Linux セキュリティ実習

IT スペシャリスト科 1 期　セキュリティ専攻

2009 年後期

目次

第 1 章

PAM

1.1 PAM とは

PAM は[*1]Sun Microsystems が自社の OS Solaris 向けに開発したもので、Linux 以外にも HP-UX、Mac OS X などの商用 UNIX にも使用されている認証用システムです。

認証を必要とする su、password、ssh などのプログラムが利用するライブラリとして提供される。PAM に対応したアプリケーションの動作を図 1.1 に示す。

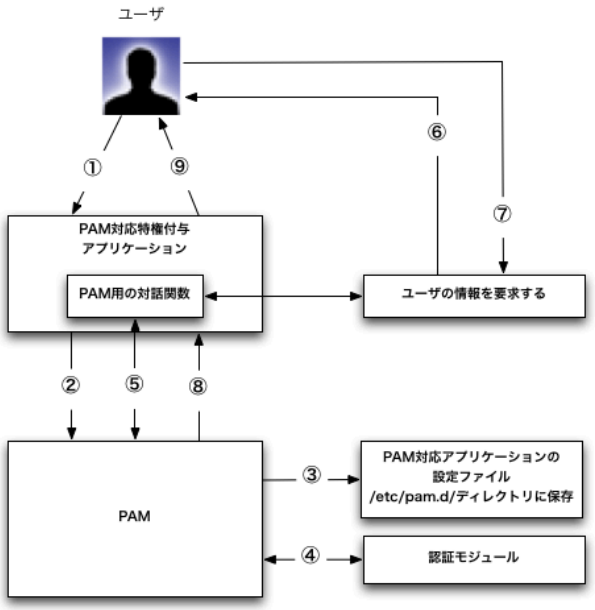

図 1.1　PAM 対応アプリケーション機能

1. ユーザが、サービスにアクセスするために PAM 対応アプリケーションを起動する。
2. PAM 対応アプリケーションが、認証を行う PAM ライブラリを呼び出す。
3. PAM ライブラリが、/etc/pam.d ディレクトリ内のアプリケーション固有の設定ファイルを検索し、そこに記されている認証タイプが使用される。設定ファイルがないときは/etc/pam.d/other ファイルの設定を使用する。
4. PAM ライブラリが、アプリケーション内で定義されている対話関数を呼び出す。

[*1] Pluggable Authentication Modules

5. 対話関数がユーザの情報を要求する。例えば、ユーザにパスワードの入力や網膜スキャンを求める。
6. ユーザが、要求された情報を提供する。
7. PAM 認証モジュールが、PAM ライブラリを介してアプリケーションに認証状況メッセージを与える。
8. 認証処理に成功すると、アプリケーションは次のいずれかの処理を行う。
   - 要求された特権をユーザに付与する
   - 処理が失敗したことをユーザに知らせる

1.2 PAM の設定

PAM の設定ファイルはアプリケーションごとに用意され、ディレクトリ/etc/pam.d に置く。設定行は次の各フィールドで構成される。設定ファイルの基本構文は次の通り。

リスト 1: 設定ファイルの基本構文

```
モジュール・タイプ　コントロール・フラグ　PAM モジュール　［引数］
```

- モジュールタイプ
- 制御フラグ
- モジュールパス
- モジュール引数

モジュールタイプは次の 4 種類がある。

表 1.1　PAM モジュールタイプ

モジュールタイプ	説明
auth	実際の認証。パスワードあるいは身分証明を求める
account	認証要求のアカウント管理を全て処理とアクセス指針を満たしているかをチェック
password	認証上の設定
session	サービスのセッション確立

表 1.2　PAM モジュール制御フラグ

制御フラグ	説明
required	成功応答を返すことを指定する。認証が失敗したときはどこで失敗したかを知られないようにし、セキュリティを高める。失敗したときに他のモジュールがあれば続ける
requisite	失敗応答を受け取ったらすぐに認証プロセスを中止
sufficient	成功応答を受け取った時点で認証プロセスを終了する
optional	ほとんど使用されない。正否応答を重視しないときに使う。

表 1.3 PAM モジュール

モジュール	機能
pam_unix	伝統的な UNIX のユーザ認証
pam_userdb	Berkley DB に格納した情報によるユーザ認証
pam_access	ユーザ、グループ単位のログイン制御
pam_time	時間帯によるログイン制御
pam_nologin	一般ユーザのログイン禁止
pam_securetty	root としてログイン可能な端末の制御
pam_rootok	root ユーザか否かによるログイン制御
pam_shells	有効なシェルが設定されているか否かによるログイン制御
pam_wheel	su コマンドで root になれるユーザの制御
pam_group	特定グループの権限付与

CentOS5.3 の/etc/pam.d/system-auth の例を示す。

リスト 2: system-auth の例

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so
auth        required      pam_nologin

account     required      pam_unix.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3
password sufficient pam_unix.so md5 shadow nullok try_first_pass use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.so
```

1.3 pam_unix

pam_unix モジュールは伝統的な UNIX ユーザ認証を実現する。このモジュールは各タイプで設定可能で様々なオプションが用意されている。

表 1.4 pam_unix モジュールのオプション

オプション	意味
debug	syslog に成否を記録する
audit	syslog に、より詳細な情報を記録する
use_first_pass	パスワードを求めるプロンプトを表示せず、すでに入力済みのパスワードを利用する
try_first_pass	パスワードが設定されている場合、パスワードを求めるプロンプトを表示しない
not_set_pass	後続のモジュールに入力済みのパスワードを引き渡さない
likeauth	後続のモジュールに入力済みの各種情報を引き渡す
noreap	子プロセスの終了を無視する
nodelay	認証などに失敗した際、2 秒間待機する処理を行わない
nullok	空パスワードを許容する

表 1.5 password タイプ固有のオプション

オプション	意味
use_authtok	try_first_pass と同様だか、新しいパスワードが設定されていない場合、失敗する。
shadow	シャドウ・パスワードを利用する
md5	ユーザがパスワードを変更する際に、md5 を用いてハッシュ化する
bigcrypt	ユーザがパスワードを変更する際に、DEC C2 アルゴリズムを用いる。
nis	新しいパスワードを設定する際に NIS RPC を利用する。
remember=X	X 個の古いパスワードを保持する

1.4 ログイン可能な時間を制御する

pam_time モジュールを使うとログインやサービスへのアクセスを元に制御できる。pam_time を使うには/etc/pam.d/system-auth に次の一行を追加する。

リスト 3: pam_time を有効に

```
account required pam_time.so
```

pam_time モジュールの設定ファイルは/etc/security/time.conf で、次の 4 つのフィールド名を順にセミコロン (;) で区切った行で制限を定義する。

services 制御対象のサービスのリスト。1 つのルールで login と su を制御するときは login|su と記す。

ttys 制限対象となるターミナルのリスト。コンソールターミナルを除き、擬似ターミナルだけを制御する場合は ttyp*&!tty*と記す。

users 制限対象となるユーザのリスト。全てのユーザを指定するときはワイルドカードの*を記す。

times 制限を提供する時間のリスト。24 時間制で範囲指定。午後 8 時から午前 6 時までを指定するには 2000-0600 と記す。曜日の指定は曜日の英語表記の先頭二文字 (Mo、Tu、We、Th、Fr、Sa、Su) と平日の Wk、土日の Wd、全ての Al が使える。曜日と時間を組み合わせられる。例えば、火曜と水曜の午前 7 時から午後 11 までは TuWe0700-2300 と記す。

これらの条件は次の書式に従い、

リスト 4: time.conf の書式

```
service;ttys;users;times
```

設定フィールドで使用できる特殊文字は次の通り。

! 否定。!login で login 以外と意味する。
| 論理和。doraemon|nobita は doraemon または nobita を意味する。
& 論理積。login & su は login かつ su を意味する。
* ワイルドカード。foo*は foo で始まる全ての文字列を意味する。

これらのルールを使い、制限を表記する。例えば、ユーザ doraemon はいつでもシステムにリモートアクセスでき、その他のユーザは毎日午後 8 時から午前 6 時まではリモートアクセスできないルールは次のように記す。

リスト 5: pam_time のルールの例

```
login;ttyp*&tty*;doraemon;!Al2000-0600
```

1.5 一般ユーザのログインを禁止する

システムメンテナンスなどでルート以外のユーザのログインを禁止したときは pam_nologin を使う。モジュールタイプには auth を使う。

リスト 6: pam_nologin を使う

```
auth required pam_nologin
```

このモジュールは/etc/nologin ファイルの有無をチェックし、そのファイルがあれば失敗を返し、そのファイルの内容を表示する。ファイルが存在しなければログインできる。

1.6 root としてログイン可能な端末を制限する

pam_securetty モジュールを使うとルートとしてログインできる仮想端末を制限できる。このモジュールには auth タイプを使用する。このモジュールを有効にするには/etc/pam.d/login に次の行を追加する。

リスト 7: pam_securetty

```
auth required pam_securetty.so
```

モジュールの設定は/etc/securetty ファイルで行い、一行に一つずつ root でログインを許可する仮想端末名を記す。このファイルがないときはモジュールによる制限は適用されない。

1.7 root か否かで実行制御

pam_rootok モジュールは root 専用ルーツで root 以外のユーザから呼び出しを禁止するときに使う。/etc/pam.d/プログラム名のファイルに auth タイプで指定する。

リスト 8: pam_rootok

```
auth sufficent pam_rootok.so
```

1.8 root になれるユーザを制限する

pam_wheel モジュールを使うと、su コマンドを実行できるユーザを特定のグループのユーザのみに限定できる。/etc/pam.d/su ファイルに auth タイプで指定する。

リスト 9: pam_wheel

```
auth required pam_wheel.so
```

このモジュールのオプションは次の通り。

debug　syslog に成否を記録する。
use_uid　処理の際に用いる uid として実 uid(ログイン時ユーザ) ではなく実効 uid(現在のユーザ) を用いる
trust　このモジュールが成功したときは無視ではなく、成功を返す
deny　指定されたグループのメンバーであるユーザのアクセスを成功させる代わりに失敗させ s る
group=**グループ名**　wheel もしくは GID が 0 のグループの代わりに指定したグループを用いる

この定義でオプションを指定しなかった場合、wheel があればそのグループの、なければ GID が 0 のグループのユーザからアクセスは無視し、それ以外のユーザからのアクセスは失敗となり、su コマンドの実行が失敗する。無視の場合、最終的なアクセスの可否はその他のモジュールに依存する。

CentOS5.3 の/etc/pam.d/su の記述のうち、pam_wheel に関する部分はリスト 10 の通りで、いずれもコメントとして記されているが、(1) を有効にすると wheel グループのユーザは無条件に su コマンドが成功し、パスワードの認証は行われない。(2) を有効にすると wheel グループのユーザのみが成功し、その他のユーザは正しいパスワードを入力しても認証失敗となり、su コマンドを利用できない。

リスト 10: CentOS の/etc/pam.d/su の内容 (抜粋)

```
#auth           sufficient      pam_wheel.so trust use_uid <- (1)
#auth           required        pam_wheel.so use_uid <- (2)
```

1.9 パスワード変更の制御

パスワード変更を制御するモジュールは pam_unix と pam_cracklib の 2 つがある。pam_unix では password タイプ (表 1.5) を指定して伝統的なパスワード変更機能を実現できる。

一方、pam_cracklib モジュールを用いると複雑なパスワードを強制できる。最近のディストリビューションではデフォルトで有効になっています。このモジュールのオプションは表 1.6 の通りで、パスワードの複雑性をチェックし、更に以下のチェックを行う。

- 回文チェック
- 大文字小文字の入れ替えのみ
- ほとんど同じパスワード
- 文字をずらしたパスワード
- 単純すぎるパスワード

表 1.6 pam_cracklib のオプション

オプション	デフォルト	意味
debug	指定無し	詳細なログを出力
type=UNIX	UNIX	ユーザに対してパスワード入力促す際のプロンプト「New UNIX password:」の UNIX の部分の文字列
retry=N	1	パスワード認証の最大試行回数
difok=N	5	新しいパスワードと古いパスワードの類似性の度合い
difignore=N	23	difok の設定を無効とする (類似していてもんだとしない) パスワードの最低長
minien=N	9	パスワードの最小複雑性。5 未満の時は 5 と見なされる。
dcredit=-N	1	数字が N 文字以上含まれていること。
ucredit=-N	1	大文字が N 文字以上含まれていること。
lcredit=N	1	小文字が N 文字以上含まれていること。
ocredit=-N	1	その他の文字 (英数字以外) が N 文字以上含まれていること。
use_authtok	なし	以前に入力されたパスワードを使用する

1.10 Apache で PAM を使うには

Apache の BASIC 認証ではユーザ名と暗号化したパスワードをファイル化し、それを利用するのが、Linux のアカウントとは別に登録作業が必要でとても煩わしい。そこで、Apache 用の PAM モジュールを導入すれば Linux のユーザアカウントを Apache の認証で利用できます。それを可能にするのが Apache のモジュール mod_authnz_external で、このモジュールは Apache の標準モジュールではないので別途ダウンロードし、インストールします。

1.11 Apache のインストール

今回の実験用に Apache の最新バージョンをソールからインストールします。ソースファイルはファイルは学内サーバ http://172.16.32.10/~sakabe/にあります。

ソースファイルを取得したら次の手順で Apache をビルド、インストールします。インストール先は/usr/local/apache2 です。

```
$ tar jxfv httpd-2.2.14.tar.bz2
$ cd httpd-2.2.14
$ ./configure --enable-rule=SHARED_CORE --enable-module=so
$ make -j2
$ sudo make install
```

図 1.2 Apache のインストール

1.12 mod_authnz_external のインストール

mod_authnz_external を http://code.google.com/p/mod-auth-external/からダウンロードします。

```
$ tar zxfv mod_authnz_external-3.2.3.tar.gz
$ cd mod_authnz_external-3.2.3
$ sudo /usr/local/apache2/bin/apxs -c mod_authnz_external.c
$ sudo /usr/local/apache2/bin/apxs -i -a mod_authnz_external.la
インストールの確認
$ grep authnz /usr/local/apache2/conf/httpd.conf
LoadModule authnz_external_module modules/mod_authnz_external.so <- 成功
```

図 1.3 mod_authnz_external のインストール

1.13 PAM 認証モジュール pwauth のインストール

pwauth は mod_authnz_external で PAM 認証するモジュールです。学内サーバよりダウンロードし、インストールします。

```
$ tar zxf pwauth-2.3.8.tar.gz
$ cd pwauth-2.3.8
$ vim config.h
267 行目を次のよう変更
#define SERVER_UIDS 72
                    ↓
#define SERVER_UIDS 2

$ make
$ sudo cp pwauth /usr/libexec
$ sudo chmod u+s /usr/libexec/pwauth
```

図 1.4 pwath のインストール

/etc/pam.d/pwauth を次の内容で作成する。

リスト 11: /etc/pam.d/pwauth ファイル

```
auth       required     /lib/security/pam_pwdb.so shadow nullok
auth       required     /lib/security/pam_nologin.so
account    required     /lib/security/pam_pwdb.so
```

Apache の設定ファイル/usr/local/apache2/conf/httpd.conf の末尾に次の二行を追加する。

リスト 12: httpd.conf への追加

```
AddExternalAuth pwauth /usr/libexec/pwauth
SetExternalAuthMethod pwauth pipe
```

httpd.conf の 152 行目を次のように変更し、.htaccess による変更を可能にする。

リスト 13: httpd.conf の変更

```
AllowOverride None
              ↓
AllowOverride All
```

1.14 PAM による認証を可能にする

PAM の認証を利用できるようにファイル.htaccess を認証したいディレクトリに設置する。今回は/usr/local/apache2/htdocs に設置し、サーバのトップページに認証を適用する。

リスト 14: .htaccess の内容

