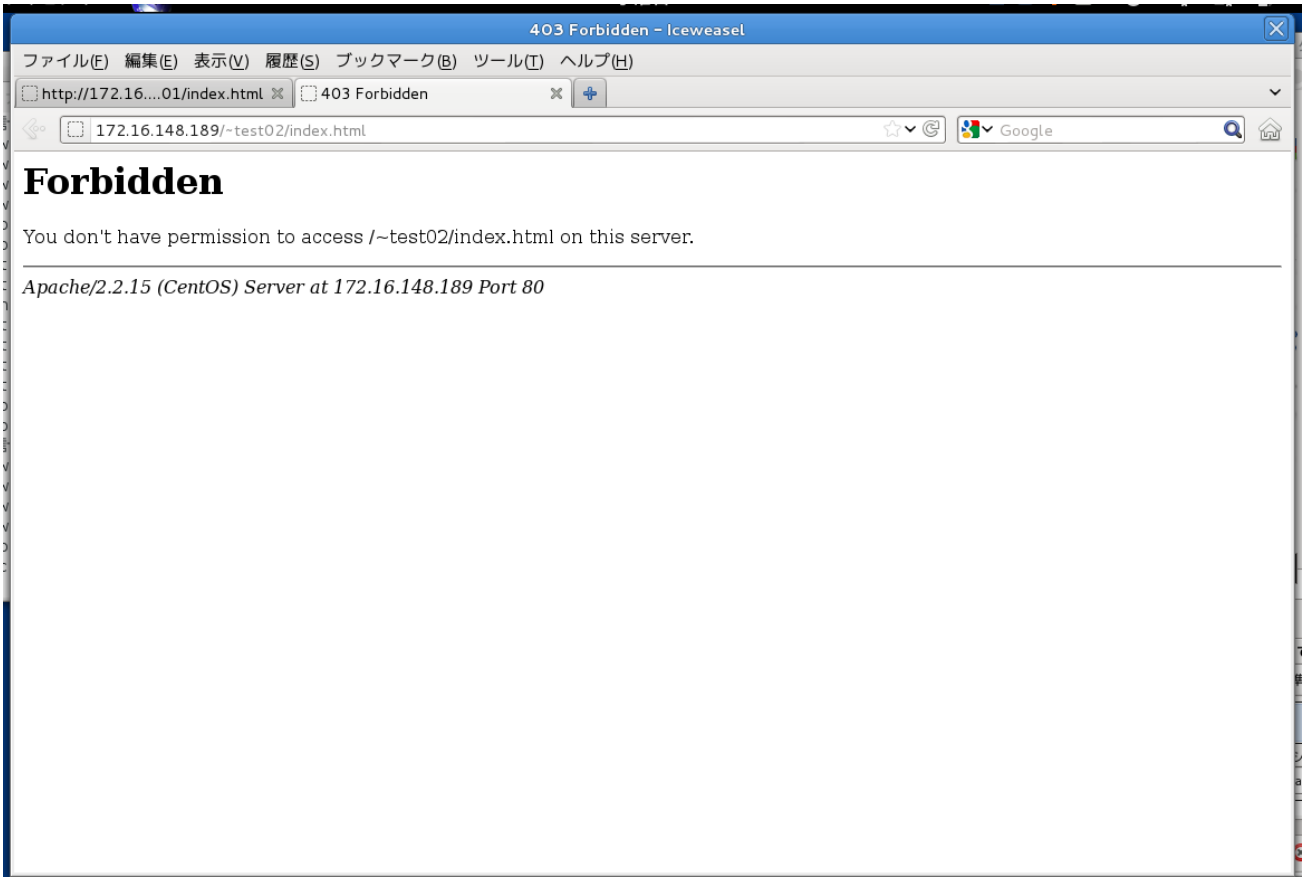


図 5.2　index.html にアクセス権がない場合の表示画面

3) この方法では、全ユーザに対して/home/test01 に x で実行権限を与えているため、セキュリティリスクが存在します。例えば、test03 ユーザでログインすると、/home/test01 以下の情報は ls で search することは出来ませんが、/home/test01/public_html の存在を test03 が知っていた場合には、/tmp 以下に/home/test01/public_html をコピーすることができます。

4) また、/home/test01 以下のファイルは、パーミッションを正しく設定していないと、その存在を知っている他のユーザ(ここでは test03)に見られてしまう可能性があります。

- パーミッションの設定が緩いと

```
 [test01@localhost ~]$ ls -l
合計 8
-rw-rw-r--. 1 test01 test01    5 11 月 29 11:39 2012 bank_password
```

他のユーザ(test03)に中身を見られてしまう!!

```
 [test03@localhost test01]$ cat /home/test01/bank_account
This is a secret file for test01!!

1234-5678-9876
[test03@localhost test01]$
```

これらを避けるには、各ユーザにパーミッションを正しく設定させるための教育が必要ですが、教育コストがかかります。

5) これを回避するために、ディレクトリ/home/[username]のパーミッションは 700 のままで、ACL を用いて httpd プロセスの UID(apache)に、/home/[username]の実行権限を与えるようにします。例として、test02 アカウントの/home/test02 に ACL を設定します。

実行例

```
[root@localhost home]# setfacl -m u:apache:x /home/test02
[root@localhost home]# ls -l /home
合計 20
drwx--x--x.  5 test01 test01 4096 11 月 29 11:40 2012 test01
drwx--x---+  5 test02 test02 4096 11 月 29 11:01 2012 test02
drwx------.  4 test03 test03 4096 11 月 27 15:48 2012 test03
[root@localhost home]# getfacl /home/test02
getfacl: Removing leading '/' from absolute path names
# file: home/test02
# owner: test02
# group: test02
user::rwx
user:apache:--x
group::---
mask::--x
other::---
```

6) ACL で apache に/home/test02 に対しての実行権限を与えたため、http://[システムのアドレス]/~test02/public.html/index.html が表示されるようになります。

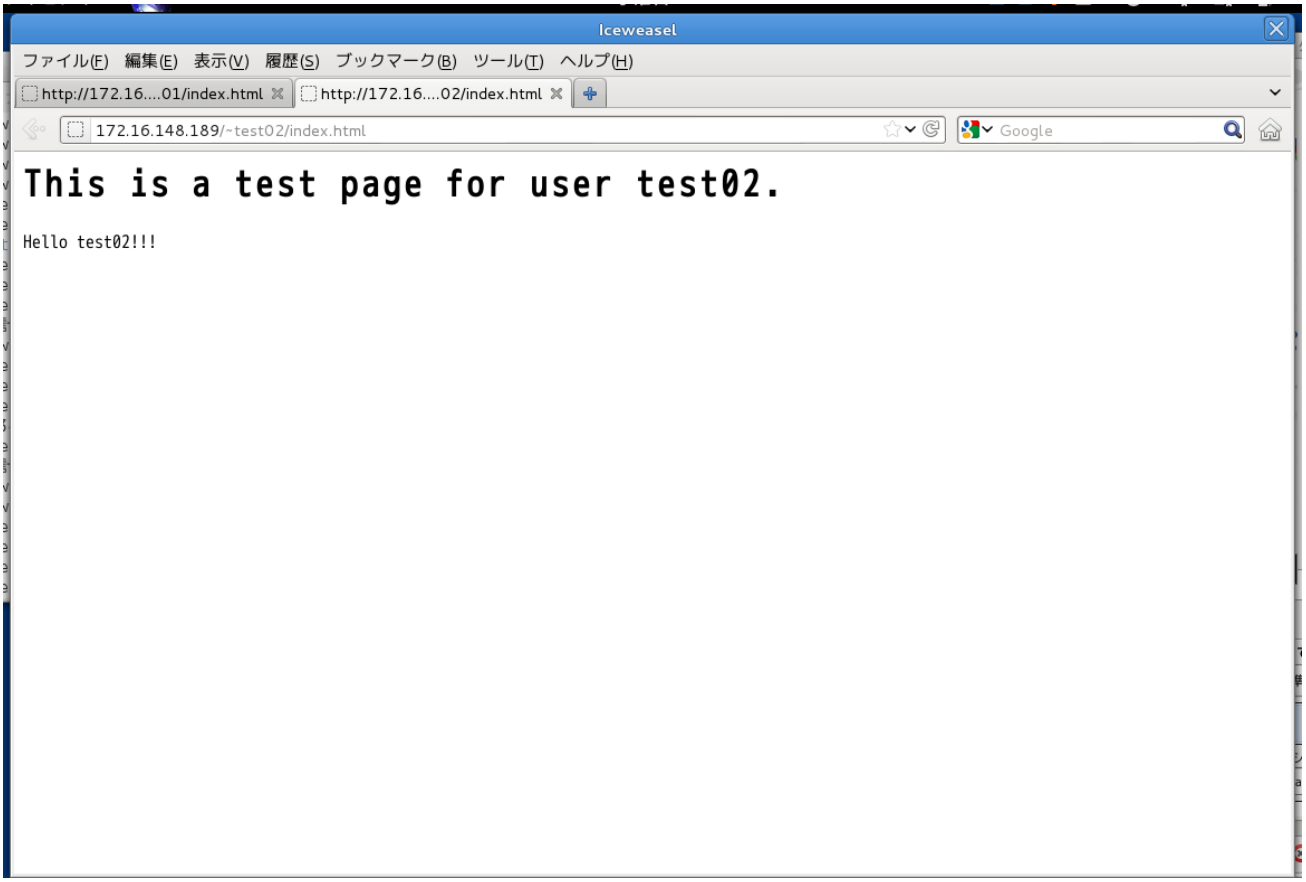


図 5.3　index.html にアクセス可能になった場合における表示画面例

7)　test01 に対してと同様に、test03 アカウントで test02 の public_html をコピーしようとしますが、「許可がありません」が出てエラーになります。また、/home/test02 以下のファイルを覗こうとしても、同様に「許可がありません」となり、失敗します。

実行例

```
[test03@localhost ~]$ cp -pr /home/test02/public_html /tmp
cp: cannot stat `/home/test02/public_html': 許可がありません
[test03@localhost ~]$ cat /home/test02/bank_account
cat: /home/test02/bank_account: 許可がありません
[test03@localhost ~]$ su -
パスワード:
[root@localhost ~]# ls -l /home/test02
合計 8
-rwxrwxr-x. 1 test02 test02   51 11月 29 11:55 2012 bank_account
drwxrwxr-x. 2 test02 test02 4096 11月 29 11:01 2012 public_html
[root@localhost ~]# cat /home/test02/bank_account
This is a secret file for test02!!

9876-5432-1234
[root@localhost ~]#
```

このように、ACL を使用すると各ユーザに適切なパーミッション設定を強制するコストを抑えて、サーバのセキュリティを高めることができます。

5.3.2 samba と ACL の関係

ACL でファイル／ディレクトリに対して追加した権限は、samba を使って共有している場合には、Windows 側から確認することができます。

前の項で、/home/test02 ディレクトリを apache が利用できるように ACL で実行権（x）を与えました。この/home/test02 の情報を Windows 側から見てみましょう。

1)　samba で/home 以下を共有しています。/home/test02 を Windows 側で確認し、セキュリティを調べると、図のようになります。/home/test02 に対して、apache アカウントが「特殊な権限」を持っていることがわかります。

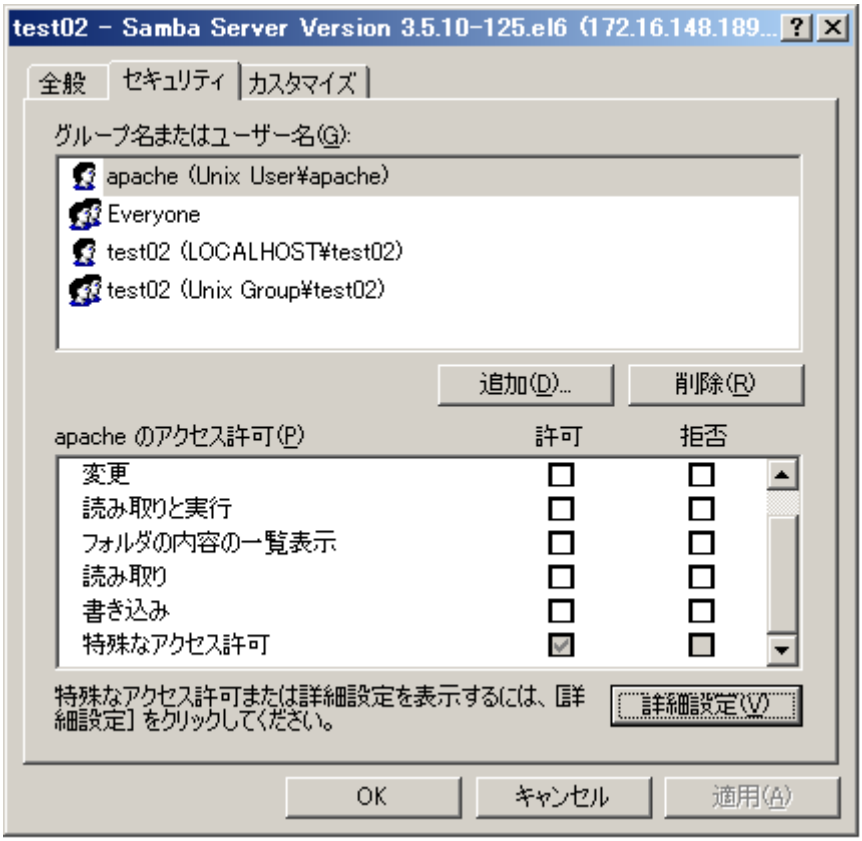


図 5.4　Windows における権限の表示

2)　特殊な権限を確認すると、apache アカウントが「スキャン／実行」の権限を持っていることがわかります。

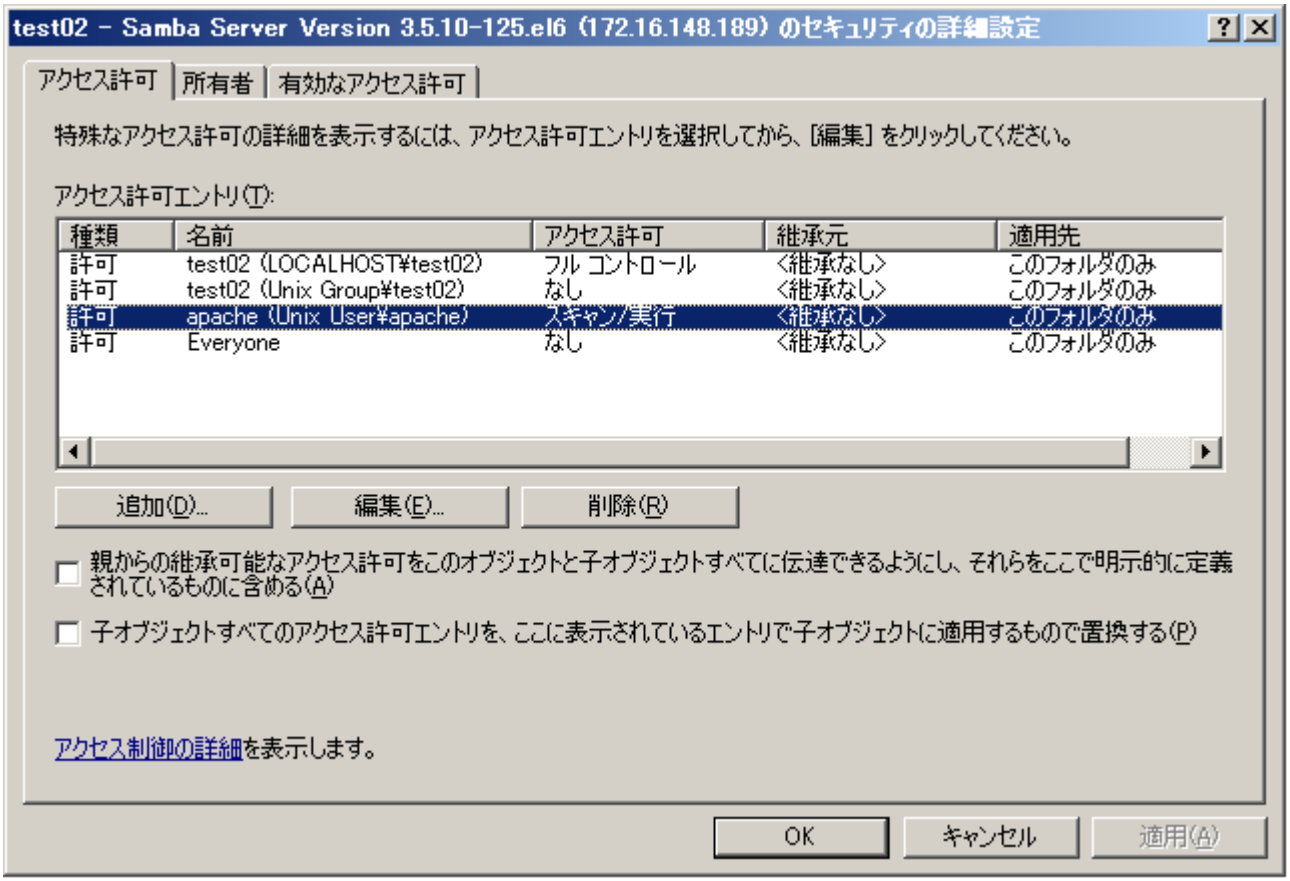


図 5.5　Windows におけるアクセス権一覧画面

3)　次に、このアクセス権を Windows 側で編集してみます。テストとして、「フォルダの一覧、データの読み取り」を付加します。

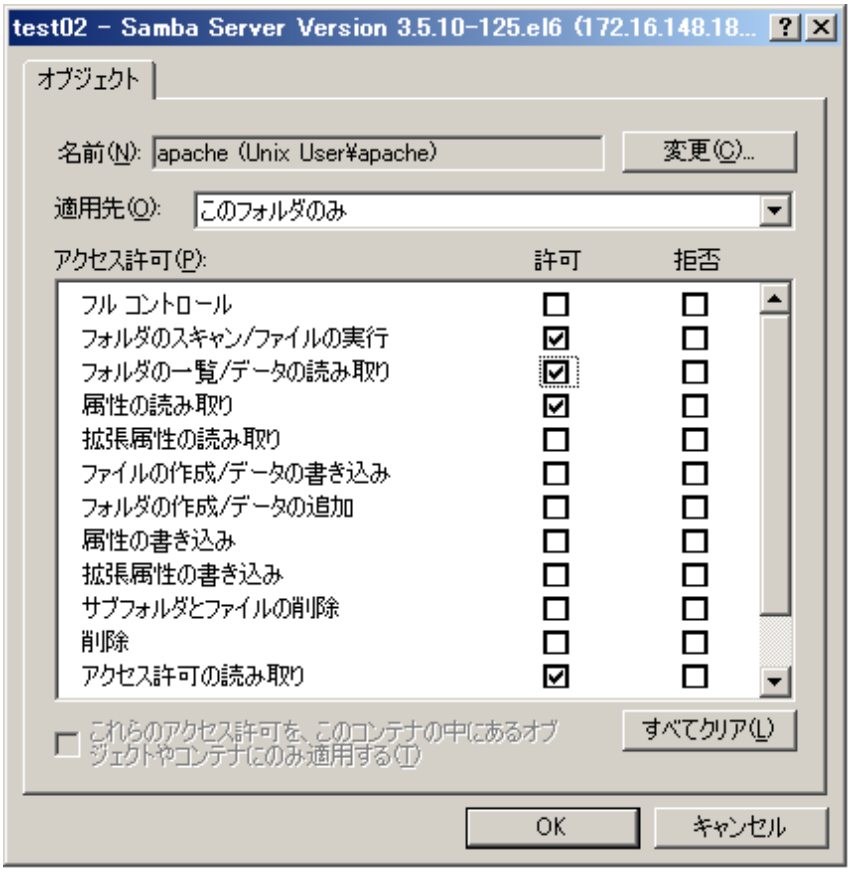


図 5.6　Windows における権限の編集画面

4)　Linux 側で getfacl を用いて ACL を確認すると、apache ユーザが x だけではなく、r-x の権限を持つようになったことがわかります。

実行例

```
[root@localhost ~]# getfacl /home/test02
getfacl: Removing leading '/' from absolute path names
# file: home/test02
# owner: test02
# group: test02
user::rwx
user:apache:r-x
group::---
mask::rwx
other::---
```

このように、Windows 側から ACL を確認するだけでなく、ACL を修正することも可能です。

5.3.3 最後に

ACL を利用することにより、今までの Linux で面倒だったユーザごとのアクセス制限が簡単にできるようになります。特に企業用途でサーバを運用していく場合には、セキュリティの完全さだけでなく、運用の簡便さも求められます。そのような場面で、ACL は非常に役立つと考えられます。

6. OpenSSH によるサーバアクセスと、サーバ管理

SSH は、リモートのクライアントからサーバへアクセスし、シェルを使って作業を行うしくみです。

SSH を使えば、会社からインターネットを経由してデータセンタ内のサーバに接続し、シェルを使ってサーバの状況確認やメンテナンス作業などを行うことができます。

SSH の重要な特徴は、全ての通信が暗号化されることです。これはインターネットを経由した SSH 通信が第三者に盗聴された場合でも、複合するためのパスワードを知らない限り、盗聴した第三者には通信内容がわからないことを意味します。

SSH が登場する以前から普及していた telnet ではパスワードは平文で送ってしまいますが、SSH はパスワードも暗号化しますので、大きなセキュリティ上のメリットがあります。

本章では、SSH を使ったサーバへの接続方法と、より安全な使い方について具体的に説明します。

6.1 OpenSSH のインストールと初期設定

Linux では、SSH 接続を行うためのソフトウェアとして OpenSSH が広く利用されています。

サーバに OpenSSH がインストールされていない場合は、yum コマンドを使って OpenSSH をインストールしてください。OpenSSH のパッケージは、以下の 3 つです。

- openssh
- openssh-clients
- openssh-server

インストールが終わったら、管理者権限で OpenSSH のデーモンを起動します。

実行例

```
[root@localhost]# /etc/init.d/sshd start
sshd を起動中:                                             [  OK  ]
```

SSH のデフォルトポートは 22 番なので、netstat を使って 22 番ポートが開いていることを確認します。また、iptables でも、22 番ポートへの接続を許可しているかを確認します。

実行例

```
[root@localhost]# netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
[root@localhost]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere        anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere        anywhere            state NEW tcp dpt:ssh

=== 以下省略 ===
```

6.2 サーバへの接続（Linux）

SSH サーバに接続するには ssh コマンドを使います。SSH クライアントに必要なパッケージは openssh と openssh-client です。インストールされていない場合は事前に yum コマンドなどでインストールします。

まずテストとして localhost に接続します。

```
[linuc@localhost ~]$ ssh localhost
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is b8:eb:8d:e4:05:77:0f:12:b6:37:b3:9d:9c:79:fe:77.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
linuc@localhost's password:                                  ←パスワードを入力
Last login: Wed May 29 16:28:36 2013 from 192.168.100.31
[linuc@localhost ~]$
```

初めて接続するホストの場合、.ssh/known_hosts ファイルに登録されていないため接続するかどうかを問い合わせるメッセージが表示されますが、接続先に間違いがなければ “yes” を入力します。

```
The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is b8:eb:8d:e4:05:77:0f:12:b6:37:b3:9d:9c:79:fe:77.
Are you sure you want to continue connecting (yes/no)?
```

一度 .ssh/known_hosts に登録されたら次回からこのメッセージは表示されません。known_hosts ファイルにはデフォルトで相手サーバ側の /etc/ssh/ssh_host_rsa_key.pub の内容が追記されます。

ssh コマンドを使ってサーバに接続すると、ホームディレクトリ配下の .ssh ディレクトリと .ssh/known_hosts ファイルは自動的に作成されます。

サーバ、クライアント共に Linux の場合はクライアント側の Linux から宛先サーバのホスト名または IP アドレスを指定して ssh コマンドを実行します。

ここではサーバ側ホスト名を server、クライアント側ホスト名を client としています。検証する際には/etc/hosts などに登録しておくと良いでしょう。

```
[linuc@client ~]$ ssh server
linuc@server's password:                                    ←パスワードを入力
Last login: Wed May 29 16:28:36 2013 from client
[linuc@server ~]$
```

別のユーザアカウントでログインしたい場合、次の例のように宛先サーバのホスト名または IP アドレスの前に@をつけてユーザを指定することが可能です。

```
[linuc@client ~]$ ssh cipl@server
```

ssh コマンドのオプションや使い方の詳しい説明は man ssh などで確認できます。

6.3 サーバへの接続（Windows＋Tera Term）

日常的に Microsoft Windows の PC を利用している場合、Windows 環境から直接 Linux サーバにリモート接続できると便利です。

Microsoft Windows の PC から、Linux サーバへ SSH を使って接続するには、Windows から SSH 接続可能なクライアントソフトを使う必要があります。Windows 用の SSH 接続可能なソフトウェアは多数ありますが、その中でも有名なのが Tera Term というソフトウェアです。

Tera Term は、寺西高氏が開発したフリーソフトウェアでしたが、現在はオープンソース化され、有志によるコミュニティ TeraTerm Project によって開発が行われています。最新版は、TeraTerm Project のホームページからダウンロードすることができます。

この節では Windows7 に TeraTerm をインストールし、Linux サーバに接続する手順を説明します。

TeraTerm Project (http://sourceforge.jp/projects/ttssh2/)から Tera Term のプログラム（exe 形式）をダウンロードし、exe ファイルを実行します。exe ファイルの実行時に、環境の設定やセキュリティ対策ソフトにより、セキュリティ警告が出る場合があります。警告の内容を確認して処理を選択してください。

exe ファイルを実行すると「セットアップに使用する言語の選択」が表示されますので、日本語で問題なければ日本語を選択し、OK ボタンを押します。

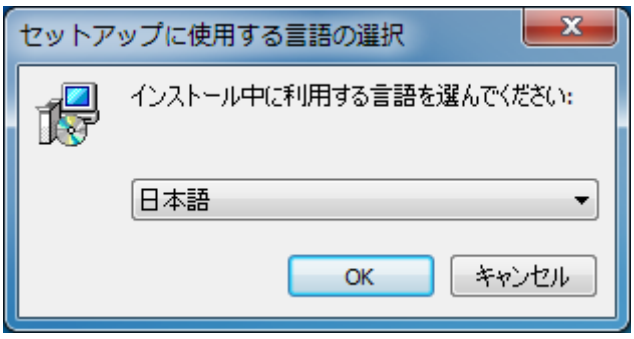


図 6.1 TeraTerm のインストール（言語設定）

言語を選択すると、セットアップウィザードの開始画面が表示されるので「次へ」を押します。

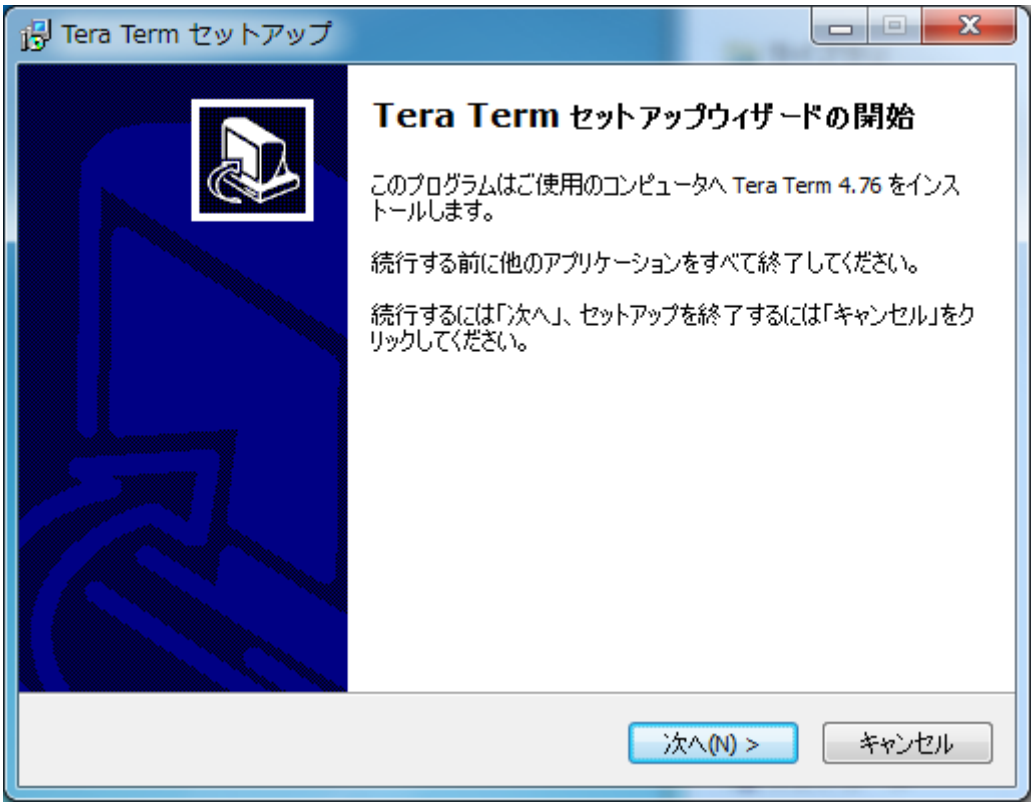


図 6.2 TeraTerm のインストール（セットアップウィザードの開始）

次に、使用承諾契約書の同意が求められますので、内容に問題がなければ「同意する」をチェックして、「次へ」を押します。

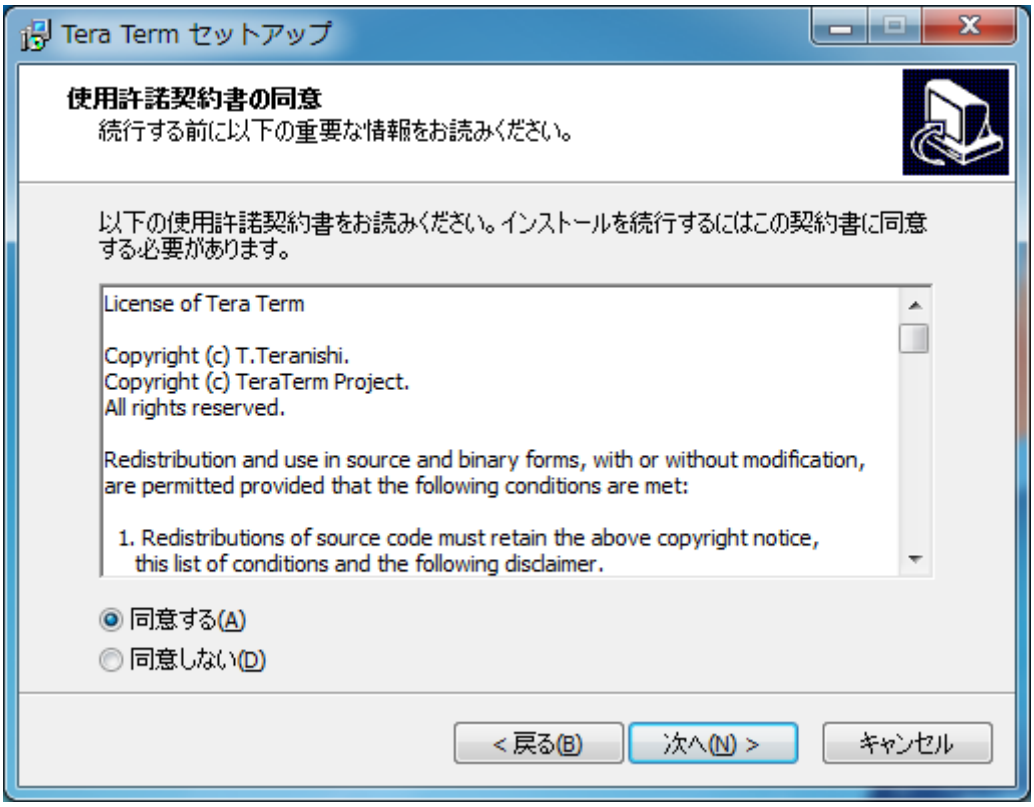


図 6.3 TeraTerm のインストール（使用承諾書の同意）

次に、インストール先の指定が表示されますので、インストール先フォルダを指定して「次へ」を押します。

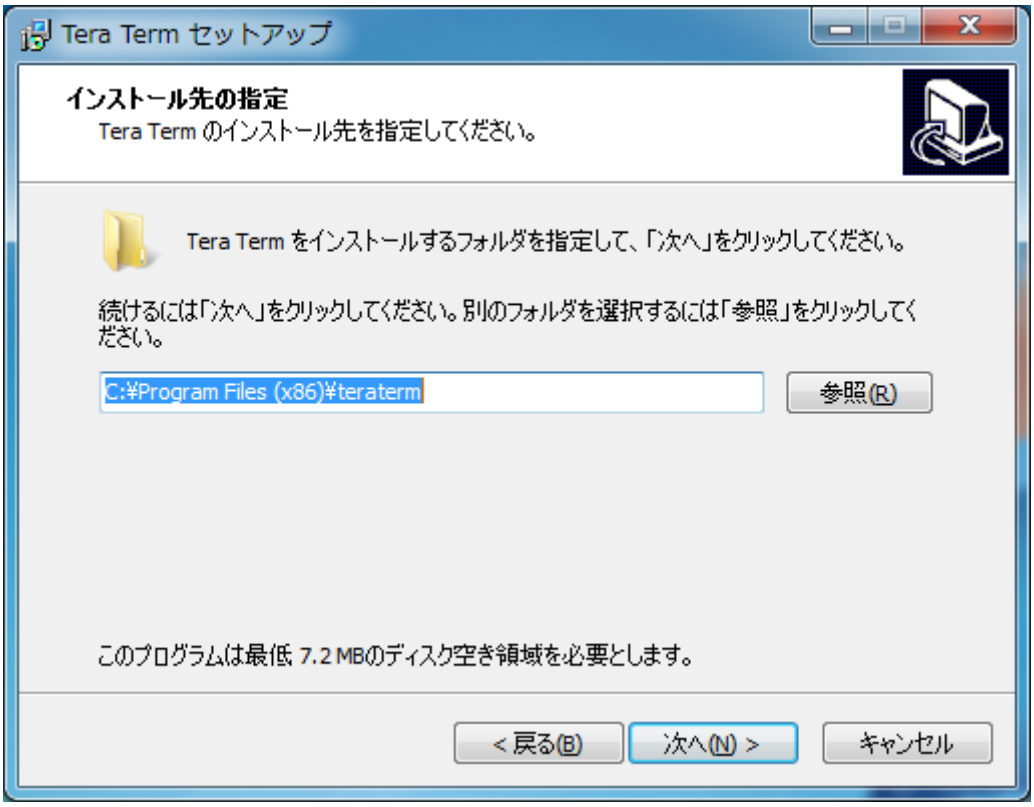


図 6.4 TeraTerm のインストール（インストール先の指定）

次に、コンポーネントの選択画面が表示されるのでインストールするものを選んでください。SSH 接続を行うためには、「Tera Term & Macro」と「TTSSH」は必ず選択します。

