Security

May 28th ,2006

Security　目次

- サーバセキュリティ
  - 不要サービス停止
  - ファイアウォール
  - SSH鍵認証
  - ・・・
- WEBセキュリティ
  - SSL
  - ログイン認証
  - メタ文字無効化
  - ・・・
- メールセキュリティ
  - 送信ドメイン認証
  - POP before SMTP
  - アンチウィルス
  - ・・・

サーバセキュリティ

- サーバのセキュリティ項目
  - サーバ分析
  - 不要サービス停止
  - ファイアウォール
  - SSHの鍵認証
  - 改ざん検知
  - rootkitの検出

不要サービス停止(1)

- 専用サーバはRHEL3のフルインストールで、
  不要なサービスもたくさん起動されていますので、
  不要サービスを停止します。

chkconfigコマンドでデフォルトで起動されるサービスを見てみると、
次のようになっています。

```
# chkconfig --list | grep オン
microcode_ctl   0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
gpm             0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
kudzu           0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
syslog          0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
netfs           0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
network         0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
random          0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
rawdevices      0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
keytable        0:オフ  1:オン  2:オン  3:オン  4:オン  5:オン  6:オフ
mdmonitor       0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
mdmpd           0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
atd             0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
apmd            0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
isdn            0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
```

不要サービス停止(2)

```
iptables        0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
ip6tables       0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
pcmcia          0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
irqbalance      0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
autofs          0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
sshd            0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
portmap         0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
nfslock         0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
rhnsd           0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
crond           0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
xinetd          0:オフ  1:オフ  2:オフ  3:オン  4:オン  5:オン  6:オフ
cups            0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
hpoj            0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
xfs             0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
firstboot       0:オフ  1:オフ  2:オフ  3:オン  4:オフ  5:オン  6:オフ
canna           0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
postfix         0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
FreeWnn         0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
arptables_jf    0:オフ  1:オフ  2:オン  3:オン  4:オン  5:オン  6:オフ
  sgi_fam:        オン
```

不要サービス停止(3)

- 専用サーバはRHEL3のフルインストールで、不要なサービスもたくさん起動されていますので、不要サービスを停止します。

```
# chkconfig gpm off
# chkconfig pcmcia off
# chkconfig isdn off
# chkconfig rhnsd off
# chkconfig canna off
# chkconfig FreeWnn off
```

セキュリティ設定あれこれ

- root権限になることができるユーザを限定す
"/etc/pam.d/su"ファイルを編集し、次の部分のコメントを外します。

▼/etc/pam.d/su

```
...
# Uncomment the following line to require a user to be in the "wheel" group.
auth required /lib/security/$ISA/pam_wheel.so use_uid
...
```

- rootユーザでのSSHログインを禁止する。（KDDI専用サーバでは、デフォルトで設定済み）
“/etc/ssh/sshd_config”ファイルの“PermitRootLogin”の行のコメントを外し、“no”に変更します

▼/etc/ssh/sshd_config

```
...
PermitRootLogin no
...
```

SSH接続のアクセス制御

▪SSHで通常使用されるOpenSSHの認証システムは強固なもので、不正ユーザによって接続される危険性は低くなっていますが、さらにセキュリティレベルを高めるには、接続するクライアントのアクセス制御を行います。
RHEL3にデフォルトでインストールされるsshdは、TCP Wrapperを利用したアクセス制御に対応するようにコンパイルされていますので、/etc/hosts.allow と /etc/hosts.deny を利用してアクセス制御を行います。
まず、すべての接続を拒否する設定をします。

▼/etc/hosts.deny

```
...
sshd: ALL
...
```

▪次に、接続を許可するホストのホスト名またはIPアドレスを記述します。

▼/etc/hosts.allow

```
...
sshd: 192.168.1.10
sshd: sshclient
...
```

SSHで鍵認証を行う(1)

SSHはセキュアシェルですが、一つ問題点として、最初のログインのパスワードは平文です。
そこで、RSA暗号化方式（秘密鍵－公開鍵方式）を使用してセキュアにログインします。
SSHにはVersion1とVersion2がありますが、ここではSSH Version1のRSA暗号を利用する場合について説明します。WindowのSSHクライアントとしてよく知られたTera Term Pro +TTSSHの従来バージョン(Version2.3)はVersion1のみに対応していますので、SSH Version1が最もよく使用されているバージョンだと思われるからです。

1.はroot権限で、2.以降は一般ユーザ権限で行います。
1.SSHのユーザ認証を鍵認証のみに限定する場合は設定を変更します。
（平文認証と併用する場合は設定変更は必要ありません。）