```
AuthType Basic
AuthName Your-Site-Name
AuthBasicProvider external
AuthExternal pwauth
require valid-user
```

1.15 動作確認

上記設定が完了したらブラウザからアクセスし、ログインできるかを確認する。うまくログインできないときは/usr/local/apache2/logs/error_log の内容を確認し、pwauth のファイル INSTALL に記されている対処法を読み、正しく動作させる。

1.16 演習

以下の演習を行いなさい。それぞれの演習を試した後、ログファイル (/var/log/messages や/var/log/secure) の内容も確認すること。

1. ログファイル/var/log/secure に記されている PAM のログを探し、どのモジュールが使用され、そこに記されているメッセージの意味を書きなさい。
2. /etc/pam.d にはどんなファイルがあるか確認しなさい。
3. 各自の Linux の/etc/pam.d/system-auth の内容を確認しなさい。
4. ユーザ doraemon は午前 10 時から午後 1 時までログインできないように制限する。
5. 全てのユーザが週末の午前 9 時 30 分から午後 6 時まで ssh でログインできないようにする。
6. root 以外のユーザがログインできないようにする。
7. ユーザ nobita を wheel グループのメンバにし、wheel グループのユーザでない doraemon と wheel グループのユーザに nobita の su コマンドの実行制限が機能するかを確認する。
8. 新しいパスワードは現在のパスワードと比べ 3 文字以上異なり、大文字が 2 文字以上含まれるように設定する。
9. chage コマンドについてマニュアルを調べ、有効期限の設定について確認しなさい。

第 2 章

Apache で PAM を利用

2.1 Apache で PAM を使うには

Apache の BASIC 認証ではユーザ名と暗号化したパスワードをファイル化し、それを利用するのが、Linux のアカウントとは別に登録作業が必要でとても煩わしい。そこで、Apache 用の PAM モジュールを導入すれば Linux のユーザアカウントを Apache の認証で利用できます。それを可能にするのが Apache のモジュール mod_authnz_external で、このモジュールは Apache の標準モジュールではないので別途ダウンロードし、インストールします。

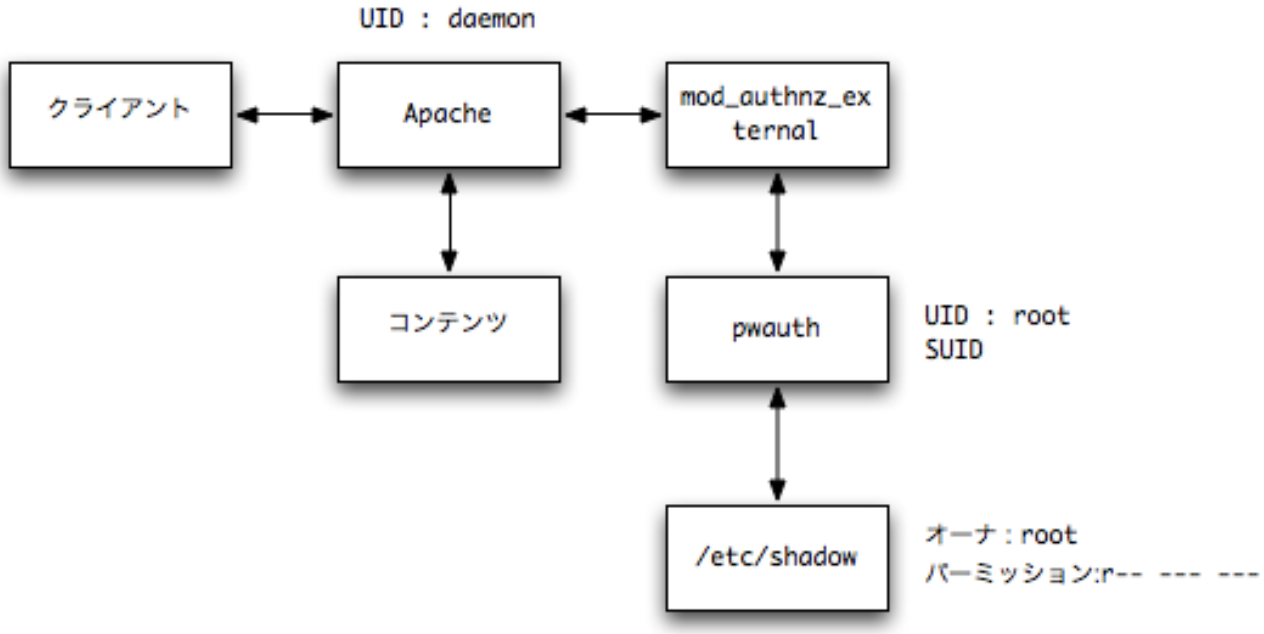

図 2.1 Apache と PAM の関係

2.2 Apache 関連ツールのインストール

Apache モジュールをソースからインストールするために必要な関連ツールをインストールします。

```
$ yum -y install httpd-devel
```

図 2.2　Apache 関連のインストール

2.3　mod_authnz_external のインストール

mod_authnz_external を http://code.google.com/p/mod-auth-external/からダウンロードし、図 2.3 の手順でインストールする。

```
# tar zxfv mod_authnz_external-3.2.5.tar.gz
# cd mod_authnz_external-3.2.5
# apxs -c mod_authnz_external.c
# apxs -i -a mod_authnz_external.la
インストールの確認
$ grep authnz /etc/httpd/conf/httpd.conf
LoadModule authnz_external_module modules/mod_authnz_external.so <- 成功
```

図 2.3　mod_authnz_external のインストール

2.4　PAM 認証モジュール pwauth のインストール

pwauth は mod_authnz_external で PAM 認証するモジュールです。
http://code.google.com/p/pwauth/よりダウンロードし、図 2.4 の手順でインストールします。

```
# tar zxf pwauth-2.3.8.tar.gz
# cd pwauth-2.3.8
# vim config.h
267 行目を次のよう変更
#define SERVER_UIDS 72
                    ↓
#define SERVER_UIDS 48

# make
# cp pwauth /usr/libexec
# chmod u+s /usr/libexec/pwauth
```

図 2.4　pwath のインストール

/etc/pam.d/pwauth をリスト 15 の内容で作成する。

リスト 15: /etc/pam.d/pwauth ファイル

```
auth       required     /lib/security/pam_pwdb.so shadow nullok
auth       required     /lib/security/pam_nologin.so
account    required     /lib/security/pam_pwdb.so
```

Apache の設定ファイル/etc/httpd/conf/httpd.conf の末尾に次の二行を追加する。

リスト 16: httpd.conf への追加

```
AddExternalAuth pwauth /usr/libexec/pwauth
SetExternalAuthMethod pwauth pipe
```

httpd.conf の変更 328 行目を次のように変更し、.htaccess による変更を可能にする。

リスト 17: httpd.conf の変更

```
AllowOverride None
               ↓
AllowOverride All
```

2.5 PAM による認証を可能にする

PAM の認証を利用できるようにファイル.htaccess を認証したいディレクトリに設置する。今回は/var/www/html/.access に設置し、サーバのトップページに認証を適用する。

リスト 18: .htaccess の内容