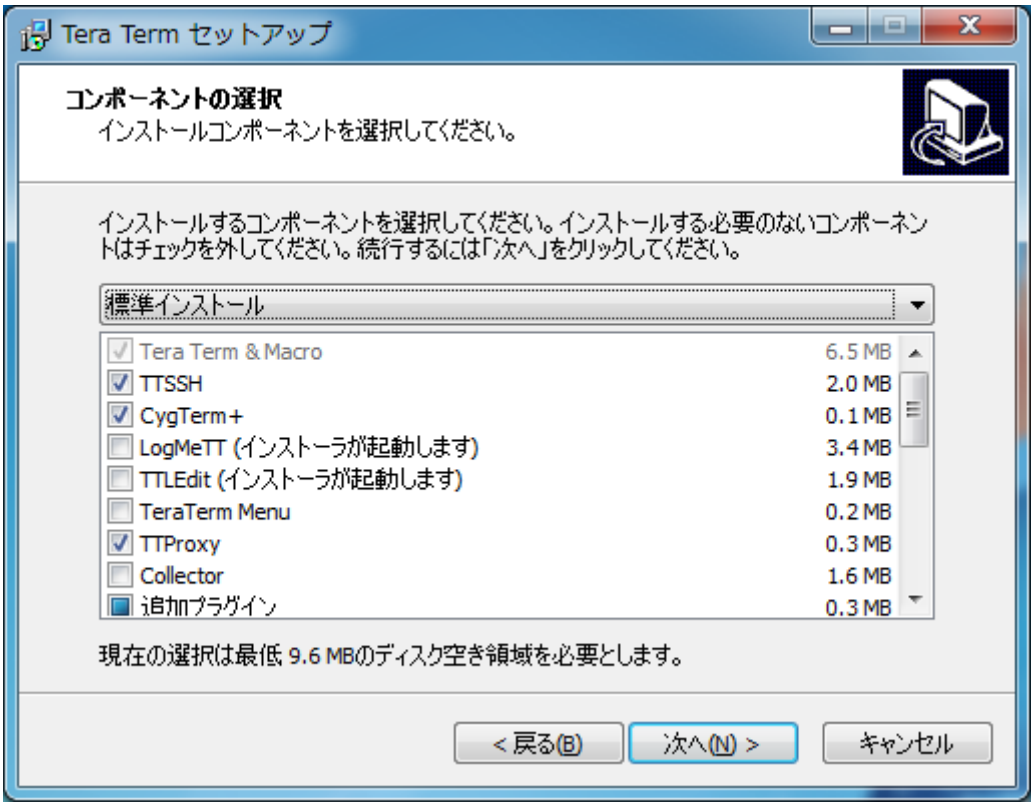


図 6.5 TeraTerm のインストール（コンポーネントの選択）

次にユーザインターフェイスの言語選択が表示されますので、日本語で問題なければ日本語を選択して「次へ」を押します。

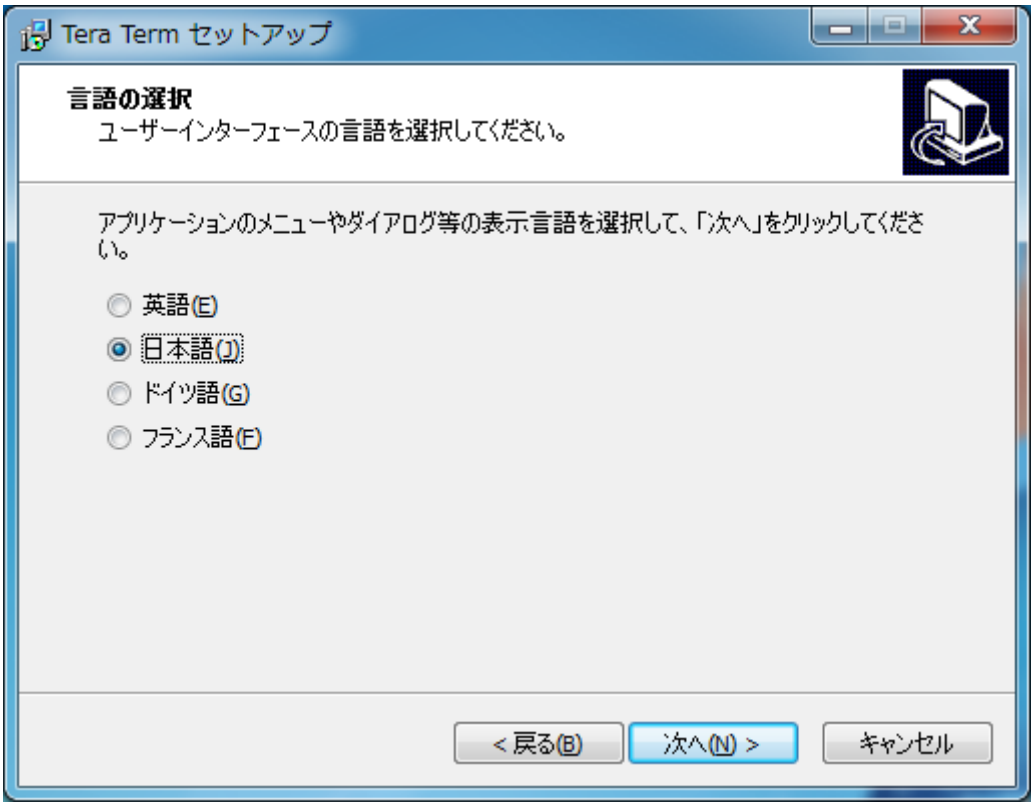


図 6.6 TeraTerm のインストール（言語の選択）

次にプログラムグループの指定が表示されますので、問題なければ「次へ」を押します。

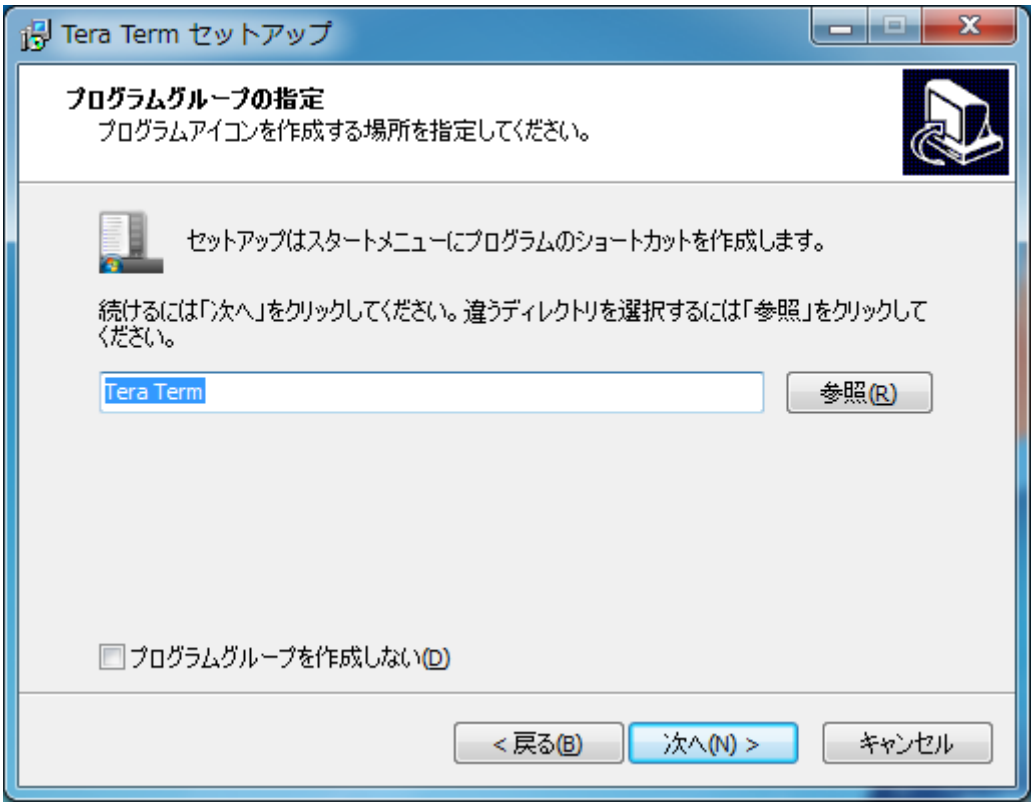


図 6.7 TeraTerm のインストール（プログラムグループの指定）

最後に追加タスクの選択が表示されますので、内容に問題がなければ「次へ」を押して、次のインストール準備完了画面で、「インストール」ボタンを押すことで、インストール作業が開始されます。

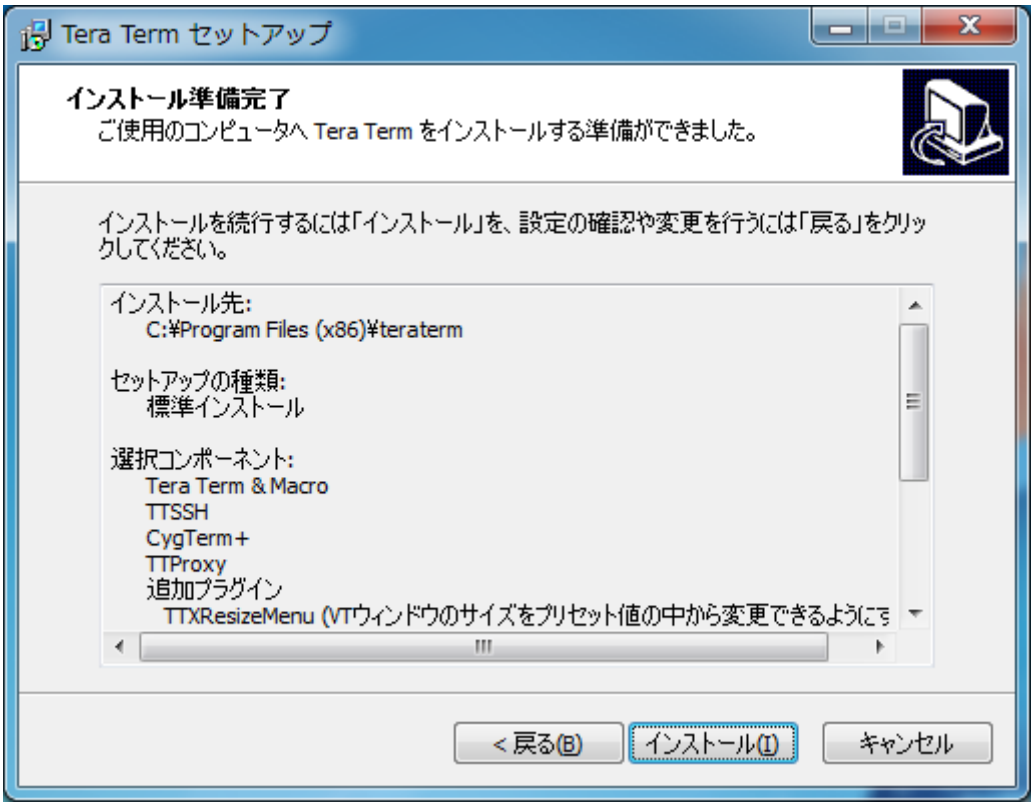
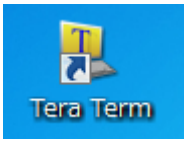


図 6.8 TeraTerm のインストール（インストール準備完了）

インストールが完了したら、Tera Term を起動します。



Tera Term が起動すると、「新しい接続」に関する入力画面が表示されますので、各項目を以下のように入力した上で「OK」ボタンを押します。

ホスト	接続サーバの IP アドレス
ヒストリ	チェックをして表示内容をログに残します。
サービス	SSH
TCP ポート	22（SSH のデフォルトポートは 22 です）
SSH バージョン	SSH2

次に、known hosts に関するセキュリティ警告が表示されます。

これは、初めて接続するサーバの場合 known hosts リストに登録されていないので、接続先に間違いがないか確認を促す警告です。接続先に間違いがなければ「このホストを known hosts リストに追加する」をチェックして「続行」を押します。次回からこの画面は表示されません。

次に SSH 認証に関する情報を入力します。

今回は、パスワード認証による接続を行いますので、「プレインテキストを使う」を選択します。

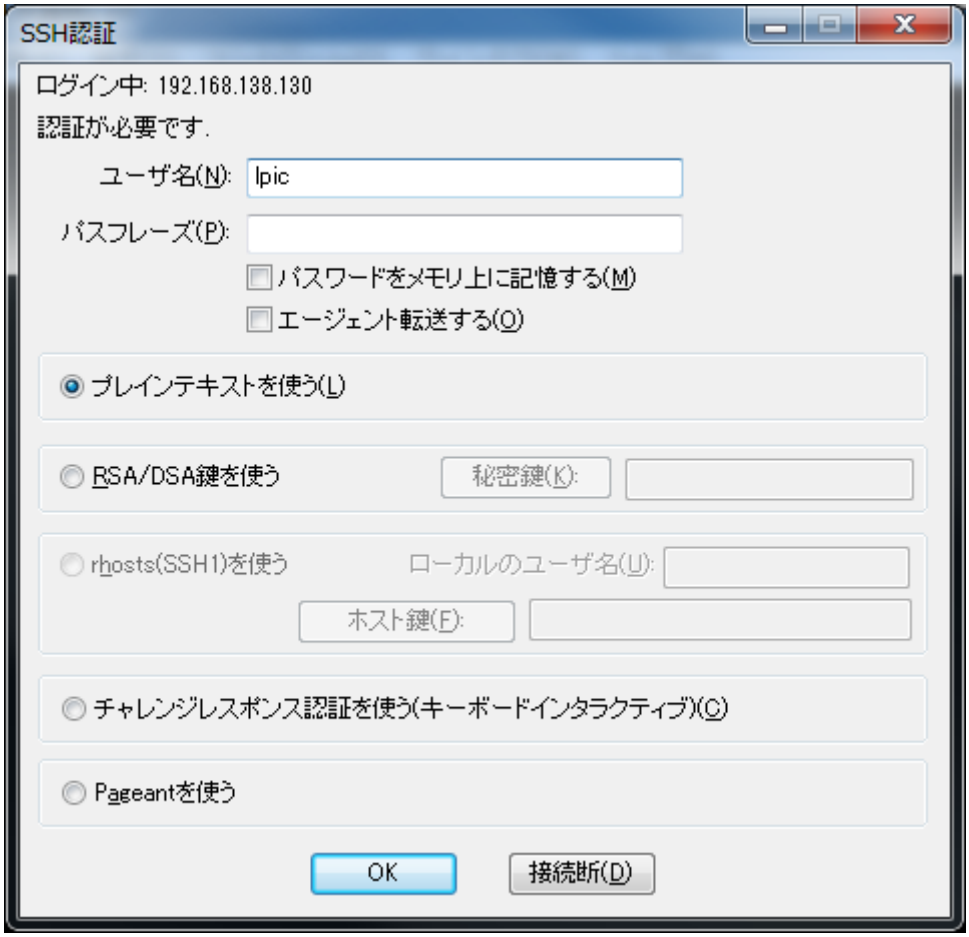


図 6.9 TeraTerm の接続画面

ユーザ名とパスフレーズ（パスワード）を入力して OK を押すと、SSH サーバへの接続が行われ、シェルプロンプトが表示されコマンドを実行できるようになります。

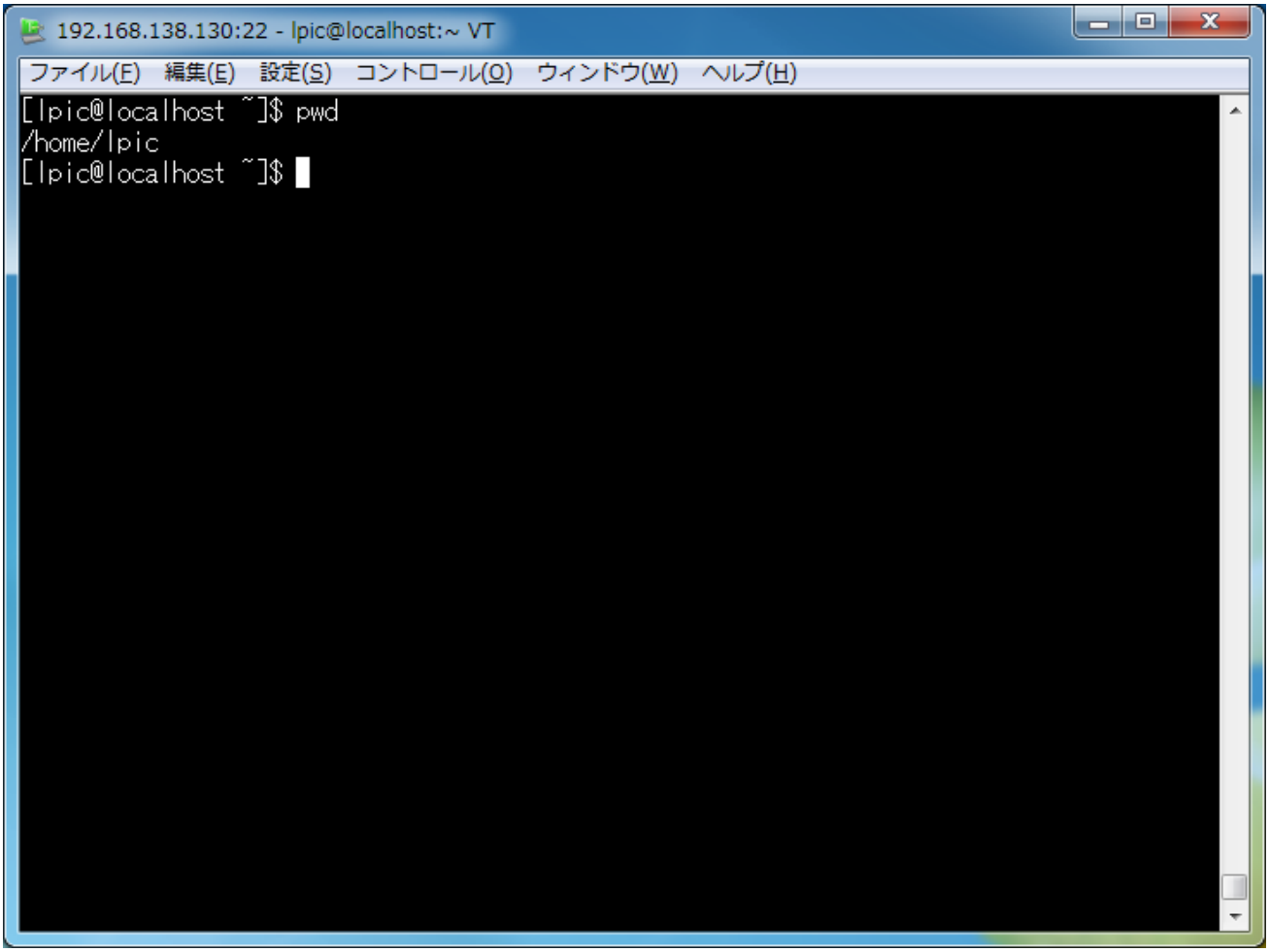


図 6.10 TeraTerm の接続完了後の画面

6.4　OpenSSH における接続制限

SSH を使えるということは、認証情報さえあれば、サーバ上でコマンドを打つなどの操作がネットワーク経由で可能になることを意味します。誰でも SSH が自由に使える状況というのはセキュリティ上好ましくありませんので、SSH を使って外部から管理作業を行うことのできるユーザや接続元 IP アドレスを必要最低限に制限すると良いでしょう。

接続制限を行うために iptables を使うこともできますが、OpenSSH の設定でも接続可能なユーザを限定することができます。ここでは OpenSSH の設定例を紹介します。

OpenSSH では/etc/ssh/sshd_config を編集することで利用可能なユーザを制限できます。

/etc/ssh/sshd_config を以下のように編集します。

設定例

```
# UsePAM no
UsePAM yes

AllowUsers linuc
```

UsePAM が no になっている場合は yes に変更し、AllowUsers に許可したいユーザ名のみを記述します。これでユーザ linuc 以外は SSH による接続ができなくなります。

ユーザ単位で許可するのではなく、ユーザと接続元 IP アドレスなどを組み合わせて許可したい場合などは、pam_access を利用する必要があります。

pam_access による接続制限を行うためには、次の 3 つのファイルを編集します。

(1) /etc/security/access.conf に制限内容を記述する。
(2) /etc/pam.d/sshd を編集する。
(3) /etc/ssh/sshd_config を編集する。

最後に sshd を再起動します。

/etc/security/access.conf には、どこからの接続を許可するかというルールを記述します。

下記は、ユーザ linuc が 192.168.138.0/24 のネットワークから接続する場合のみ接続を許可し、その他は全て拒否するというものです。

設定例

```
+ : linuc : 192.168.138.0/24
- : ALL : ALL
```

設定例

/etc/pam.d/sshd には pam_access に関する記述を追加します。

（太字が追加部分）

```
%PAM-1.0
auth       required     pam_sepermit.so
auth       include      password-auth
account    required     pam_nologin.so
account    required     /lib/security/pam_access.so     ←追加
account    include      password-auth                ←この行より前に追加する

===  省略 ===
```

pam_access に関する記述は、” account　include　password-auth” より前に記述する必要がありますので、記述位置に注意してください。

/etc/ssh/sshd_config では、UsePAM を yes とします。

ここまでの設定が完了すれば、sshd を再起動することで、新しい設定が反映されます。

6.5　鍵認証による SSH 接続

SSH には通常のパスワードを使った認証以外にも鍵ファイル利用した認証方法（公開鍵暗号方式）が用意されています。

6.5.1 sshd の設定

公開鍵暗号方式を利用するにはサーバ側の sshd_confg を以下のように編集します。

設定例

root 権限で /etc/sshd_config を編集します。

```
[root@localhost]# vi /etc/ssh/sshd_config
```

以下の 2 行のコメントを外して有効にします。

＜変更前＞

```
#PubkeyAuthentication yes
#AuthorizedKeysFile     .ssh/authorized_keys
```

＜変更後＞

```
PubkeyAuthentication yes
AuthorizedKeysFile     .ssh/authorized_keys
```

設定変更が完了したら、sshd を再起動します。

実行例

```
[root@localhost]# /etc/init.d/sshd restart
sshd を停止中:                                             [  OK  ]
sshd を起動中:                                             [  OK  ]
```

6.5.2 鍵ファイルの生成と使用

SSH で公開鍵暗号方式を利用するには、最初に鍵を生成します。

Linux での鍵ファイルの生成と使用

Linux 上で SSH の鍵ファイルを生成するには ssh-keygen コマンドを使います。

```
$ ssh-keygen
```

最初に鍵ファイルの保存場所とファイル名を質問されます。

```
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
```

デフォルトではホームディレクトリの .ssh ディレクトリ配下に秘密鍵は id_rsa、公開鍵は id_rsa.pub というファイル名で生成されます。変更する必要がなければそのまま Enter を押して処理を続けます。

次にパスフレーズの入力を求めるメッセージが 2 回表示されるので 2 回とも同じパスフレーズを入力します。パスフレーズはパスワードよりも自由度が高いので 10～30 文字程度のアルファベット・数字・記号を使った複雑で長いフレーズを設定すると良いでしょう。

```
Enter passphrase (empty for no passphrase):
```

生成された鍵セットのうち、公開鍵暗号方式の特性上、秘密鍵は他人に知られてはいけません。ファイルのパーミッションや保存場所など取り扱いには十分な注意が必要です。デフォルトで ~/.ssh ディレクトリのパーミッションは rwx------ 、id_rsa のパーミッションは rw-------、id_rsa.pub のパーミッションは rw-r--r-- です。

ssh-keygen のオプションや使い方の詳しい説明は man ssh-keygen などで確認できます。

サーバ、クライアント共に Linux の場合は、次のようにクライアント側で ssh-keygen を実行すると良いでしょう。

実行例

```
[linuc@client ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/linuc/.ssh/id_rsa):  ←そのまま Enter
Created directory '/home/linuc/.ssh'.
Enter passphrase (empty for no passphrase):                     ←パスフレーズを入力
Enter same passphrase again:                                    ←再度パスフレーズを入力
Your identification has been saved in /home/linuc/.ssh/id_rsa.
Your public key has been saved in /home/linuc/.ssh/id_rsa.pub.
The key fingerprint is:
f6:02:6b:73:05:95:62:c7:90:47:2d:31:ab:aa:15:d8 linuc@client
The key's randomart image is:
+--[ RSA 2048]----+
|        .==+     |
|        +.*o.    |
|       ..+..     |
|     o   o       |
|    . E S .      |
|       * o       |
|      * o .      |
|     + o .       |
|    .            |
+-----------------+
[linuc@client ~]$ ls .ssh
id_rsa  id_rsa.pub                                      ←秘密鍵と公開鍵が作成されている
[linuc@client ~]$ ls -ld .ssh
drwx------. 2 linuc linuc 4096  5月 29 16:32 2013 .ssh
[linuc@client ~]$ ls -l .ssh
合計 8
-rw-------. 1 linuc linuc 1751  5月 29 16:32 2013 id_rsa
-rw-r--r--. 1 linuc linuc  395  5月 29 16:32 2013 id_rsa.pub
[linuc@client ~]$
```

次に、クライアント側で生成した鍵ペアのうち、公開鍵をサーバ側に転送します。

なお、サーバ側のホームディレクトリに .ssh ディレクトリが作成されていない場合は事前に作成します。.ssh ディレクトリのパーミッションは 700 に設定します。

```
[linuc@client ~]$ ssh server
linuc@server's password:
Last login: Wed May 29 19:02:17 2013 from client
[linuc@server ~]$ ls -a
[linuc@server ~]$ mkdir .ssh
[linuc@server ~]$ chmod 700 .ssh
[linuc@server ~]$ ls -ld .ssh
drwx------. 2 linuc linuc 4096  5月 29 19:35 2013 .ssh
```

.ssh/authorized_keys ファイルが作成されていない場合は事前に作成します。authorized_keys ファイルのパーミッションは 600 に設定して他人から見ることができないようにします。

```
[linuc@server ~]$ touch .ssh/authorized_keys
[linuc@server ~]$ chmod 600 .ssh/authorized_keys
[linuc@server ~]$ ls -l .ssh
合計 4
-rw-------. 1 linuc linuc 395  5月 29 19:35 2013 authorized_keys
```

すでに .ssh/authorized_key が存在している場合はクライアント側で次のように ssh コマンドを実行することで公開鍵ファイルを転送することが可能です。

```
[linuc@client ~]$ cat .ssh/id_rsa.pub | ssh linuc@server 'cat >> .ssh/authorized_keys'
linuc@server's password:
[linuc@client ~]$
```

ssh 経由の作業で、サーバ側とクライアント側で使用するユーザ名が異なる場合は ssh のオプションでユーザ名を指定し忘れないように注意します。

サーバ側で authorized_keys を使用する設定が済んでいれば、authorized_keys に公開鍵を追加した時点で次の ssh 接続から公開鍵暗号方式で通信が行われます。

authorized_keys を設定する前は password の入力を求められていました。

```
[linuc@client ~]$ ssh linuc@server
linuc@server's password:
[linuc@server ~]$
```

authorized_keys を設定すると passphrase の入力が求められるようになります。

```
[linuc@client ~]$ ssh linuc@server
Enter passphrase for key '/home/linuc/.ssh/id_rsa':
Last login: Wed May 29 19:33:48 2013 from client
[linuc@server ~]$
```