▼/etc/ssh/sshd_config

```
...
PasswordAuthentication no # 平文パスワード認証を禁止する
PermitEmptyPasswords no # 空文字パスワードの鍵認証を禁止する
...
```

sshdを再起動します。

```
# service sshd restart
```

SSHで鍵認証を行う(2)

2. 公開鍵ー秘密鍵ペを作成します。(RSA公開鍵暗号Ver.1の場合)

```
$ ssh-keygen -t rsa1 Generating public/private rsa1 key pair. Enter file in which to save
the key (/home/username/.ssh/identity): [通常はこのままリターン] Enter passphrase
(empty for no passphrase): [パスフレーズを入力] Enter same passphrase
again: [確認のためパスフレーズを再入力]
Your identification has been saved in /home/username/.ssh/identity.
Your public key has been saved in /home/username/.ssh/identity.pub.
The key fingerprint is: b7:a1:38:2a:ea:fd:0f:c1:e8:30:d1:63:81:52:40:ad username@sensaba.net
```

~/.ssh 以下に、identity, identity.pubの二つのファイルが生成されます。それぞれ、
自分の秘密鍵と公開鍵です。パーミッションは、それぞれ600(-rw-------), 644(-rw-r--r--)です。

3. authorized_keysを用意する
~/.ssh/authorized_keys ファイルに信頼できる自分の公開鍵を追加します。
ssh-keygenによって先ほど作成した自分の公開鍵identity.pubを追加します。

```
$ chmod 700 ~/.ssh $ cd ~/.ssh $ touch authorized_keys $ cat identity.pb >> authorized_keys $ chmod 600
authorized_keys
```

また、接続先ホストの~/.ssh/authorized_keysにも接続元ホストにある自分の公開鍵を追加します。
identiry.pubは他人に見られても問題ありませんので、ftpなど非セキュアな方法で接続先ホストに
コピーすることができます。

SSHで鍵認証を行う(3)

4. 接続先ホストのホスト公開鍵を登録する
RSA公開鍵暗号による認証を利用するには、以下の条件が必要です。
a. 接続元ホストでRSA秘密鍵を作成してある(~/.ssh/identityが存在する)。
b. 接続元ホストの~/.ssh/known_hostsファイルに接続先ホストのホスト公開鍵が登録されている。
c. 接続先ホストの~/.ssh/authorized_keysファイルに接続元ホストにある自分の公開鍵
(接続元ホストの~/.ssh/identity.pubの内容)が登録されている。
一度もSSHを使って接続した事のない接続先ホストにSSHによって接続すると、
~/.ssh/known_hostsに相手のホスト公開鍵を登録するかどうかを訪ねられます。
これに`yes'と答えれば登録されます。

作成した秘密鍵は世界でたった一つしかない重要な鍵です。この秘密鍵を誤って削除したりして失くしてしまうと、二度と同じ鍵を作成できませんので、注意してください。

5. ホストに接続する

```
$ ssh -1 dummyhost.sensaba.net Enter passphrase for RSA key '/home/username/.ssh/identity': [パスフレーズを入力]
```

単にsshとコマンドをうつとVersion2として判断されることがありますので、その場合には、“-1”オプションを付与してVersion1として接続します。

SSH Version2での鍵の使用法(1)

●DSA暗号を利用する場合

1. 公開鍵ー秘密鍵ペアを作成します。(DSA暗号)

```
$ ssh-keygen -t dsa Generating public/private dsa key pair. Enter file in which to save the key
    (/home/username/.ssh/id_dsa): [通常はこのままリターン] Enter passphrase (empty for no
    passphrase): [パスフレーズを入力] Enter
    same passphrase again: [確認のためパスフレーズを再入力] Your identification has been saved
    in /home/kddidev/.ssh/id_dsa. Your public key has been saved in /home/kddidev/.ssh/id_dsa.pub.
    The key fingerprint is: f6:1a:2b:7c:44:51:6b:6e:27:5e:40:fe:c4:7d:5f:49 username@sensaba.net
```

~/.ssh 以下に、id_dsa,
id_dsa.pubの二つのファイルが生成されます。それぞれ、自分の秘密鍵と公開鍵です。パーミッションは、それぞれ600(-rw-------), 644(-rw-r--r--)です。

2. authorized_keys2を用意する

~/.ssh/authorized_keys2 ファイルに信頼できる自分の公開鍵を追加します。
ssh-keygenによって先ほど作成した自分の公開鍵id_dsa.pubを追加します。

```
$ cd ~/.ssh
$ cat id_dsa.pub >> authorized_keys2
$ chmod 600 authorized_keys2
```

また、接続先ホストの~/.ssh/authorized_keys2にも接続元ホストにある自分の公開鍵を追加します。

3. ホストに接続する

```
$ ssh dummyhost.sensaba.net Enter passphrase for key '/home/username/.ssh/id_dsa': [パスフレーズを入力]
```