```
AuthType Basic
AuthName Your-Site-Name
AuthBasicProvider external
AuthExternal pwauth
require valid-user
```

2.6 動作確認

上記設定が完了したらブラウザからアクセスし、ログインできるかを確認する。うまくログインできないときは/var/logs/httpd/error_log の内容を確認し、pwauth のファイル INSTALL に記されている対処法を読み、正しく動作させる。

2.7 演習

1. サーバトップページへの認証を外す。
2. ディレクトリ/var/www/secret を作り、http://localhost/secret/でアクセスできるように設定する。適当な HTML ファイルを作り、設置する。
3. このディレクトリに PAM 認証を設定する。

第 3 章

SELinux

3.1 SELinux とは

SELinux はゼロデイアタックなどのソフトウェアの脆弱性を悪用した攻撃からシステムを守るための様々なメカニズムを持ったソフトウェアこのと。現在では Linux 標準の機能として搭載されている。

SELinux 製品としては新しいが過去の数多くの遺産を受け継いでいる。

3.2 アクセス制御モデル

MAC : Mandatory Access Control　強制アクセス制御で root も例外ではなく、制限した許可しか持たない専用のサンドボックスの中でそれぞれのプログラム実行される。プログラムが侵入されても、そのプログラムに与えられた権限の範囲内だけが危険にさらされるだけです。SELinux はこの MAC を採用して強固なセキュリティを提供する。

DAC : Discretionary Access Control　任意アクセス制御で普通の Linux が提供する機能。プログラムはそれを実行したユーザ権限下で実行され、侵入された場合、そのユーザの権限が及ぶ範囲に影響が出る。

TE : Type Enforcement　プロセスごとのアクセス制御　リソースに対する

RBAC : Role Based Access Control　ユーザごとのアクセス制御

3.3 SELinux 特徴

SELinux は次に上げるような不正使用もしくは許可されない行為からシステムを守る。

1. 許可のないデータやプログラムの読み出し
2. 許可のないデータやプログラムの変更
3. アプリケーションのセキュリティ機構の回避
4. 他のプロセスへの干渉
5. 権限の昇格行為
6. 情報セキュリティに背く行為

3.4 SELinux の仕組み

SELinux ではプログラムやプロセスをドメインと呼ぶサンドボックスに割り当て、ドメインには適切な機能は利用できるがそれ以外は何にもできない最小限の許可セットを設定する。アクセスできるファイル、そのファイルに対する動作を制限する。この許可の種類をセキュリティコンテキストという。

SELinux ではある条件下で実行しているプログラムのプロセスが現在のドメインから新しいドメインに移っていく。これをドメイン遷移という。

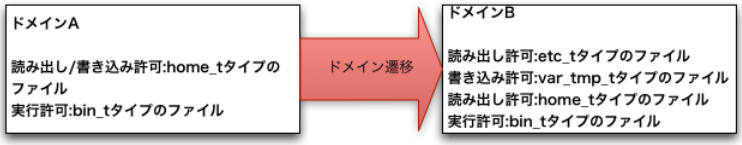

図 3.1　SELinux の動作概念

SELinux には TE[*1]というドメインに設定されたアクセス設定が常に守られるように型にはめる機能がある。TE の設定は拡張子 te をもポリシーファイルで設定する。

さらに RBAC[*2]というドメインにアクセスするユーザに制限を加える機能もある。

3.5 SELinux のセキュリティモデル

SELinux のセキュリティモデルは次の 3 つの要素で成り立っている。

サブジェクト　コンピュータ内の主体であるプロセスのこと。サブジェクトに対して動作する。

オブジェクト　サブジェクトによって操作される対象。ディレクトリ、ファイル、ファイルシステム、リンク、プロセスなど。オブジェクトクラスというオブジェクトのグループを規定している。

アクション　サブジェクトがオブジェクトに対する操作のこと。追加、作成、削除、書き込み、実行、属性取得、I/O 制御など。

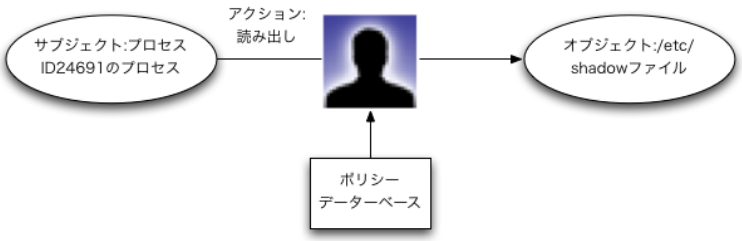

図 3.2　SELinux 基本アクセス

[*1] Type Enforcement
[*2] Role Based Access Control

3.6 セキュリティコンテキスト

SELinux では、サブジェクトとオブジェクトにセキュリティ属性というラベル情報を結びつけ、セキュリティ属性の値に基づいてアクセス可否を判定している。SELinux では次の 3 つのセキュリティ属性を使用する。

ユーザ属性 サブジェクトまたはオブジェクトと結びつける SELinux のユーザ ID でユーザアカウントが ID になる。

ロール属性 ユーザに 1 つまたは複数のロール (役割) を割り当てられる。ロールはユーザが SELinux で利用できるドメインを保持していて、ユーザに許可するアクセス権を定めている。ログインした時点でユーザは 1 つのロールだけを使用できる。

タイプ属性 SELinux がアクセス可否を判定する時に使用する属性で、特定のサブジェクトは特定のオブジェクトへアクセス許可することができる。サブジェクトに付加するとドメイン、オブジェクトに付加するとタイプと呼ぶ。

セキュリティ属性の命名規則

セキュリティ属性	命名規則	記述例
ユーザ属性	なし	root
ロール属性	ロール名_r	sysadm_r
タイプ属性	タイプ名_t	sysadm_t
	ドメイン名_t	

3.7 一時的なオブジェクトと永続的なオブジェクト

一時的なオブジェクトは存続期間が短くカーネル空間内では単なるデータ構造として存在するオブジェクトでプロセスがその例である。

永続的なオブジェクトは存続時間は無限になる。ファイルやディレクトリがその例である。

3.8 アクセス判定

SELinux セキュリティポリシーでは、以下の 2 種類の基本的な動作について判定する。

アクセス判定 特定のサブジェクトが特定のオブジェクトに対して特定の操作を実行する許可があるかを判断する。使用頻度が高く重要。

遷移判定 新しく作成したオブジェクトに割り当てるタイプを決定する。

アクセス判定にはアクセスベクタというパーミッションをそれぞれのオブジェクトクラスに割り当てられる。アクセスベクタポリシーには次の機能がある。

allow サブジェクトがオブジェクトに対して実行することを許可する操作を定義し、操作してもログを残さない。

auditallow 指定した操作を実行すると、操作を実行せずにログのみを残す。

dontaudit サブジェクトがオブジェクトに対して実行を拒否した時、その拒否ログは残さない。

表 3.1 オブジェクトへのアクセスポリシー

結果	アクセスの許可	ログの記録
一致するポリシールールがない	拒否	しない
allow	許可	しない (auditallow を指定していない場合)
auditallow	拒否	する (allow を指定していない場合)
dontallow	拒否	しない

3.9 ドメイン遷移、タイプ遷移

SELinux では、基本的に全てのオブジェクトにセキュリティコンテキストを割り当てなければならないので新しく作成したオブジェクトに何らかのセキュリティコンテキストを付ける必要がある。これは遷移判定によりどんなセキュリティコンテキストを付与するのかを指定できる。次の 2 つの遷移判定を行う。

プロセス (サブジェクト) の生成 子プロセスに親プロセスのドメインをそのまま継承するか、新しいドメインで動作させるかの判定。新しいドメインで動作する時をドメイン遷移という。

ファイル (オブジェクト) 作成 新しいファイルを作成した時、それを格納するディレクトリのセキュリティコンテキストを継承するか、新しいタイプを付与させるかの判定。新しいタイプを付与する時をタイプ遷移という。

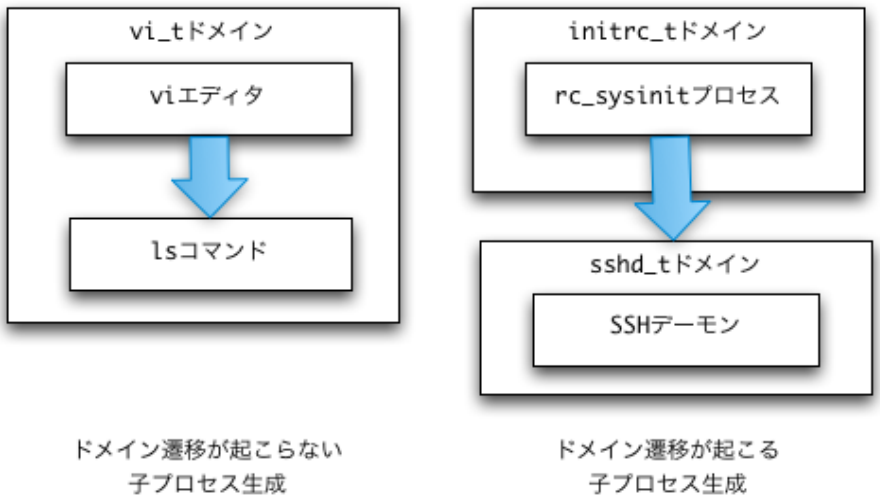

図 3.3 ドメイン遷移

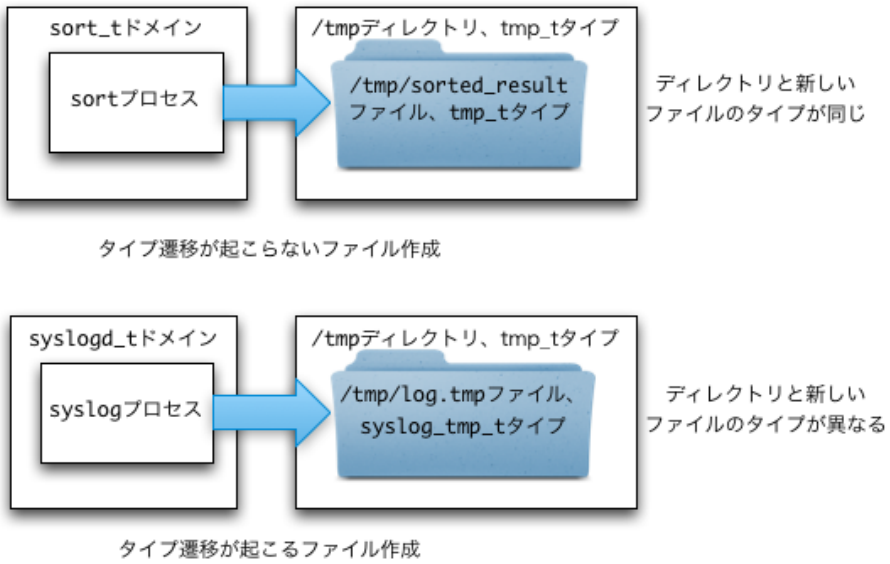

図 3.4 タイプ遷移

3.10 SELinux のアーキテクチャ

SELinux は次の要素で構成される。

1. カーネルレベルのコード
2. SELinux 共有ライブラリ
3. セキュリティポリシー
4. ツール
5. ラベル付の SELinux ファイルシステム

3.11 SELinux セキュリティポリシー

SELinux はシステム管理者の設定したポリシーに基づいてアクセス可否を判定します。ポリシーファイルは柔軟性がとても高く、あらかじめ用意された万能型のポリシーを使う以外に現場の必要に応じてカスタマイズもできる。

ポリシーファイルは/etc/selinux に格納されているバイナリファイルである。CentOS では targeted というポリシーがあらかじめインストールされている。

ポリシファイルの作成とカーネルへのロードは図 3.5 に示す手順で実行される。

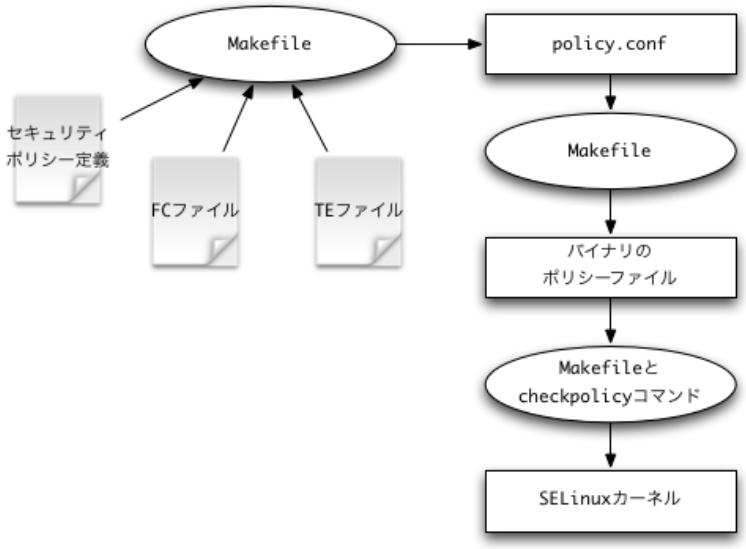

図 3.5 ポリシーの作成とロード

3.12 SELinux のツール

chcon 指定したファイルに指定したセキュリティコンテキストを付与する

setfile FC(コンテキスト) ファイルの定義内容に基づいて、指定のディレクトリとそのサブディレクトリにセキュリティコンテキストを付与する

checkpolicy ポリシーファイルをバイナリ形式にコンパイルしたり、様々なポリシー関連のアクションを実行する。ユーザが直接実行するコマンドではない。

getenforce SELinux の現在の動作モードを表示する。

setenforce SELinux の動作モードを設定する。

newrole 現在のロールを利用許可されている範囲内で別のロールに切り換える。

run_init init と同じセキュリティコンテキストを使用して、サービスを起動/停止する。

getsebool SELinux の論理値を取得する

setsebool SELinux の論理値を設定する

seinfo SELinux ポリシーの問い合わせツール

3.13 SELinux 用に修正されたコマンドやプログラム

次のコマンドやプログラムは SELinux 用に修正されている。

id ユーザの現在のセキュリティコンテキストを表示する

ls ファイルの現在のセキュリティコンテキストを表示する

ps プロセスの現在のセキュリティコンテキストを表示する

cron すべての cron ジョブにセキュリティコンテキストを設定できる

login ログイン時にセキュリティコンテキストを付加できる

logrotate　循環するログファイルのセキュリティコンテキストを維持できる
pam　ユーザにセキュリティコンテキストを付加し、SELinux API を使ってパスワード情報にアクセスできる
ssh　ログイン時にセキュリティコンテキストを付与できる

3.14 SELinux の動作モード

SELinux の動作モードは次の 2 つがある。

permissive　従来の Linux の DAC の状態で動作し、ログを残す。SELinux のトラブルシューティングのために使う。
enforcing　SELinux のデフォルトの状態でセキュリティポリシーに従ったアクセス制御を実施する。

3.15 SELinux のロール

SELinux のロールは次の 4 つが定義されている。

staff_r　sysadm_r ロールに遷移する許可のあるユーザが通常使用するロール
sysadm_r　システム管理者用のロール
system_r　システムのプロセスが使用するロール
user_r　いわゆる一般ユーザのロール

3.16 演習

各自の Linux で以下の操作等の演習を行いなさい。

1. 各自の SELinux の設定とポリシータイプを確認する
2. HTTPD に対するポリシーでホームディレクトリに対する操作はどう設定されているかを確認する。
3. SELinux の為に拡張されたコマンドで SELinux に関する情報を表示するオプションを確認しなさい。
4. root と学籍番号のユーザの SELinux に対するユーザ、ロール、ドメインを調べなさい。
5. コマンド ls、su、sudo、httpd のユーザ、ロール、ドメインを調べなさい。
6. 実行中のプロセス xinetd、dhclien、bash のユーザ、ロール、ドメインを調べなさい。
7. SELinux のログに何が記されているかを確認する。
8. SELinux を enfocing に設定し、Samba サーバを起動、実在するユーザが登録されていなければ登録し、そのユーザのホームディレクトリへのアクセスを制御する。

SELinux のポリシー・ファイルの構造と設定内容を確認する。

3.17 ポリシー・ファイル

CentOS ではインストールされた SELinux のポリシー・ファイルは/etc/selinux/targeted/policy/policy.21 で、このファイルはバイナリ形式である。

ポリシー・ファイルは図 3.6 の様にモジュールパッケージの集合である。そのモジュールはバイナリ形式のファイルで必要に応じ、追加・削除できる。初期状態では base モジュール・パッケージが組み込まれている。

ユーザがポリシーを追加したいときはテキスト形式の設定ファイルを記述し、バイナリ形式のモジュールに変換し、

ポリシーファイルに組み込む。

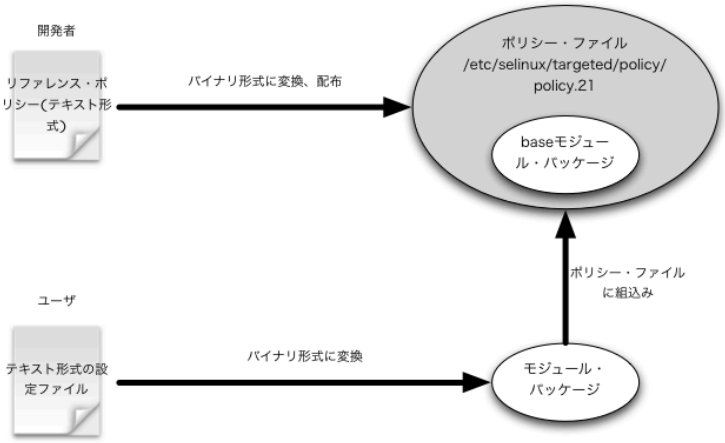

図 3.6 ポリシー・ファイル

3.18 アクセス許可の書式

ポリシーファイル設定の中核をなすのが「ドメインかタイプにどのようなアクセスが可能か」を示す、アクセス許可の設定である。その書式はリスト 19 の通り。

リスト 19: アクセス許可の allow 書式

```
allow <ドメイン> <タイプ>:<オブジェクトクラス> <パーミッション>;
```

オブジェクトクラス、パーミッションを表 3.2 と表 3.3 に示す。

表 3.2 オブジェクトクラス

オブジェクト・クラス	意味
file	通常のファイル
dir	ディレクトリ
lnk_file	シンボリック・リンク
chr_file	キャラクタ型デバイス・ファイル
blk_file	ブロック型デバイス・ファイル
tcp_socket	TCP ソケット、TCP ポート番号
udp_socket	UDP ソケット、UDP ポート番号

表 3.3 主なパーミッション

パーミッション	意味
read	読込み、受信
execute_no_trans	実行。実行されたプログラムのドメインは実行元から継承
execute	実行。実行されたプログラムのドメインは実行元から変更。共有ライブラリの利用
append	追記
write	書き込み、送信
create	書き込み、送信
unlink	ファイル消去
name_bind	ポートを接続待ちに利用
name_connect	ポートに接続

たとえば、次の文は smb_t ドメインが user_home_dir_t タイプが付与されたディレクトリに対する読み込みを可能にするという意味になる。

リスト 20: allow 文の例

```
allow smbd_t user_home_dir_t:dir read;
```

複数のパーミッションを付与したいときはそのパーミッションを {} で囲む。

3.19 モジュール・パッケージの設定書式

モジュール・パッケージを作成するときはアクセス許可設定とファイルタイプ設定の 2 つを記述したファイルが必要で、その拡張子は前者ファイルが.te、後者ファイルが.fc である。

リスト 21: te ファイルの書式

```
module local 1.0;

require {
        type httpd_t;
        class file { write create append execute };
        class dir { write rmdir read };
}

allow httpd_t httpd_sys_content_t:file { write append create };
```

リスト 22: fc ファイルの書式

```
# squid
/usr/sbin/squid          --   system_u:object_r:squid_exec_t
/usr/sbin/squid(/.*)?         system_u:object_r:squid_cache_t
/var/spool/squid(/.*)?        system_u:object_r:squid_cache_t
```

3.20 モジュール・パッケージの作成

自分でモジュールパッケージを作成し、使いする方法を確認する。SELinux が有効 (enforcing) になっている状態で fswiki が動作するようにする。

3.20.1 準備

http://172.16.32.10/~sakabe/LinuxSecurity/SELinux/から fswiki のファイル wiki3_6_3_1.zip をダウンロード、動作するように設定する (図 3.7 参照)。さらにモジュール・パッケージ作成に必要なパッケージを導入する。

```
# cd /var/www/html
# unzip wiki3_6_3_1.zip
# mv  wiki3_6_3_1 wiki
# cd wiki
# sh ./setup.sh
# setsebool -P httpd_builtin_scripting 0
# yum -y install selinux-policy-devel
# cd
# mkdir work
# cp /usr/share/selinux/devel/Makefile work
# cd work
```

図 3.7 fswiki などの準備

今回は CentOS 添付の httpd を使う。/etc/httpd/conf/httpd.conf をリスト 23 のように変更 (DocumentRoot のディレクトリ内ので CGI の実行を許可する。) し、起動する。(apachectl -k start)

リスト 23: httpd.conf の修正

```
319: Options Indexes FllowSymLinks ExecCGI
779: AddHandler cgi-script .cgi
```

SELinux が Enforcing モードの時、wiki(hhtp://サーバアドレス/wiki/wiki.cgi) にアクセスすると Internal Server Error と表示される。

3.20.2 モジュール・パッケージ作成手順

モジュール・パッケージは次の手順で作成する。

1. SELinux を permissive モードにし、ブラウザで wiki のページを表示する
2. audit2allow コマンドで te ファイルを作成
3. モジュール・パッケージに変換し、設定を反映
4. 正しく動作するまで上記手順の繰り返し

3.20.3 permissive モードで動作テスト

全ての準備が整ったら httpd を起動し、Web ブラウザから http://サーバアドレス/wiki/wiki.cgi にアクセスし、新規ページを作る。

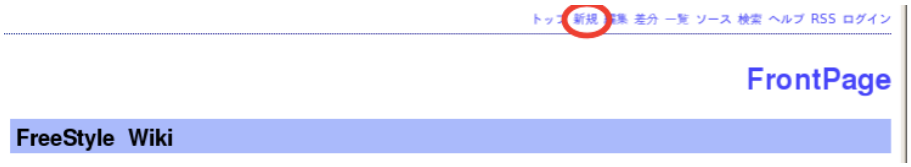

図 3.8 fswiki フロントページ

新規ページを作成したら SELinux トラブルシュータで SELinux のログを確認すると、ポリシーに違反している旨を伝えるメッセージの存在をを確認する。

3.20.4 audit2allow で te ファイルの作成

アクセス許可設定の te ファイルを SELinux のログを元に作成するのが audit2allow コマンドである。audit サービスが起動していてログが出力されているのが前提である。

図 3.9 で audit サービスのログから直前のポリシー適用後の新しいものだけを取り出し、モジュール名 local でファイル local.te を作成する。

```
# audit2allow -a -l -m local >_local.te
```

図 3.9 audit2allow コマンドで te ファイル作成

audit2allow コマンドのオプションを表 3.4 に示す。

表 3.4 audit2allow

オプション	意味
-a	audit と message ログから読む。
-d	dmesg コマンド出力から読み込む。
-h	ヘルプを表示。
-i ファイル	ファイルから読み込む。
-l	最後のポリシーをロード後から読み込む。
-m	module/require 出力を作る。
-M	ロード可能なモジュールを作る。
-o ファイル	ファイルに出力を追加する。
-r	ロード可能なモジュール向けの require 出力を作る。
-R	インストールされているマクロを使い、参照ポリシーを作る。
-v	冗長な出力を作る。

3.20.5 モジュール・パッケージへの変換

local.te ファイルをモジュール・パッケージ変換するのは make コマンドを実行するだけでよい。local.te ファイルから local.pp ファイルが生成されるのでこれをモジュール・パッケージにインストールすには semodule コマンドを使う。

```
# make
モジュールのインストール
# semodule -i local.pp
モジュールのリスト表示
# semodule -l
..
iscsid  1.0.0
local   1.0
milter  1.0.0
..
モジュールの削除
# semodule -r local
```

図 3.10 モジュール・パッケージの作成とインストールなど

以上でモジュール・パッケージのインストールが完了したので SELinux を enforcing モードに切り換え、Web ブラウザからの wiki にアクセスし、書き込みができるように調整していく。うまく動作しないときは SELinux トラブルシュータでログを確認しながら足りない設定を追加していく。

3.21 タイプの付与

上記の設定で wiki への SELinux 設定ができたがセキュリティ面の問題が残っている。/var/www/html/wiki には/var/www/html と同じタイプ (httpd_sys_content) が付与されているため、/var/www/html は読み込みのみで/var/www/html/wiki には読み書き可能な設定ができない。これを実現するには新しいタイプを設定し、/var/www/html/wiki に付与すればよい。

新しいタイプを付与する手順は次の通り。

1. 設定の見直し
2. タイプの付与 (fc、te ファイル)
3. Permissive モードでテスト
4. audit2allow で設定生成

タイプを付与する前に、te ファイルを削除し、ロードしたモジュールを削除しておく。