Windows での鍵ファイルの生成と利用

Windows では Tera Term を使って鍵ファイルを生成することができます。

Tera Term のメニューから「設定」－「SSH 鍵生成」を選択します。

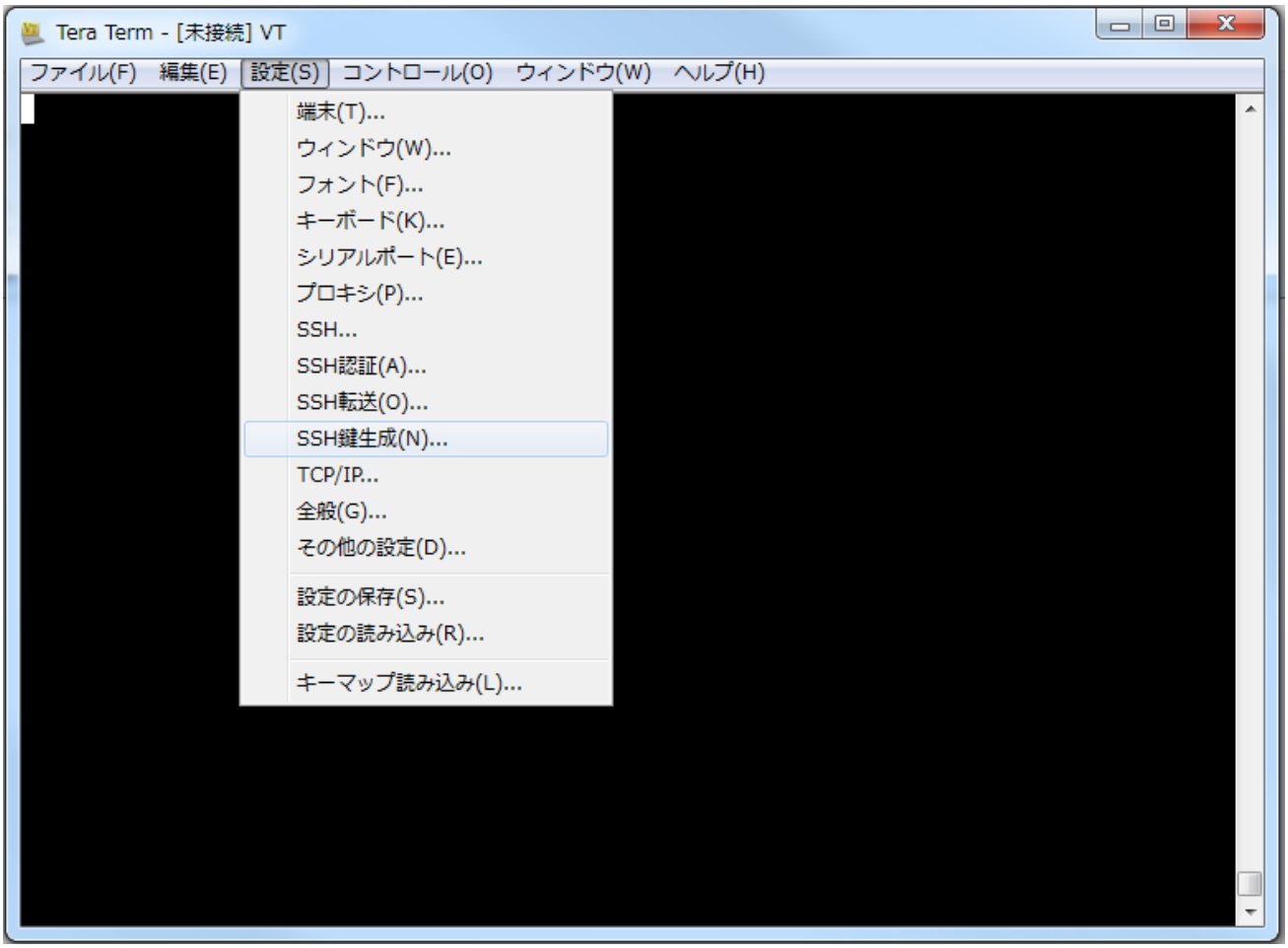


図 6.11 TeraTerm の SSH 鍵生成メニュー

鍵生成の情報入力画面が表示されますので、鍵の種類が「RSA」ビット数が「2048」になっていることを確認して、「生成」ボタンを押します。

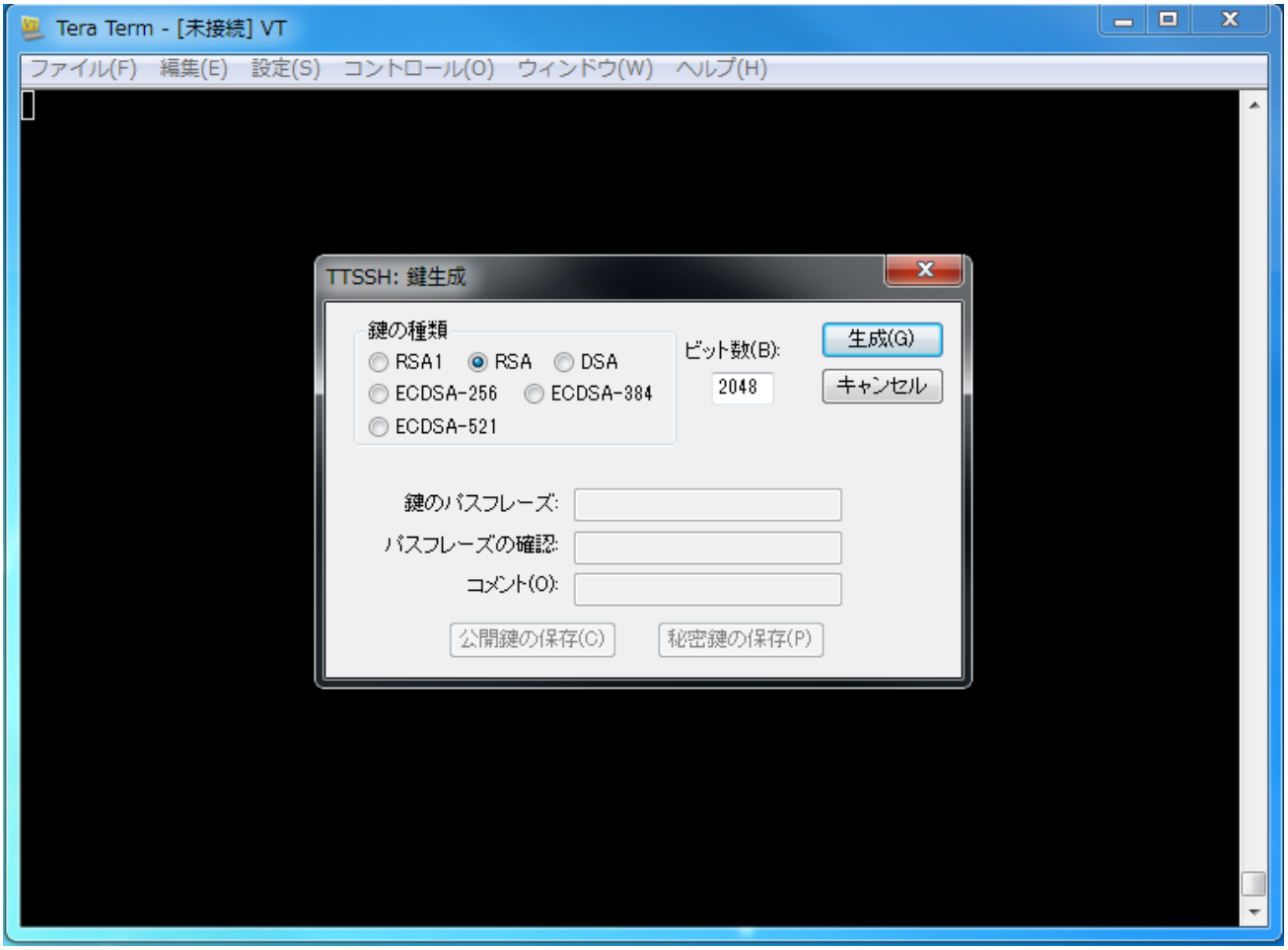


図 6.12 TeraTerm の SSH 鍵生成情報入力画面

鍵の生成が完了したら、パスフレーズを入力します。コメントは任意に入力することができます。

「公開鍵の保存」「秘密鍵の保存」それぞれのボタンを押して、保存先を指定して二つの鍵を保存します。この例では、公開鍵を id_rsa.pub、秘密鍵を id_rsa という名前で保存しています。

秘密鍵が外部に漏れると鍵認証の意味がなくなってしまうため、秘密鍵には厳重な管理が必要です。

次に、公開鍵をサーバへアップロードします。

Tera Term で SSH 接続を行った上で、保存した公開鍵ファイルを Tera Term の画面へドラッグしてください。

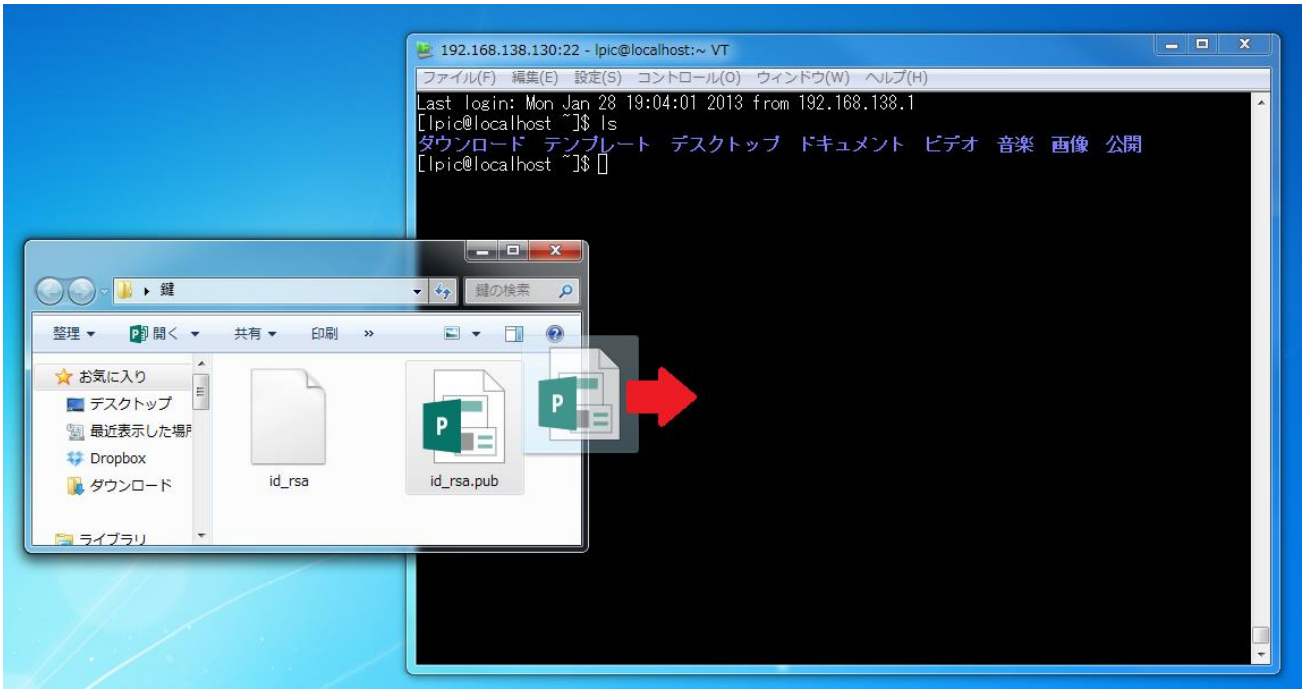


図 6.13 TeraTerm のファイル転送

ファイル転送を行いますか？と聞かれますので、「SCP」ボタンを押してファイルをサーバへアップロードしてください。SCP は暗号化された状態でファイルを安全に転送してくれます。

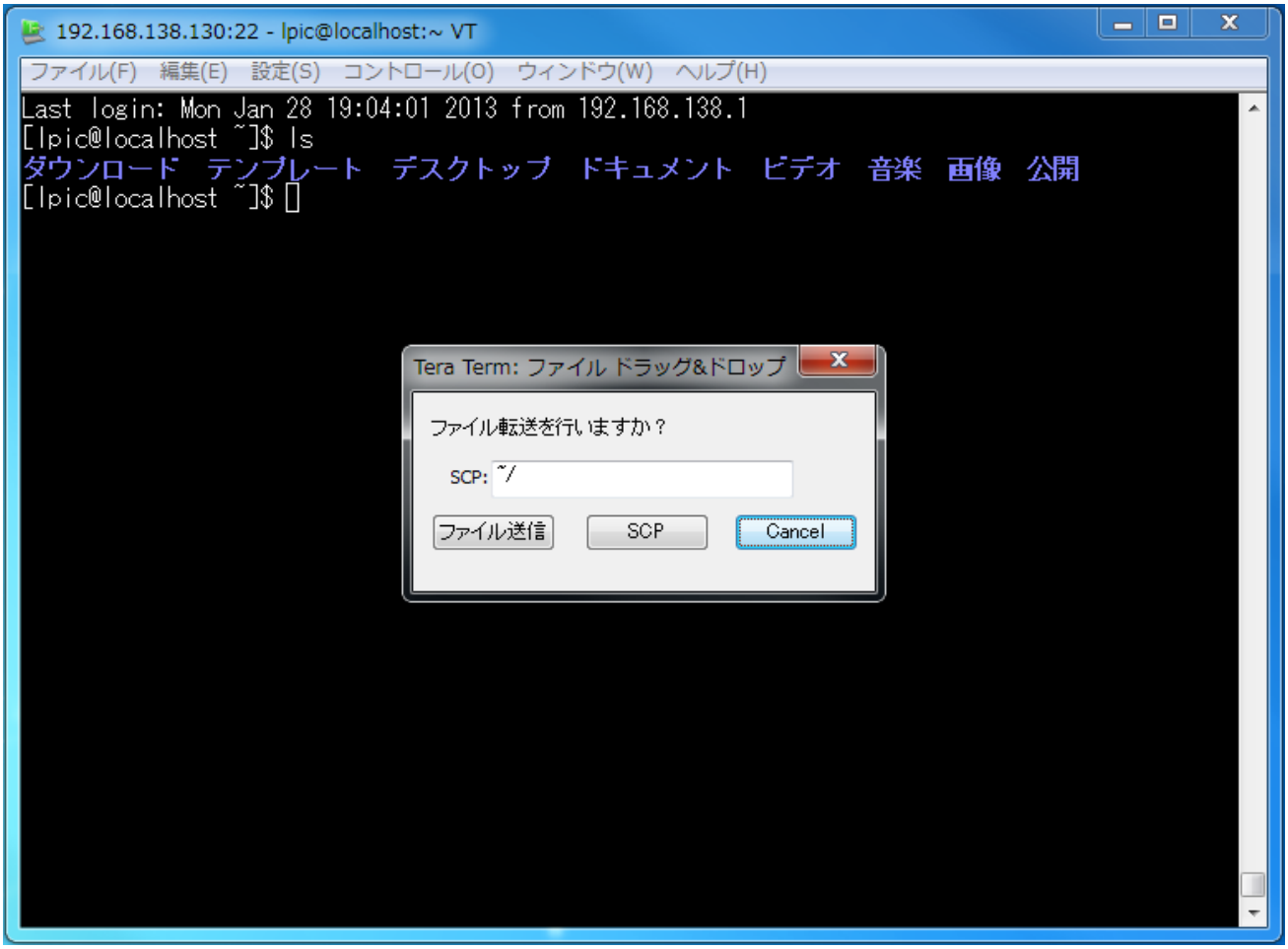


図 6.14 TeraTerm のファイル転送確認画面

公開鍵をホームディレクトリ配下の.ssh ディレクトリに、authorized_keys という名前で保存します。公開鍵のパーミッションは 600 にしてください。

実行例

```
[root@localhost]# ls
id_rsa.pub     テンプレート   ドキュメント   音楽   公開
ダウンロード   デスクトップ   ビデオ         画像
[root@localhost]# mv id_rsa.pub ~/.ssh/authorized_keys
[root@localhost]# chmod 600 ~/.ssh/authorized_keys
[root@localhost]# ls -l ~/.ssh
合計 4
-rw-------. 1 linuc linuc 381  1月 28 19:22 2013 authorized_keys
```

クライアント側では、Tera Term において秘密鍵を使ってサーバへアクセスを行います。

基本的に、パスワードで認証する時と同じですが、認証の画面で「プレインテキストを使う」ではなく、「RSA/DSA 鍵を使う」を選択した上で、「秘密鍵」ボタンを押して、保存している秘密鍵を指定してください。

画面上のパスフレーズには、鍵を生成した際に指定したパスフレーズを入力してください。

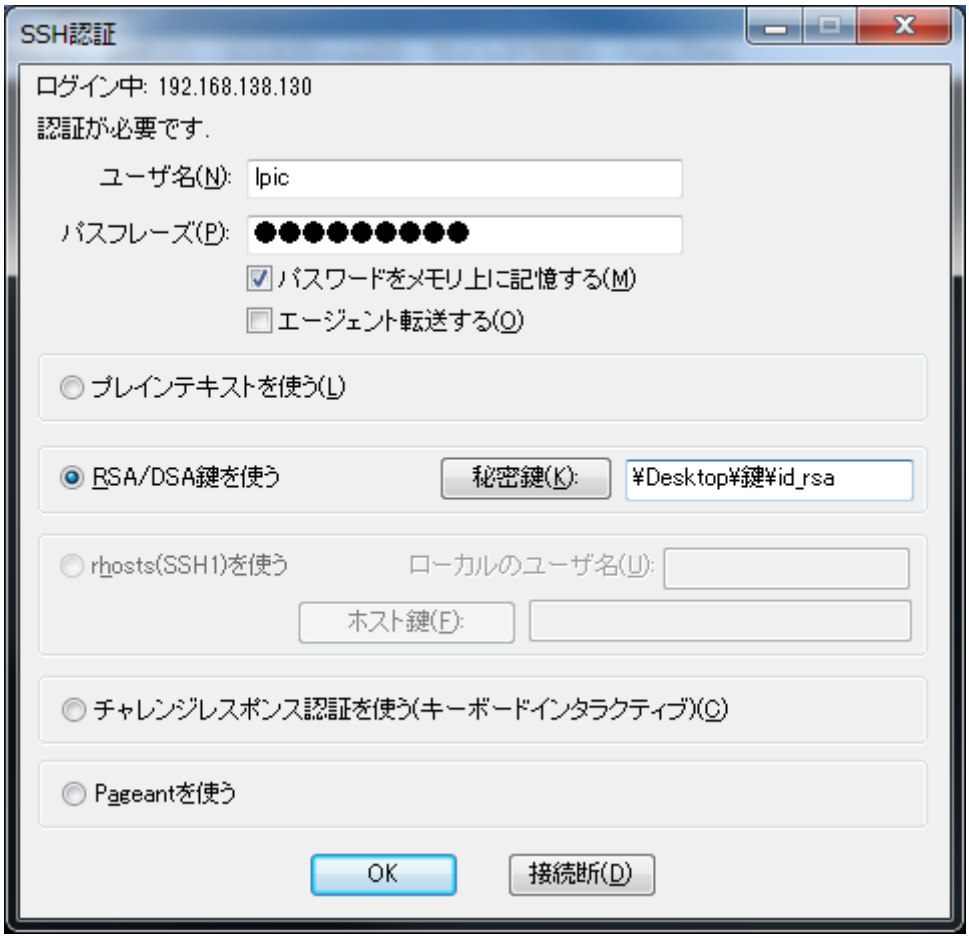


図 6.15 TeraTerm の認証鍵接続

6.6. PermitRootLogin の設定

デフォルトの SSH の設定では root ユーザとして直接ログインすることが可能なので、root でのログインは許可しないように設定すると良いでしょう。root でのログインを許可しないようにするには /etc/ssh/sshd_config を次のように修正します。

＜変更前＞

```
#PermitRootLogin yes
```

＜変更後＞

```
PermitRootLogin no
```

変更後は sshd を再起動します。

PermitRootLogin に設定する値は次のとおりです。

パラメータ	意味
yes	root での SSH ログインを許可します。(デフォルト)
no	root での SSH ログインを許可しません。
without-password	パスワード認証を許可しません。
forced-commands-only	指定されたコマンドのみ許可します。その他の認証方法は許可されません。

forced-commands-only は、リモートでバックアップなどの処理をさせたいときに便利です。後で紹介するように authorized_keys ファイルに command オプション設定することで実行したいコマンドを指定することができます。PermitRootLogin のオプションについては man sshd_config で確認することができます。

6.7. パスワードによる認証を無効にする

デフォルトの設定では通常のパスワード認証が許可されていますが、通常のパスワード認証では辞書攻撃のような手法でユーザ名とパスワードを割り出されてしまうと、不正なアクセスを許す恐れがあります。

これに対して鍵ファイルを使った認証の場合、仮にユーザ名とパスフレーズを正しく推測できたとしても、秘密鍵を持っていなければ不正なアクセスを許すことはありません。また、秘密鍵を盗まれたとしてもパスフレーズが分からなければ秘密鍵を使った認証はできません。

このように通常のパスワード認証よりも鍵ファイルを使った認証の方がセキュリティ強度を高く保つことができます。特に、接続元の制限を緩めたい場合は、通常のパスワード認証は許可しないで、鍵ファイルを用いた認証だけを許可する設定にすると良いでしょう。

鍵ファイルでの認証だけを許可する際は、事前に鍵ファイルを使った認証で正しくアクセスできることを確認します。**リモートからアクセスして作業を行っている場合、設定を間違えると SSH でのアクセスができなくなる可能性があるので、十分に注意してください。**

鍵認証でのアクセスを確認できたら、sshd_config を編集して、パスワードによる認証ができないようにします。

公開鍵認証のみを許可するには/etc/ssh/sshd_config を次のように修正します。

＜変更前＞

```
PasswordAuthentication yes
```

＜変更後＞

```
PasswordAuthentication no
```

再度 sshd を再起動すると、鍵認証以外でのアクセスはできなくなり、より安全な SSH 運用が可能になります。

6.8. パスフレーズなしの運用について

リモートのサーバで自動バックアップのような処理でパスワードなしで実行したい場合に、パスフレーズなしの秘密鍵を用意することで比較的安全にリモートサーバの操作が可能となります。パスフレーズなしの秘密鍵で運用する場合、その秘密鍵が盗まれてしまうとパスフレーズなしで認証されてしまうため、鍵ファイルは厳重に管理する必要があります。root 権限でパスフレーズなしの鍵ファイルを利用したい場合は /etc/ssh/sshd_config の PermitRootLogin で forced-commands-only を指定すると良いでしょう。

パスフレーズなしの鍵ファイルは ssh-keygen を実行したときに空のパスフレーズを入力するか、後からパスフレーズを変更することで作成できます。

秘密鍵のパスフレーズを変更するには ssh-keygen コマンドの -p オプションを使います。-f オプションで鍵ファイルを指定することができます。

```
[linuc@localhost ~]$ ssh-keygen -p -f .ssh/id_rsa_nopass
Enter old passphrase:    ←古いパスフレーズを入力
Key has comment '.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):        ←そのまま Enter
Enter same passphrase again:                           ←そのまま Enter
Your identification has been saved with the new passphrase.
```

パスフレーズなしで運用する場合は専用の鍵ファイルを用意し、authorized_keys の設定で下記のように command オプションで使用可能なコマンドを制限すると良いでしょう。

＜authorized_keys で実行できるコマンドを制限する設定の例＞

```
command="ls /bin" ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCcpn8vg0cdbmi5bpyCpO8DB5xtLqu5UC0AQT37/J2g8OK5D0pwI9I4+z5qx7bu
UzF7znbX6/iATVyhdllD+uC3K0TBg/tcNQ/n3MLqrv・・・＜省略＞
```

クライアント側の ~/.ssh に保存された秘密鍵 id_rsa_nopass を使ってサーバに SSH 接続するには、“-i” オプションで鍵ファイルを指定します。

```
[linuc@client ~]$ ssh -i .ssh/id_rsa_nopass linuc@server
alsaunmute            dnsdomainname  lsblk           rview
arch                  domainname     lscgroup        sed
awk                   dumpkeys       lssubsys        setfont
basename              echo           mail            setserial
bash                  ed             mailx           sh
cat                   egrep          mkdir           sleep
＜以下略＞
[linuc@client ~]$
```

サーバ側の ~/.ssh/authorized_keys の command オプションで指定されたコマンドだけが実行されます。

authorized_keys ファイルの詳しい説明は man authorized_keys などで確認してください。

7. セキュリティツールによる改ざん検知と侵入検知

7.1. システムの改ざん検知および侵入検知について

一般的に、外部のインターネットにシステムを接続する際は、ファイアウォールを配置することで不正アクセスや第三者の侵入などの驚異に対してセキュリティ対策を行います。

しかし、ファイアウォールを配置することで、ポートや IP アドレスでアクセスを制限できるサービスに対する不要な通信を遮断することはできますが、不特定多数に対して公開しているサービスへの攻撃（たとえば、ウェブアプリケーションの脆弱性をついた攻撃）については、完全に防御することが難しいと言えます。

適切なセキュリティ対策がなされていないことにより、万が一、不正アクセスの被害にあった場合、公開サービスの停止といった単純な被害だけではなく、Web コンテンツの改ざんなどにより、知らずに他人を攻撃してしまう（たとえば、公開している Web コンテンツがウィルス配布サイトへ転送するよう改ざんされたり、フィッシングサイトへの誘導するよう改ざんされたりする）事態が起こる可能性があります。

上述のような、ファイアウォール等のセキュリティ対策を回避して侵入された不正なアクセスを検知する有効な手段として、改ざん検知や侵入検知が挙げられます。

本章では、改ざん検知ソフトウェアである Tripwire と、侵入検知ソフトウェアである Snort を紹介します。

7.1.1 改ざん検知について

システム内のファイルに対する追加・変更・削除といった事象を検知することで、不正アクセス等によるファイルの改ざんを検知します。

改ざん検知を行うことで、予期せぬ Web コンテンツの変更等を検知することができ、不正アクセスを検出することができるようになります。

7.1.2. 侵入検知について

ネットワークを流れるパケットを監視して、ワームやサービス拒否攻撃(DoS)などのパケットが持つ特徴的なパターンをシグネチャというパターンファイルで管理し、シグネチャを基に悪意のあるパケットを検知します。

侵入検知を行うことで、外部から不正なアクセスやその傾向を検知することができるようになります。

7.2. Tripwire による改ざん検知

7.2.1. Tripwire の概要

本節では、改ざん検知ソフトウェアである Tripwire の概要を説明します。

Tripwire は、システムに加えられた変更を検知するオープンソースの改ざん検知ソフトウェアです。GPL(GNU General Public License)の元に無償で利用することができます。

Tripwire は、1992 年に、アメリカパデュー大学で、スパフォード博士とジーン・キム氏らにより開発されました。その後、米 Tripwire 社により商用化され、対応するプラットフォームの拡張などが行われています。現在、Linux のみ対応するオープンソース版と Windows などにも対応する米 Tripwire 社による商用版が存在します。オープンソース版は、http://sourceforge.net/projects/tripwire/より入手することができます。

Tripwire は、正常な状態でのシステムのスナップショット（ベースラインデータベース）をデータベースに保存し、現在の状態でのシステムのスナップショットと比較することで改ざんを検知します。このため、システムが正常な状態でベースラインデータベースを作成しておくことが重要です。

Tripwire では、以下のような事象を発見することができます。

- ファイルの内容が変更された
- ファイルやディレクトリが追加された
- ファイルやディレクトリが削除された
- ファイルやディレクトリのオーナー・パーミッションが変更された

一方、Tripwire では、以下のようなことはできません。

- リアルタイムで変更を検知する
- ファイル・ディレクトリの変更者を特定する
- ファイル・ディレクトリの変更箇所を特定する
- ファイル・ディレクトリを変更前の状態に戻す