```
# rm local.te
# semodule -r local
```

図 3.11 local モジュールの削除

3.21.1 タイプの付与

まず、te ファイルにタイプ宣言を記す。local.te ファイルにリスト 24 の内容を入力する。

リスト 24: te ファイルでタイプ宣言

```
module local 1.0;
type wiki_write_t;
files_type(wiki_write_t)
```

3.21.2 ファイルとタイプの関連づけ

次に、fc ファイルを作成し、ファイルとタイプの関連づけを記す。local.fc ファイルの内容をにリスト 25 示す。一行で入力する。

リスト 25: ファイルとタイプの関連づけ

```
/var/www/html/wiki(/.*)? gen_context(system_u:object_r:wiki_write_t, s0)
```

3.21.3 タイプ付与の反映

te、fc ファイルを用意できたらモジュールを作り、ファイルやディレクトリにタイプを付与する。

```
# make
# semodule -i local.pp
# restorecon -RF /var/www/html/wiki
# ls -Z /var/www/html
```

図 3.12 モジュールの作成とタイプの付与

以上の操作でタイプの付与が完了した。

3.22 モジュールの更新

タイプの付与が完了したので次に付与したいタイプ (wiki_write_t) に読み書きの権限を当たる。SELinux を permissive モードにし、ブラウザでアクセスする。SELinux トラブルシュータのアイコンが表示され、警告が記録される。audit2allow コマンドを使い、ログから設定を生成し、local.te に追加する。

```
# audit2allow -a -l -r >> local.te
# make
# semodule -i local.pp
```

図 3.13 audit2allow コマンドで設定追加

以上の操作で wiki に対する新しい設定が適用される。

3.23 リファレンス・ポリシー

リファレンス・ポリシーは一定のルールに基づいて新しく策定されたマクロである。従来の SELinux が無秩序にマクロが利用され、マクロの寄せ集め的な状態でポリシーの開発でマクロの内容が理解不能になっていた。

リファレンス・ポリシーを利用して新しいポリシーを開発できる。設定が足りない時はマクロを利用してモジュールを作る。

リファレンス・ポリシーのマクロ定義ファイルは/usr/share/selinux/devel/include ディレクトリにある。

リファレンス・ポリシーは次の特徴を持つ。

- ポリシー・モジュールに対応
- ポリシーのソースファイルの構造の整理
- 様々な種別のポリシーを一括記述できる
- ドキュメントの自動生成

3.24 アプリケーションドメインの付与

ドメインの用意されていないアプリケーションにリファレンス・ポリシーを使ってドメインを付加する。次の手順でドメインを付与する。

1. te ファイルの作成
   (a) モジュール・パッケージの命名
   (b) 実行モジュールのタイプ宣言
   (c) ドメインの宣言
   (d) 提携マクロの選択と適用
2. fc ファイルの作成
   (a) 実行ファイルへのタイプ付与
3. 設定反映
4. ドメインの確認
5. 足りない設定を追加

アプリケーションとして次のシェルスクリプトを使う。ファイルを入力し、実行権を付与する。作業ディレクトリは/root/work1、付与するドメインを testapp_t とする。

リスト 26: サンプルシェルスクリプト/usr/local/bin/testapp

```
#!/bin/sh
mkdir -p /etc/testapp
while read i;
do
  echo $i > /etc/testapp/hoge
done
```

3.24.1 te ファイルの作成

testapp モジュールパッケージを作るため、testapp.te というファイルを作る。

リスト 27: testapp.te

```
# ポリシーモジュール名
policy_module(testapp, 1.0)

# 実行ファイルのタイプ宣言
type testapp_exec_t;
files_type(testapp_exec_t)

# ドメイン宣言
type testapp_t;

# ドメイン付与の定型マクロ
domain_type(testapp_t)
domain_entry_file(testapp_t, testapp_exec_t)
unconfined_domtrans_to(testapp_t, testapp_exec_t)
```

ドメイン付与設定のリファレンス・ポリシーマクロには次の物がある。

システム起動スクリプトの中で使うコマンド init_system_domain(ドメイン、実行ファイルのタイプ)

デーモン init_daemon_domain(ドメイン、実行ファイルのタイプ)

コマンドラインアプリケーション
- domain_type(ドメイン)
- domain_entry_type(ドメイン、実行ファイルのタイプ)
- unconfined_domtrans_to(ドメイン、実行ファイルのタイプ)

3.24.2 fc ファイルの作成

実行ファイル/usr/local/bin/testapp に testapp_exec_t タイプを付与する設定を testapp.fc に記す。

リスト 28: testapp.fc

```
/usr/local/bin/testapp gen_context(system_u:object_r:testapp_exec_t,s0)
```

3.24.3 設定の反映

前回と同じ方法で設定を反映する。

```
# mkdir work1
# cd work1
# cp /usr/share/selinux/devel/Makfile .
# make
# semodule -i testapp.pp
# restorecon -RF /usr/local/bin/testapp
```

図 3.14 設定の反映

3.24.4 ドメインの確認

設定を正しく反映したら、permissive モードでスクリプトを実行、ドメインが正しく付与されているかを確認する。

```
# setenforce 0
# /usr/local/bin/testapp
```

図 3.15 ドメインの確認

testapp は入力待ちとなる。別端末で ps -eZ | grep testapp を実行すれば確認できる。スクリプト testapp を終了するには Ctrl+D とキー操作すればよい。

```
# ps -eZ | grep testapp
user_u:system_r:testapp_t         719 pts/1    00:00:00 testapp
```

図 3.16 ドメインの確認

3.24.5 足りない設定の追加

このままでは enforcing モードでは動作しないので前回と同じ方法で足りない設定を追加する。

リスト 29: te および fc ファイルへの追加

● te ファイルへの追加

```
type testapp_write_t;files_type(testapp_write_t)
allow testapp_t testapp_write_t:dir create_dir_perms;
allow testapp_t testapp_write_t:file create_file_perms;
```

● fc ファイルへの追加

```
/etc/testapp(/.*)? gen_context(system_u:object_r:testapp_write_t,s0)
```

これで/etc/testapp 以下に testapp_write_t というタイプが付加される。続いて設定を反映する。

```
# make
# semodule -i testapp.pp
# restorecon -RF /etc/testapp
```

図 3.17 設定の反映

設定を反映できたらスクリプトを実行し、SELinux トラブルシュータの表示を確認し、警告が出なくなるように調節する。

```
# audit2allow -a -l -r >> testapp.te
# make
# semodule -i testapp.pp
```

図 3.18 設定の追加

3.25 MCS でユーザやファイルを分類

MCS(Multi Category Security) を使うとユーザの権限をカテゴリという分類分けでアクセス制御できる。ユーザやファイルにはカテゴリと呼ぶラベルを付加し、ファイルのカテゴリの全てに一致するカテゴリのユーザのみがそのファイルにアクセスできるというルールで動作する。つまり、root にもアクセスできないファイルを作ることができるわけである。

カテゴリには c0,c1,c2,..c255 と 256 種類の名前が使える。

3.25.1

SELinux は初期上程ではカテゴリを設定していないのでユーザやファイルにカテゴリを設定する必要がある。カテゴリの設定は chcat コマンドを使う。

ファイルにカテゴリを設定 chcat カテゴリファイル名
ファイルのカテゴリ設定を解除 chcat -d ファイル名
ユーザにカテゴリを設定 chcat -l カテゴリユーザ名
ユーザのカテゴリ設定確認 chcat -l -L ユーザ名

コマンドの使用例を示す。コマンドの実行には少し時間がかかる。

```
# chcat -l c0 root
# chcat -l c1 nobita
# chcat -L -l root
root: s0:c0
# chcat -L -l nobita
nobita: s0:c1

ユーザ nobita で
$ chcat c1 test.txt
$ ls -Z test.txt
-rw-r--r-- sakabe wheel user_u:object_r:user_home_t:s0:c2   test.txt

root で
# cat ~sakabe/test.txt
SELinux カテゴリ設定
# setenforce 1
# cat ~sakabe/test.txt
cat: /home/sakabe/test.txt: 許可がありません
```

図 3.19　カテゴリ設定例

ユーザにカテゴリが反映されるのは再ログイン後になる。id コマンドでユーザのカテゴリを確認できる。

```
$ id -Z
user_u:system_r:unconfined_t:s0:c2
```

図 3.20　ユーザのカテゴリ確認

カテゴリは別名設定を設定できる。別名は/etc/selinux/targeted/setrans.conf ファイルに記す。別名を設定すれば、カテゴリ設定のその名前を使える。

リスト 30: カテゴリの別名設定

```
s0:カテゴリ=別名
s0:c1=is02
```

3.26 演習

ドメインの付与、カテゴリの設定を実際に行い、その効果を確認しなさい。

第 4 章

Tripwire

オープンソース整合性チェックプログラム Tripwire による IDS を構築する。

4.1 改ざんの検出

UNIX コマンドでファイルの改ざんを検出する方法は次の通り。これらの操作は別々に実行しなければならない。

1. diff コマンドでファイルの差分
2. md5 コマンドのハッシュ値
3. ls コマンドでファイルの属性

4.2 Tripwire とは

Tripwire は米 Purdue 大学の COAST プロジェクト (http://www.cerias.purdue.edu/coast/) で開発されたソフトウェアで UNIX の整合性チェックを行うもので、次のような特徴がある。

1. 複数の手法による整合性チェック (ハッシュ、ファイル属性チェック)
2. 監視対象ファイルの細かな制御
3. 監査結果のレポートのメール送信
4. 設定ファイルの暗号化

Tripwire は商用化されたものとフリーウェアのものがあり、今回はフリーウェアの Tripwire を導入する。

4.3 インストール

学内サーバよりファイル tripwire-2.4.2-src.tar.bz2 をダウンロードし、インストールする。

```
$ su -
# tar jxf tripwire-2.4.2-src.tar.bz2
# cd tripwire-2.4.2-src
# ./configure --prefix=/opt/tripwire
# make
# vim install/install.cfg
96 行目のコメントを外す
TWMAILMETHOD=SMTP
# make install
Installer program for:
Tripwire(R) 2.4 Open Source
(中略)
Press ENTER to view the License Agreement.
ライセンスの表示、スペースキーを押し全てを表示する。

Please type "accept" to indicate your acceptance of this
license agreement. [do not accept] accept と入力
(中略)
Continue with installation? [y/n]y と入力

-------------------------------------------
Creating key files...
(中略)
Enter the site keyfile passphrase:キーパスフレーズを入力
Verify the site keyfile passphrase:キーパスフレーズを再入力
Generating key (this may take several minutes)...Key generation complete.
(中略)
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再入力
Generating key (this may take several minutes)...Key generation complete.
(中略)
-------------------------------------------
Creating signed policy file...
Please enter your site passphrase: サイトパスフレーズを入力
Wrote policy file: /opt/etc/tw.pol
(中略)
make[1]: ディレクトリ ‘/home/sakabe/LinuxSecurity/tripwire-2.4.2-src’ から出ます
```

図 4.1 tripwire のインストール

インストールが完了するとディレクトリ/opt/tripwire/etc に設定に関する twcfg.txt と twpol.txt が、/opt/tripwire/sbin に twadmin と twprint の実行ファイルがインストールされる。

4.4 設定

インストール完了時点で次のファイルが生成されているが、データベースがないので生成する。

- 暗号・署名鍵の生成
- 設定ファイルの生成
- ポリシーファイルの生成

4.5 ポリシーの最適化

ポリシーファイルには存在しないファイルのチェックが有効だったり、存在しないファイルのチェックが無効だったりするのでこれを修正し、最適化する。学内サーバから perl スクリプト twpolmake.pl をダウンロードしておく。

```
# PATH=$PATH:/opt/tripwire/sbin
# cd /opt/tripwire/etc
# perl twpolmake.pl twpol.txt > twpol.txt.new
# twadmin -m P -c tw.cfg -p tw.pol -S site.key twpol.txt.new
```

図 4.2 ポリシーの最適化

4.5.1 データベースの新規作成

ポリシーファイルを元に整合性チェック用のデータベースを作成する (図 4.3)。データベース作成に必要な時間はシステムの性能、ファイル数に左右される。

```
# tripwire -m i -c tw.cfg
Please enter your local passphrase: ローカルパスフレーズを入力
Parsing policy file: /opt/etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
The object: "/misc" is on a different file system...ignoring.
The object: "/net" is on a different file system...ignoring.
The object: "/selinux" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
The object: "/var/lib/nfs/rpc_pipefs" is on a different file system...ignoring.
The object: "/mnt/hgfs" is on a different file system...ignoring.
Wrote database file: /opt/tripwire/lib/tripwire/beta.localdomain.twd
The database was successfully generated.
```

図 4.3 データベースの新規作成

4.6 整合性の確認

データベースの作成が完了したら整合性をチェックする。(図 4.4) コマンドの実行が終わるとレポートファイルが /opt/tripwire/lib/tripwire/report に生成される。

```
# tripwire -m c -c tw.cfg
Parsing policy file: /opt/tripwire/etc/tw.pol
*** Processing Unix File System ***
Performing integrity check...
Wrote report file: /var/lib/tripwire/report/XXXXXX.twr
(中略)
All rights reserved.
Integrity check complete.
```

図 4.4 Tripwire の整合性のチェック

整合性の確認は定期的に実施するべきで cron でスケジューリングするのがよい。

4.7 データベースの更新

整合性の確認後、レポートの内容を基にデータベースを更新する (図 4.5)。レポートの内容から適切な変更か否かを判断し、その情報を更新する。[x] の表示の x を消すとこの情報は更新の対象にならない。レポートの内容を保存すれば、データベースは更新される。