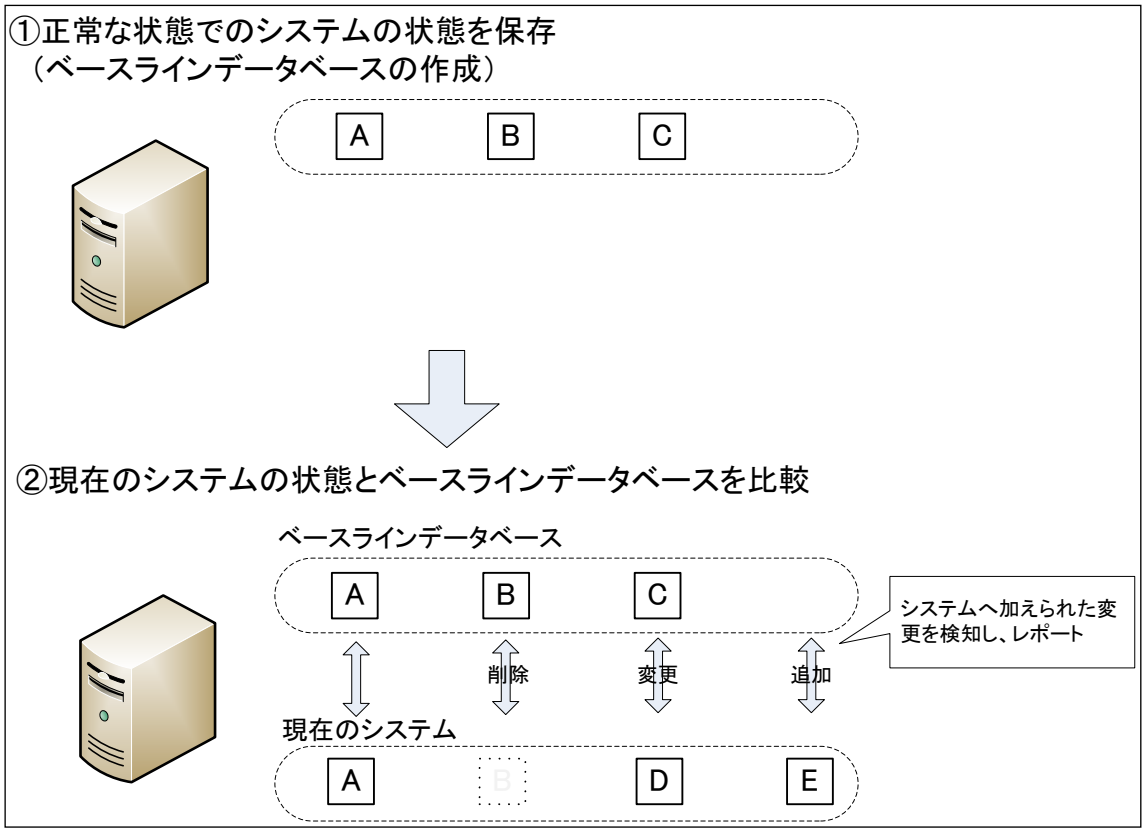


図 7.1 Tripwire の概要

7.2.2. Tripwire の導入

Tripwire 導入の流れを以下に示します。

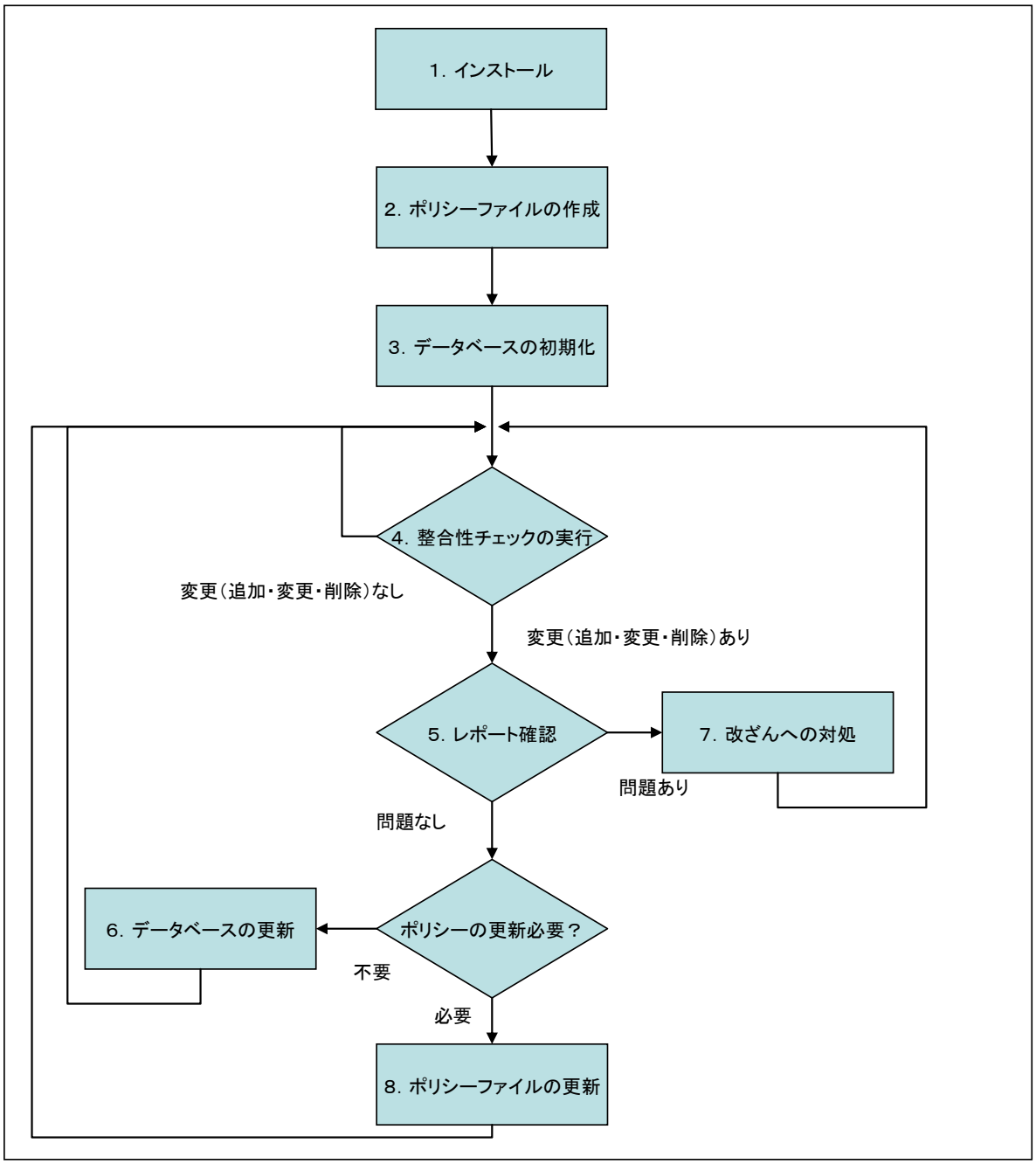


図 7.2　Tripwire 導入の流れ

7.2.2.1. Tripwire のインストール

Tripwire のインストール手順について示します。

以下の手順は、執筆時点での最新バージョンである 2.4.2.2 についてのものです。

① パスフレーズの準備

Tripwire では 2 つの鍵ファイル（サイトキー、および、ローカルキー）を利用してファイルの保護を行っています。インストール中にこれら 2 つの鍵ファイルのパスフレーズの入力を求められるため、インストール作業を開始する前に準備しておく必要があります。各パスフレーズは 8 文字以上とする必要があります。

表 7.1 鍵ファイルと使用用途

鍵ファイル	使用用途
サイトキー	システム設定ファイル、ポリシーファイルの保護
ローカルキー	データベースファイル、レポートファイルの保護

② インストール

CentOS 用の Tripwire パッケージは、執筆時点において用意されておりませんので、wget でソースコードをダウンロードして、make などを用いてコンパイルを行います。

コンパイルには、make, gcc といったおなじみのパッケージの他に、cpp および gcc-c++パッケージが必要になりますので、事前に yum コマンドを用いてインストールしておいてください。

実行例

```
[root@localhost ~]# wget
http://jaist.dl.sourceforge.net/project/tripwire/tripwire-src/tripwire-2.4.2.2/tripwire-2.4.2.2-src.tar.bz2
[root@localhost ~]# tar jxvf tripwire-2.4.2.2-src.tar.bz2
[root@localhost ~]# cd tripwire-2.4.2.2-src
[root@localhost ~]# ./configure
[root@localhost ~]# make
[root@localhost ~]# make install
:
（省略）
:
Installer program for:
```

```
Tripwire® 2.4 Open Source
Copyright © 1998-2000 Tripwire ® Security Systems, Inc.  Tripwire ®
is a registered trademark of the Purdue Research Foundation and is
licensed exclusively to Tripwire ® Security Systems, Inc.
LICENSE AGREEMENT for Tripwire® 2.4 Open Source
Please read the following license agreement.  You must accept the
agreement to continue installing Tripwire.
Press ENTER to view the License Agreement.
```
ENTER キーを押下すると、使用許諾契約が表示されます。SPACE キーを押下し、使用許諾契約の最後まで進みます。
```
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
 :
 (省略)
 :
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.


Please type “accept” to indicate your acceptance of this
license agreement. [do not accept] accept (accept と入力)
```
ここで ENTER キーを押下すると、インストーラーが終了してしまうので、注意してください。
```

Using configuration file ./install/install.cfg

Checking for programs specified in install configuration file....

/usr/sbin/sendmail -oi -t exists.  Continuing installation.

/bin/vi exists.  Continuing installation.


-------------------------------------------
Verifying existence of binaries...

./bin/siggen found
./bin/tripwire found
./bin/twprint found
./bin/twadmin found

This program will copy Tripwire files to the following directories:
```

```
        TWBIN: /usr/local/sbin
        TWMAN: /usr/local/man
     TWPOLICY: /usr/local/etc
     TWREPORT: /usr/local/lib/tripwire/report
         TWDB: /usr/local/lib/tripwire
 TWSITEKEYDIR: /usr/local/etc
TWLOCALKEYDIR: /usr/local/etc

CLOBBER is false.

Continue with installation? [y/n] y (y と入力)

---------------------------------------------
Creating directories...

/usr/local/sbin: already exists
/usr/local/etc: already exists
/usr/local/lib/tripwire/report: created
/usr/local/lib/tripwire: already exists
/usr/local/etc: already exists
/usr/local/etc: already exists
/usr/local/man: created
/usr/local/doc/tripwire: created

---------------------------------------------
Copying files...

/usr/local/doc/tripwire/COPYING: copied
/usr/local/doc/tripwire/TRADEMARK: copied
/usr/local/doc/tripwire/policyguide.txt: copied
/usr/local/etc/twpol-Linux.txt: copied

---------------------------------------------
The Tripwire site and local passphrases are used to
sign a variety of files, such as the configuration,
policy, and database files.

Passphrases should be at least 8 characters in length
and contain both letters and numbers.

See the Tripwire manual for more information.

---------------------------------------------
```

```
Creating key files...

(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)

Enter the site keyfile passphrase: （サイトパスフレーズを入力）
Verify the site keyfile passphrase: （サイトパスフレーズを入力（確認用））
Generating key (this may take several minutes)...Key generation complete.

(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)

Enter the local keyfile passphrase: （ローカルパスフレーズを入力）
Verify the local keyfile passphrase: （ローカルパスフレーズを入力（確認用））
Generating key (this may take several minutes)...Key generation complete.

---------------------------------------
Generating Tripwire configuration file...

---------------------------------------
Creating signed configuration file...
Please enter your site passphrase: （サイトパスフレーズを入力）
Wrote configuration file: /usr/local/etc/tw.cfg

A clear-text version of the Tripwire configuration file
/usr/local/etc/twcfg.txt
has been preserved for your inspection.  It is recommended
that you delete this file manually after you have examined it.

---------------------------------------
Customizing default policy file...

---------------------------------------
Creating signed policy file...
Please enter your site passphrase: （サイトパスフレーズを入力）
Wrote policy file: /usr/local/etc/tw.pol

A clear-text version of the Tripwire policy file
/usr/local/etc/twpol.txt
has been preserved for your inspection.  This implements
a minimal policy, intended only to test essential
Tripwire functionality.  You should edit the policy file
```

```
to describe your system, and then use twadmin to generate
a new signed copy of the Tripwire policy.



-----------------------------------------------
The installation succeeded.

Please refer to
for release information and to the printed user documentation
for further instructions on using Tripwire 2.4 Open Source.

make[3]: Leaving directory `/root/tripwire-2.4.2.2-src'
make[2]: Leaving directory `/root/tripwire-2.4.2.2-src'
make[1]: Leaving directory `/root/tripwire-2.4.2.2-src'
[root@localhost tripwire-2.4.2.2-src]#
```

7.2.2.2. ポリシーファイルの作成

Tripwire はポリシーファイルに従い、ファイル・ディレクトリの検査を行います。

ポリシーファイルは、ポリシーを記述したテキストファイル（以降、ポリシーファイル（平文）と呼ぶ）の内容を元に、twadmin コマンドを利用して作成します。

ポリシーの記述内容については、07.2.3.4. ポリシーファイル～07.2.3.7. 属性を参照してください。

ポリシーファイルの作成は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# twadmin --create-polfile --site-keyfile /usr/local/etc/site.key（サイトキーファイルのパス） /tmp/twpol.txt（ポリシーファイル（平文）のパス）
```

または

```
[root@localhost ~]# twadmin -m P -S /usr/local/etc/site.key（サイトキーファイルのパス） /tmp/twpol.txt（ポリシーファイル（平文）のパス）
```

※ポリシーファイル（平文）のパスは、環境に応じて適切な値に置き換えてください。

※ポリシーファイル(平文)のパスとして、Tripwire インストール時に生成されるファイル(/usr/local/etc/twpol.txt)を指定することも可能です。

コマンド実行時にサイトパスフレーズの入力を求められますので、Tripwire をインストールした時に入力したサイトパスフレーズを入力してください。

なお、サイトキーファイルには、Tripwire インストール時に作成された/usr/local/etc/site.key を指定します。

実行例

```
[root@localhost ~]# twadmin --create-polfile --site-keyfile /usr/local/etc/site.key /tmp/twpol.txt
Please enter your site passphrase:（サイトパスフレーズを入力）
Wrote policy file: /usr/local/etc/tw.pol
[root@localhost ~]#
```

コマンドを実行することにより、ポリシーファイル（平文）から署名暗号化されたポリシーファイル（usr/local/etc/tw.pol）が作成されます。

7.2.2.3. データベースの初期化

ポリシーファイルを元に、現時点でのシステムのファイル・ディレクトリ情報をデータベース化します。

データベースの初期化は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# tripwire --init
```

または

```
[root@localhost ~]# tripwire -m i
```

コマンド実行時にローカルパスフレーズの入力を求められます。

以下は、Tripwire のインストール時に生成されるファイル（/usr/local/etc/twpol.txt）をそのまま使用した場合の例です。

実行例

```
 [root@localhost ~]# tripwire --init
Please enter your local passphrase: （ローカルパスフレーズを入力）
Parsing policy file: /usr/local/etc/tw.pol
Generating the database...
*** Processing Unix File System ***
The object: "/selinux" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
### Warning: File system error.     ※エラー（/usr/local/sysinfo が存在しない）
### Filename: /usr/local/sysinfo    ※エラー（/usr/local/sysinfo が存在しない）
### No such file or directory       ※エラー（/usr/local/sysinfo が存在しない）
### Continuing...
### Warning: File system error.     ※エラー（/usr/X11R6/lib が存在しない）
### Filename: /usr/X11R6/lib        ※エラー（/usr/X11R6/lib が存在しない）
### No such file or directory       ※エラー（/usr/X11R6/lib が存在しない）
 :
 （省略）
```

```
 :
Wrote database file: /usr/local/lib/tripwire/localhost.localdomain.twd
The database was successfully generated.
 [root@localhost ~]#
```

上記コマンド実行例のようにエラーが出力された場合は、ポリシーファイルを修正し、再度データベースの初期化を行います。

1．ポリシーファイル（平文）の修正

エラーとなったファイル・ディレクトリのパスを確認し、正しいパスに修正するか、監視対象から外してもいい場合は、ポリシーファイル（平文）の該当箇所をコメントアウトします。

2．ポリシーファイルの作成（twadmin --create-polfile）

3．データベースの初期化（tripwire --init）

7.2.2.4. 整合性チェック

ベースラインデータベースに保持しているシステムのファイル・ディレクトリ情報と現在のシステムのファイル・ディレクトリ情報をポリシーファイルに従いチェックします。

実運用においては、定期的にシステムの整合性チェックを行うことになるため、整合性チェックのコマンドを cron に登録すると便利です。

整合性チェックは、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# tripwire --check
```

または

```
[root@localhost ~]# tripwire -m c
```

整合性チェックの結果は、/usr/local/lib/tripwire/report/ホスト名-日付（YYYYMMDD）-時刻（HHMMSS）.twr というレポートファイルして出力されます。

監視対象のコンテンツに、変更を加えず、整合性チェックを実行した結果は以下の通りです。

「Total violations found: 0」、「No violations.」と表示され、違反は報告されません。

出力例

```
[root@localhost ~]# tripwire --check
Parsing policy file: /usr/local/etc/tw.pol
*** Processing Unix File System ***
Performing integrity check...
The object: "/selinux" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
The object: "/var/run/vmblock-fuse" is on a different file system...ignoring.
Wrote report file: /usr/local/lib/tripwire/report/localhost.localdomain-20130113-154402.twr


Open Source Tripwire(R) 2.4.2.2 Integrity Check Report

Report generated by:          root
Report created on:            Sun 13 Jan 2013 03:44:02 PM JST
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    localhost.localdomain
Host IP address:              127.0.0.1
Host ID:                      None
Policy file used:             /usr/local/etc/tw.pol
Configuration file used:      /usr/local/etc/tw.cfg
Database file used:           /usr/local/lib/tripwire/localhost.localdomain.twd
Command line used:            tripwire --check

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                    Severity Level    Added    Removed  Modified
  ---------                    --------------    -----    -------  --------
  Tripwire Data Files          0                 0        0        0
  Monitor Filesystems          0                 0        0        0
```

```
  User Binaries and Libraries     0                 0       0       0
  Tripwire Binaries               0                 0       0       0
  OS Binaries and Libraries       0                 0       0       0
  Temporary Directories           0                 0       0       0
  Global Configuration Files      0                 0       0       0
  System Boot Changes             0                 0       0       0
  RPM Checksum Files              0                 0       0       0
  OS Boot Files and Mount Points  0                 0       0       0
  OS Devices and Misc Directories 0                 0       0       0
  Root Directory and Files        0                 0       0       0

Total objects scanned:  107042
Total violations found:  0

===============================================================================
Object Summary:
===============================================================================

-------------------------------------------------------------------------------
# Section: Unix File System
-------------------------------------------------------------------------------

No violations.

===============================================================================
Error Report:
===============================================================================

No Errors

-------------------------------------------------------------------------------
*** End of report ***

Open Source Tripwire 2.4 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.
 [root@localhost ~]#
```

監視対象のファイルに、改ざんされたことを想定した変更を加えて、整合性チェックを実行した結果は以下の通りです。

検知された変更が、違反として報告されます。

以下の出力例から以下の 5 つに関して変更が検出されていることが、わかります。

・/usr/local/etc/twpol.txt.bak が削除

・/usr/local/etc が変更

・/usr/local/etc/twpol.txt が変更

・/home/test/twpol.txt.bak が追加

・/home/test/が変更

出力例

```
[root@localhost ~]# tripwire --check
Parsing policy file: /usr/local/etc/tw.pol
*** Processing Unix File System ***
Performing integrity check...
The object: "/selinux" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
The object: "/var/run/vmblock-fuse" is on a different file system...ignoring.
Wrote report file: /usr/local/lib/tripwire/report/localhost.localdomain-20130113-160810.twr


Open Source Tripwire(R) 2.4.2.2 Integrity Check Report

Report generated by:          root
Report created on:            Sun 13 Jan 2013 04:08:10 PM JST
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    localhost.localdomain
Host IP address:              127.0.0.1
Host ID:                      None
Policy file used:             /usr/local/etc/tw.pol
Configuration file used:      /usr/local/etc/tw.cfg
Database file used:           /usr/local/lib/tripwire/localhost.localdomain.twd
Command line used:            tripwire --check

===============================================================================
Rule Summary:
```

```
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Tripwire Data Files           0                 0        0        0
* Monitor Filesystems           0                 1        0        1
* User Binaries and Libraries   0                 0        1        2
  Tripwire Binaries             0                 0        0        0
  OS Binaries and Libraries     0                 0        0        0
  Temporary Directories         0                 0        0        0
  Global Configuration Files    0                 0        0        0
  System Boot Changes           0                 0        0        0
  RPM Checksum Files            0                 0        0        0
  OS Boot Files and Mount Points 0                0        0        0
  OS Devices and Misc Directories 0               0        0        0
  Root Directory and Files      0                 0        0        0

Total objects scanned:  107042
Total violations found:  5

================================================================================
Object Summary:
================================================================================

--------------------------------------------------------------------------------
# Section: Unix File System
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
Rule Name: User Binaries and Libraries (/usr/local/etc)
Severity Level: 0
--------------------------------------------------------------------------------

Removed:
"/usr/local/etc/twpol.txt.bak"

Modified:
"/usr/local/etc"
"/usr/local/etc/twpol.txt"
```

```
---------------------------------------------------------------------------
Rule Name: Monitor Filesystems (/home)
Severity Level: 0
---------------------------------------------------------------------------

Added:
"/home/test/twpol.txt.bak"

Modified:
"/home/test"

===========================================================================
Error Report:
===========================================================================

No Errors

---------------------------------------------------------------------------
*** End of report ***

Open Source Tripwire 2.4 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
Integrity check complete.
[root@localhost ~]#
```

7.2.2.5. レポートの確認

整合性チェックにより出力されたレポートファイルの表示は、以下のコマンドにより行います。

書式

```
  twprint  --print-report --report-level（レポートレベル,0～4）  --twrfile（レポートファイルのパス）
```

　　または

```
twprint  -m r -t（レポートレベル,0～4）-r（レポートファイルのパス）
```

※レポートレベルは、0 が簡易表示、4 が詳細表示です。

※レポートファイルのパスは、環境に応じて適切な値に置き換えてください。

レポートレベルが 0 の場合の表示内容は、以下の通りです。

実行例

```
[root@localhost ~]# twprint --print-report --report-level 0 --twrfile
/usr/local/lib/tripwire/report/localhost.localdomain-20130113-160810.twr
Note: Report is not encrypted.
TWReport localhost.localdomain 20130113160810 V:5 S:0 A:1 R:1 C:3
[root@localhost ~]#
```

レポート結果の各項目は以下の通りです。

V：違反の数
S：重要度
A：追加されたファイル・ディレクトリの数
R：削除されたファイル・ディレクトリの数
C：変更されたファイル・ディレクトリの数

上記の実行例では、違反が 5 個（ファイル・ディレクトリの追加が 1 個、ファイル・ディレクトリの削除が 1 個、ファイル・ディレクトリの変更が 3 個）そのうち重要度の最も高い値は 0 であることがわかります。

レポートレベルが 4 の場合における表示内容は、以下の通り詳細部分まで表示されます。

実行例

```
[root@localhost ~]# twprint --print-report --report-level 4 --twrfile
/usr/local/lib/tripwire/report/localhost.localdomain-20130113-160810.twr
Note: Report is not encrypted.
Open Source Tripwire(R) 2.4.2.2 Integrity Check Report
```

```
Report generated by:          root
Report created on:            Sun 13 Jan 2013 04:08:10 PM JST
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    localhost.localdomain
Host IP address:              127.0.0.1
Host ID:                      None
Policy file used:             /usr/local/etc/tw.pol
Configuration file used:      /usr/local/etc/tw.cfg
Database file used:           /usr/local/lib/tripwire/localhost.localdomain.twd
Command line used:            tripwire --check

===============================================================================
Rule Summary:
===============================================================================


-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Tripwire Data Files           0                 0        0        0
* Monitor Filesystems           0                 1        0        1
* User Binaries and Libraries   0                 0        1        2
  Tripwire Binaries             0                 0        0        0
  OS Binaries and Libraries     0                 0        0        0
  Temporary Directories         0                 0        0        0
  Global Configuration Files    0                 0        0        0
  System Boot Changes           0                 0        0        0
  RPM Checksum Files            0                 0        0        0
  OS Boot Files and Mount Points 0                0        0        0
  OS Devices and Misc Directories 0               0        0        0
  Root Directory and Files      0                 0        0        0

Total objects scanned:  107042
Total violations found:  5

===============================================================================
Object Summary:
```

```
================================================================================


--------------------------------------------------------------------------------
# Section: Unix File System
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
Rule Name: User Binaries and Libraries (/usr/local/etc)
Severity Level: 0
--------------------------------------------------------------------------------

Removed:
"/usr/local/etc/twpol.txt.bak"

Modified:
"/usr/local/etc"
"/usr/local/etc/twpol.txt"

--------------------------------------------------------------------------------
Rule Name: Monitor Filesystems (/home)
Severity Level: 0
--------------------------------------------------------------------------------

Added:
"/home/test/twpol.txt.bak"

Modified:
"/home/test"
"/home/test/.xsession-errors"

================================================================================
Object Detail:
================================================================================


--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
Rule Name: User Binaries and Libraries (/usr/local/etc)
Severity Level: 0
--------------------------------------------------------------------------------
  ----------------------------------------
  Removed Objects: 1
```

```
  ----------------------------------------

Removed object name:  /usr/local/etc/twpol.txt.bak

  Property:            Expected                    Observed
  -------------        -----------                 -----------
* Object Type          Regular File                ---
* Device Number        2050                        ---
* Inode Number         939253                      ---
* Mode                 -rw-r-----                  ---
* Num Links            1                           ---
* UID                  root (0)                    ---
* GID                  root (0)                    ---
* Size                 13715                       ---
* Modify Time          Thu 03 Jan 2013 10:12:35 PM JST
                                                  ---
* Blocks               32                          ---
* CRC32                CsX19s                      ---
* MD5                  DL30rgh65sv60UpWFXtLYo      ---



  ----------------------------------------
  Modified Objects: 2
  ----------------------------------------

Modified object name:  /usr/local/etc

  Property:            Expected                    Observed
  -------------        -----------                 -----------
  Object Type          Directory                   Directory
  Device Number        2050                        2050
  Inode Number         912759                      912759
  Mode                 drwxr-xr-x                  drwxr-xr-x
  Num Links            2                           2
  UID                  root (0)                    root (0)
  GID                  root (0)                    root (0)
  Size                 4096                        4096
* Modify Time          Sun 13 Jan 2013 03:35:57 PM JST
                                                  Sun 13 Jan 2013 04:07:32 PM JST
  Blocks               8                           8


Modified object name:  /usr/local/etc/twpol.txt
```

```
  Property:            Expected                    Observed
  -------------        -----------                 -----------
  Object Type          Regular File                Regular File
  Device Number        2050                        2050
* Inode Number         914985                      915102
  Mode                 -rw-r-----                  -rw-r-----
  Num Links            1                           1
  UID                  root (0)                    root (0)
  GID                  root (0)                    root (0)
* Size                 13819                       13820
* Modify Time          Sun 13 Jan 2013 09:50:00 AM JST
                                                   Sun 13 Jan 2013 04:07:32 PM JST
  Blocks               32                          32
* CRC32                DZLw4z                      BI/W5b
* MD5                  Arzkb+NG9RsbMeMoijbcZr      BOXj6YdVtiSromrvl/EuMB



--------------------------------------------------------------------------------
Rule Name: Monitor Filesystems (/home)
Severity Level: 0
--------------------------------------------------------------------------------
  ----------------------------------------
  Added Objects: 1
  ----------------------------------------

Added object name:  /home/test/twpol.txt.bak

  Property:            Expected                    Observed
  -------------        -----------                 -----------
* Object Type          ---                         Regular File
* Device Number        ---                         2050
* Inode Number         ---                         939253
* Mode                 ---                         -rw-r-----
* Num Links            ---                         1
* UID                  ---                         root (0)
* GID                  ---                         root (0)
* Size                 ---                         13715
* Modify Time          ---                         Thu 03 Jan 2013 10:12:35 PM JST
* Blocks               ---                         32
* CRC32                ---                         CsX19s
* MD5                  ---                         DL30rgh65sv60UpWFXtLYo
```

```
  ---------------------------------------
  Modified Objects: 1
  ---------------------------------------

Modified object name:  /home/test

  Property:            Expected                    Observed
  -------------        ------------                ------------
  Object Type          Directory                   Directory
  Device Number        2050                        2050
  Inode Number         651527                      651527
  Mode                 drwx------                  drwx------
  Num Links            27                          27
  UID                  test (500)                  test (500)
  GID                  test (500)                  test (500)
  Size                 4096                        4096
* Modify Time          Sun 13 Jan 2013 08:41:18 AM JST
                                                   Sun 13 Jan 2013 04:07:05 PM JST
  Blocks               8                           8



================================================================================
Error Report:
================================================================================

No Errors

--------------------------------------------------------------------------------
*** End of report ***

Open Source Tripwire 2.4 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
[root@localhost ~]#
```

追加・変更・削除されたファイル・ディレクトリに関して、パスおよびプロパティ情報が表示されます。
変更となったプロパティについては、先頭に*がついた状態で表示されます。

7.2.2.6. データベースの更新

整合性チェックはベースラインデータベースと現在のシステムの状態を比較して行われます。このため、なにも実施しなければ、プログラムのアップデートなどに伴い、監視対象のファイルに対して正規のユーザが追加・変更・削除を行った場合にも、整合性チェックで変更が検知されます。

正しいファイルの更新（追加・変更・削除）を次回以降の整合性チェックで検知されないようにするためには、ベースラインデータベースの更新が必要となります。

データベースの更新は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# tripwire  --update --twrfile
/usr/local/lib/tripwire/report/localhost.localdomain-20130103-222802.twr（レポートファイルのパス）
```

または

```
[root@localhost ~]# tripwire  -m u -r
/usr/local/lib/tripwire/report/localhost.localdomain-20130103-222802.twr（レポートファイルのパス）
```

※レポートファイルのパスは、環境に応じて適切な値に置き換えてください。

コマンドを実行すると、vi エディタが起動します。

実行例

```
[root@localhost ~]# tripwire  --update --twrfile
/usr/local/lib/tripwire/report/localhost.localdomain-20130103-222802.twr
```

編集方法や、ファイルの保存方法は、通常の vi エディタと同じです。

スクロールしていくと、以下のように違反を検出したファイル・ディレクトリのパスの横に[x]がついた行が表示されます。

出力例

```
:
 (省略)
:
------------------------------------------------------------------------------
# Section: Unix File System
------------------------------------------------------------------------------


------------------------------------------------------------------------------
Rule Name: User Binaries and Libraries (/usr/local/etc)
Severity Level: 0
------------------------------------------------------------------------------

Remove the "x" from the adjacent box to prevent updating the database
with the new values for this object.

Removed:
[x] "/usr/local/etc/twpol.txt.bak"

Modified:
[x] "/usr/local/etc"
[x] "/usr/local/etc/twpol.txt"

------------------------------------------------------------------------------
Rule Name: Monitor Filesystems (/home)
Severity Level: 0
------------------------------------------------------------------------------

Remove the "x" from the adjacent box to prevent updating the database
with the new values for this object.

Added:
[x] "/home/test/twpol.txt.bak"

Modified:
[x] "/home/test"

==============================================================================
Object Detail:
==============================================================================
:
 (省略)
```

```
:
```

[x]とチェックがついたファイル・ディレクトリは、データベースに反映されるファイル・ディレクトリ（正常な運用におけるファイル変更とするもの）です。
逆に、データベースに反映したくないファイル・ディレクトリ（正常な変更か判断つかないもの）について、チェックを外し、保存します。

なお、保存時に「### Error: Report file could not be parsed. Report may be corrupt.」と表示される場合には、レポートファイル内の日付が日本語になっていることが原因の可能性があります。
export LANG=C を実行して、再度レポートファイルを作成しなおしてください。

実行例

```
[root@localhost ~]# export LANG=C
[root@localhost ~]# tripwire --check
  == 省略 ==
[root@localhost ~]# tripwire --update -r
/usr/local/lib/tripwire/report/localhost.localdomain-20130103-222802.twr
```

ファイルの保存時に、ローカルパスフレーズの入力を求められます。

実行例

```
[root@localhost ~]# tripwire --update --twrfile
/usr/local/lib/tripwire/report/localhost.localdomain-20130113-160810.twr
Please enter your local passphrase:   (ローカルパスフレーズの入力)
Wrote database file: /usr/local/lib/tripwire/localhost.localdomain.twd
[root@localhost ~]#
```

7.2.2.7. 改ざんへの対処

Tripwire の実行によりシステムへの不正な変更（ファイル・ディレクトリの追加・変更・削除）が検知された場合は、その不正な変更が行われた原因を特定した上で、正常な状態への復元作業として、バックアップデータからのリストアなどの対処を行います。

7.2.2.8. ポリシーファイルの更新

運用中に、新たなファイル・ディレクトリをチェック対象に追加したい、チェックする内容（プロパティ）を変更したいという状況が生じた場合は、ポリシーファイルの更新を行います。ポリシーファイルの更新方法については、0 7.2.3. Tripwire の基本設定を参照してください。

7.2.3. Tripwire の基本設定

Tripwire の設定ファイルには、システム設定ファイル、および、ポリシーファイルの２つがあります。インストール時に暗号署名化された設定ファイル、および、設定ファイルの平文のコピーが生成されます。

表 7.2　Tripwire の設定ファイル

ファイル名	説明
/usr/local/etc/tw.cfg	システム設定ファイル
/usr/local/etc/twcfg.txt	システム設定ファイルのコピー（平文）
/usr/local/etc/tw.pol	ポリシーファイル
/usr/local/etc/twpol.txt	ポリシーファイルのコピー（平文）

7.2.3.1. システム設定ファイル

システム設定ファイルで設定可能な設定項目は表 7.3　Tripwire のシステム設定の通りです。

表 7.3　Tripwire のシステム設定

設定項目	初期値	説明
ROOT	/usr/local/tripwire/sbin	Tripwire のパス
POLFILE	/usr/local/tripwire/etc/tw.pol	ポリシーファイルのパス
DBFILE	/usr/local/tripwire/lib/tripwire/$(HOSTNAME).twd	データベースファイルのパス
REPORTFILE	/usr/local/tripwire/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr	レポートファイルのパス
SITEKEYFILE	/usr/local/tripwire/etc/site.key	サイトキーファイルのパス
LOCALKEYFILE	/usr/local/tripwire/etc/localhost.localdomain-local.key	ローカルキーファイルのパス
EDITOR	/bin/vi	エディタの指定
TMPDIRECTORY	/tmp　*1	一時ファイルの書き込み先ディレクトリのパス
GLOBALEMAIL	none　*1	レポートの送信先メールアドレス
LATEPROMPTING	False	メモリにパスフレーズが保持さ

		れる時間を短くする場合は、「true」を設定する。
LOOSEDIRECTORYCHECKING	False	ファイルを追加・削除した際のレポート内容の設定 false：ファイル・ディレクトリ両方の変更をレポート true：ファイルの変更のみレポート
REPORTLEVEL	3	レポートレベルのデフォルト値（0～4）
SYSLOGREPORTING	False	データベース初期化・整合性チェック完了・データベース更新・ポリシー更新時にシスログにメッセージを送信するかどうかの設定
MAILMETHOD	SENDMAIL	メールを送信する際のプロトコル（SENDMAIL/SMTP）
SMTPHOST	mail.domain.com *1	SMTP サーバアドレス ※MAILMETHOD が SMTP の場合に指定
SMTPPORT	25 *1	SMTP ポート番号 ※MAILMETHOD が SMTP の場合に指定
MAILPROGRAM	/usr/sbin/sendmail -oi -t	メール送信に利用するプログラムのパス ※MAILMETHOD が SENDMAIL の場合に指定
EMAILREPORTLEVEL	3	メールで送信するレポートレベルのデフォルト値（0～4）
MAILNOVIOLATIONS	True	整合性チェックでルール違反が検出されない時もメールを送信するかどうかの設定

*1　インストール時に生成される twcfg.txt に記載のない項目

7.2.3.2. システム設定ファイルの内容表示・編集

システム設定ファイルは署名暗号化されているため、そのまま内容を表示・編集することができません。

システム設定ファイルの内容の表示は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# twadmin --print-cfgfile
```

または

```
[root@localhost ~]# twadmin -m f
```

システム設定ファイルを編集する場合は、以下のようにシステム設定ファイルの内容をテキストファイルに出力し、出力したテキストファイルを編集します。

実行例

```
[root@localhost ~]# twadmin --print-cfgfile > /tmp/twcfg.txt（出力先ファイルのパス）
```

※出力先ファイルのパスは、環境に応じて適切な値に置き換えてください。

7.2.3.3. システム設定ファイルの作成・更新

テキストファイルを元に設定ファイルの更新は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# twadmin --create-cfgfile --site-keyfile /usr/local/etc/site.key（サイトキーファイルのパス）  /tmp/twcfg.txt（テキストのシステム設定ファイルのパス）
```

または

```
[root@localhost ~]# twadmin -m F -S /usr/local/etc/site.key（サイトキーファイルのパス）
/tmp/twcfg.txt（システム設定ファイル（平文）のパス）
```

※システム設定ファイル（平文）のパスは、環境に応じて適切な値に置き換えてください。

コマンド実行時に、サイトパスフレーズの入力を求められます。

実行例

```
[root@localhost ~]# twadmin --create-cfgfile --site-keyfile /usr/local/etc/site.key
/tmp/twcfg.txt
Please enter your site passphrase:（サイトパスフレーズを入力）
Wrote configuration file: /usr/local/etc/tw.cfg
[root@localhost ~]#
```

7.2.3.4. ポリシーファイル

ポリシーファイルとは、検査対象（ファイル・ディレクトリ）と検査内容を定義したファイルです。

ポリシーファイルのルールの記述形式を以下に示します。

```
検査対象オブジェクト名 -> プロパティマスク（属性 1,属性 2,・・・）；
```

または

```
(
属性 1,属性 2,・・・
)
{
検査対象オブジェクト名 -> プロパティマスク;
検査対象オブジェクト名 -> プロパティマスク;
：
}
```

検査対象オブジェクト名には、検査対象とするファイルまたはディレクトリの絶対パスを記述します。プロパティマスクには、チェックを行う内容、または、チェックを行わない内容を記述します。属性には、ルールの名前や違反が検知された場合の送信先メールアドレスなどを記述します。

ルールの例は以下の通りです。#から行末までの部分はコメントとして扱われます。

編集例

```
# サンプルルール
(
rulename = “Sample”,
)
{
/tmp/sample.txt -> +p;  # /tmp/sample.txt に対して、ファイルのアクセス権を検査
}
```

7.2.3.5. 検査対象オブジェクト名

検査対象オブジェクト名にディレクトリのパスを指定した場合は、配下のファイル・ディレクトリに対してプロパティマスクで指定したチェックが行われます。

ファイルのパスを記述した場合、記述したファイルに対してのみプロパティマスクで指定したチェックが行われます。

また、特定のファイルまたはディレクトリを検査対象外としたい場合は、検査対象オブジェクト名の前に!をつけます。

```
# /etc 配下のすべてのファイルに対してプロパティマスク mask1 での検査を行う。
# ただし、/etc/passwd に対してはプロパティマスク mask2 での検査を行う。
# また、/etc/rc.d に対しては検査を行わない。
mask1 = +pug;
mask2 = +pugsM;
/etc         -> $(mask1);
/etc/passwd -> $(mask2);
!/etc/rc.d;
```

7.2.3.6. プロパティマスク

プロパティマスクは、検査内容を表す文字（プロパティ）の集合です。プロパティの一覧は、表 7.4　プロパティの一覧の通りです。

プロパティの前に+があれば検査を行い、プロパティの前に-があれば検査をスキップします。プロパティの前に+-いずれも存在しない場合は検査を行います。

表 7.4　プロパティの一覧

プロパティ	説明
a	アクセス時刻
b	割り当てられたブロック数
c	i ノード作成/修正時刻
d	i ノードが登録されるデバイス ID
g	オーナーのグループ ID
i	i ノード番号
l	サイズの大きくなるファイル
m	修正時刻
n	リンク数
p	パーミッション
r	i ノードで指定されるデバイス ID
s	ファイルサイズ
t	ファイルタイプ
u	オーナーのユーザ ID
C	CRC-32 ハッシュ値

H	Haval ハッシュ値
M	MD5 ハッシュ値
S	SHA ハッシュ値

Tripwire のインストール時に生成されるファイル（/usr/local/etc/twpol.txt）では、いくつかのプロパティマスクが変数として定義されています。(表 7.5　定義済みプロパティマスク参照)

例えば、Dynamic は、p,i,n,u,g,t,d に関して検査を行い、s,r,l,b,a,m,c,C,M,S,H に関しては検査を行いません。従って、パーミッションやオーナーの改ざんについては検知できますが、ファイルの内容やサイズの変更については検知できません。

表 7.5　定義済みプロパティマスク

変数	値（プロパティマスク）	使用用途など
Device	+pugsdr-intlbamcCMSH	デバイスファイル
Dynamic	+pinugtd-srlbamcCMSH	動的に変化するファイル
Growing	+pinugtdl-srbamcCMSH	時間とともにサイズが大きくなるファイル
IgnoreAll	-pinugtsdrlbamcCMSH	存在のみをチェックすればよいファイル
IgnoreNone	+pinugtsdrbamcCMSH-l	すべてのプロパティをチェックするファイル
ReadOnly	+pinugtsdbmCM-rlacSH	読み取り専用ファイル
Temporary	+pugt	一時ファイル

定義済みプロパティマスクを活用し、ポリシーファイルを簡単に作成することが可能です。

例えば、WEB サーバの場合、HTML ファイルや画像ファイルなど静的コンテンツのプロパティマスクを$(ReadOnly)とし、ファイルのパーミッションやファイルの内容の変更を検査することで、WEB サイトの改ざんを検知可能となります。

ログファイルなど運用中に更新が発生するファイルについては、ファイルの内容が改ざんされたことを検知することはできません。サイズが小さくなることはなく大きくなるだけであればプロパティマスクを$(Growing)とするなど、ファイルの特性に応じたプロパティマスクを設定します。

7.2.3.7. 属性

属性の適用範囲はルールの記述形式により異なります。

以下の記述形式では、属性はひとつのルールに対して適用されます。

```
検査対象オブジェクト名 -> プロパティマスク（属性名 1 = 属性値,・・・）；
```

以下の記述形式では、属性は複数のルールに対して適用されます。

```
(
属性名 1 = 属性値,属性名 2 = 属性値,・・・
)
{
検査対象オブジェクト名 -> プロパティマスク;
検査対象オブジェクト名 -> プロパティマスク;
：
}
```

属性として、使用可能なものは、表 7.6 の通りです。

表 7.6　属性一覧

属性	説明
rulename	ルールに名前をつけます。 つけた名前は、整合性チェックや整合性チェック結果のレポートの中で使用されます。
emailto	整合性チェック結果のメール送信先を指定します。 --email-report オプション付きで整合性チェックを実施した際に使用されます。 アドレスを複数指定する場合は、emailto=”メールアドレス A;メールアドレス B”と指定します。
severity	ルールに重要度(0～1,000,000)をつけます。重要度のデフォルト値は 0 です。 整合性チェックを実施する際、ある重要度以上のルールのみを対象とすることが可能です。
recurse	ディレクトリを何階層下まで再帰的に検査するかを指定します。 true、false、または、-1～1,000,000 の数字を指定します。デフォルト値は true です。 true（または-1）：ディレクトリ配下のすべてのファイル、サブディレクトリが検査対象となります。 false（または 0）：ディレクトリのみが検査対象となります。

7.2.3.8. ポリシーファイルの内容表示・編集

ポリシーファイルは署名暗号化されているため、そのまま内容を表示・編集することができません。

ポリシーファイルの内容の表示は、以下のコマンドにより行います。

実行例

```
[root@localhost ~]# twadmin --print-polfile
```

または

```
[root@localhost ~]# twadmin -m p
```

ポリシーファイルを編集する場合は、以下のようにポリシーファイルの内容をテキストファイルに出力し、出力したテキストファイルを編集します。

```
[root@localhost ~]# twadmin --print-polfile > /tmp/twpol.txt（出力先ファイルのパス）
```

※出力先ファイルのパスは、環境に応じて適切な値に置き換えてください。

7.2.3.9. ポリシーファイルの更新

テキストファイルを元にポリシーファイルの更新は、以下のコマンドにより行います。

実行例

```
[root@localhost tripwire-2.4.2.2-src]# tripwire --update-policy /tmp/twpol.txt（ポリシーファイル（平文）のパス）
```

または

```
[root@localhost tripwire-2.4.2.2-src]# tripwire -m p /tmp/twpol.txt（ポリシーファイル（平文）のパス）
```

※ポリシーファイル（平文）のパスは、環境に応じて適切な値に置き換えてください。

コマンド実行時にローカルパスフレーズ、サイトパスフレーズの入力を求められます。

実行例

```
[root@localhost tripwire-2.4.2.2-src]# tripwire --update-policy /tmp/twpol.txt
Please enter your local passphrase: （ローカルパスフレーズを入力）
Please enter your site passphrase: （サイトパスフレーズを入力）
======== Policy Update: Processing section Unix File System.
======== Step 1: Gathering information for the new policy.
The object: "/selinux" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
The object: "/var/run/vmblock-fuse" is on a different file system...ignoring.
======== Step 2: Updating the database with new objects.
======== Step 3: Pruning unneeded objects from the database.
Wrote policy file: /usr/local/etc/tw.pol
Wrote database file: /usr/local/lib/tripwire/localhost.localdomain.twd
[root@localhost ~]#
```

ポリシーファイルを更新する際、ベースラインデータベースも更新する必要があります。例えば、ポリシーファイルの変更によりファイルAをチェックするというルールが追加となったとします。ポリシーファイルを変更しただけでは、ベースラインデータベースにファイルAが存在しないため、整合性チェックでファイルAに関して、ベースラインデータベースと現在の状態を比較できなくなってしまいます。

tripwireコマンド（tripwire --update-policy）は、新しいポリシーファイルに従い、ポリシーファイルおよびベースラインデータベースの更新を行います。

7.3. Snort による侵入検知

7.3.1. Snort の概要

本節では、侵入検知ソフトウェアである Snort の概要を説明します。

Snort は、UNIX および UNIX 互換マシン、Windows マシン上で動作する、IP ネットワーク上のリアルタイムでのトラフィック解析、およびパケット収集に対応したオープンソースの侵入検知ツールです。GPL (GNU General Public License) の元に無償で利用することができます。

Snort は、アメリカの Marty Roesch 氏により、1998 年にその初期バージョンが開発されました。その後、世界中のボランティアの手により様々な改良や検出ルールセットの作成が行われ、現在も日々進化し続けています。

近年の不正アクセス事件の増加を受け、これら危険なアクセスを早期に検出することができる信頼性の高いソフトウェアとして、またルールセットのカスタマイズにより様々な運用ポリシーに適応できる柔軟性を持つソフトウェアとして多くの組織で導入されています。

7.3.2. Snort の導入

本節では、Snort の導入手順を示します。

Snort の必須ライブラリとして libpcap、PCRE、libdnet、Barnyard2、DAQ が必要となります。詳細は公式サイト　http://www.snort.org/start/requirements　を参照してください。

また、CentOS 6.3 minimal 構成（インストール時に最小構成を選択した場合）では、これらをコンパイルするのに以下のパッケージの追加インストールが必要となります。まずは CentOS のディスク から依存関係で必要なパッケージをインストールします。

yum が利用できる場合は、yum を使ってインストールしても問題ありません。

（以下は CentOS 6.3 64bit 版環境のため rpm 名等は、利用の環境に合わせて読み換えてください。）

実行例

```
[root@localhost /]# mount -r -t iso9660 /dev/cdrom /mnt
[root@localhost /]# cd /mnt/Packages/
[root@localhost /]# rpm -ihv gcc-4.4.6-4.el6.x86_64.rpm ¥
           cpp-4.4.6-4.el6.x86_64.rpm ¥
           glibc-devel-2.12-1.80.el6.x86_64.rpm ¥
           libgomp-4.4.6-4.el6.x86_64.rpm ¥
           cloog-ppl-0.15.7-1.2.el6.x86_64.rpm ¥
           glibc-headers-2.12-1.80.el6.x86_64.rpm ¥
           kernel-headers-2.6.32-279.el6.x86_64.rpm ¥
           mpfr-2.4.1-6.el6.x86_64.rpm ¥
           ppl-0.10.2-11.el6.x86_64.rpm
[root@localhost /]# rpm -ihv make-3.81-20.el6.x86_64.rpm
[root@localhost /]# rpm -ihv flex-2.5.35-8.el6.x86_64.rpm ¥
bison-2.4.1-5.el6.x86_64.rpm
[root@localhost /]# rpm -ihv gcc-c++-4.4.6-4.el6.x86_64.rpm ¥
libstdc++-devel-4.4.6-4.el6.x86_64.rpm
[root@localhost /]# rpm -ihv zlib-devel-1.2.3-27.el6.x86_64.rpm ¥
           pkgconfig-0.23-9.1.el6.x86_64.rpm
```

続けて、必須ライブラリ **Libpcap** を導入します。

Libpcap とは、ネットワークを流れるパケットを解析するためのライブラリです。

Snort ではこのライブラリを利用して、ネットワークパケットの解析を行っています。

http://www.tcpdump.org/ から Libpcap のソースコードをダウンロードしてコンパイル・インストールを行います。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget http://www.tcpdump.org/release/libpcap-1.3.0.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ tar xvzf libpcap-1.3.0.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ cd libpcap-1.3.0
[lpij@localhost tmp]$ ./configure
    ==  省略 ==
[lpij@localhost tmp]$ make
    ==  省略 ==
[lpij@localhost tmp]$ su -
パスワード:
[root@localhost ~]# cd /tmp/libpcap-1.3.0
[root@localhost libpcap-1.3.0]# make install
    ==  省略 ==
```

次に **Perl Compatible Regular Expressions** をインストールします。

Perl Compatible Regular Expressions (PCRE) とは、Perl 互換の正規表現を C 言語で実装したライブラリです。Snort だけではなく、Apache 等の様々なソフトウェアでも使われています。

http://www.pcre.org/ から tar ファイルをダウンロードします。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget ¥
http://sourceforge.net/projects/pcre/files/pcre/8.32/pcre-8.32.tar.gz/download
    ==  省略 ==
[lpij@localhost tmp]$ tar xvzf pcre-8.32.tar.gz
    ==  省略 ==
```

```
[lpij@localhost tmp]$ cd pcre-8.32
[lpij@localhost tmp]$ ./configure
    ==  省略 ==
[lpij@localhost tmp]$ make
    ==  省略 ==
[lpij@localhost tmp]$ su -
パスワード:
[root@localhost ~]# cd /tmp/ pcre-8.32
[root@localhost pcre-8.32]# make install
    ==  省略 ==
```

Libdnet とは、ネットワーク関連の操作を簡素化するライブラリです。

http://libdnet.sourceforge.net/ から tar ファイルをダウンロードします。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget http://prdownloads.sourceforge.net/libdnet/libdnet-1.11.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ tar zxvf libdnet-1.11.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ cd libdnet-1.11
[lpij@localhost libdnet-1.11]$ ./configure
    ==  省略 ==
[lpij@localhost libdnet-1.11]$ make
    ==  省略 ==
[lpij@localhost libdnet-1.11]$ su -
パスワード:
[root@localhost ~]# cd /tmp/ libdnet-1.11
[root@localhost libdnet-1.11]# make install
    ==  省略 ==
```

インストールは以上で完了ですが、この Libdnet ライブラリを Snort 利用できるようにするには、/usr/local/lib/に、.so ファイル（Unix 系 OS の共有ライブラリの拡張子が.so である）が必要となります。

執筆のバーションでは、.so ファイルは自動作成されず、ライブラリが/usr/local/lib/libdnet.1.0.1 として存在しますので、ファイル名の変更と、シンボリックリンクの作成を行い、Snort が利用できるようにします。

実行例

```
[root@localhost ~]# cd /usr/local/lib
[root@localhost lib]# cp -p libdnet.1.0.1 libdnet.so.1.0.1
[root@localhost lib]# ln -s libdnet.so.1.0.1 libdnet.so.1
[root@localhost lib]# ln -s libdnet.so.1.0.1 libdnet.so
```

Barnyard2 とは、Snort の結果を出力するためのライブラリです。Snort 単体では、結果はバイナリファイルとして出力されます。

本ライブラリは、Sort の結果データを読み込み、MySQL などのデータベースへ保存します。

今回は、データベースとの連携は行いませんが、Sort のコンパイルに必要になるため、インストール手順を紹介します。

http://www.securixlive.com/barnyard2/download.php から tar ファイルをダウンロードします。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget http://www.securixlive.com/download/barnyard2/barnyard2-1.9.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ tar zxvf barnyard2-1.9.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ cd barnyard2-1.9
[lpij@localhost barnyard2-1.9]$ ./configure
    ==  省略 ==
[lpij@localhost barnyard2-1.9]$ make
    ==  省略 ==
[lpij@localhost barnyard2-1.9]$ su -
パスワード:
[root@localhost ~]# cd /tmp/ barnyard2-1.9
[root@localhost barnyard2-1.9]# make install
    ==  省略 ==
```

DAQ とは、Snort が各種データを取得するために必要なライブラリおよびモジュールです。バージョン 2.9.0 以降の Snort では必須となります。

http://www.snort.org から tar ファイルをダウンロードします。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget http://www.snort.org/dl/snort-current/daq-2.0.0.tar.gz -O
daq-2.0.0.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ tar zxvf daq-2.0.0.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ cd daq-2.0.0
[lpij@localhost daq-2.0.0]$ ./configure
    ==  省略 ==
[lpij@localhost daq-2.0.0]$ make
    ==  省略 ==
[lpij@localhost daq-2.0.0]$ su -
パスワード:
[root@localhost ~]# cd /tmp/ daq-2.0.0
[root@localhost daq-2.0.0]# make install
    ==  省略 ==
```

続いて、インストールしたライブラリを、以下の手順でシステムに認識させます。

実行例

```
[root@localhost ~]# ldconfig -v /usr/local/lib
[root@localhost ~]# ldconfig -p | grep daq
        libdaq.so.2 (libc6) => /usr/local/lib/libdaq.so.2
        libdaq.so (libc6) => /usr/local/lib/libdaq.so
```

なお、オプションなしの ldconfig コマンドを実行した際に、自動的に/usr/local/lib をライブラリフォルダとして認識させたい場合は、/etc/ld.so.conf の設定に、/usr/local/lib を追加するようにしてください。

実行例

```
[root@localhost ~]# vi /etc/ld.so.conf
include ld.so.conf.d/*.conf
/usr/local/lib
```

最後に Snort のインストールを行います。

http://www.snort.org から tar ファイルをダウンロードします。

実行例

```
[lpij@localhost ~]$ cd /tmp
[lpij@localhost tmp]$ wget http://www.snort.org/dl/snort-current/snort-2.9.4.tar.gz  ¥
-O snort-2.9.4.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ tar zxvf snort-2.9.4.tar.gz
    ==  省略 ==
[lpij@localhost tmp]$ cd snort-2.9.4
[lpij@localhost snort-2.9.4]$ ./configure
    ==  省略 ==
[lpij@localhost snort-2.9.4]$ make
    ==  省略 ==
[lpij@localhost snort-2.9.4]$ su -
パスワード:
[root@localhost ~]# cd /tmp/ snort-2.9.4
[root@localhost snort-2.9.4]# make install
    ==  省略 ==
```

ユーザ作成

Snort が利用するユーザを作成します。

実行例

```
[root@localhost ~]# useradd snort
```

サービス設定

Snort の設定ファイルは、ソースコードのパッケージに含まれていますので、/etc/sysconfig 配下にコピーして、必要な箇所の設定を変更してください。

まず、設定ファイルをコピーします。(パッケージを/tmp に解凍したと仮定します)

実行例

```
[root@localhost ~]# cp /tmp/snort-2.9.4/rpm/snort.sysconfig /etc/sysconfig/snort
```

必要に応じて/etc/sysconfig/snort を編集します。

設定例

```
# /etc/sysconfig/snort
# $Id$

==  省略 ==

# Listen する（サービスとしてポートを開く）ネットワークインターフェイスを指定します。
INTERFACE=eth0

==  省略 ==

# snort を実行するユーザとグループを指定します。
USER=snort
GROUP=snort

==  省略 ==

# ログ出力ディレクトリを指定します。
LOGDIR=/var/log/snort

==  省略 ==
```

起動スクリプト

サービスとして自動起動させる場合は、ソースパッケージに起動スクリプトが含まれていますので、それを利用することが可能です。

実行例

```
[root@localhost ~]# cp /tmp/snort-2.9.4/rpm/snortd /etc/init.d/
[root@localhost ~]# chmod 755 /etc/init.d/snortd
```

なお、デフォルトの設定では、起動スクリプト snortd の実行パスは/usr/sbin/snort となっており、デフォルトのインストールパス/usr/local/bin/snort と違っていますので注意してください。

環境に合わせて/etc/init.d/snortd を編集する、もしくは/usr/sbin/snort から/usr/local/bin/snort へのシンボリックリンクを作成するなどの対応を行ってください。

次にルールの初期配置を行います。

ルールファイルは、どのようなパターンを侵入として検知するかを定義しているファイルです。

ルールには、様々なものがありますが、本説明においては、公式サイトで紹介している無料で使えるコミュニティ版のルールを適用する方法について説明します。

ルールの初期配置

実行例

```
[root@localhost ~]# cd /tmp
[root@localhost tmp]# wget
https://s3.amazonaws.com/snort-org/www/rules/community/community-rules.tar.gz
    ==  省略 ==
[root@localhost tmp]# tar zxvf community-rules.tar.gz
    ==  省略 ==
[root@localhost tmp]# cd community-rules

[root@localhost community-rules]# mkdir /etc/snort
[root@localhost community-rules]# cp sid-msg.map /etc/snort/
[root@localhost community-rules]# mkdir /etc/snort/rules
[root@localhost community-rules]# cp community.rules /etc/snort/rules/
```

次に、その他、snort の動作に必要な設定ファイルなどをソースの解凍ディレクトリから、/etc/snort 配下にコピーします。

実行例

```
[root@localhost ~]# cd /tmp/snort-2.9.4/etc
[root@localhost etc]# cp attribute_table.dtd¥
 classification.config¥
 gen-msg.map¥
 reference.config¥
 snort.conf¥
 threshold.conf¥
 unicode.map /etc/snort/
[root@localhost etc]# chown -R snort.snort /etc/snort
```

次に、ログなど snort の動作に必要な、ディレクトリを作成します。

実行例

```
ライブラリが利用するディレクトリの作成
[root@localhost ~]# mkdir /usr/local/lib/snort_dynamicrules

ログ領域の作成（/etc/sysconfig/snort で設定したディレクトリ）
[root@localhost ~]# mkdir /var/log/snort
[root@localhost ~]# chown -R snort.snort /var/log/snort
```

次にログローテーションの設定を行います。ソースパッケージに含まれるファイルを設置します。

デフォルトでは、/var/log/snort 配下のログファイルがローテーション対象となっています。必要に応じて、/etc/sysconfig/snort で設定したディレクトリへ変更してください。

実行例

```
[root@localhost ~]# cp /tmp/snort-2.9.4/rpm/snort.logrotate /etc/logrotate.d/snort
```

7.3.3. Snort の基本設定

本節では、Snort の基本設定を説明します。ここでは Snort を動作させることを目標とし、独自ルールの設定や設定項目の詳細説明は割愛します。

ディレクトリ構成一覧

ファイル名	内容
/etc/sysconfig/snort	サービス設定
/etc/init.d/snortd	起動スクリプト
/etc/snort	Snort 関連設定
/etc/snort/snort.conf	メイン設定ファイル
/etc/snort/attribute_table.dtd	属性の定義
/etc/snort/classification.config	クラスの定義。system-call-detect や web-application-attack といったグループ分けと、その通知レベル（priority）を設定します。
/etc/snort/gen-msg.map	メッセージ定義
/etc/snort/reference.config	外部 URL の参照設定。ルール中で外部 URL を参照する際に指定します。
/etc/snort/threshold.conf	通知制限設定。指定した時間内で同じアラートを何度も通知しないといった設定で通知を抑制できます。
/etc/snort/unicode.map	Unicode の変換マップ
/etc/snort/rules	ルール設定
/etc/snort/so_rules	動的ルール設定
/usr/local/lib	追加した関連ライブラリ
/usr/local/lib/snort	Snort のライブラリ
/usr/local/lib/snort_dynamic*	動的なルール実行に関するライブラリ
/etc/logrotate.d/snort	ログローテーション設定
/var/log/snort	ログ領域
/var/log/snort/alert	アラートログ
/var/log/snort/snort.log.*	検査ログ

/etc/snort/snort.conf を編集して、システム設定を行います。

設定例

```
[root@localhost ~]# vi /etc/snort/snort.conf

# HOME, EXTERNAL などのネットワークを設定します。
ipvar HOME_NET 192.168.0.0/24
ipvar EXTERNAL_NET any

# 以下の変数の値を修正します。パスを親ディレクトリからの相対パスではなく、カレントディレクトリ# からの相対パスとします。(../xxx を./xxx とします。)
var RULE_PATH ./rules
var SO_RULE_PATH ./so_rules
var PREPROC_RULE_PATH ./preproc_rules
var WHITE_LIST_PATH ./rules
var BLACK_LIST_PATH ./rules

# 読み込むルールの設定を行います。
# ファイル最下部の include 行で読み込むファイルを指定します。
# デフォルトで用意されているルールは存在しないためコメントアウトします。
# 先ほどダウンロードしたコミュニティルールを追加します。
include $RULE_PATH/local.rules
include $RULE_PATH/community.rules

#include $RULE_PATH/app-detect.rules
  ==  省略 ==
#include $RULE_PATH/web-php.rules
#include $RULE_PATH/x11.rules
```

次に/etc/snort/rules/local.rules を編集してルールの設定を行います。

動作確認のためにすべての PING を検知させるルールを追加します。

設定例

```
[root@localhost ~]# vi /etc/snort/rules/local.rules
alert icmp any any -> any any (msg: "ICMP Packet detected"; sid:999999;)
```

ここまでの作業が完了したら、デーモンを起動して、動作確認を行います。

実行例

```
[root@localhost ~]# /etc/init.d/snortd start
```

なお、サービス起動に失敗する場合は、/var/log/messages にエラーメッセージが出力されていないかを確認してください。

以下は、local.rules ファイルが存在しない場合のエラーメッセージです。

出力例

```
FATAL ERROR: /etc/snort/rules/local.rules(0) Unable to open rules file
"/etc/snort/rules/local.rules": No such file or directory.

  local.rules ファイルが存在しない
```

大量のログが出力され、起動できない場合やログ出力されない場合があれば、rsyslog の設定を変更します。

実行例

```
[root@localhost ~]# vi /etc/rsyslog.conf
2 分間で 500 回まで出力を許可する場合は以下の通りです。
$SystemLogRateLimitInterval 2
$SystemLogRateLimitBurst 500

[root@localhost ~]# /etc/init.d/rsyslog restart
```

なお、ネットワーク型 IDS では他ホスト宛のパケットを受け取る必要があるためプロミスキャスモードで動作する必要があります。

そのために、ネットワークインターフェイスの設定を以下のように設定してください。

実行例

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
PROMISC=yes

[root@localhost ~]# /etc/init.d/network restart
```

Snort が起動したら他ホストから PING を実行し、検証用のルールが動作するか確認します。

検知した場合、/var/log/snort/alert に以下のような出力が追加されます。

出力例

```
12/18-01:51:33.985838  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.1 -> 192.168.10.10
12/18-01:51:33.985900  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.10 -> 192.168.10.1
12/18-01:51:34.989393  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.1 -> 192.168.10.10
12/18-01:51:34.989442  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.10 -> 192.168.10.1
12/18-01:51:35.994032  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.1 -> 192.168.10.10
12/18-01:51:35.994066  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.10 -> 192.168.10.1
12/18-01:51:36.996208  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.1 -> 192.168.10.10
12/18-01:51:36.996251  [**] [1:999999:0] ICMP Packet detected [**] [Priority: 0] {ICMP}
192.168.10.10 -> 192.168.10.1
```