```
# env LANG=C tripwire -m u -r /opt/tripwire/lib/tripwire/report/XXXXXX.twr
レポートの内容が表示される。必要に応じ編集し、終了。
Please enter your local passphrase: 設定したパスフレーズを入力
Wrote database file: /opt/tripwire/lib/tripwire/fc6.localdomain.twd
```

図 4.5 データベースの更新

4.8 演習

1. 自分のシステムに Tripwire をインストール、動作させる。
2. 整合性のチェックを実行する前に次のコマンドを実行し、その結果がレポートに表示されていることを確認する。

   ```
   touch /etc/hosts
   touch test00.txt
   ```

3. 定期的に Tripwire を実行するにはどうすればよいか。
4. Tripwire の問題点は何か。

4.9 プロパティマスク

Tripwire のポリシーファイルの冒頭部にも記されているファイルやディレクトリのプロパティをチェックするか否かを指定する値をプロパティマスクといい、プロパティの前に + があればチェック、−があれば無視します。プロパティマスクの一覧を表 4.1 に示す。

表 4.1 プロパティマスク

プロパティ	説明
a	アクセス時間 (+CMSH とは排他的に指定)
b	割り当てられてブロック数
c	i ノードの作成・修正時刻
d	i ノードのあるデバイス ID
g	オーナのグループ ID
i	i ノード番号
l	サイズの大きくなるファイル (例えばログファイル)
m	修正時刻
n	リンク数
p	パーミッションとファイルのモードビット
r	i ノードで指定されるデバイスの ID
s	ファイルサイズ
t	ファイルタイプ
u	オーナのユーザ ID
C	CRC-32 ハッシュ
H	HAVAL ハッシュ
M	MD5 ハッシュ
S	SHA ハッシュ

4.10 定義済みプロパティマスク

ポリシーファイルには表 4.2 のプロパティマスクが定義済みである。表 4.1 のプロパティマスクそのものを使うより、定義済みマスクを使う方が簡単です。

表 4.2 定義済みプロパティマスク

名前	説明	マスク
Device	内容ではなく属性をチェックすべきデバイスやファイル	+pugsdr-intlbamcCMSH
Dynamic	定期的に変更が発生するユーザディレクトリなど	+pinugtd-srlbamcCMSH
Growing	サイズは巨大化するが他の方法では変更されないファイル	+pinugtdl-srbamcCMSH
IgnoreAll	ファイルの存在のみをチェックすればよいファイル	-pinugtsdrlbamcCMSH
IgnoreNone	あらゆるプロパティをチェックする。カスタムマスクを定義するときに使う。	+pinugtsdrbamcCMSH-l
ReadOnly	読み込み専用ファイル	+pinugtsdbmCM-rlacSH
Temporary	一時ファイル	+pugt

4.11 カスタムポリシーの作成

既存のテキスト形式のポリシファイル twpol.txt.new を編集し、カスタムポリシーを追加する。

二年次実習の利用した Apache に関連するファイルやディレクトリ (/home/httpd と/www) に対するポリシーを次の要件に従い追加する。

- ルール名:MyWeb
- 変数 WEBROOT を定義し、その値は Apache の DocumentRoot のディレクトリにし、ルールは ReadOnly でサブディレクトリは 1 階層下までをチェックする。
- 変数 CGIBIN を定義し、その値は Apache の cgi-bin ディレクトリにし、ルールは ReadOnly を適用する。
- Apache のログディレクトリ/www/logs にルール Growing を適用する。

以上の修正をし、前回も実行したポリシファイル作成コマンドを実行し、データベースを再初期化し、成功性をチェックする。

```
# twadmin -m P -c tw.cfg -p tw.pol -S site.key twpol.txt.new
# データベースの再初期化
Apache の起動、ブラウザでのアクセスを実行
Linux の再起動などの操作
# 整合性チェック
# twprint -m r --twrfile レポートファイル
```

図 4.6 データベース再初期化など

4.12 まとめ

以上の実習が終了したら下記の内容をまとめ、次回の実習開始時までにファイルを学内サーバ (http://172.16.32.10/~sakabe/) にアップロードするかメール (sakabe@hac.neec.ac.jp) で提出する。ファイル形式はテキスト、Word、OpenOffice、HTML の何れかとし、ファイル名は**学籍番号_LS_100427** とする。

- Tripwire でできることは何か。
- Tripwire で定期的に、例えば毎日午前 4 時に整合性をチェックするためにはどんな設定が必要か。
- Tripwire の欠点は何か。
- 今回の実習で作ったカスタムポリシーとレポート

第 5 章

Snort

Tripwire に続き、IDS の snort を導入し、着信した要求の監視を体験する。

snort はシステムに着信したネットワーク要求を監視し、怪しげな動きのあった場合にアラートを発生し、その証拠の痕跡を残すソフトウェアです。

5.1 snort の導入

snort の導入には以下のライブラリやプログラムが必要になる。ソースコードからのビルド、あるいはパッケージのインストールをしておく。

snort のソースファイルは http://www.snort.org/からダウンロードできるが、学内サーバ (http://172.16.32.10/~sakabe/) に置いてある。

- libpcap :パケットキャプチャライブラリ
- libpcre : Perl 互換正規表現ライブラリ
- mysql、mysql-server、mysql-devel : MySQL サーバ

以下に snort の導入例を示す。

```
$ rpm -q libcap
# rpm -ivh snort-2.8.6-1.RH5.i386.rpm
# rmp -ivh snort-mysql-2.8.6-1.RH5.i386.rpm
MySQL で snort を使うためのユーザ、データベース作成
$ mysql -u root -p
> GRANT ALL PRIVILEGES ON *.* TO snort@localhost IDENTIFIED BY 'snort' WITH GRANT OPTION;
>exit;
$ echo "CREATE DATABASE snort;" | mysql -u snort -p
Enter password:
$ mysql -D snort -u snort -p < /usr/share/snort-2.8.6/schemas/create_mysql
Enter password:
# mkdir -p /etc/snort/rules
# cp ./etc/*.{conf,config,map} /etc/snort
# tar zxf snort_rules.tar.gz -C /etc/snort
```

図 5.1 snort の導入例

snort を動作させるにはルールファイルが必要で snort のサイトでユーザ登録し、入手する。

5.1.1 snort.conf の編集

snort.conf の次の箇所を変更する。コメントになっている場合はそれを外す。

監視対象の NIC あるいはホストアドレス。 var HOME_NET $eth0_ADDRESS

データベースとホスト指定 ルールファイルの所在指定

var RULE_PATH /etc/snort/rules output database: log, mysql, user=root password=root dbname=snort host=localhost

アラートの出力を指定 output alert_syslog: LOG_AUTH LOG_ALERT

5.1.2 動作確認

インストールと設定が完了したら次のコマンドで動作を確認する。出力最後の方にバージョン、コピーライト表示が出ればよい。

```
# snort -A full -c /etc/snort/snort.conf
(中略)
        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.8.5.1 (Build 114)
'''' By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2009 Sourcefire, Inc., et al.
           Using PCRE version: 7.8 2008-09-05
```

図 5.2 snort の動作確認

つぎに検証用 PC の端末から nmap コマンドを使い、自分の Linux に対してポートスキャンを実行する。コマンド実行後、Ctrl+C で snort の実行を停止し、var/log/snort に alert という名前のファイルが生成されていることを確認し、その内容を見る。もし、nmap がインストールされていなければ、インストールする。

```
# nmap -P0 172.16.11.xx
自分の Linux で開いているポートの一覧が表示される。
```

図 5.3 ポートスキャンの検出

snort の出力したアラートの内容は次のようになり、どの IP のホストからどの IP のホストへポートスキャンが実行されたかが分かる。

リスト 31: /var/log/snort/alert の内容の一部

```
[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
12/09-11:38:04.669027 172.16.114.1 -> 172.16.114.129
PROTO:255 TTL:0 TOS:0x0 ID:0 IpLen:20 DgmLen:162 DF
```

5.1.3 snort の使用例

1. ネットワークトレース情報を整形して表示する

   # snort -v [-d|-X] [-C] [-e] [フィルタ条件式]

2. ネットワーク上のパケットを傍受する

   # snort [-i インターフェース] [-P スナップショットの最大長] [フィルタ条件式]

3. 以前保存したネットワークトレース情報を参照する

```
# snort -r ファイル名［フィルタ条件式］
```

4. ログをバイナリ形式で作成し、警告はシステムロガーに送信する

```
# snort -c /etc/snort/snort.conf -b -s
```

5. snort をデーモンとして起動し、バックグランドで実行する

```
# snort -D [-u user] [-g group] [-m umask] -c ...
```

表 5.1 snort のオプション

オプション	意味
-v	パケットごとのプロトコル情報の要約
-d	16 進形式や ASCII 形式表示
-X	プロトコルヘッダ表示
-e	Ethernet ヘッダ表示
-i	インターフェス指定
-r	保存されている libcap 形式ファイルの読み込み
-c	設定ファイルの指定
-D	デーモンモード指定
-b	バイナリの libcap 形式でログを取る
-s	警告メッセージをシステムロガーに送信する

5.2 BASE

BASE(Basic Analysis and Security Engine) を使うと snort の出力を Web ブラウザを使って GUI で参照できる。BASE を使うには次のプログラム類をインストールする。

base と adodb は学内サーバ (http://172.16.32.10/~sakabe/) からダウンロードできる。

1. base-1.4.5.tar.gz
2. adodb511.tgz
3. php-gd パッケージ
4. php-pearp パッケージ
5. php-mysql パッケージ

5.2.1 BASE インストール

次の手順でインストールする。

```
# tar zxf base-1.4.5.tar.gz
# mv base-1.4.4 /var/www/base
# unzip adodb511.tgz
# mv adodb51 /var/www/base/adodb
# chown -R apache:apache /var/www/base
# cp /var/www/base/base_conf.php.dist /var/www/base/base_conf.php
```

図 5.4 BASE 等のインストール

5.2.2 BASE の設定

コピーした BASE の設定ファイル (/var/www/base/base_conf.php) の次の五項目を修正する。

言語指定 $BASE_Language = 'japanese';
BASE **関連ファイルの** URL $BASE_urlpath = '/base';
データベースのパス $DBlib_path = '/var/www/base/adodb';
データベース名 $archive_dbname = 'snort'; $alert_dbname = 'snort';
データベースパスワード $alert_password = 'snort';

5.2.3 パッケージのインストール

BASE は PHP のスクリプトで PHP でグラフを作る為のパッケージ類が必要になる。

```
# yum -y install php-mysql
# yum -y install php-gd php-pear
# pear config-set http_proxy http://cache2.st1.hac.neec.ac.jp:8080/
config-set succeeded <- 成功したというメッセージを確認する
# pear upgrade PEAR-1.5.4
# pear upgrade PEAR-1.9.0
# pear channel-update pear.php.net
# pear upgrade-all
# pear install --alldeps Image_Graph-alpha
(中略)
install ok: channel://pear.php.net/Image_Canvas-0.3.2
install ok: channel://pear.php.net/Numbers_Roman-1.0.2
install ok: channel://pear.php.net/Image_Graph-0.7.2
```

図 5.5 PHP 関連パッケージのインストール

5.2.4 httpd 用 BASE 設定ファイルの作成

BASE へのアクセスできるホストを限定するために httpd の設定ファイル/etc/httpd/conf.d/base.conf を次の内容で作成する。

リスト 32: base.conf の内容

```
Alias /base /var/www/base
<Directory "/var/www/base">
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
  Allow from 172.16.10.0/24
</Directory>
```

5.2.5 PHP の設定

PHP のエラーなどのメッセージ出力範囲を設定し直す。/etc/php.ini ファイルを編集する。

リスト 33: php.ini の編集

```
error_reporting = E_ALL & ~E_NOTICE  ← 行頭の; を削除してコメント解除

;error_reporting  =  E_ALL  ← 行頭に; を追加してコメントアウト
```

5.2.6 動作確認

BASE を動作させる設定が完了したら MySQL、Apache、snort を起動し、Web ブラウザから BASE(http://localhost/base/) にアクセスし、snort の検出結果を確認する。

```
# service mysqld start
# service httpd start
# snort -Dd -c /etc/snort/snort.conf
```

図 5.6 サービス類の起動

初回アクセス時は図 5.7 のように表示される。この時点ではセットアップが完了していないので**セットアップページ**というリンクをクリックし、設定を続ける。

図 5.7 BASE、初回アクセス時の表示

セットアップ-ページをクリックすると表示は図 5.8 に変わる。
次にボタン **Create BASE AG** をクリックし、BASE に関わるデータベースのテーブルの作成する。

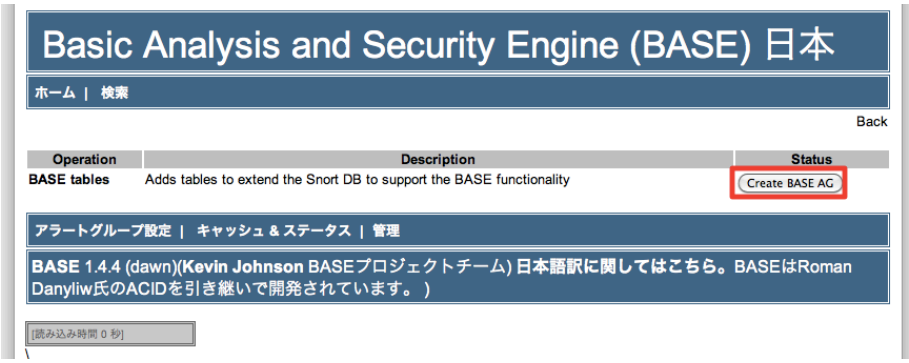

図 5.8　BASE、セットアップ-ページ

画面は図 5.9 のように成功したことを伝えるメッセージが表示される。
最後にページ下部の **Main page** のリンクをクリックすればこれまでのデータを元にした TCP、UDP、ICMP、ポートスキャンの情報を表示するメインページに変わる (図 5.10)

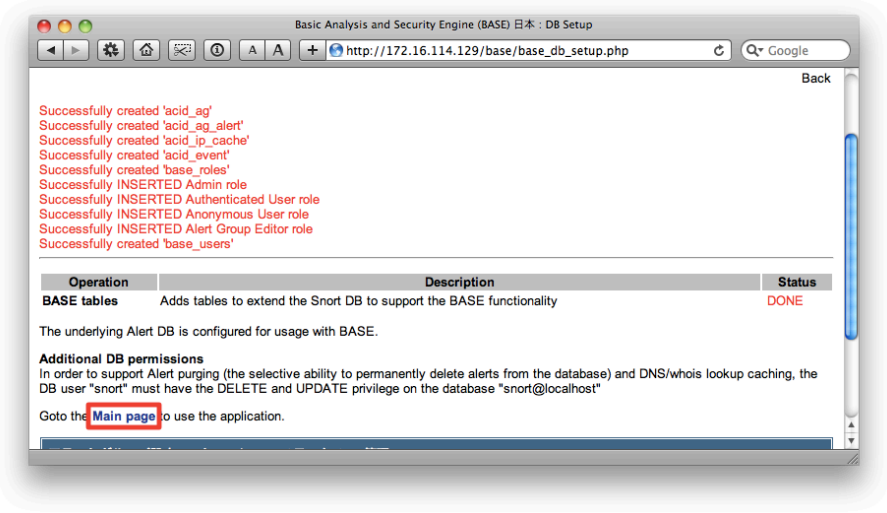

図 5.9　BASE、セットアップ完了

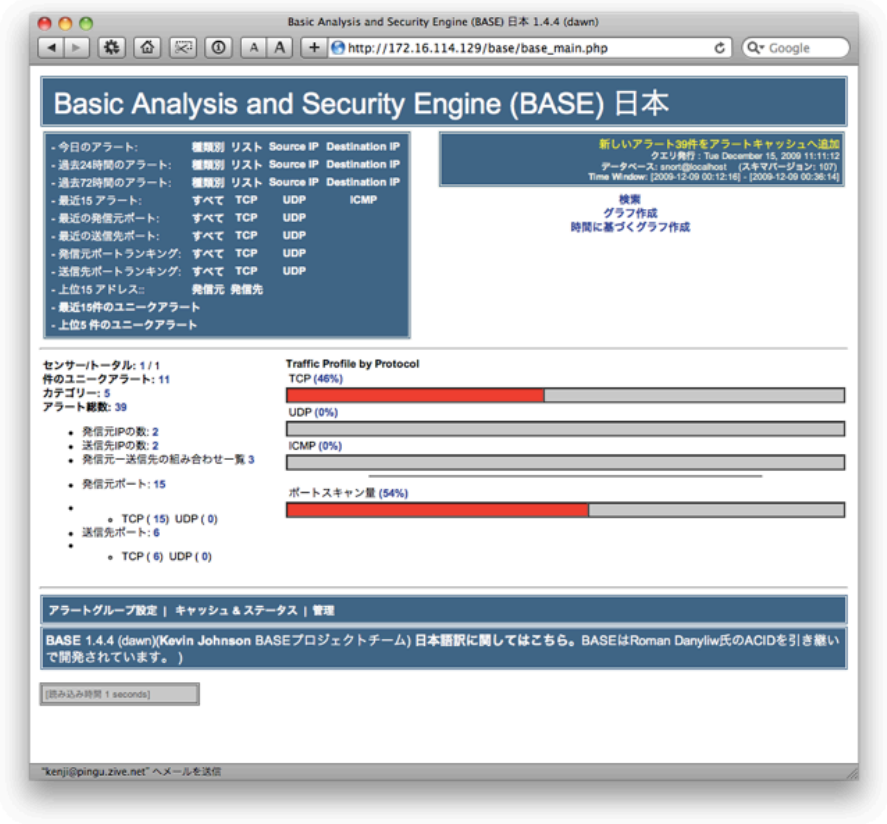

図 5.10　BASE、メインページ

メインページ左上にはアラートを表示するためのリンクがある。これらをクリックすればアラートを細かく見られる。また、ページ右側にあるグラフ作成リンクのページに移動するればアラートをいろいろは形で見られる。以下の出力例を示す。

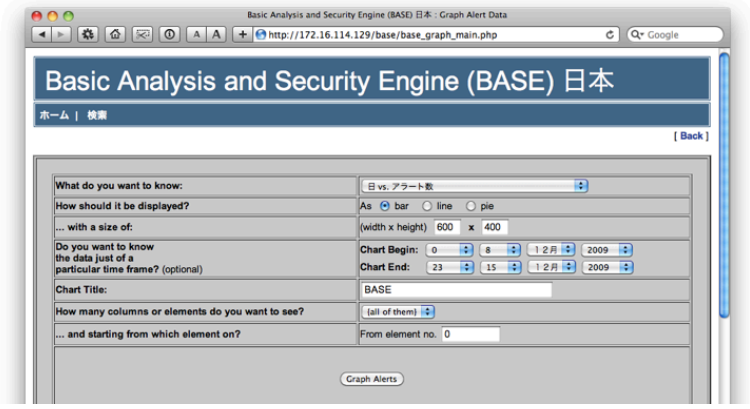

図 5.11　BASE グラフ作成設定

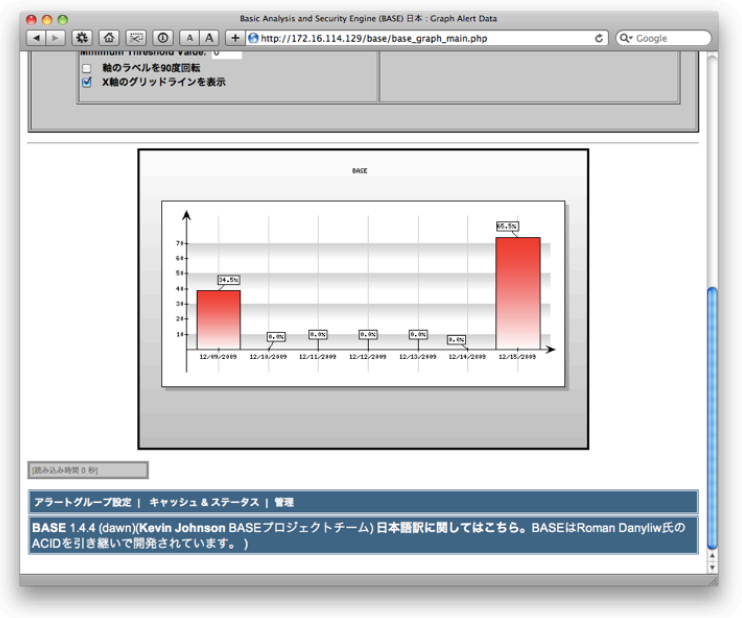

図 5.12 BASE グラフ

5.3 SnortALog

BASE にはレポート機能がないのでそれを補う SnortALog を導入し、テキスト、HTML、PDF 形式でのレポートを作成する。

5.3.1 インストール