7.3.4. GUI のセットアップ

前節で Snort をセットアップし動作を確認することができましたが、大量のログをチェックするのは大変であり、検出すべき内容が大量の情報の中に埋もれてしまっては困ります。そこで GUI をセットアップし確認作業を軽減することができます。

公式ページでは、3 種類のソフトウェアが比較されています。

以下の URL にある資料で、GUI ツールの比較表がありますので参考にしてください。

http://www.snort.org/assets/187/Snort_Frontend_Compare.pdf

7.3.5.ルールのチューニング

日頃の運用として気を付けるべきことを説明します。

- **ルールの調整**

不正な通信であるのにも関わらず検知できないことをフォルス・ネガティブといいます。この反対で、本来は正常な通信であるにも関わらず検知してしまうことをフォルス・ポジティブといいます。

攻撃の手段は日々新しいものが登場するため誤検知は避けられませんが、ルールの閾値等が適切でない場合に、このような誤検知を引き起こしてしまう可能性があります。フォルス・ネガティブ、フォルス・ポジティブの割合を把握しながら、継続的にチューニングすることが重要となります。

また、あまりにも大量のログが出力されていると、悪意のある通信を見逃してしまう可能性があります。適宜ルールのチューニングを行うことが必要となります。

- **最新ルールの自動適用**

ウィルス対策として、自身の使っている PC のウィルス対策ソフトにおいて、定期的にパターンファイルをアップデートしていると思います。

Snort においても同様で、新種の攻撃を検知するためにはルールを最新化する必要があります。ここでは、ルールを定期的に自動アップデートするツール Oinkmaster とその設定について説明します。

事前準備

Oinkmaster は root ユーザでは実行できません。

snort ユーザを使うか、oinkmaster 用のユーザを用意してください。

ルールのアップデート時、Snort のルールを格納するディレクトリにアクセスするため oinkmaster ユーザを作成する場合は snort グループへ所属させてください。

実行例

```
[root@localhost ~]# useradd oinkmaster -g snort

バックアップ用ディレクトリを作成します。
[root@localhost ~]# mkdir /etc/snort/rules_backup
[root@localhost ~]# chown snort.snort /etc/snort/rules_backup
```

必須パッケージの導入

Oinkmaster は tar, gzip, wget コマンドが導入されている必要があります。CentOS のディスクや yum を利用し、インストールしてください。

インストール

http://oinkmaster.sourceforge.net/download.shtml から tar ファイルをダウンロードしてください。

実行例

```
[root@localhost ~]# tar zxvf oinkmaster-2.0.tar.gz
[root@localhost ~]# cd oinkmaster-2.0

実行ファイルの設置
[root@localhost oinkmaster-2.0]# cp -p oinkmaster.pl /usr/local/bin/
[root@localhost oinkmaster-2.0]# chown oinkmaster.snort /usr/loca/bin/oinkmaster.pl

設定ファイルの設置（以下に設置すればデフォルトで参照されます）
[root@localhost oinkmaster-2.0]# cp -p oinkmaster.conf /usr/local/etc/
[root@localhost oinkmaster-2.0]# chown oinkmaster.snort /usr/loca/etc/oinkmaster.conf

マニュアルファイルの設置
[root@localhost oinkmaster-2.0]# cp oinkmaster.1 /usr/local/man/man1/
```

設定

実行例

```
[root@localhost ~]# vi /usr/local/etc/oinkmaster.conf

最新のルールをダウンロードするための URL を設定します。
url = https://s3.amazonaws.com/snort-org/www/rules/community/community-rules.tar.gz

なお、設定ファイルや公式サイトに記載のある以下の URL は、執筆時点では無効な URL となっています。
http://www.snort.org/pub-bin/downloads.cgi/Download/comm_rules/Community-Rules.tar.gz
```

このままでは、自分で設定した内容が上書きされてしまいます。

解凍したディレクトリに含まれる template-examples.conf に記載のある通り、ファイルや SID 単位（ルール単位）で、スキップ（最新ファイルを適用しない）や、内容の置換ができます。詳細は割愛しますが、例を参考に設定を行います。

実行（動作確認）

実行例

```
[root@localhost ~]# /usr/bin/perl /usr/local/bin/oinkmaster.pl -o /etc/snort/rules -b
/etc/snort/rules_backup -C /usr/local/etc/oinkmaster.conf

/usr/local/bin/oinkmaster.pl -h でオプションの説明が確認できます。
-o 出力先
-b バックアップ先
-C 設定ファイル
-c 更新を反映せずに実行します（設定を試す場合などで利用します。）
```

最後に、これを自動実行するために cron へ登録します。

実行（cron による自動実行）

実行例

例）oinkmaster ユーザで毎日 1 時に実行します。

```
[root@localhost ~]# crontab -e -u oinkmaster
0 1 * * * /usr/bin/perl /usr/local/bin/oinkmaster.pl -o /etc/snort/rules -b
/etc/snort/rules_backup -C /usr/local/etc/oinkmaster.conf  2>&1 > /dev/null
```

- **その他のツール**

  運用を助けるツールが公式サイトで紹介されていますので、参照してください。

  http://www.snort.org/snort-downloads/additional-downloads/

8. セキュリティスキャンツールによる脆弱性のチェック

8.1. システムの脆弱性のチェックについて

不正アクセスやコンピュータウィルスの攻撃、操作ミスなどにより、アプリケーションやシステムが意図しない動作をしてしまい、結果として組織の情報資産（個人情報など）の漏洩やシステムの機能や性能が損なわれる原因となり得るセキュリティ上の問題点のことを、システムの脆弱性と呼びます。

脆弱性対策とは、使用している OS やミドルウェア、アプリケーションの脆弱性に関する最新情報の収集や診断を行うことにより、脆弱性を検出し、適切に対策を講じることです。脆弱性対策を適切に行わない場合、情報流出事故やホームページの不正侵入事件など予期せぬ情報セキュリティ事故に巻き込まれ、取り返しのつかない問題に発展する危険性がありますので、脆弱性の対策は非常に重要です。

本節では脆弱性のチェックとして非常に有効なセキュリティスキャンツールである nmap および Nessus を紹介します。

（注意事項）

セキュリティスキャンツールを利用することで、すべての脆弱性がなくなったことが保証できるわけではありません。OS やミドルウェア、アプリケーションには未知の脆弱性が存在し、脆弱性に関する情報は日々アップデートされています。利用するセキュリティスキャンツールを最新に保ち、定期的な脆弱性チェックを行なってください。

8.2. nmap によるセキュリティスキャン

8.2.1. nmap の概要

nmap は、UNIX および UNIX 互換マシン、Windows マシン上で動作する、IP ネットワーク上の機器に対するポートスキャン機能、およびポート情報からサービスアプリケーションの種類やバージョンの検出、OS の種類の検出を可能とするオープンソースのセキュリティスキャンツールです。GPL（GNU General Public License）の元に無償で利用することができます。

現在ではほとんどの Linux ディストリビューションに収録されており、最新バージョンやソースコードは http://nmap.org/ より入手できます。

8.2.2. nmap の実行環境

セキュリティスキャンを有効に行うためには、ネットワーク構成と目的に応じて nmap の設置環境を検討する必要があります。

例）

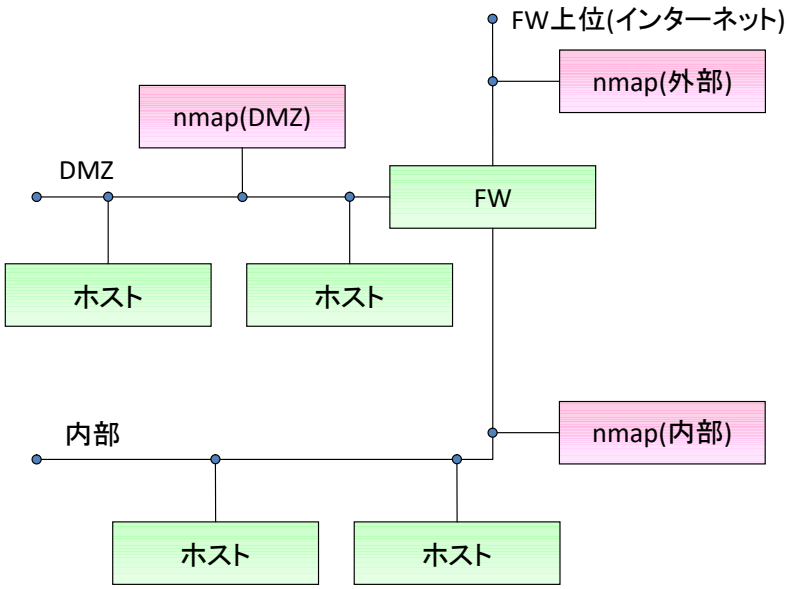


図 0.1 ネットワーク内における nmap の設置例

- FW 上位に nmap を配置することで、外部に公開されたホストおよびサービスを調査することが可能です。これは、外部からの脅威を想定した脆弱性のチェックとして有効です。
- DMZ 上に nmap を配置することで、DMZ 上のホストおよびサービスの調査、DMZ から内部にアクセス可能なホストおよびサービスの調査が可能です。これは、外部に公開している DMZ 上のホストが不正

侵入された場合の外部からの脅威や、DMZ 上のホストからの操作ミスといった内部からの脅威を想定した脆弱性のチェックとして有効です。

- 内部ネットワークに nmap を配置することで、内部のホストおよびサービスの調査が可能です。これは、内部のホストからの操作ミスなどの内部からの脅威を想定した脆弱性のチェックとして有効です。

8.2.3. nmap のインストール

本手順では、CentOS 6.3 に収録されている nmap-5.51-2 を利用します。

実行例

```
[root@localhost disk]# rpm -ivh nmap-5.51-2.el6.x86_64.rpm
Preparing...                ########################################### [100%]
   1:nmap                   ########################################### [100%]
```

8.2.4. nmap によるネットワークスキャンの実行

nmap は、複数のコマンドオプションが用意されています。詳細なコマンドオプションについては、nmap(1)の manpage を参照してください。

1）ネットワーク上で起動しているホストの調査

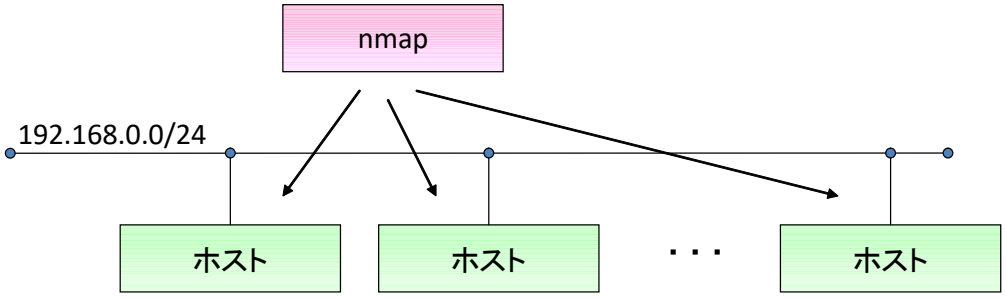


図 0.2 nmap における起動しているホストの調査

以下のコマンドでは、192.168.198.0/24 のネットワーク上で起動しているホストを調査しています。

実行例

```
[lpij@localhost ~]$ nmap -sn 192.168.198.0/24

Starting Nmap 5.51 ( http://nmap.org ) at 2013-01-13 05:36 PST
Nmap scan report for 192.168.198.1
Host is up (0.00015s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.198.2
Host is up (0.00010s latency).
MAC Address: 00:50:56:E3:64:AD (VMware)
Nmap scan report for 192.168.198.129
Host is up.
Nmap scan report for 192.168.198.254
Host is up (0.00011s latency).
MAC Address: 00:50:56:FA:D3:1F (VMware)
Nmap done: 256 IP addresses (4 hosts up) scanned in 5.38 seconds
```

上記の例においては、実行結果より、192.168.198.1, 192.168.198.2, 192.168.198.129, 192.168.198.254 の 4 ホストが動作していることが確認されました。

2）ホストで動作しているサービスの調査

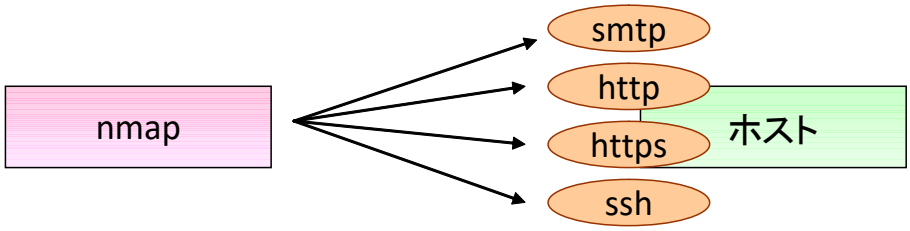


図 0.3 nmap における動作しているサービスの調査

以下のコマンドでは、特定ホスト(192.168.198.129)の TCP および UDP の全ポート(0-65535)に対しスキャンを実施し、起動しているサービスを調査しています。

実行例

```
[lpij@localhost ~]$ nmap -sT -sU -p 0-65535 192.168.198.129

Starting Nmap 5.51 ( http://nmap.org ) at 2013-01-13 05:58 PST
Nmap scan report for 192.168.198.129
Host is up (0.00045s latency).
Not shown: 131066 closed ports
PORT      STATE         SERVICE
22/tcp    open          ssh
25/tcp    open          smtp
80/tcp    open          http
68/udp    open|filtered dhcpc
5353/udp  open|filtered zeroconf
34224/udp open|filtered unknown

Nmap done: 1 IP address (1 host up) scanned in 6.53 seconds
```

実行結果より、このホストでは 3 つの TCP サービス(ポート番号：22,25,80)および 3 つの UDP サービス（ポート番号：68,5353,3422）が動作していることが確認されました。

セキュリティスキャンの実行結果から、想定していないホストやサービスが公開されていないか確認してください。本ツールを利用することで、想定していなかった FW の設定ミスやホストのサービス起動ミスによる、不要なポートの開放を検出することができるでしょう。

Nessus によるセキュリティスキャン

8.3.1. Nessus の概要

Nessus は、UNIX および UNIX 互換マシン、Windows マシン上で動作する、IP ネットワーク上で動作しているサービスに対する既知のアプリケーション脆弱性検査、リモートホストが利用している OS やソフトウェアのバージョンに既知の脆弱性検査を可能とするセキュリティスキャンツールです。以前は GPL (GNU General Public License) に基づいたオープンソースソフトウェアでしたが、現在は TENABLE Network Security 社が所有しており、非商用環境でのみ無償で利用することができます。

最新バージョンは、http://www.nessus.org/ より入手できます。

8.3.2. Nessus の実行環境

脆弱性スキャンを有効に行うためには、ネットワーク構成と目的に応じて Nessus の設置環境を検討する必要があります。

例)

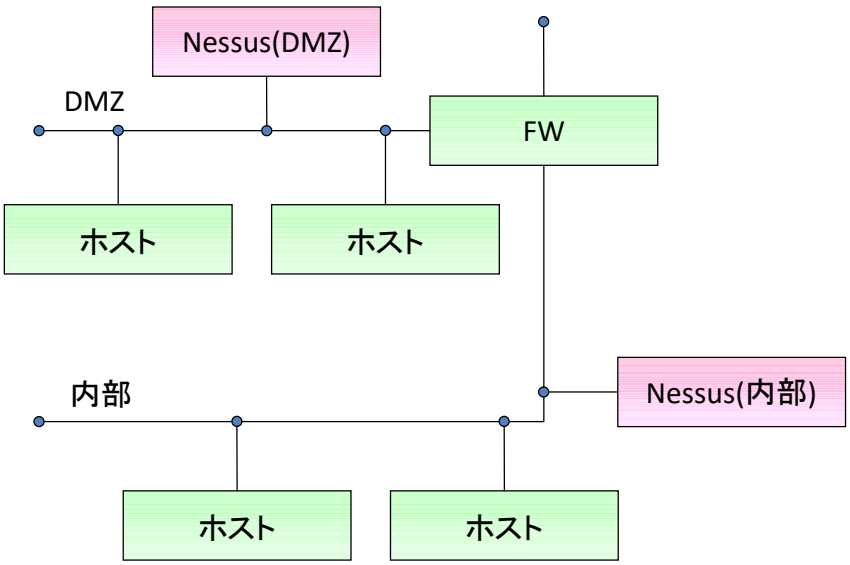


図 0.4 ネットワーク内における Nessus の設置例

- DMZ 上に Nessus を配置することで、DMZ 上のホストにおける脆弱性の調査が可能です。DMZ 上のホストにて検出された脆弱性は、内部ネットワーク内のホストと比較してより慎重に対応すべきです。
- 内部ネットワークに Nessus を配置することで、内部のホストにおける脆弱性の調査が可能です。

8.3.3. Nessus のインストール

本手順では、http://www.nessus.org/ より Nessus-5.0.2 をダウンロードし、利用します。

① Nessus パッケージのインストール

実行例

```
[root@localhost disk]# rpm -ivh Nessus-5.0.2-es6.x86_64.rpm
Preparing...                ########################################### [100%]
   1:Nessus                 ########################################### [100%]
nessusd (Nessus) 5.0.2 [build R23205] for Linux
(C) 1998 - 2012 Tenable Network Security, Inc.

Processing the Nessus plugins...
[##########################################################]

All plugins loaded
 - You can start nessusd by typing /sbin/service nessusd start
 - Then go to https://hostname:8834/ to configure your scanner
```

② Nessus サービスの起動

実行例

```
[root@localhost disk]# /etc/init.d/nessusd start
Starting Nessus services:                                  [  OK  ]
```

③ ブラウザより https://hostname:8834 へアクセスし、[Get started]を選択します。
※hostname 部分は、インストールする環境に応じて適切な値に置き換えてください。

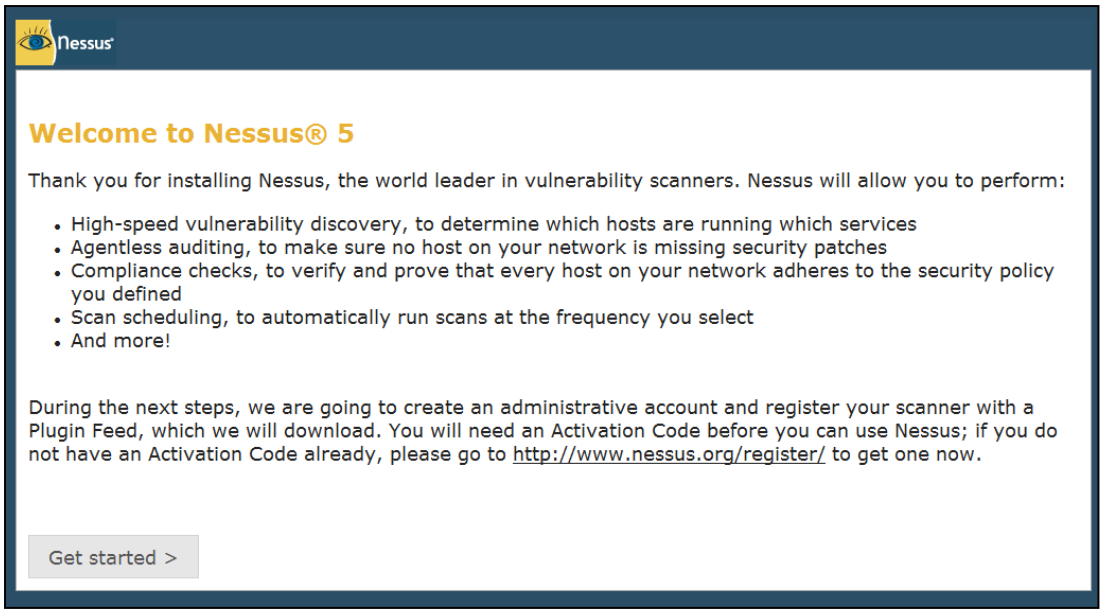


図 0.5 Nessus の設定ウィザード起動画面

④ Nessus アカウントのセットアップ

アカウントおよびパスワードを入力し、[next >]を選択します。

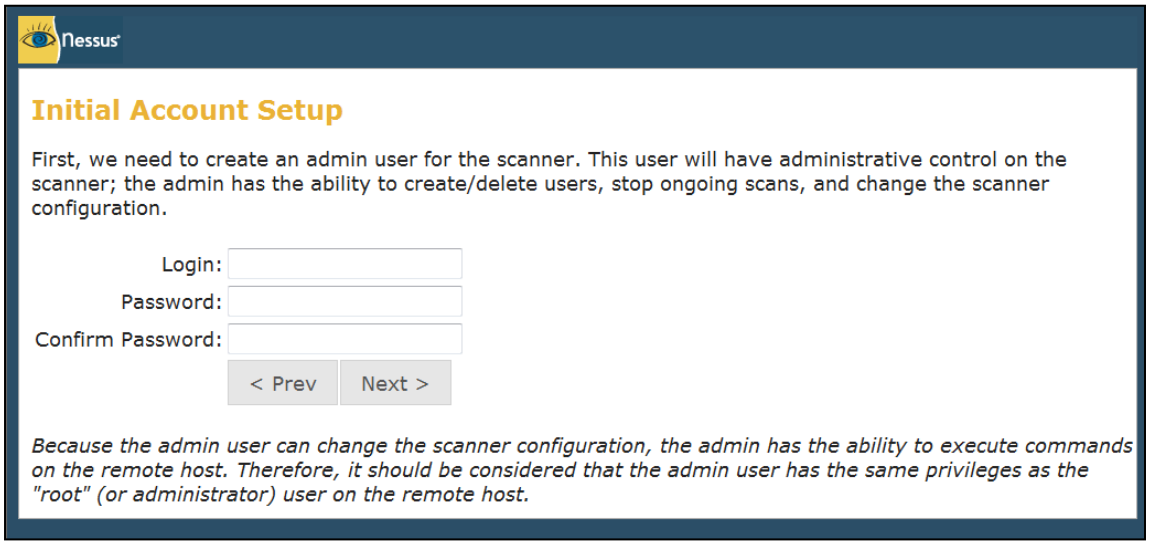


図 0.6 Nessus の設定ウィザード（アカウントセットアップ）

⑤ アクティベーションコード入力およびプラグインのセットアップ

アクティベーションコードは、http://www.nessus.org/register/ より入手することができます。Nessus インストール環境がインターネット接続可能な場合、入手したアクティベーションコードを入力し、[Next >]を選択します。また、Nessus インストール環境がインターネット接続不可の場合、[offline]を入力し、[Next >]を選択します。

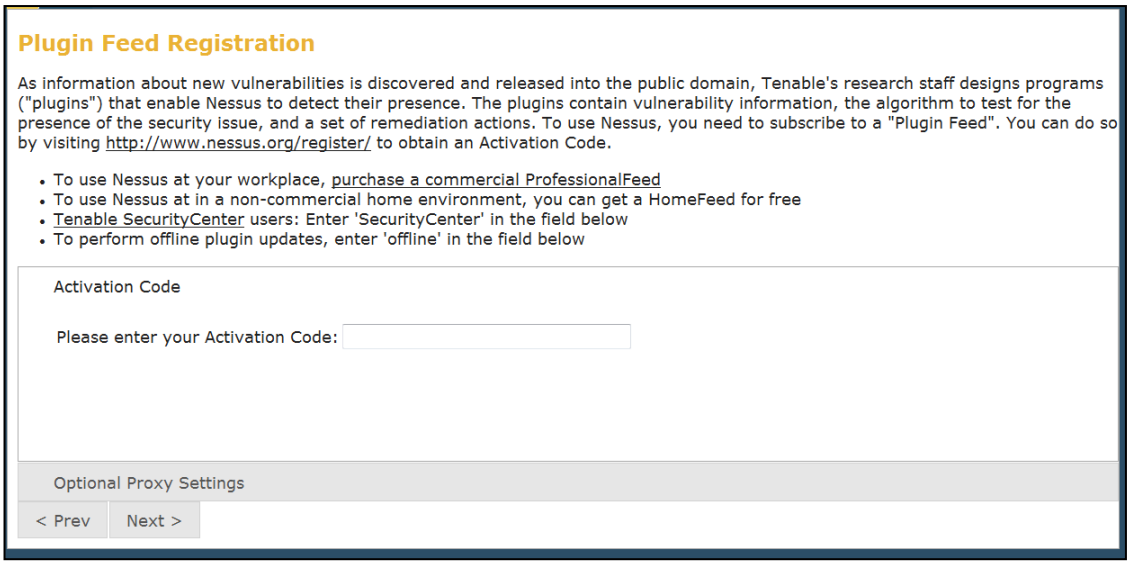


図 0.7 Nessus の設定ウィザード（アクティベーションコード入力）

- Nessus インストール環境がインターネット接続可能な場合

アクティベーションが完了すると以下の表示となりますので、[Next: Download plugins >]を選択します。

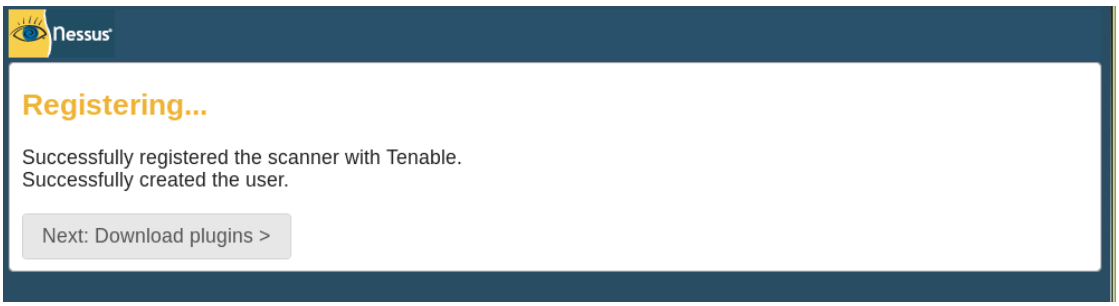


図 0.8 Nessus の設定ウィザード（アクティベーション完了後の画面）

プラグインのダウンロードおよびセットアップが開始されます。

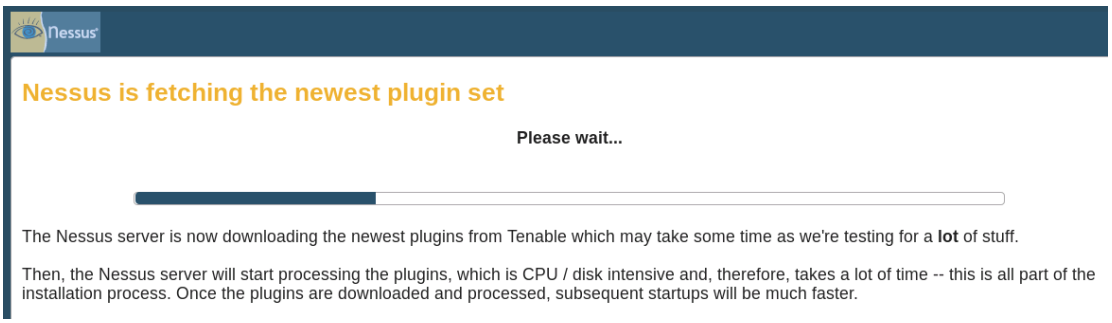


図 0.9 Nessus の設定ウィザード（セットアップ実行画面）

プラグインのセットアップが完了すると、ログイン画面に遷移しますので、手順 ⑨(セットアップ完了)に進んでください。

- Nessus インストール環境がインターネット接続不可の場合

[Next: Download plugins >]を選択します。

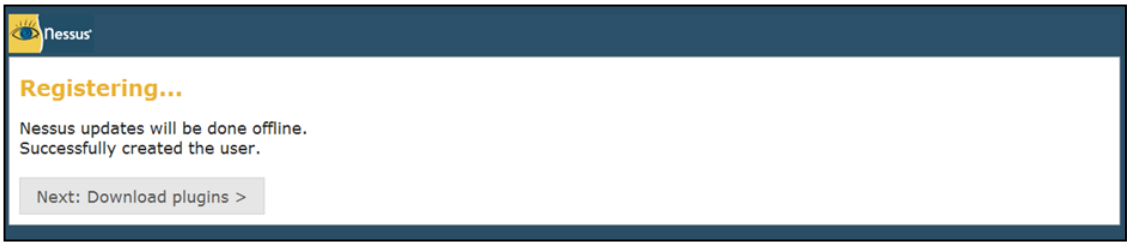


図 0.10 Nessus の設定ウィザード（オフライン時に表示される画面）

[No plugins are installed]と表示されます。

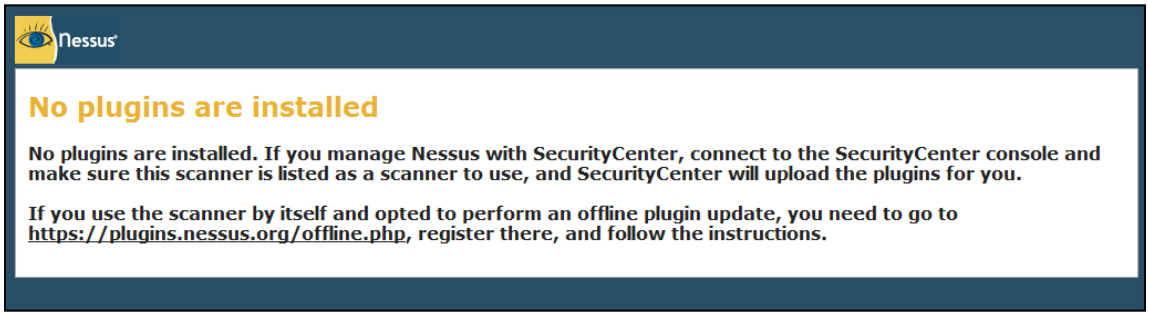


図 0.11 Nessus の設定ウィザード（プラグインがインストールできない時の画面）

インターネット接続不可のため、プラグインは手動でセットアップする必要があります。手順 No. ⑥、⑦、⑧に従って、Challenge code の取得およびプラグインのダウンロード、セットアップを行ってください。

⑥　Challenge code の取得

インターネット接続が不可能な環境の場合、プラグインをダウンロードするために必要な Challenge code を取得します。

実行例

```
[lpij@localhost ~]$ /opt/nessus/bin/nessus-fetch --challenge

Challenge code: 7e6ae00c9364aab49a90d0a5013e428a54712b70

You can copy the challenge code above and paste it alongside your
Activation Code at:
https://plugins.nessus.org/offline.php
```

⑦　プラグインのダウンロード

https://plugins.nessus.org/offline.php へアクセスし、Challenge code およびアクティベーションコード(※)を入力します。

※アクティベーションコードは、以下より取得可能です。(ホームユース向け)

http://www.tenable.com/products/nessus/nessus-homefeed

氏名およびメールアドレスを登録することで、アクティベーションコードが登録したメールアドレスに送信されます。

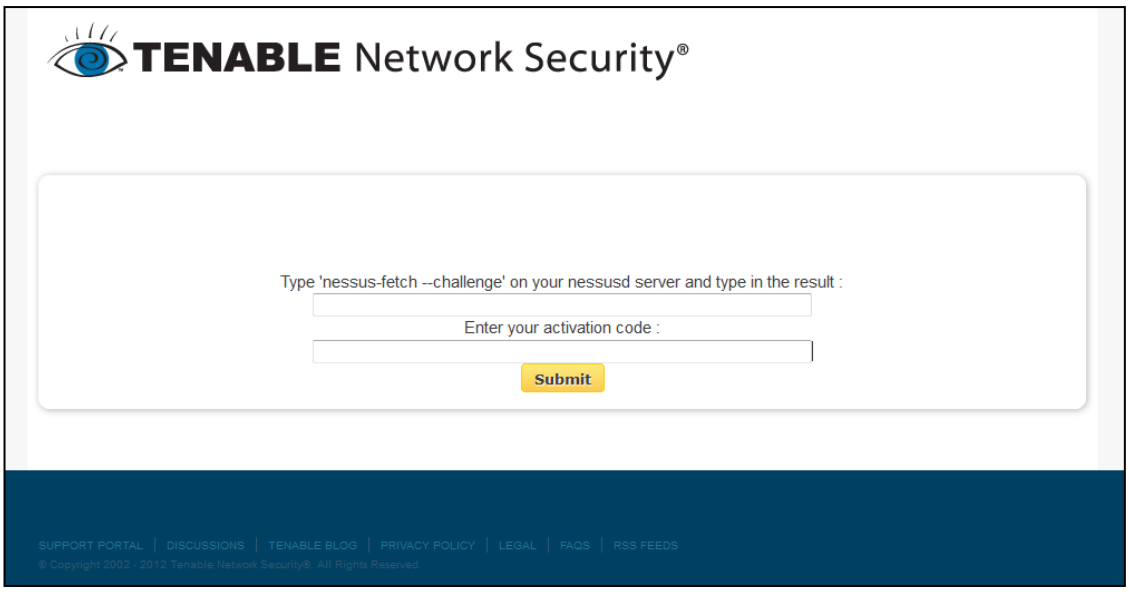


図 0.12 WEB からのチャレンジコード取得（入力画面）

ダウンロードリンクが表示されますので、プラグイン圧縮ファイル(例. all-2.0.tar.gz)および、nessus-fetch.rc ファイルをダウンロードします。

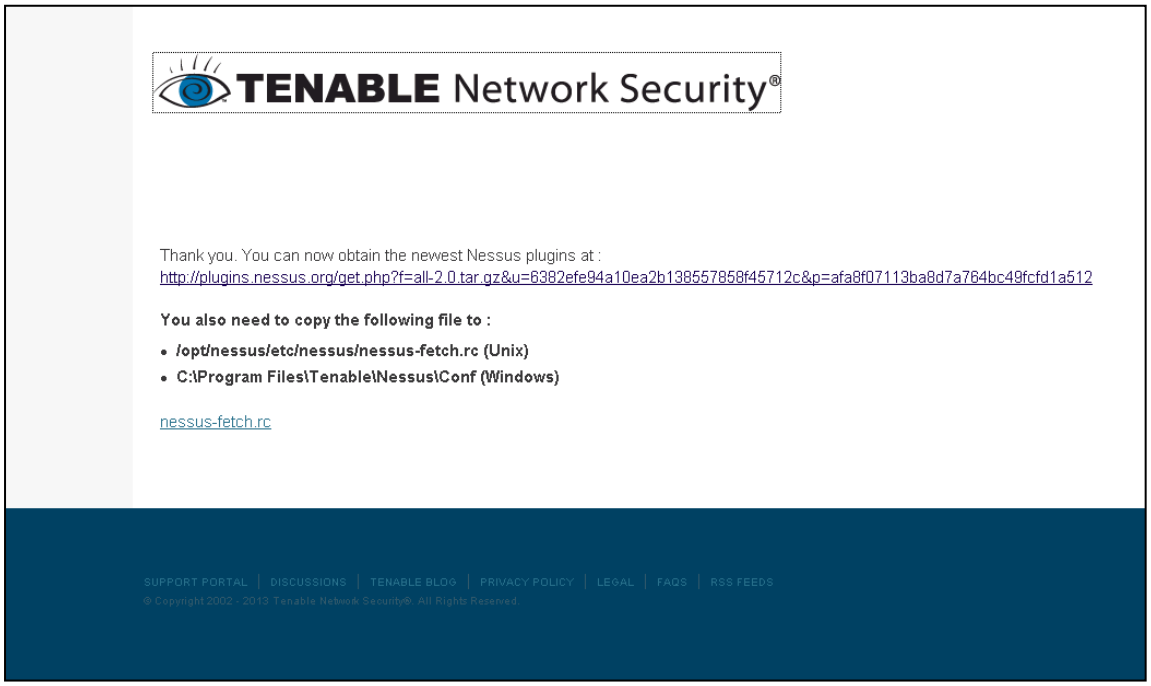


図 0.13 WEB からのチャレンジコード取得（入力完了画面）

⑧ プラグインのセットアップ

ダウンロードした nessus-fetch.rc ファイルは、/opt/nessus/etc/nessus/ディレクトリにコピーします。

実行例

```
[root@localhost ~]# ls -l /opt/nessus/etc/nessus/
total 28
-rw-r-----. 1 root root 5736 Jan 13 09:25 nessusd.conf.imported
-rw-------. 1 root root 6144 Jan 13 09:25 nessusd.db
-rw-r--r--. 1 root root  474 Jan 13 09:25 nessusd.rules
-rw-------  1 root root 4096 Jan 13 10:11 nessus-fetch.db
-rw-r--r--  1 root root  593 Jan 13 11:05 nessus-fetch.rc
```

以下のコマンドにより、ダウンロードしたプラグインの組み込みを行います。

実行例

```
[root@localhost ~]# /opt/nessus/sbin/nessus-update-plugins all-2.0.tar.gz
Expanding all-2.0.tar.gz...
Done. The Nessus server will start processing these plugins within a minute
```

プラグインの組み込み処理は、コマンドプロンプトが戻った後もバックグラウンドで動作しています。

⑨　セットアップ完了

プラグインの組み込み完了後は、ブラウザより https://hostname:8834 へアクセスすると、ログイン画面が表示されるようになります。

図 0.14 Nessus の管理画面（ログイン画面）

8.3.4. Nessus による脆弱性スキャンの実行

Nessus は、管理画面よりポリシーの作成、スキャンの実行、レポートの確認ができます。

① Nessus 管理画面にログイン

ブラウザより https://hostname:8834 へアクセスし、インストール時に作成したアカウントにてログインします。

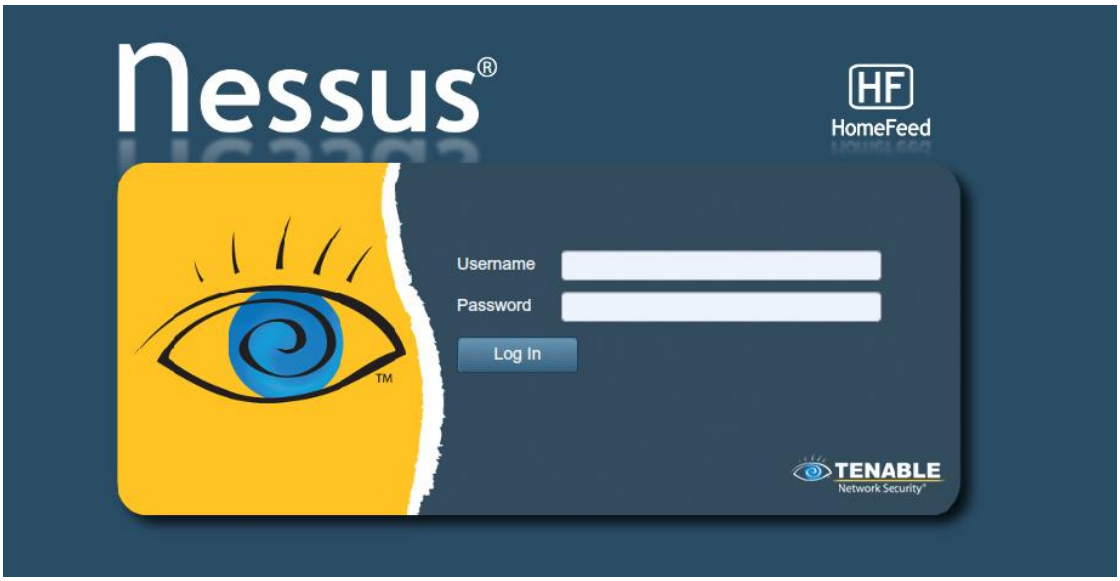


図 0.15 Nessus の管理画面（ログイン画面）

② スキャンに利用するポリシーの準備

ポリシーの設定は、ログイン後[Policies]タブより設定します。

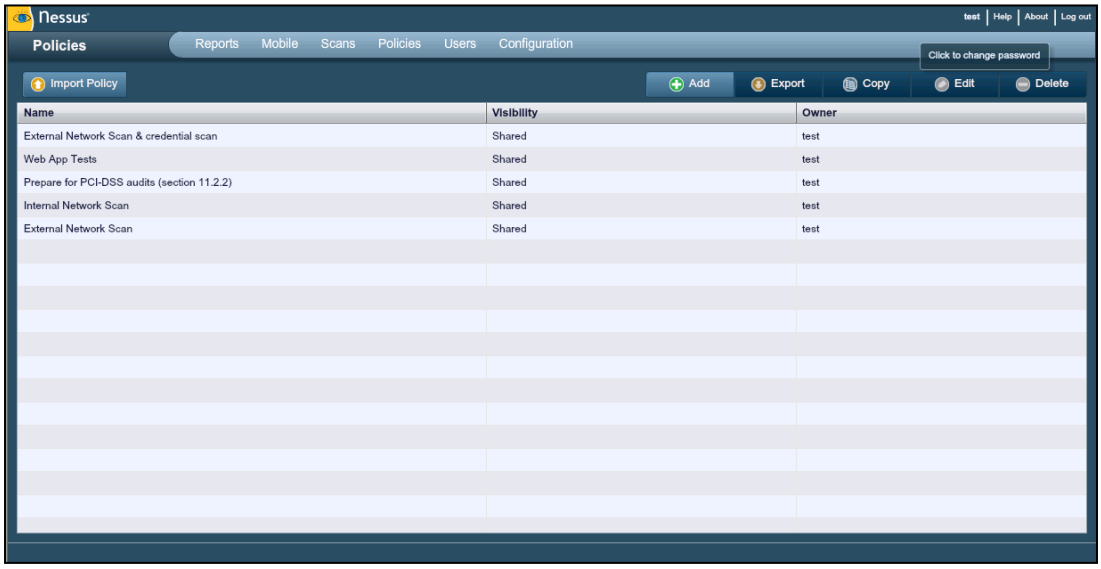


図 0.16 Nessus の管理画面（ポリシー設定）

ポリシーはデフォルトで 4 つのポリシーが準備されています。デフォルトの設定は、基本的なセキュリティスキャンを行うための設定がされたものなので、このデフォルトポリシーを基準にカスタマイズすることをお勧めします。

図 0.17 Nessus の管理画面（デフォルトのポリシー）

表 0.1 デフォルトポリシーの概要一覧

ポリシー名	説明
External Network Scan	外部（インターネット）に公開されたホスト向けのポリシー。65,536 個の全ポートに対しスキャンを実施します。このポリシーを基準にセキュリティスキャンを行うのがよい

	でしょう。
Internal Network Scan	内部ネットワーク内のホスト向けポリシー。標準的なポートに対しスキャンを実施します。
Web App Tests	ウェブアプリケーションサーバ向けポリシー。
Prepare for PCI DSS audits	情報セキュリティ基準である PCIDSS 基準の要件を満たしているかを確認するために作成されたポリシー。

③　ポリシーのカスタマイズ

デフォルトポリシーをコピーして、カスタマイズをします。[Policies]タブにて、デフォルトポリシー[External Network Scan] を選択し、[Copy] を選択すると、[Copy of External Network Scan] というポリシーが新規に作成されます。

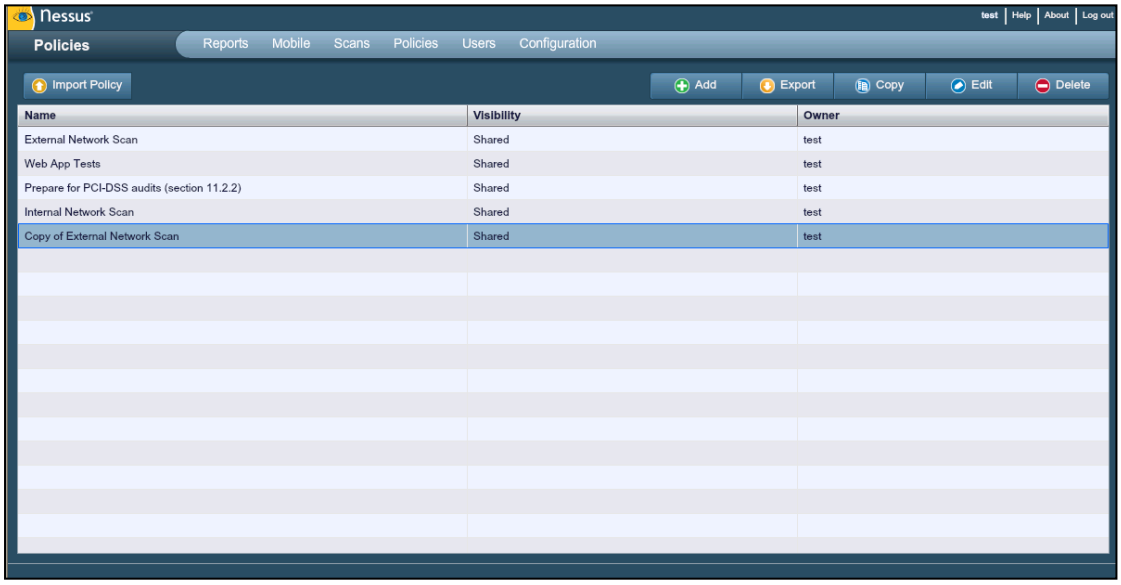


図 0.18 Nessus の管理画面（ポリシーの一覧）

ポリシー [Copy of External Network Scan] をダブルクリックまたはポリシー選択後に[Edit] を選択すると、ポリシー編集画面が表示されます。

今回は、デフォルトポリシーに Credential スキャンを追加設定します。

※Credential スキャンは、リモートホストに SSH 等を利用してリモートログインし、パッケージバージョンのチェック等による脆弱性検査をする機能です。Linux システムにおける脆弱性は OSS パッケージに

含まれる不具合等に起因するものが多く、パッケージバージョンのチェックによる脆弱性検査は非常に有効です。

左サイドバーの［Credentials］を選択し、［Credential Type］のドロップダウンリストから［SSH Settings］を選択してください。

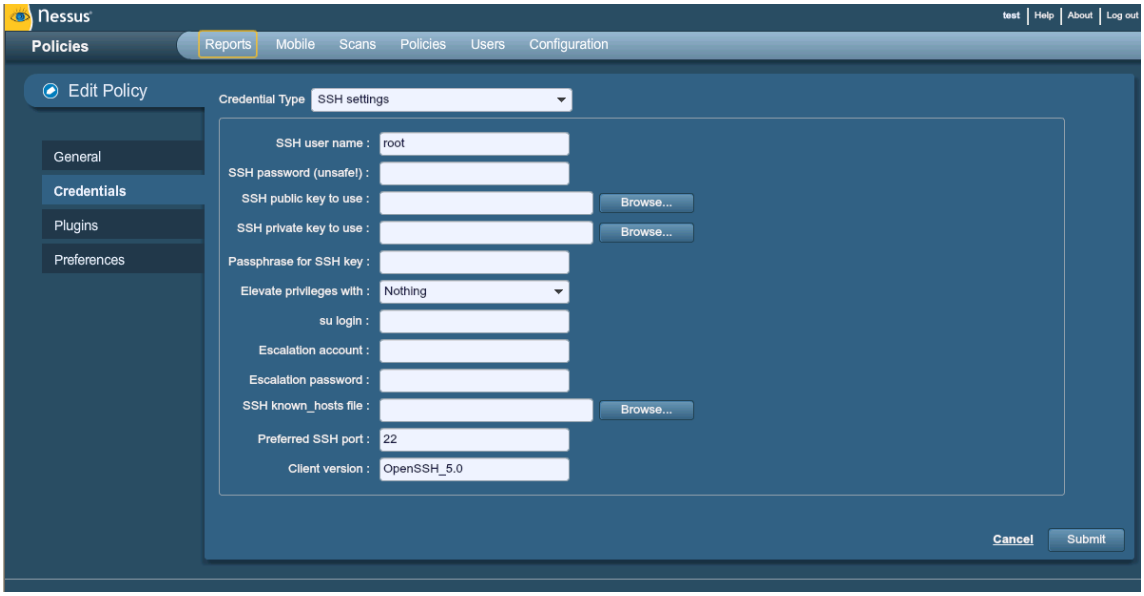


図 0.19 Nessus の管理画面（ポリシー編集画面）

表 0.2 Credentials スキャン SSH パラメーター一覧

パラメータ	説明
SSH user name	ssh ログイン ID を入力。
SSH password	ssh パスワードを入力。
SSH public key to use	ssh 公開鍵を選択。
SSH private key to use	ssh 秘密鍵を選択。
Passphrase for SSH key	鍵のパスフレーズを入力。
Elevate privileges with	管理者権限へ移行する方式を選択。 Nothing：ssh ログイン ID が管理者権限の場合に選択。 sudo：sudo により管理者権限へ移行する場合に選択。 su：su により管理者権限へ移行する場合に選択。 su+sudo：su 実行後に sudo 実行することで管理者

	権限へ移行する場合に選択。
su login	Elevate privileges with にて、su または su+sudo を選択した場合に入力。未入力の場合は、root となります。
Escalation account	Elevate privileges with にて、sudo または su+sudo を選択した場合に入力。未入力の場合は、root となります。
Escalation password	su または sudo により管理者権限を持つアカウントへ移行する際のパスワードを入力。

今回の手順にてスキャンを行うリモートホストは、root ユーザにてパスワードログインが可能な環境のため、[SSH password] に root ログインパスワードを入力し、[Submit] を選択します。

なお、Credential スキャンは root 権限が必要となります。root ユーザでの SSH アクセスを禁止している場合は、su または sudo を利用します。[Elevate privileges with]で認証方式を選択し、[su login]、[Escalation account]、[Escalation password]に認証情報を入力して下さい。

④　スキャンの実行

スキャンの実行は、[Scans] タブより行います。

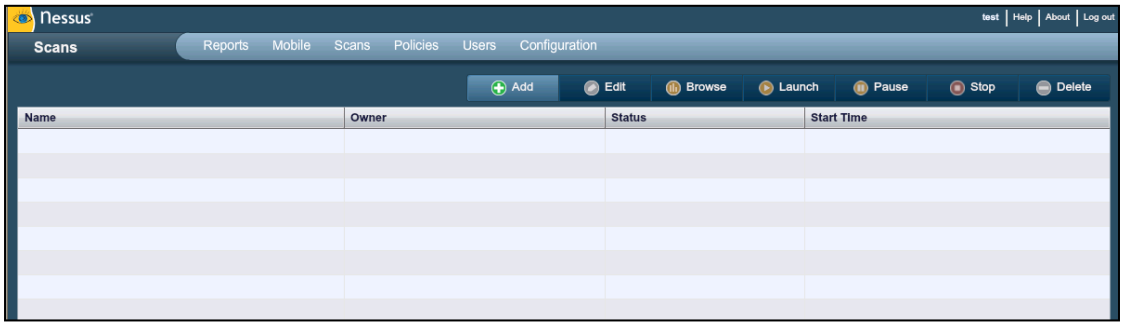


図 0.20 Nessus の管理画面（スキャンの実行）

[Add] を選択すると、スキャンのパラメータ入力画面が表示されます。

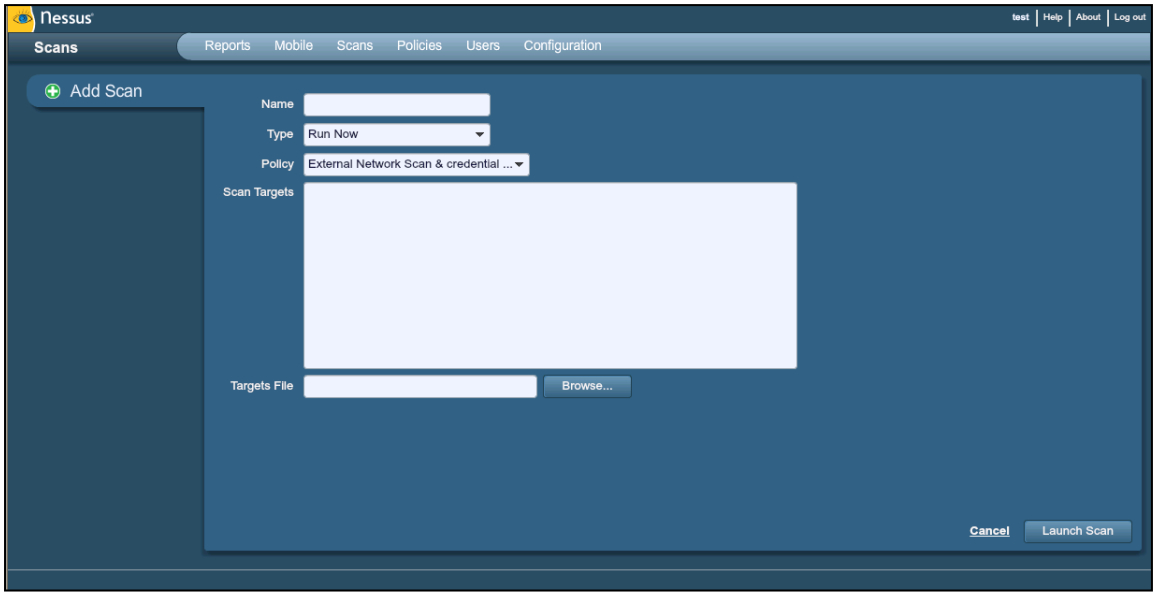


図 0.21 Nessus の管理画面（スキャンのパラメータ入力画面）

表 0.3 スキャンの実行パラメーター一覧

パラメータ	説明
Name	スキャン名称を指定する。
Type	実行タイプを以下のいずれかから選択する。 Run Now(Launch Scan 選択後に即時実行) Scheduled(開始日時を指定) Template(テンプレートとして作成).
Policy	[Policies] タブにて作成したポリシーから選択する。
Scan Targets	スキャン対象のリモートホストを以下のいずれかの形式で指定する。 IP アドレス(例. 192.168.0.1) IP レンジ(例. 192.168.0.1-192.168.0.255) CIDR 形式(例. 192.168.0.0/24) ホスト名(例. www.nessus.org)
Targets File	スキャン対象のリモートホストのリストを指定する。※[Scan Targets]を入力する場合は指定不要。

今回の手順では、以下のパラメータを入力し、[Launch Scan]を選択しました。

Name: [test scan]

Type: [Run Now].

Policy: [Copy of External Network Scan]

Scan Targets: [192.168.198.129]

図 0.22 Nessus の管理画面（スキャンの実行画面）

⑤　レポートの確認

スキャン実行の結果作成されるレポートの確認は、[Reports] タブより行います。

実行したスキャンが完了すると、[Status] が Completed と表示されます。

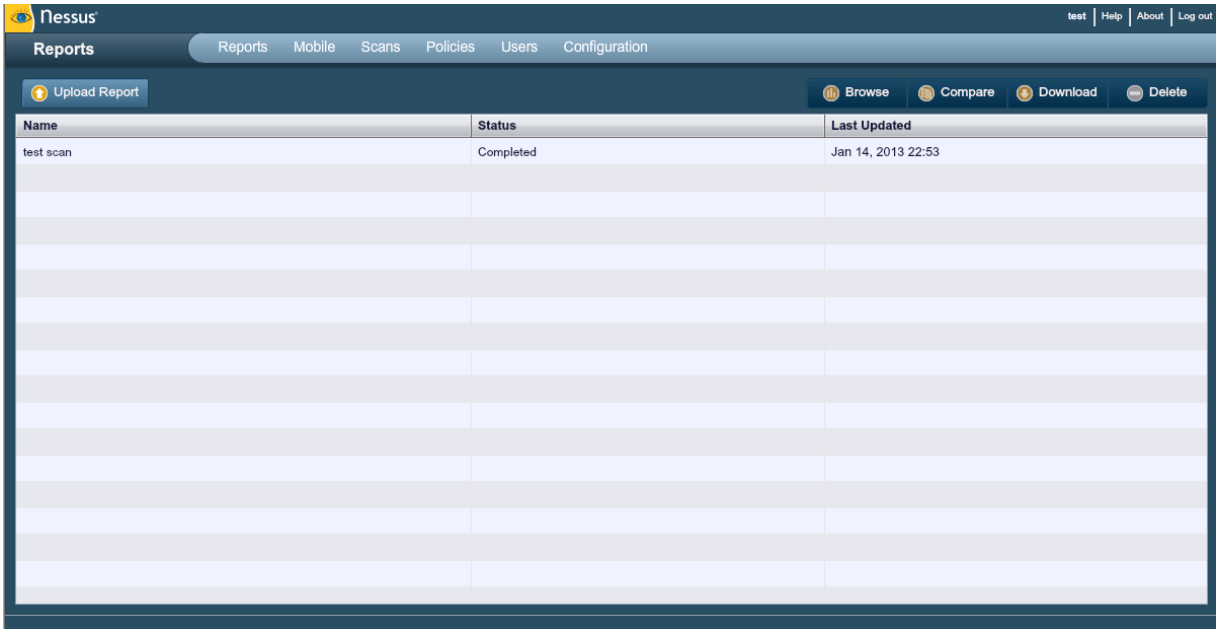


図 0.23 Nessus の管理画面（スキャンの実行画面：実行完了後）

対象のスキャンをダブルクリックまたは、対象のスキャンを選択後に[Browse] を選択することで、スキャン実行結果のレポートサマリが表示されます。

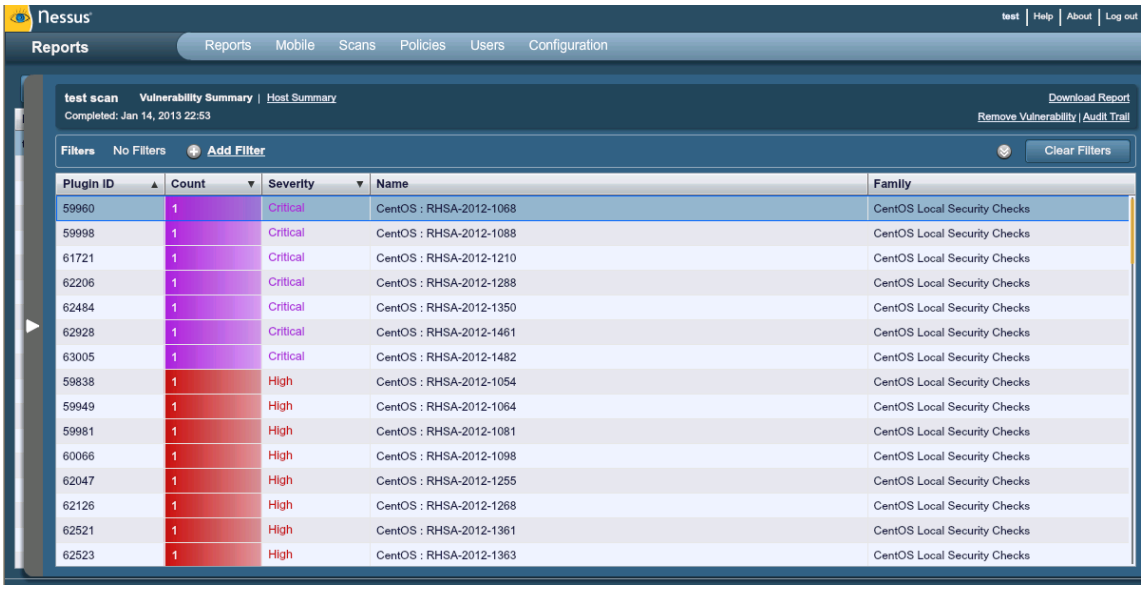


図 0.24 Nessus の管理画面（レポートサマリ）

レポートサマリに表示された指摘項目が脆弱性の可能性があるものです。指摘項目を選択すると、詳細な情報が表示されますので、必要に応じて対処してください。

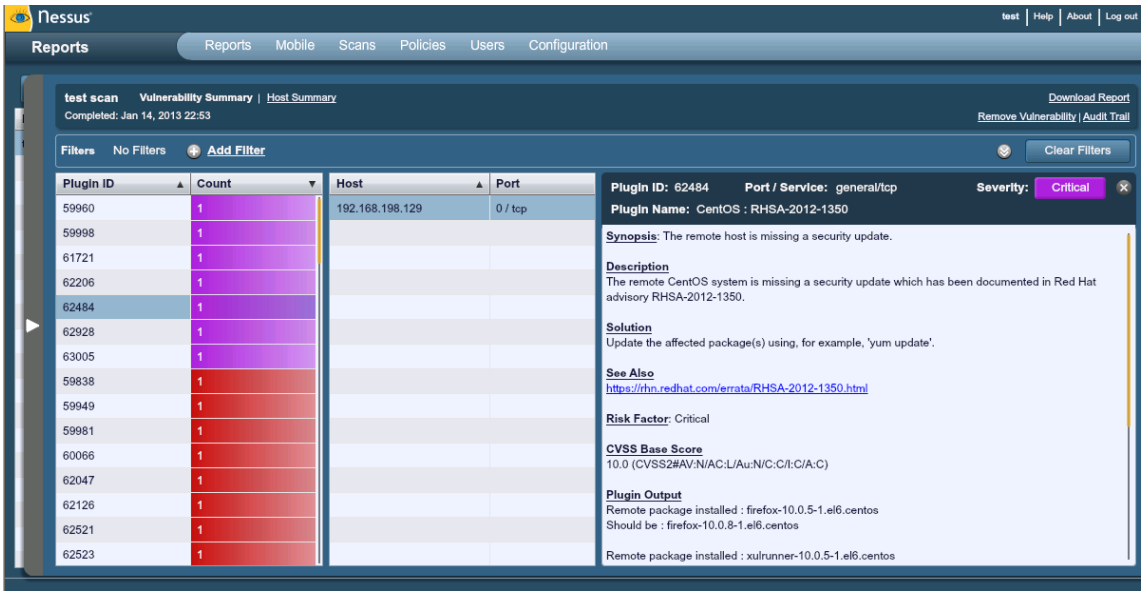


図 0.25 Nessus の管理画面（結果の詳細表示）

8.3.5. Nessus の定期運用

Nessus の Plugin は日々アップデートされています。

その為、脆弱性スキャンを行う際には、Plugin を最新のバージョンにアップデートしたうえで行う必要があります。

Nessus 実行環境がインターネットアクセス可能な場合は、自動で Nessus が Plugin のアップデートを行いますが、オフライン環境で利用されている場合は、必ず Plugin の最新情報（https://plugins.nessus.org/offline.php）を確認して、更新があればダウンロードしてください。