```
# export http_proxy=http://cache2.st1.hac.neec.ac.jp:8080/
# perl -MCPAN -e 'install GD::Graph'

# tar zxf snortalog_v2.4.2.tgz
# cd snortalog
# cp -r picts /var/www/html
```

図 5.13 インストール

5.3.2 実行

snortalog で snort の出力レポートは次のコマンドで作ります。アラートファイルを/var/www/html/snort.html に日本語 (EUC) で出力します。出力されたログは http://localhost/snort.html で表示できる。

```
# cat /var/log/snort/alert | ./snortalog.pl -n 100 -report
  -o /var/www/html/snort.html -g png -l ja -pictsdir ./picts/
```

図 5.14　レポート作成

図 5.15　SnortALog 出力例

第 6 章

SNMP

SNMP[*1]によるサーバやネットワークのトラフィックの監視方法を学ぶ。今回は Net-SNMP を利用する。

6.1 SNMP

サーバのディスクやメモリ容量、CPU に対する高負荷などリソース不足を監視するのが SNMP である。SNMP は UDP で動作し、管理用の機器のマネージャと管理対象の機器のエージェントに分かれており、マネージャとエージェント間での通信プロトコルが SNMP である。エージェントはルータ、プリンタなどの機器が該当するが全ての機器で SNMP が利用できるわけではない。

エージェントには MIB[*2]と呼ぶ管理情報データベースを持っている。MIB はツリー状のデータで数字列を使って表現していて RFC で提唱されている標準 MIB とメーカ独自の拡張 MIB がある。

SNMP マネージャは MIB 情報の取得、MIB 情報の設定、SNMP トラップの受信の 3 つの機能を実現している。SNMP トラップはエージェントにあらかじめ設定した異常値を検出したときにそれをエージェントに伝える機能のことである。

MIB は番号をピリオドでつなげたものを OID(Object ID) という。

図 6.1 の OID1.3.6.1.2.1.1 は iso.org.dod.internet.mgmt.mib-2.system とも表記できる。Net-SNMP の MIB は http://net-snmp.sourceforge.net/docs/mibs/に記されている。

SNMP はバージョンにより次のような違いがある。

SNMPv1　コミュニティ名による平文のパスワードで認証する。セキュリティ上の問題から接続元 IP アドレスを制限する必要がある。SNMP トラップでの再送確認はない

SNMPv2c　v1 と同じ平文での認証。SNMP トラップの再送確認あり

SNMPv3　ユーザレベルの暗号化された認証。SNMP トラップでの再送確認あり。

[*1] Simple Network Managemento Protocol

[*2] Management Information Base、ミブ

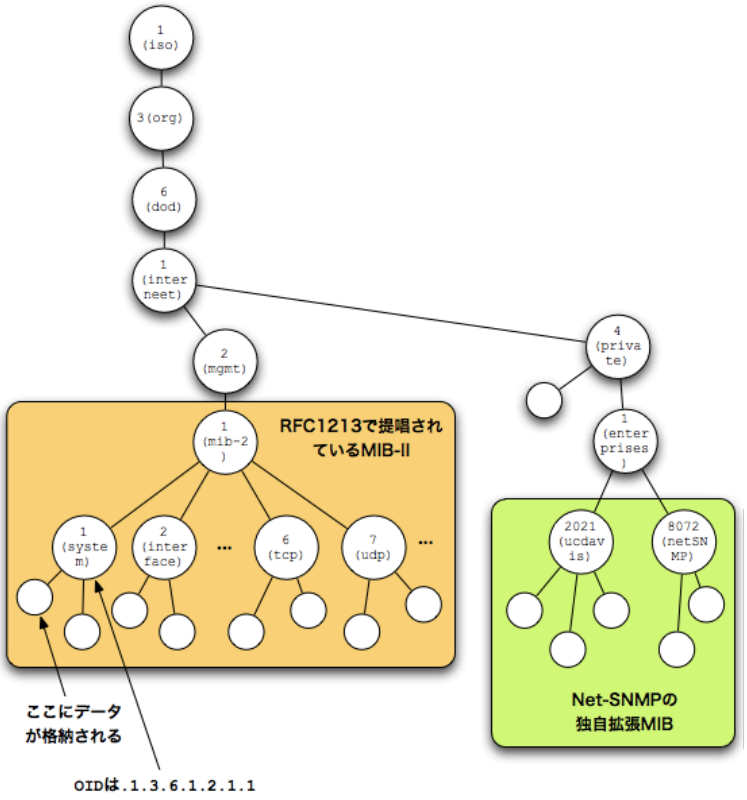

図 6.1　MIB のツリー

6.2　SNMP のインストール

SNMP の実装例として net-snmp があり、yum でインストールできる。

```
$ su -
# yum -y install net-snmp net-snmp-utils
# mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.org
```

図 6.2　net-snmp のインストール

6.3　net-snmp の設定

net-snmp の設定ファイルは/etc/snmp/snmpd.conf で、ファイルがインストールされるが、このファイルには手を加えず、新しいファイルを作る。オリジナルのファイルは名前を変更しておく。図 6.2

設定ファイルの内容をリスト 34 に示す。ファイルを保存したら root のみが読み書きできるようにパーミッションを変更しておく。

リスト 34: snmpd.conf の内容

```
# システムに関する設定
syslocation  myserver
# 管理者の連絡先
syscontact  admin@example.com
## アクセスコントロール
# SNMPv3 に関する設定
# mywriter は読み書き可能で認証を必要とする
# 認証を必須とする
rwuser  mywriter auth
# myreader は読み取りのみで認証を必要とする
rouser  myreader
# SNMPv1/v2c に関する設定
# myprivate は読み書き可能。localhost だけに制限する
rwcommunity myprivate localhost
# mypublic は読み取りのみ可能。172.16.11.0/255.255.255.0 だけに制限する
rocommunity  mypublic 172.16.11.0/255.255.255.0
```

認証のパラメータ noauth、auth、priv はそれぞれ認証無し、暗号化不要の認証、暗号化必須の認証を意味する。

6.4 動作確認

snmpd.conf を保存できたら SNMP のサービスを起動する。

```
# service snmpd start
```

図 6.3 SNMP サービスの起動

サービスが起動できたら SNMPv1 および v2c でコミュニティ名 myprivate で OID1.3.6.1.2.1.1.4.0 に設定されている情報を snmpget コマンドで表示してみる。この OID は管理者の連絡先である。

```
# snmpget -c myprivate -v 1 localhost .1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING: "admin@example.com"
```

図 6.4 OID1.3.6.1.2.1.1.4.0 の情報取得

snmpget コマンドは OID ツリーに相当するオブジェクトの値を表示するコマンドで、ツリーを展開したいときは snmpwalk コマンドを使う。SNMP 関連コマンドの共通オプションを示す。

表 6.1 SNMP コマンドの共通オプション

オプション	意味
-a	SNMPv3 の認証方法の指定。MD5 か SHA
-A	SNMPv3 の認証パスワードの指定。
-c	SNMPv1 および v2c のコミュニティ名の指定
-l	SNMPv3 のセキュリティレベル指定。認証と暗号をしない noAuthNoPriv、認証のみの authNoPriv、認証と暗号化をする authPriv
-x	暗号化プロトコル指定。DES か AES
-X	authPriv の指定の時の暗号化パスワード指定
-v	SNMP のバージョン指定 1、2c、3 のいずれか
-O	表示形式の指定。-On で OID、-Os でシンボル出力

snmpwalk コマンドを使い、OID 1.3.6.1.2.1.1(iso.org.dod.internet.mgmt.mib-2.system) のツリーを表示する。なお、SNMP のバージョンは 1 で実行する。

```
# snmpwalk -v 1 -c myprivate localhost .1.3.6.1.2.1.1
SNMPv2-MIB::sysDescr.0 = STRING: Linux fc6.localdomain .6.22.14-72.fc6
 #1 SMP  Wed Nov 21 15:12:59 EST 2007 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
（中略）
SNMPv2-MIB::sysORUpTime.7 = Timeticks: (2) 0:00:00.02
SNMPv2-MIB::sysORUpTime.8 = Timeticks: (2) 0:00:00.02
```

図 6.5 snmpwalk コマンドでツリーを展開

実習室のレーザプリンタに同じ OID ツリーを表示してみる。なお、コミュニティ名は public を指定する。

```
# snmpwalk -v 1 -c public 172.16.11.201 .1.3.6.1.2.1.1
SNMPv2-MIB::sysDescr.0 = STRING: EPSON Type-B 10Base-T/100Base-TX Print Server
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::
  enterprises.1248.1.1.2.1.3.5.69.73.80.69.54
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (292056116) 33 days, 19:16:01.16
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: LP-9100-D6DDD9
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 72
```

6.5 MIB の値を設定する

MIB の値を設定するときは snmpset コマンドを使う。設定する値の型の指定が必要。

表 6.2 MIB の型指定

型名	型	型名	型
i	整数	u	符号無し整数
s	文字列	x	16 進文字列
d	10 進文字列	n	NULL
o	オブジェクト ID	t	時間
a	IP アドレス	b	2 進数

システム名を www.example.com に変更するときは図 6.6 のように実行する。

```
# snmpset -v 1 -c myprivate localhost .1.3.6.1.2.1.1.5.0 s www.example.com
SNMPv2-MIB::sysName.0 = STRING: www.example.com
```

図 6.6 システム名の変更

全ての値を変更できるわけではない。

6.6 ユーザ登録

SNMPv3 用のユーザ mywriter と myreader を使うには snmpd.conf にユーザ名、認証種別、パスワード、暗号化種別、暗号化パスワードをリスト 35 の用に登録する。

リスト 35: SNMPv3 用ユーザ登録

```
createUser mywriter MD5 mypasswordforwriter DES myencpassforwrite
createUser myreader MD5 mypasswordforread DES myencpassforread
```

ユーザ登録後、snmpd サービスを再起動すると/var/net-snmp/snmpd.conf に記したユーザ名とパスワードが登録され、利用できるようになる。

6.7 SNMPv3 での接続

ユーザ名の登録が完了したら snmpget コマンドで動作を確認する。認証に成功し、sysContact の値が得られる。
暗号化無しの認証の場合は図 6.7 のように指定する。

```
# snmpget -c myprivate -v 3 -l authNoPriv -u mywriter -a MD5 -A mypasswordforwriter
localhost .1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING: admin@example.com
```

図 6.7 SNMPv3 での接続

6.8 OID を調べる

SNMP コマンドを使って情報の取得や設定をするためには MIB の OID が必要である。前述の通り、MIB はツリー構造をしており、RFC1213[*3]の 12 ページから MIB-II の定義が記されている。

MIB-II のツリーは表 6.3 の分類がある。

表 6.3 MIB-II の分類

OID	意味
system	システム名、場所、提供するサービス、稼働時間などが含まれる。
interfaces	インタフェースの情報。そのインタフェースを通じて送受信したパケット総数などの情報がある。
at	アドレス変換情報。例えば、TCP/IP の IP アドレスと Ethernet の MAC アドレスの対応などが含まれる。
ip	IP に関する情報。ルーティング情報や IP パケットの送受信数などの情報がある。
icmp	ICMP に関する情報。送受信した ICMP パケットの情報がある。
tcp	TCP コネクションに関する情報。接続中（ESTABLISHED）や待ち受け中（LISTEN）のポート番号や接続先、通信エラー数などの情報がある。
udp	UDP コネクションに関する情報。待ち受け中（LISTEN）のポート番号などがある。
egp	EGP（Exterior Gateway Protocol）に関する情報。
transmission	変換タイプを定義するグループ。
snmp	SNMP 自身に関する情報。例えば、SNMP パケットの送受信数、認証エラー数などが含まれる。

OID の分類名を使って MIB のツリーや値を取得できる。例えば、管理者の連絡先は OID に system.4 を指定すればよい。system が 1.3.6.1.2.1.1 と等価である。

[*3] http://tools.ietf.org/html/rfc1213

```
# snmpget -c myprivate -v 1 localhost system.4.0
または
# snmpwalk -c myprivate -v 1 localhost system.4
```

図 6.8 OID 名で値を取得

6.9 異常の検出

Net-SNMP の独自 MIB ツリーの UCD-SNMP を参照すると機器のメモリ、プロセス、ディスクなどのサーバの状況を参照できる。

6.10 メモリ情報の取得

メモリ情報は ucdavis.4.XXX に含まれている。snmpget コマンドで取得できる。