9. WEB サーバ（apache）におけるセキュリティ

Linux は、サーバ用 OS として、Web、メール、データベースなど様々なサービスを提供することができます。その中でも、Web サーバは常に不特定多数のアクセスを受け付けるため、外部からの攻撃に晒されやすいと言えます。

Web サーバに対して攻撃を行い、サーバを乗っ取った上で内容を改ざんするという多くの事例があります。Web サーバに対する攻撃手法は様々ですが、天才的なプログラマーがクラッキングを行うことは希と言えるでしょう。多くの場合は、クラッキングツールを使った単純な攻撃によるものなので、本章で紹介するような比較的簡単な方法でも一定以上の効果が期待できます。

Web コンテンツを改ざんされることは、利用者に迷惑をかけるだけではなく、企業や組織の社会的信用をなくすことにもつながります。管理者としては万全の対策を行い未然に防ぐ必要があります。本章では、Web サーバのセキュリティを高めることを目的に、apache の設定を中心に説明します。

9.1. ユーザ認証の実施

Web サーバにアクセス制限を設ける代表的な方法には、アクセス元によって制限する方法とユーザ認証によって制限する方法があります。本設ではユーザ認証による方法を説明します。

アクセス制限の方法は様々ありますが、iptables や apache の設定などによって、アクセスできる IP アドレスを制限する事ができます。しかし、この方法では、個々の利用者を制限する事はできません。

この機能は、リリース前の Web サーバにアクセス制限を設けたい場合に利用できます。リリース前の Web サーバは、設定変更を行う、最終テスト前のプログラムを置くなど、セキュリティホールができやすい状況にあると言えるでしょう。

また、ある Web ページを一部のユーザだけに見せたい場合、接続元 IP アドレスによる制限ではなく、ユーザ認証を行う必要があります。

apache においてユーザ認証を実施する場合、Basic 認証と Digest 認証の二つの方法が存在します。広く利用されているのは Basic 認証です。Basic 認証はパスワードが平文で送信されるのに対し Digest 認証はパスワードが暗号化されるため、Basic 認証は Digest 認証に比べセキュリティ面で劣ります。本章で

は、より安全な Digest 認証を実現する方法を説明します。ただし、古いブラウザの場合 Digest 認証に対応していない場合があるので注意が必要です。

ここでの例は、/var/www/html ディレクトリ配下のアクセスに対して Digest 認証を行うことを想定します。

最初に apache の設定ファイル/etc/httpd/conf/httpd.conf を編集します。

/var/www/html に関する<Directory> ディレクティブに、Digest 認証に関する記述を追加します。
（太字が追加部分）

設定例

```
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all

    AuthType Digest
    AuthName "Digest Auth Name"
    AuthUserFile /etc/httpd/conf/.htdigest
    Require valid-user
</Directory>
```

各項目の説明は以下の通りです。

表 9.1　Digest 認証に関する Apache の設定項目

AuthType	認証の種類を指定します。Digest 認証の場合は、Digest を指定します。
AuthName	認証名です。パスワードファイルと同じものを指定する必要があります。パスワードファイルではファイルを作る際に指定します。
AuthUserFile	パスワードファイルのパスを指定します。
Require	認証を使えるユーザを指定します。パスワードファイルに登録されているユーザすべての場合は valid-user とします。linuc というユーザのみにしたい場合は、user linuc とします。

次にパスワードファイルを作成します。

Digest 認証におけるパスワードファイル作成には、htdigest コマンドを利用します。

```
htdigest -c パスワードファイルへのパス レルム ユーザ名
```

-c オプションは、ファイル新規作成のオプションです。すでにパスワードファイルが存在する場合は-c は必要ありません。レルムには認証の識別名として http.conf で追加した AuthName の値を指定します。

下記は、linuc というユーザをパスワードファイルに登録する例です。コマンドを実行するとパスワードを聞かれますので、ユーザのパスワードを入力してください。

実行例

```
[root@localhost]# htdigest -c /etc/httpd/conf/.htdigest "Digest Auth Name" linuc
Adding password for linuc in realm Digest Auth Name.
New password:
Re-type new password:
```

ここまでの作業が完了したら設定変更を反映させるために、apache を再起動します。

実行例

```
[root@localhost]# /etc/init.d/httpd restart
httpd を停止中:                                            [  OK  ]
httpd を起動中:
```

再起動が完了し、ブラウザでアクセスすると、ホームページが表示される前に認証ボックスが表示されます。設定したユーザ名とパスワードを入力して、動作を確認します。

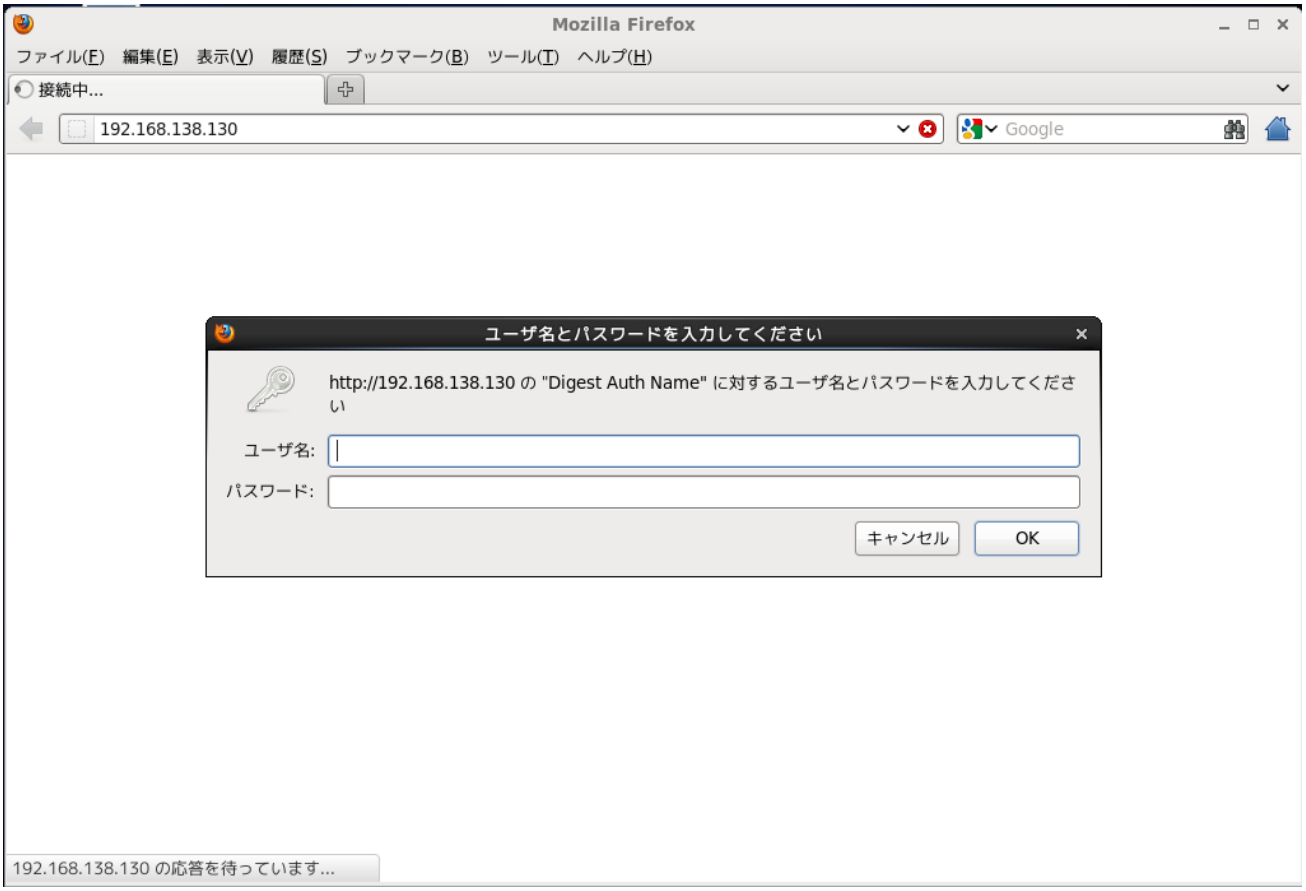


図 9.1 ユーザ認証の表示

本節では、Digest 認証でアクセス可能なユーザを限定する方法を紹介しました。

Digest 認証ではパスワードファイルでパスワードを管理しているため、開発中のサイトを関係者だけにアクセスさせたいときや、一部の部署のユーザだけが閲覧可能なページを用意したい場合などに適しています。SNS のような多くのユーザを抱えるサイトでのユーザ認証には適していません。SNS のような不特定多数のユーザが利用するような CGM(Consumer Generated Media)サイトでは、認証機能を Web アプリケーションの中で実装し、パスワードは暗号化した上でデータベースに保管する方法が適しています。

9.2. サーバに関する情報を非表示にする

外部からのサイバー攻撃には様々な方法がありますが、セキュリティホールを見つけ、そこを攻撃するのが基本です。セキュリティホールを見つけるために、攻撃者は断片的な情報を拾い集めます。

下記は第三者が存在しないファイルを指定した時に表示される Not Found 画面です。適当なファイル名を入力すればよいので誰でも簡単に表示できます。

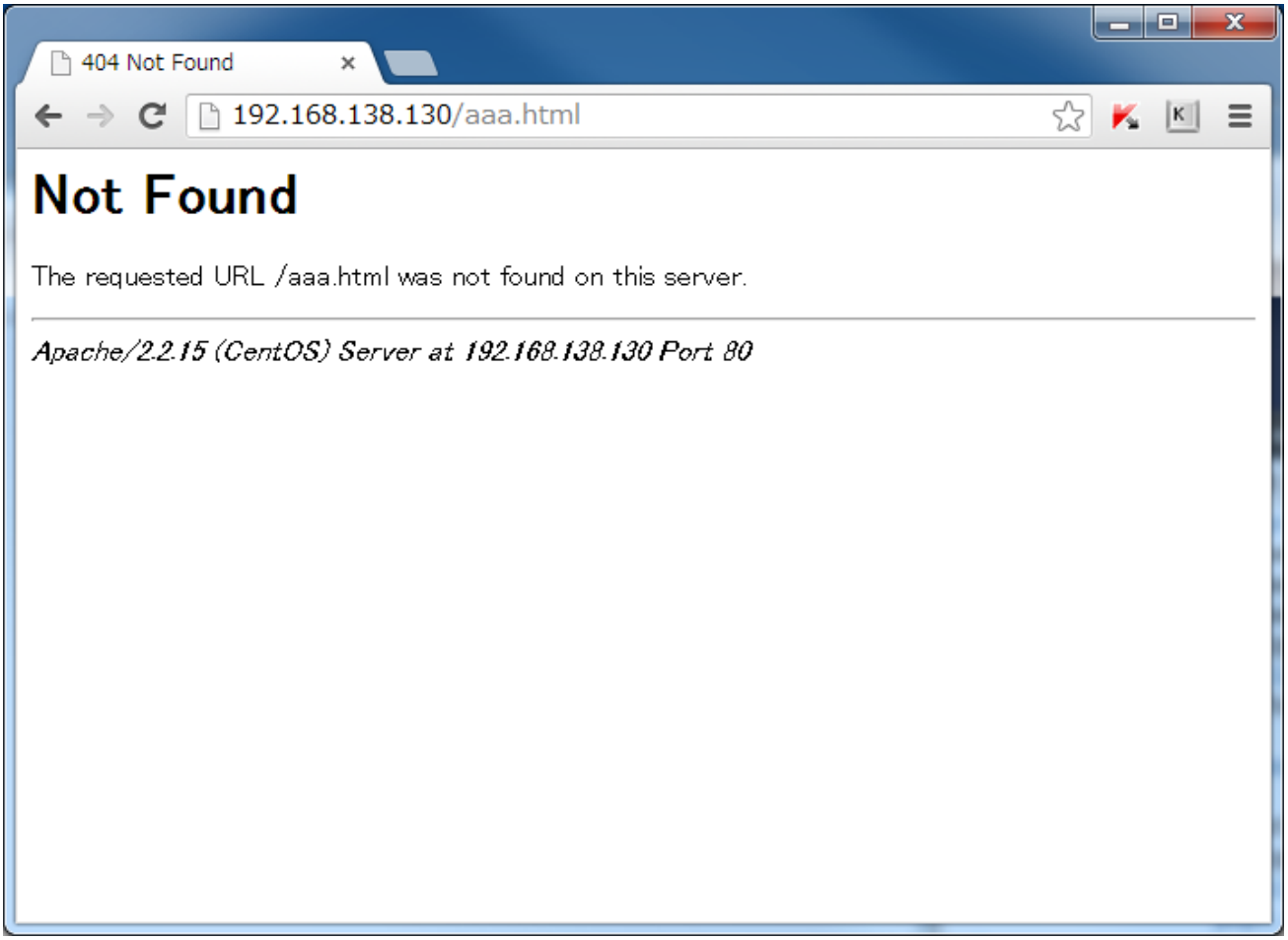


図 9.2 デフォルトの not found 画面

一見なにもない画面に見えるかもしれませんが、ここには様々な情報が表示されています。

1. Web サーバのプログラムとして apache を利用している
2. apache のバージョンが 2.2.15 である
3. OS は、CentOS である

攻撃者はこれらの情報から apache のバージョン 2.2.15 に関して脆弱性の情報を調べることが可能です。したがって、Web サーバのバージョンや OS の種類など、システムに関する情報を公開することは、セキュリティ上のリスクとなります。

apache に関する情報（シグネチャ）を表示しないようにするには、/etc/httpd/conf/httpd.conf の

```
ServerSignature On
```

となっている部分を

```
ServerSignature Off
```

と修正します。

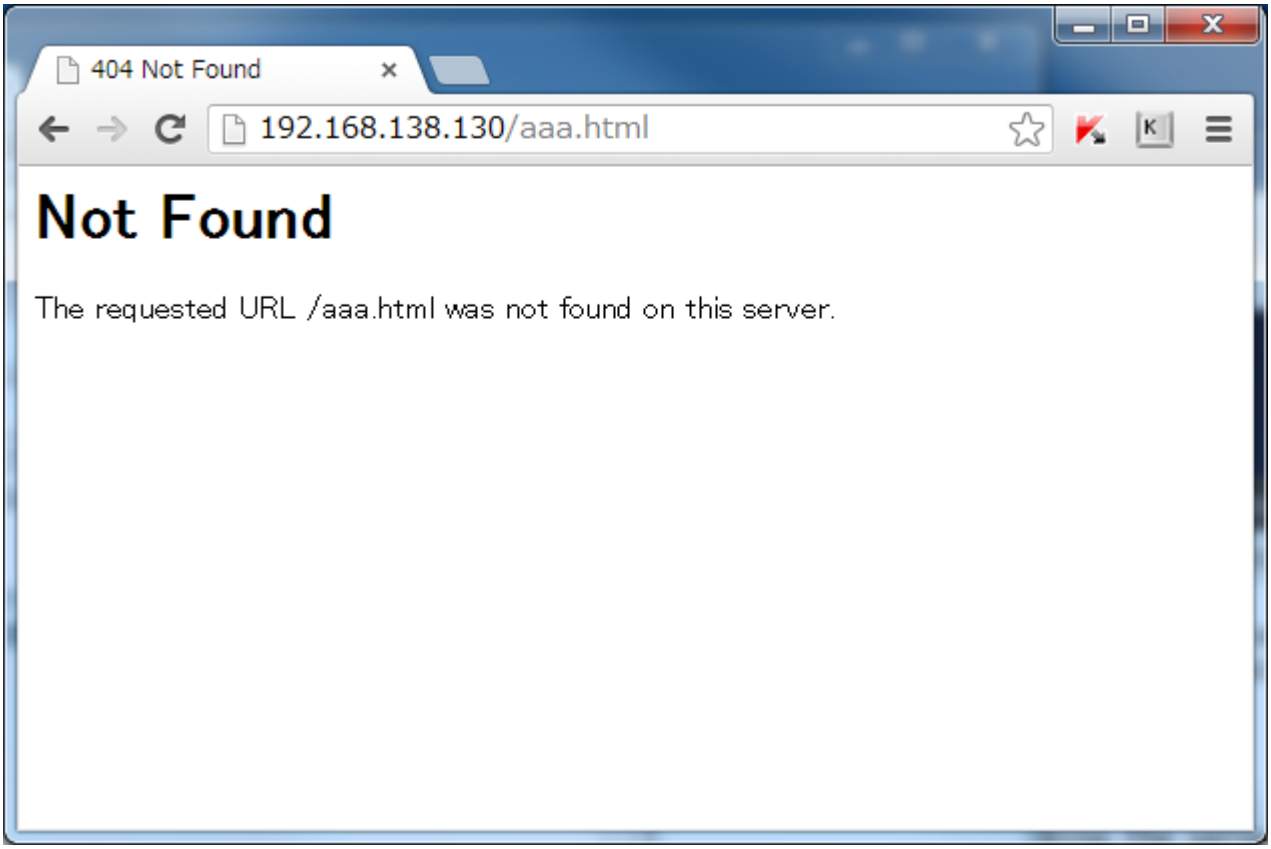


図 9.3 表示情報を限定した not found 画面

また、よくある事例として次のように URL にディレクトリを指定することで、そのディレクトリの一覧が見える状態があります。これも第三者に不要な情報を与えるためセキュリティ上のリスクとなり得ます。

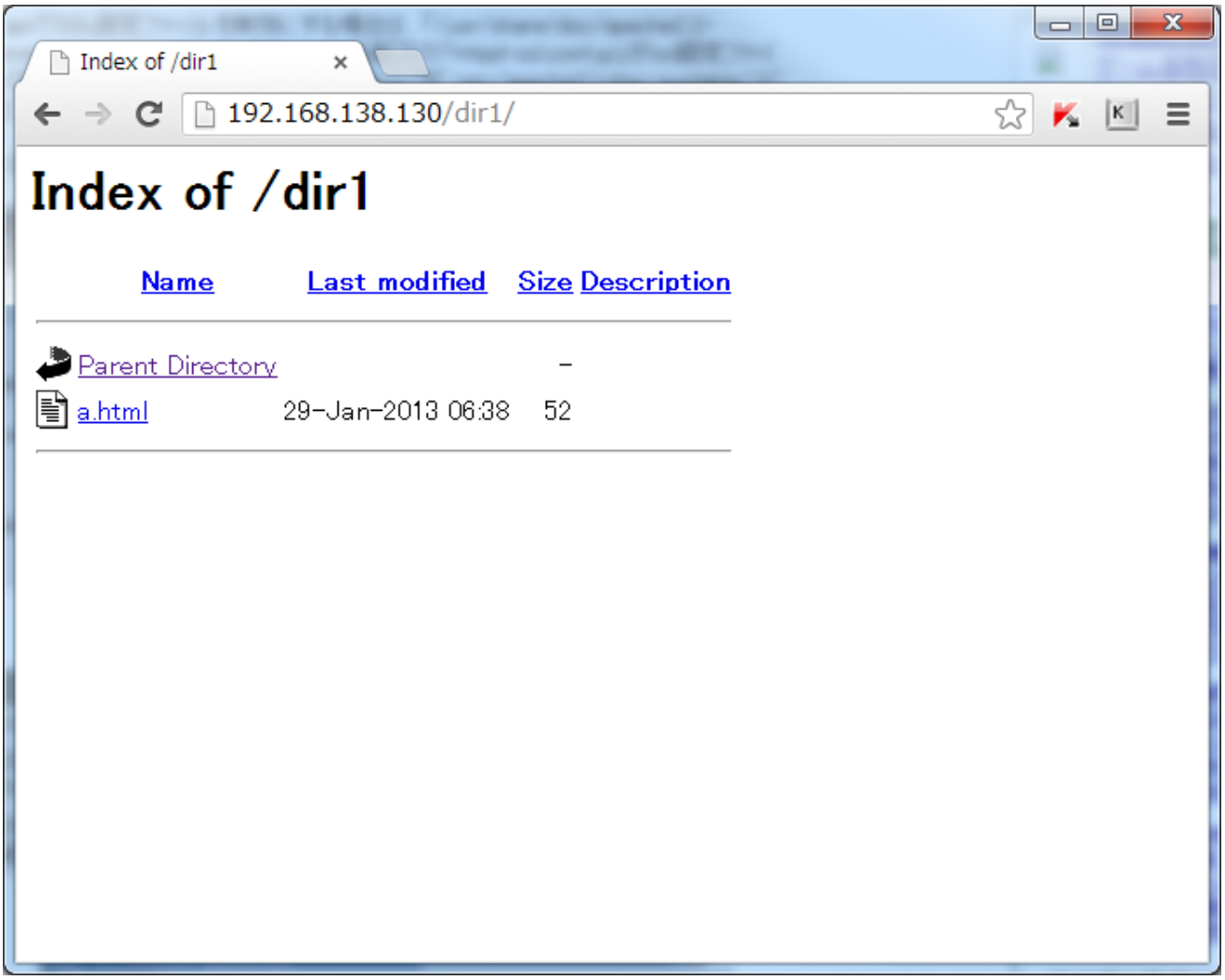


図 9.4 ブラウザによるディレクトリの中を表示する例

この場合、httpd.conf において、<directory>ディレクティブ内の Options に Indexes が設定されていると考えられます。

設定例

```
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

ディレクトリの中身一覧を表示させないようにするには、Indexes の記述を以下のように削除します。

設定例

```
<Directory "/var/www/html">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

このように、不特定多数のユーザからアクセスされる Web サーバでは、不要な情報を外部に公開しないように設定の見直しをすると良いでしょう。

Apache の設定以外にも、例えば PHP の動作確認を行うために phpinfo()メソッドを使った PHP や Apache に関する情報一覧のページを表示するファイルを作成した場合、そのファイルをそのまま放置することは好ましくありません。また、システムのログファイルを、/var/www 配下など第三者がアクセスできるような Web コンテンツと同じディレクトリに置くべきではありません。

9.3. Web コンテンツの所有者とパーミッションの設定

ここで一つ問題です。

> Web サーバにインターネットからリクエストがあった時、index.html などのコンテンツファイルはどの Linux ユーザ・グループの権限でアクセスされるでしょうか？

答えは、httpd.conf における設定されているユーザおよびグループです。

httpd.conf において User および Group が設定されている箇所を確認してください。
多くの場合は、下記のような設定になっています。

設定例

```
User apache
Group apache
```

上記の設定の場合、Web コンテンツファイルへのアクセスは全てユーザ apache の権限でアクセスされます。

apache ユーザは、Web コンテンツにアクセスするための最低限の権限しか与えられておらず、通常のパスワードによる認証ができないユーザとして登録されています。このように apache ユーザが制限されているのは、外部から不正な操作で、システムの重要な設定が書き換えられないようにするためです。

User や Group に、存在しないユーザやログイン可能な権限を持つユーザが設定されている場合は、設定を見直した方が良いでしょう。

特に CGI プログラムなどの Web コンテンツの所有者は、上記の User/Group の設定に合わせて apache ユーザに設定するべきです。Web コンテンツの所有者に、Web コンテンツ以外にアクセス可能なユーザを設定するべきではありません。Web コンテンツを利用して、不正に他のファイルを書き換えられることを防ぐためです。

静的な HTML コンテンツであれば、ファイルのパーミッションは読み込み権限のみ設定すれば十分です。CGI プログラムなど実行が必要なファイルであれば実行権限のみ設定すれば十分です。どちらの場合でも、

書き込み権限を与えることは、セキュリティ上好ましくありません。そして、所有者以外には特別な理由がない限り権限を与える必要はありません。

コンテンツファイルは所有者だけでなく、グループもすべて apache グループにします。そのために、ウェブコンテンツを置くディレクトリには SGID を設定して、配下のファイルはすべて apache グループになるようにしておくと良いでしょう。

下記は、ウェブコンテンツを置くディレクトリが/var/www/html の場合における実行例です。

実行例

```
[root@localhost]# chmod 2755 /var/www/html
[root@localhost]# ls -l /var/www
合計 16
drwxr-xr-x. 2 apache apache 4096  1月 29 07:32 2013 cgi-bin
drwxr-xr-x. 3 apache apache 4096  9月  3 05:17 2012 error
drwxr-sr-x. 3 apache apache 4096  1月 29 06:39 2013 html
drwxr-xr-x. 3 apache apache 4096  1月 29 06:30 2013 icons
```

9.4. CGI における注意点

例外はありますが、基本的に CGI プログラムによって、Web コンテンツのディレクトリにファイルを作成することは、好ましくありません。Web コンテンツのディレクトリに置かれたファイルは、http 経由で誰でも内容を閲覧することが可能です。

ユーザから入力された、住所・メールアドレス・パスワードなどの情報は一時的にでも、Web コンテンツのディレクトリ配下のファイルに書き込むことは、情報漏洩につながる可能性があります。

また、ユーザからの入力などの外部から来るデータは、そのまま受け取るべきではありません。コマンドインジェクション、SQL インジェクションなどの脆弱性を生むことになり、サーバを不正に操作される、または情報漏洩という事態になり得るためです。入力されたデータは内容をチェックし適切に加工するようにコーディングする必要があります。

参考：コマンドインジェクション、SQL インジェクションに関する IPA の情報

http://www.ipa.go.jp/security/vuln/vuln_contents/sql_solution.html

http://www.ipa.go.jp/security/vuln/vuln_contents/oscmd_solution.html

これはアプリケーション内部の話で、サーバ管理には関係のない話に聞こえるかもしれません。

しかし脆弱性のある CGI プログラムをサーバに置いた場合、それが原因でサーバが乗っ取られてしまう可能性がある以上、CGI プログラムなどの Web アプリケーションのセキュリティに十分注意し、ログを監視するなどのリスク管理を行うことはセキュアなサーバを運用する上で非常に重要な要素です。

9.5. apache の DoS 攻撃対策

Web サーバに対する攻撃手法として代表的なものが DoS 攻撃です。DoS 攻撃とは、サーバに対して大量のパケットを送信することで負荷をかけ、サービスを利用停止に追い込むものです。

DoS 攻撃の対策として、接続数を制限する方法があります。例えば、httpd.conf には、下記のように設定します。

設定例

```
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
ServerLimit      256
MaxClients       256
MaxRequestsPerChild  4000
</IfModule>
```

MaxClients は、リクエストに応答するために作成される子プロセスの最大数を設定しており、MaxRequestsPerChild は子プロセス 1 個が処理できるリクエスト数などを設定しています。

これらの数字を小さくすると、多くのリクエストが届いても、サーバ全体が動作不能になることを防ぐことができます。

しかし、これだけでは DoS 攻撃への対策としては十分ではありません。サーバ全体が機能停止することは防げても、リクエスト数が最大になると、正規のユーザまでが Web コンテンツを閲覧することができなくなってしまいます。

接続数を制限する以外の DoS 攻撃への対策としては、DoS 対策モジュールを利用する方法があります。これらのモジュールを利用することで大量のアクセスを行う発信元のみをブロックし、その他の正規ユーザには引き続きサービスを提供することが可能となります。

apache の DoS 対策モジュールは、いくつか存在しますが、ここでは mod_evasive を紹介します。

mod_evasive は、Jonathan A. Zdziarski 氏が作成した DoS 対策モジュールで、GPL バージョン 2 の下で配布されているオープンソースソフトウェアです。

mod_evasive をインストールするには、Jonathan A. Zdziarski 氏のサイトから wget コマンドを使ってソースコードを取得し、コンパイルする必要があります。

コンパイルは apxs コマンドで行うことが可能ですが、apxs を利用するには httpd-devel パッケージが必要です。httpd-devel パッケージがインストールされていない場合は yum コマンドで事前にインストールする必要があります。

以下、mod_evasive のソースコードを/tmp に展開してインストールする例です。

実行例

```
[root@localhost]# cd /tmp
[root@localhost]# yum install httpd-devel
[root@localhost]# wget
http://www.zdziarski.com/blog/wp-content/uploads/2010/02/mod_evasive_1.10.1.tar.gz
[root@localhost]# tar xvzf mod_evasive_1.10.1.tar.gz
=== 省略 ===
[root@localhost]# cd mod_evasive
[root@localhost]# apxs -cia mod_evasive20.c
[root@localhost]# vi /etc/httpd/conf.d/mod_evasive20.conf
```

上記の処理により、/etc/httpd/modules に mod_evasive20.so が作成されます。

次に mod_evasive の設定ファイルを/etc/httpd/conf.d/mod_evasive20.conf として以下の内容で作成します。

設定例

```
<IfModule mod_evasive20.c>
DOSHashTableSize    3097
DOSPageCount        2
DOSSiteCount        50
DOSPageInterval     1
DOSSiteInterval     1
```

```
DOSBlockingPeriod    10
</IfModule>
```

表 9.2　mod_evasive の設定例

DOSHashTableSize	ハッシュテーブルサイズ。パフォーマンスに問題がないかぎり、3000 くらいを目安に設定する。
DOSPageCount DOSPageInterval	DOSPageInterval で設定した秒数以内で、同じページにアクセスできる上限を設定します。設定例では、同ページに 1 秒間に 2 回以上アクセスがあると、そのクライアントをブラックリストへ登録し接続を制限します。
DOSSiteCount DOSSiteInterval	DOSSiteCount は DOSSiteInterval で設定された秒数における、サイト全体に対するアクセス数の上限。
DOSBlockingPeriod	ブラックリストへ登録された接続元を遮断する時間を秒数で指定。

動作確認を行うには、mod_evasive のソースコードがあるディレクトリ（上記の場合/tmp/ mod_evasive）内の test.pl ファイルを使います。perl コマンドで実行することで確認できます。

このコマンドは、自身（127.0.0.1）の 80 番ポートへ 100 回リクエストを送信します。mod_evasive が有効であれば、途中から応答が 200（成功）から 403（拒否）へと変わります。

実行例

```
[lpij@localhost ~]$ perl test.pl
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden

===  以下省略  ===
```

9.6. その他のセキュリティ対策

本章ではWebサーバのセキュリティを高める方法の一部を紹介しました。サーバへの攻撃手法や、防御策は日々変化しており、これだけやっておけば安心という方法はありません。また、環境やサービスの種類によって行うべきセキュリティ対策も変わります。

例えば、ECサイトなど顧客情報や決済情報を扱うサイトであれば、盗聴対策としてHTTPSで通信内容を暗号化すべきです。その際、なりすまし対策として、信頼性の高い認証局から発行された電子証明書を使い、サイトの正当性を証明すると良いでしょう。

9.5節では、Webサーバ単体でのDoS対策を紹介しましたが、Webサーバ単体の対策ではネットワーク帯域が占有されてしまう状況までは防げません。ネットワークの入り口となるファイアーウォールでの侵入防御システム（IPS：Intrusion Prevention System）などを有効に活用することでDoS攻撃対策などを実施し、ネットワークシステム全体でセキュリティ対策を行うことが重要です。

また、設定のミスやパッケージのアップデートを怠ったなどの運用上のミスが原因という事故は少なくありません。日常における運用管理・監視を計画的に行うことが高いセキュリティの維持に重要なことを理解していただければ幸いです。

www.lpi.or.jp