```
# snmpget -c myprivate -v 1 localhost memAvailReal.0
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 4460
```

図 6.9 空きメモリの状態取得

表 6.4 にメモリ情報取得の OID などを示す。

表 6.4 メモリ情報

名称	OID	意味
memIndex	cdavis.4.1	常に 0
memErrorName	ucdavis.4.2	常に「swap」という文字列
memTotalSwap	ucdavis.4.3	利用できる総スワップ容量
memAvailSwap	ucdavis.4.4	未使用のスワップ容量
memTotalReal	ucdavis.4.5	搭載されている実メモリ容量
memAvailReal	ucdavis.4.6	未使用の実メモリ容量
memTotalSwapTXT	ucdavis.4.7	テキストページとして割り当てられているスワップ総容量
memTotalRealTXT	ucdavis.4.9	テキストページとして割り当てられている実メモリ容量
memTotalFree	ucdavis.4.11	メモリ空き容量
memMinimumSwap	ucdavis.4.12	このホストにおける最小限必要なスワップ残量。memAvailSwap が、この値を下回ると、memSwapError に 1 が設定される
memShared	ucdavis.4.13	共有メモリとして割り当てられているメモリ容量
memBuffer	ucdavis.4.14	バッファメモリとして割り当てられているメモリ容量
memCached	ucdavis.4.15	キャッシュメモリとして割り当てられているメモリ容量
memUsedSwapTXT	ucdavis.4.16	テキストページとして現在使われているスワップ容量
memUsedRealTXT	ucdavis.4.17	テキストページとして現在使われている実メモリ容量
memSwapError	ucdavis.4.100	スワップ残量が少なくなったときの警告フラグ。memAvailSwap が memMinimumSwap を下回ったときに 1 になる
memSwapErrMsg	ucdavis.4.101	memSwapError が 1 になったときのエラーメッセージ

6.11 プロセスの監視

SNMP を使ってプロセスを監視するには設定ファイル snmpd.conf にどのプロセスを監視するかを記述しておく。例えば、httpd と sendmail を監視したいときはリスト 36 のように設定を記述する。

リスト 36: httpd と sendmail プロセスを監視する設定

```
proc httpd
proc sendmail
```

snmpd.conf の変更を反映するときにも senmset コマンドが使える。

```
# snmpset -c myprivate -v 1 localhost versionUpdateConfig.0 i 1
```

図 6.10 snmpd.conf 設定変更の反映

snmpd.conf の proc ディレクティブの書式は次の通り。

リスト 37: proc ディレクティブの書式

```
proc プロセス名 最大値 最小値
```

最大値と最小値は幾つ起動していないとエラーとするかの指定で、最小値を下回ったり、最大値上回った場合にエラーとする。

監視結果は snmpwalk コマンドで確認できる (図 6.11)。この出力のうち、prErrorFlag の値に注目する。1 がエラーである。

```
# snmpwalk -c myprivate -v 1 localhost prTable
UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: senmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 0
UCD-SNMP-MIB::prMin.2 = INTEGER: 0
UCD-SNMP-MIB::prMax.1 = INTEGER: 0
UCD-SNMP-MIB::prMax.2 = INTEGER: 0
UCD-SNMP-MIB::prCount.1 = INTEGER: 9
UCD-SNMP-MIB::prCount.2 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 1
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING: No senmail process running.
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

図 6.11 プロセス監視結果の表示

prTable のサブツリーは表 6.5 の通り。

表 6.5 prTable のサブツリー

名称	OID	意味
Index	ucdavis.2.1	インデックス番号
Names	ucdavis.2.2	プロセス名。proc ディレクティブに指定した名称
Min	ucdavis.2.3	存在しなければならない最小値。prCount がこの値を下回ったときには、prErrorFlag が 1 になる
Max	ucdavis.2.4	存在できるプロセスの上限値。prCount がこの値を上回ったときには、prErrorFlag が 1 になる
Count	ucdavis.2.5	現在存在しているプロセス数
ErrorFlag	ucdavis.2.100	プロセスが異常であることを告げるフラグ。プロセス数が prMin < prCount または prMax < prCount になったときに 1 になる
ErrMessage	ucdavis.2.101	prErrorFlag が 1 であるときのエラーメッセージ
ErrFix	ucdavis.2.102	このフラグを 1 に設定すると、prErrFixCmd で指定したコマンドが実行される
ErrFixCmd	ucdavis.2.103	prErrFix が 1 になったときに実行されるコマンド。procfix ディレクティブで指定する (関連記事)。実行結果は、extTable(ucdavis.8) から参照できる

6.12 ディスクの監視

ディスクの状況監視には dskTable(ucdavis.9) サブツリーを使う。設定ファイルの書式は次の通り。

リスト 38: ディスクの監視設定

```
disk［マウント先のパス］［最小値あるいはパーセンテージ］
```

例えば、ルートディレクトリの空き容量の閾値を 20% に設定するときはリスト 39 と指定する。

リスト 39: ディスク監視の記述例

```
disk / 20%
```

```
[root@fc6 ~]# snmpset -c myprivate -v 1 localhost versionUpdateConfig.0 i 1
UCD-SNMP-MIB::versionUpdateConfig.0 = INTEGER: 1
# snmpwalk -c myprivate -v 1 localhost dskTable
UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/mapper/VolGroup00-LogVol00
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 20
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 14603080
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2789376
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 11061768
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 80
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 12
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 1
UCD-SNMP-MIB::dskErrorMsg.1 = STRING: /: less than 20% free (= 80%)
```

図 6.12 ディスク監視例

dskTable サブツリーを示す。

表 6.6 dskTable サブツリー

名称	OID	意味
dskIndex	ucdavis.9.1	インデックス番号
dskPath	ucdavis.9.2	マウント先のパス名。disk ディレクティブで指定した値
dskDevice	ucdavis.9.3	デバイスのパス名
dskMinimum	ucdavis.9.4	disk ディレクティブにおいて、キロバイト単位で指定したとき (後ろに「%」が付かないとき) のエラー扱いとする設定値
dskMinPercent	ucdavis.9.5	disk ディレクティブにおいて、パーセント単位で指定したとき (後ろに「%」を付けたとき) のエラー扱いとする設定値
dskTotal	ucdavis.9.6	そのパスに対応するディスクの総量 (キロバイト単位)
dskAvail	ucdavis.9.7	そのパスに対応するディスクの残量 (キロバイト単位)
dskUsed	ucdavis.9.8	そのパスに対応するディスクの使用量 (キロバイト単位)
dskPercent	ucdavis.9.9	そのパスに対応するディスクの使用割合
dskPercentNode	usdavis.9.10	そのパスに対応するディスクの i ノードの使用割合
dskErrorFlag	usdavis.9.100	ディスク容量不足のエラーフラグ。ディスク残量が dskMinimum または dskMinPercent を下回ったときに 1 が設定される
dskErrorMsg	ucdavis.9.101	dskErrorFlag が 1 になったときのエラーメッセーシ

6.13 ロードアベレージの監視

サーバの負荷状態を監視するときは laTbale サブツリーを使う。設定ファイルの記述はリスト 40 のように load ディレクティブを使う。

リスト 40: ロードアベレージ監視設定

```
load [1 分の閾値] [5 分の閾値] [15 分の閾値]
```

例えば、1 分間のロードアベレージが 10 を超えたときにエラーとしたいときは

リスト 41: ロードアベレージ監視設定例

```
load 10
```

```
# snmpwalk -c myprivate -v 1 localhost laTable
UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.01
UCD-SNMP-MIB::laLoad.2 = STRING: 0.03
UCD-SNMP-MIB::laLoad.3 = STRING: 0.00
UCD-SNMP-MIB::laConfig.1 = STRING: 10.00
UCD-SNMP-MIB::laConfig.2 = STRING: 10.00
UCD-SNMP-MIB::laConfig.3 = STRING: 10.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 1
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 2
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 0
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.010000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.030000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.000000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

図 6.13 ロードアベレージ監視例

laTable のサブツリーを示す。

表 6.7 laTable サブツリー

名称	OID	意味
laIndex.n	ucdavis.10.1.n	インデックス番号
laNames.n	ucdavis.10.2.n	監視しているロードアベレージ番号「Load-1」「Load-5」「Load-15」
laLoad.n	ucdavis.10.3.n	現在のロードアベレージ値 (文字列)
laConfig.n	ucdavis.10.4.n	エラー扱いとする下限値。load ディレクティブで指定した値。この値を laLoad が超えたときには laErrorFlag が 1 になる
laLoadInt.n	ucdavis.10.5.n	現在のロードアベレージ値を 100 倍して整数化したもの
laLoadFloat.n	ucdavis.10.6.n	現在のロードアベレージ値を浮動小数点として示したもの
laErrorFlag.n	ucdavis.100.n	エラーを伝えるフラグ。ロードアベレージが laConfig を超えているときには 1 になる
laErrMessage.n	ucdavis.101.n	laErrorFlag が 1 であるときのエラーメッセーシ

6.14 ファイルサイズ監視

ログファイルなど肥大化しやすいファイルサイズを監視するときは file ディレクティブを使う。
書式と使用例を示す。

リスト 42: file ディレクティブの書式

```
file [ファイルのフルパス] [警告サイズ]
```

例えば、ファイル/var/log/messages のサイズが 10MB を超えたときにエラーとしたいときは次のように設定を記述する。

リスト 43: file ディレクティブ

```
file /var/log/messages 102400
```

表 6.8 fileTable サブツリー

名称	OID	意味
fileIndex	ucdavis.15.1	インデックス番号
fileName	ucdavis.15.2	監視するファイル名。file ディレクティブで指定したもの
fileSize	ucdavis.15.3	現在のファイルサイズ (キロバイト)
fileMax	ucdavis.15.4	エラー扱いとするサイズ (キロバイト)。この値よりもファイルサイズが大きくなったときに fileErrorFlag が 1 になる
fileErrorFlag	ucdavis.15.100	エラーを示すフラグ。fileSize が fileMax を上回ったときに 1 になる
fileErrorMsg	ucdavis.15.101	fileErrorFlag が 1 であるときのエラーメッセージ

6.15 モニタのグラフ化

コマンドを使って様々な状況を確認できるが、それは瞬間の値で、統計的な情報を把握できない。大量の数値データを眺めるのは大変である。そこでデータをグラフ化すると傾向を把握しやすくなる。

データのグラフ化ツールには MRTG と RRDTool がある。今回は RRDTool を使って SNMP で取得した情報をグラフ化する。RRDTool だけではグラフを作るのに手間がかかるのでその作業を簡略化するツール cacti を使う。

6.16 RRDTool とは

RRDTool の RRD は Round Robin Database のことでデータベースに時系列のデータを格納し、それをグラフ化するツールである。RRD はデータサイズが一定で、データが一杯になったら先頭に戻って最も古いデータを上書きしていくという仕組みで、データベースが肥大化せずに済む。

RRDTool は SNMP とは全く関係のないツールで、SNMP の snmpget コマンドで取得したデータを RRD に格納し、そのデータを使って描画する。

6.17 RRDTool と Cacti のインストール

RRDTool と Cacti はいずれもパッケージが用意されているので yum でインストールする。cacti は MySQL を使用するため、コマンドによる設定が必要である。

```
# yum -y install rrdtool cacti
# mysqladmin -u root -p create cacti
# mysql -u root -p cacti < /var/www/cacti.sql
# mysql -u root -p mysql
mysql>grant all privileges on cacti.* to cactiuser@localhost identified by 'cacti';
mysql>exit;
# vim /var/www/cacti/include/config.php
$database_password = "cacti"
# crontab -e
*/5 * * * * /usr/bin/php /var/www/cacti/poller.php > /dev/null 2>&1
```

図 6.14 RRDTool と Cacti のインストール

6.18 動作確認

インストールが終わったら、http://localhost/cacti/にアクセスする。初回は設定が必要。

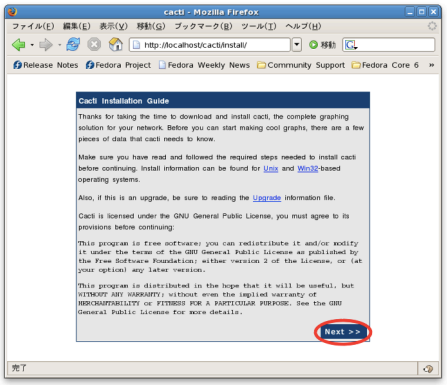

図 6.15 Cacti 初回画面 1

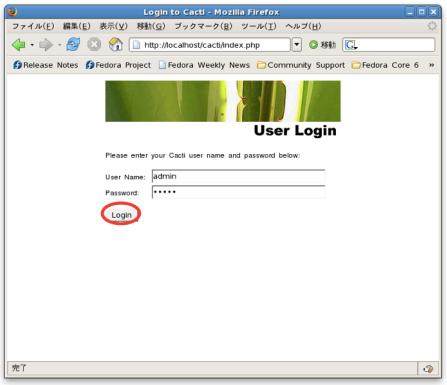

図 6.18 Cacti ログイン

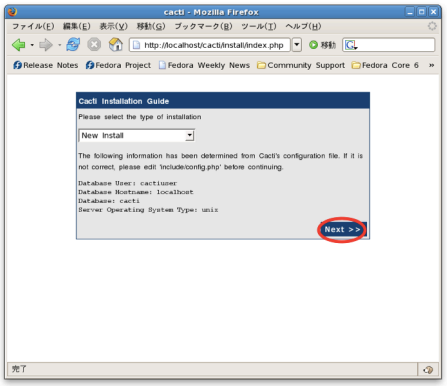

図 6.16 Cacti 初回画面 2

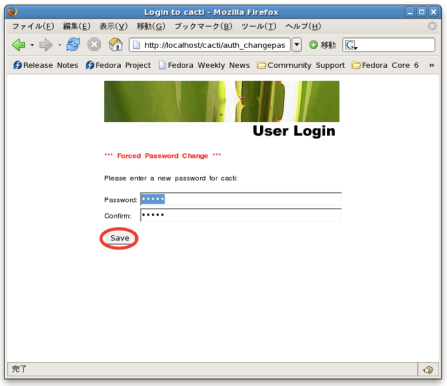

図 6.19 パスワード変更

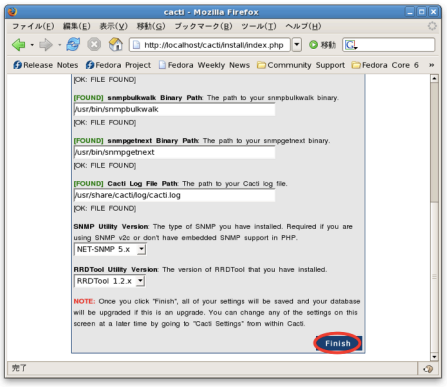

図 6.17 Cacti 初回画面 3

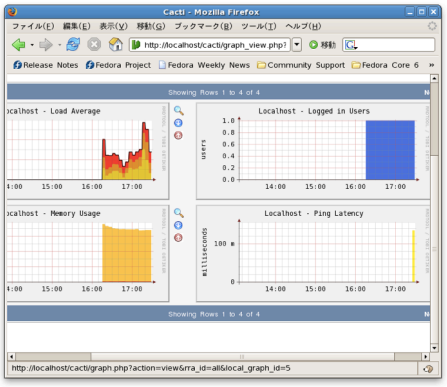

図 6.20 グラフ表示例

6.19 演習

1. Net-SNMP を導入し、設定する

2. snmpwalk コマンドの出力が OID の数値を表示するにはどうすればよいか
3. SNMPv3 の暗号化を利用して接続するにはどうすればよいか
4. SNMP でシステムやネットワークの状態を調べたい。RFC1213 の文書を読み、どんな OID を指定すればよいかを考えなさい。
5. 実習室のレーザプリンタも SNMP が搭載されている。システムの情報、インターフェースなどの情報を確認しなさい。コミュニティ名は public です。
6. Smaba、DHCP クライアント、X ウィンドウのプロセスを監視する。

第 7 章

MRTG

SNMP による監視対象の状況を MRTG で可視化する。

7.1 SNMP の設定

今回は SNMP の設定を次の用に変更し、snmpd を実行する。既存の設定ファイルの名前を変更し、新たにファイルを作る。

```
# mv /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.1
# vim /etc/snmp/snmpd.conf
# service snmpd start
```

図 7.1　設定ファイルの名前変更

リスト中の 10.211.55.0 は自分環境に合わせて変更する。

リスト 44: /etc/snmp/snmpd.conf の内容

```
com2sec local localhost private
com2sec mynetwork 10.211.55.0/24 public

group MyROGroup v1 mynetwork
group MyROGroup v2c mynetwork

view all included .1 80

access MyROGroup "" any noauth exact all none none
access MyROGroup "" any noauth exact all all none

proc httpd
disk / 10000
load 12 14 14
```

7.2 MRTG のインストールと設定

MRTG を yum でインストールし、設定ファイルを修正する。

```
# yum -y install mrtg
# vim /etc/mrgt/mrtg.cfg
```

図 7.2 MRTG のインストール

リスト中の 10.211.55.4 は自分の Linux の IP アドレスに変更する。

リスト 45: /etc/mrtg/mrtg.cfg

```
#  for UNIX
WorkDir: /var/www/mrtg

### Global Defaults

#  to get bits instead of bytes and graphs growing to the right
# Options[_]: growright, bits
Options[_]: growright, noinfo

EnableIPv6: no
Refresh: 300
Language:eucjp

######################################################################
# System:
# Description:
# Contact:
# Location:
######################################################################

Target[eth0]: \eth0:public@10.211.55.4:
SetEnv[eth0]: MRTG_INT_IP="10.211.55.4" MRTG_INT_DESCR="eth0"
MaxBytes[eth0]: 12500000
Title[eth0]: eth0 トラフィック
PageTop[eth0]: <h1>eth0 トラフィック</h1>

### CPU Load Average ###
Target[cpu]: .1.3.6.1.4.1.2021.10.1.5.1&.1.3.6.1.4.1.2021.10.1.5.2:public@10.211.55.4
MaxBytes[cpu]: 100
Unscaled[cpu]: dwmy
Options[cpu]: gauge, absolute, growright, noinfo, nopercent
YLegend[cpu]: CPU Load(%)
ShortLegend[cpu]: (%)
LegendI[cpu]: 1 分間平均
```

```
LegendO[cpu]: 5 分間平均
Legend1[cpu]: 1 分間平均 (%)
Legend2[cpu]: 5 分間平均 (%)
Title[cpu]: CPU 使用率
PageTop[cpu]: <h1>CPU 使用率</h1>
### Memory Free ####
Target[mem]: .1.3.6.1.4.1.2021.4.6.0&.1.3.6.1.4.1.2021.4.4.0:public@10.211.55.4
MaxBytes1[mem]: 1035060
MaxBytes2[mem]: 2097144
Unscaled[mem]: dwmy
Options[mem]: gauge, absolute, growright, noinfo
YLegend[mem]: Mem Free(Bytes)
ShortLegend[mem]: Bytes
kilo[mem]: 1024
kMG[mem]: k,M,G,T,P
LegendI[mem]: Real
LegendO[mem]: Swap
Legend1[mem]: 空き物理メモリ [MBytes]
Legend2[mem]: 空きスワップメモリ [MBytes]
Title[mem]: 空きメモリ量
PageTop[mem]: <H1>空きメモリ量</H1>
### Disk Used ####
Target[disk]: .1.3.6.1.4.1.2021.9.1.9.1&.1.3.6.1.4.1.2021.9.1.9.1:public@10.211.55.4
MaxBytes[disk]: 100
Unscaled[disk]: dwmy
Options[disk]: gauge, absolute, growright, nopercent, noinfo
YLegend[disk]: Disk Used(%)
ShortLegend[disk]: (%)
LegendI[disk]: / Disk used
LegendO[disk]: / Disk Used
Legend1[disk]: / Disk used
Legend2[disk]: / Disk used
Title[disk]: ディスク使用率
PageTop[disk]: <H1>ディスク使用率</H1>
```

設定ファイルの文字コードを EUC に変換する。

```
# cp /etc/mrtg/mrtg.cfg /etc/mrtg/mrtg.cfg.org
# nkf -e /etc/mrtg/mrtg.cfg.org > /etc/mrtg/mrtg.cfg
```

図 7.3 文字コードの変換

次に下記の内容の MRTG 起動スクリプト mrtg.sh を作り、実行権を付与する。

リスト 46: mrtg.sh の作成

```
#!/bin/bash
LOCK=/var/lock/mrtg/mrtg_l
```

```
CONFCACHE=/var/lib/mrtg/mrtg.ok

export LANG=ja_JP.eucJP
mrtg /etc/mrtg/mrtg.cfg --lock-file $LOCK --confcache-file $CONFCACHE
```

MRTG の設定棟が終わったら MRTG を起動する。3 回起動すること。

次に、/etc/httpd/conf.d/mrtg.conf に仮想マシンと同一ネットワークからのアクセスを許可する文を追加する。

```
Allow from 172.16.11
```

MRTG のインデックスファイルを作る。

```
# indexmaker --columns=1 \
--addhead="<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text/html; \
    charset=euc-jp\">" /etc/mrtg/mrtg.cfg > \
    /var/www/mrtg/index.html
```

MRTG の表示の確認は http://localhost/mrtg/を開けばよい。

第 8 章

GNUPG

今回は GnuPG を使ってファイルを暗号化する方法を確認する。

8.1 GnuPG とは

GnuPG(Gnu Private Gurad) は共通鍵暗号と公開鍵暗号をサポートする暗号化ソフトウェアです。共通鍵には通常のパスワードを使い、公開鍵には公開鍵と秘密鍵の 2 つを使います。公開鍵で暗号化されたデータは秘密鍵で復号化できるが、公開鍵から秘密鍵を得ることはできません。名前の通り公開鍵を公開・配布し、秘密鍵は厳重に管理し、他人には秘密にします。

GnuPG はコマンド gpg を使い、暗号化、復号化などの操作をします。

8.2 共通鍵暗号を使う

8.2.1 パスワードを使ってファイルを暗号化する

パスワードを使ってファイルを暗号化し、自分だけがファイルを復号化できるようにするには c オプションを使います。ASCII ファイルにする時は a オプションを併用します。

```
$ gpg -c ファイル名
$ gpg -c -a ファイル名
```

図 8.1 パスワードでファイルを暗号化

c オプションのみの時はバイナリファイルで、a オプションを併用するとアスキーファイルが生成され、それぞれの拡張子 gpg と asc が基のファイルに付加されます。

8.2.2 ファイルの復号化

暗号化されたファイルを復号化する時はオプションは不要で、ファイル名のみを指定します。復号結果を標準出力に出力する時は decrypt オプションを指定します。

```
$ gpg ファイル名
$ gpg --decrypt ファイル名
```

図 8.2 ファイルの復号化

8.3 公開鍵暗号を使う

他のユーザがファイルを復号化できる暗号を作ったり、電子署名する場合には公開鍵暗号方式を使います。

8.3.1 公開鍵暗号を使えるようにする

公開鍵暗号方式を使うには–gen-key オプションを使って公開鍵と秘密鍵を作ります。コマンドを実行すると幾つかの質問が表示されるので必要に応じて設定します。

鍵の種類、長さなどはデフォルトのままでも大丈夫でしょう。

```
$ gpg --gen-key

ご希望の鍵の種類を選択してください:
   (1) DSA と Elgamal (既定)
   (2) DSA (署名のみ)
   (5) RSA (署名のみ)
選択は? <ー Enter キーを押す
```

図 8.3 鍵の生成

次に鍵の長さを指定します。最低でも 1024 を指定します。

```
DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) <ー Enter キーを押す
要求された鍵長は 2048 ビット
```

図 8.4 鍵の長さ指定

次は鍵の有効期限を指定します。ここでは無期限を設定します。

```
鍵の有効期限を指定してください。
         0 = 鍵は無期限
       <n>  = 鍵は n 日間で満了
       <n>w = 鍵は n 週間で満了
       <n>m = 鍵は n か月間で満了
       <n>y = 鍵は n 年間で満了
鍵の有効期間は? (0) <ー Enter キーを押す
Key does not expire at all
これで正しいですか? (y/N) y <ー y、Enter キーを押す
```

図 8.5 鍵の有効期限指定

次にユーザの名前、メールアドレス、コメントを入力します。

```
あなたの鍵を同定するためにユーザー ID が必要です。
このソフトは本名、コメント、電子メール・アドレスから
次の書式でユーザー ID を構成します: "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

本名: Kazuhisa Sakabe <ー 氏名の入力
電子メール・アドレス: k-sakabe@ca2.so-net.ne.jp <ー メールアドレスの入力
コメント: My Key. <ー コメントの入力
次のユーザー ID を選択しました:
    " Kazuhisa Sakabe (My Key.) <k-sakabe@ca2.so-net.ne.jp> "

名前 (N)、コメント (C)、電子メール (E) の変更、または OK(O) か終了 (Q)? o <ー o を入力
```

図 8.6 ユーザ情報の指定

最後に秘密鍵を保護するパスフレーズを入力し、キーを生成させます。

```
秘密鍵を保護するためにパスフレーズがいります。<- パスフレーズを入力します。

今から長い乱数を生成します。キーボードを打つとか、マウスを動かす
とか、ディスクにアクセスするとかの他のことをすると、乱数生成子で
乱雑さの大きないい乱数を生成しやすくなるので、お勧めいたします。
++++++++++++++++++
今から長い乱数を生成します。キーボードを打つとか、マウスを動かす
とか、ディスクにアクセスするとかの他のことをすると、乱数生成子で
乱雑さの大きないい乱数を生成しやすくなるので、お勧めいたします。
++++++++++.+++++++
gpg: /home/sakabe/.gnupg/trustdb.gpg: 信用データベースができました
gpg: 鍵 981365BF を絶対的に信用するよう記録しました
公開鍵と秘密鍵を作成し、署名しました。

gpg: 信用データベースの検査
gpg: 最小の「ある程度の信用」3、最小の「全面的信用」1、PGP 信用モデル
gpg: 深さ: 0  有効性:  1  署名:  0  信用: 0-, 0q, 0n, 0m, 0f, 1u
pub   1024D/981365BF 2010-09-06
指紋 = 0EC1 73F0 6BDB C8AD D50A 6747 DCFE 8500 9813 65BF
uid Kazuhisa Sakabe (My Key.) <k-sakabe@ca2.so-net.ne.jp>
sub   2048g/78C7203B 2010-09-06
```

図 8.7 パスフレーズ指定とキーの生成

以上の操作で公開鍵と秘密鍵の 2 つが生成され、ディレクトリ~/.gnupg に保存されます。

8.3.2 キーリングの表示

生成したキーや他人の公開鍵の一覧を表示するには–list-secret-keys、–list-public-keys オプションを使用します。

表示には公開鍵 (pub) か秘密鍵 (sec) か、鍵の長さ、暗号化アルゴリズム、鍵の ID、生成日、ユーザ ID が表示されます。

```
$ gpg --list-secret-keys
/home/sakabe/.gnupg/secring.gpg
------------------------------
sec   1024D/981365BF 2010-09-06
uid Kazuhisa Sakabe (My Key.) <k-sakabe@ca2.so-net.ne.jp>
ssb   2048g/78C7203B 2010-09-06
```

図 8.8 秘密鍵のリスト

```
$ gpg --list-public-keys
/home/sakabe/.gnupg/pubring.gpg
------------------------------
pub   1024D/981365BF 2010-09-06
uid Kazuhisa Sakabe (My Key.) <k-sakabe@ca2.so-net.ne.jp>
sub   2048g/78C7203B 2010-09-06
```

図 8.9 公開鍵のリスト

8.3.3 公開鍵のエクスポート

公開鍵をファイル化するには export オプションを使います。

```
$ gpg -a --export ユーザの ID など > ファイル
```

図 8.10 公開鍵のエクスポート

8.3.4 キーリングに鍵を追加する

他人の公開鍵を自分のキーリングに追加する時は–import オプションを使います。

```
$ gpg --import キーファイル
```

図 8.11 鍵のファイルのインポート

鍵を使いした後、そのリストの例を図 8.12 に示す。

```
$ gpg --list-public-keys
/home/sakabe/.gnupg/pubring.gpg
------------------------------
pub   1024D/981365BF 2010-09-06
uid Kazuhisa Sakabe (My Key.) <k-sakabe@ca2.so-net.ne.jp>
sub   2048g/78C7203B 2010-09-06

pub   1024D/EA079F13 2010-09-06
uid                  Nobita Nobi (test)
sub   2048g/CE26C2E7 2010-09-06
```

図 8.12 鍵を追加した後のリスト

8.3.5 特定の受信者のためにファイルを暗号化する

特定の相手が復号化できるファイルを作るには次の事前準備が必要である。

1. 相手の公開鍵を入手する
2. 入手した鍵をキーリングに追加する
3. 自分の秘密鍵と相手の公開階を使ってファイルを暗号化する

暗号化には e と r オプションを使うとバイナリ形式が更に a オプションを使いするとアスキー形式のファイルが作られる。

```
$ gpg -e -r 相手の公開鍵 ID 暗号化するファイル
$ gpg -e -r 相手の公開鍵 ID -a 暗号化するファイル
```

図 8.13 公開鍵でファイルの暗号化

8.3.6 テキストファイルに署名する

–clearsign オプションでテキストファイルに署名を追加できる。署名を付けることでファイルの信頼性を検証できる。

```
$ gpg --clearsign ファイル名
$ cat test1.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

このファイルは暗号化されています。
秘密の情報が記されている。
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.5 (GNU/Linux)

iD8DBQFMhQJx3P6FAJgTZb8RAi3kAKCgEhXjr56pUh2zpJbc+Bvg4rOiZACeL5WB
oXuUiFbXpBGPvsHRqoPNo08=
=mrEs
-----END PGP SIGNATURE-----
```

8.4 演習

資料を参考にクラスの他の人と暗号化したファイルのやりとりを体験する。

第 9 章

iptables

先週配布した資料を参考に組織間ファイアウォールを作る。

9.1 LAN カードをインストール

組織間のルータとして仮想マシンの Linux を動作させるので LAN カードを増設する。

Windows 起動後、配布された PC カードの LAN カードをスロットに差し込み、ドライバが自動的にインストールされるのを待つ。

9.2 VirtualBox の設定

VirtualBox で増設した LAN カードを使うために

1. VirtualBox の設定でネットワークを選ぶ
2. アダプタ 2 を有効にする
3. ブリッジを選択、デバイスは I-O DATA の製品を選択する。

設定できたら Linux を起動する。

9.3 Linux の設定

Linux も 2 枚目の LAN カードを自動的に認識し、IP アドレスを DHCP から得ようとするが失敗する。root 権限で次のコマンドを実行する。

新しいインターフェース eth1 の IP アドレスを設定し、ネットワーク間の伝送を可能にする。

```
# ifconfig eth1 192.168.1.254/24
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

図 9.1　Linux の設定

9.4 Windows の設定

Windows のコマンドプロンプトで次のコマンドを実行し、ルーティングできるようにする。

```
$ route add 192.168.1.0 mask 255.255.255.0 172.16.11.xx
```

図 9.2 ホスト OS 上のルーティング設定

9.5 動作確認

LAN カード 2 枚差しの状態でホスト OS、ゲスト OS、内部ホストのそれぞれの間で ping が通ることを確認しておくこと。

9.6 実習環境

資料図 1 のネットワークを実習室の環境に合わせる。

ネットワーク A 172.16.11.0/24
ネットワーク B 192.168.1.0/24
Web **サーバの変換後アドレス** 172.16.11.190
ネットワーク A **のクライアントのネットワーク** B **でのアドレス** 192.168.1.100