SSH Version2での鍵の使用法(2)

●RSA暗号を利用する場合

1. 公開鍵ー秘密鍵ペアを作成します。(RSA暗号)

```
$ ssh-keygen -t rsa Generating public/private rsa key pair. Enter file in which to save the key
    (/home/username/.ssh/id_rsa): [通常はこのままリターン] Enter passphrase (empty for no
    passphrase): [パスフレーズを入力] Enter same passphrase again:
    [確認のためパスフレーズを再入力] Your identification has been saved in
    /home/kddidev/.ssh/id_rsa. Your public key has been saved in /home/kddidev/.ssh/id_rsa.pub. The
    key fingerprint is: 3a:63:89:27:7b:19:90:91:64:29:4f:47:00:9c:50:c4 username@sensaba.net
```

~/.ssh 以下に、id_rsa,
id_rsa.pubの二つのファイルが生成されます。それぞれ、自分の秘密鍵と公開鍵です。パーミッションは、それぞれ600(-rw-------), 644(-rw-r--r--)です。

2. authorized_keys2を用意する
~/.ssh/authorized_keys2
ファイルに信頼できる自分の公開鍵を追加します。ssh-keygenによって先ほど作成した自分の公開鍵id_rsa.pubを追加します。

```
$ cd ~/.ssh
$ cat id_rsa.pub >> authorized_keys2
$ chmod 600 authorized_keys2
```

また、接続先ホストの~/.ssh/authorized_keys2にも接続元ホストにある自分の公開鍵を追加します

3. ホストに接続する

```
$ ssh dummyhost.sensaba.net Enter passphrase for key '/home/username/.ssh/id_rsa': [パスフレーズを入力]
```

SSH Version2での鍵の使用法(3)

●複数の鍵がある場合

複数の鍵がある場合には、RHEL3の標準では、RSA、DSA、平文パスワードの順番でパスワード認証を試します。

```
$ ssh dummyhost.sensaba.net Enter passphrase for key '/home/username/.ssh/id_rsa':
        [そのままリターン] Enter passphrase for key '/home/username/.ssh/id_dsa': [そのままリターン]
        kddidev@210.230.64.32's password:
```

●Windowsでの利用

SSH Version1およびVersion2に対応し、UTF-8文字コードにも対応した、UTF-8 TeraTerm Pro with TTSSH2を以下よりダウンロードして利用してください。
http://sourceforge.jp/projects/ttssh2/

Tripwire－改ざん検知(1)

Tripwireは指定したファイルの情報をデータベースに保存して、ファイルが改竄されていないかどうかを確認するためのソフトウェアです。Tripwireは本来商用ソフトですが、開発元のTripwire, Inc.がオープンソースコミュニティへの協力しており、ソースファイルの配布が行われています。
Linux版にはRPMも用意されています。

Tripwireの詳細は次のページを参照してください。
Tripwire Japan K.K.
http://www.tripwire.co.jp/

ここでは、Tripwireをすでにインストールしたとして最低限の設定について説明します。

Tripwireでは、設定ファイルやポリシーファイルで利用されるサイトキーファイルを暗号化するための「サイトパスフレーズ」、データベースファイルなどで利用されるローカルキーファイルを暗号化するための「ローカルパスフレーズ」が必要になります。各パスフレーズを設定するために、
/etc/tripwire/twinstall.sh を実行します。

Tripwire－改ざん検知(2)

 # /etc/tripwire/twinstall.sh

The Tripwire site and local passphrases are used tosign 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: [サイトキーファイル用のパスフレーズを入力]
Verify the site keyfile passphrase: [パスフレーズを再入力]
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: [ローカルキーファイル用のパスフレーズを入力]
Verify the local keyfile passphrase: [パスフレーズを再入力]
Generating key (this may take several minutes)...Key generation complete.

Signing configuration file...
Please enter your site passphrase: [設定したサイトパスフレーズを入力]
Wrote configuration file: /etc/tripwire/tw.cfg
A clear-text version of the Tripwire configuration file
/etc/tripwire/twcfg.txt
has been preserved for your inspection. It is recommended that you delete this file manually after you have examined it.

Signing policy file...
Please enter your site passphrase: [設定したサイトパスフレーズを入力]
Wrote policy file: /etc/tripwire/tw.pol
A clear-text version of the Tripwire policy file
/etc/tripwire/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.

設定が終わると、/etc/tripwireディレトリに
以下の2つのキーファイルが作成されます。
* ホスト名-local.key
* site.key

Tripwire－改ざん検知(3)

Tripwireのインストールが完了したら、まずデータベースファイルの作成を行います。

```
# /usr/sbin/tripwire --init Please enter your local passphrase:
        [ローカルパスフレーズを入力] Parsing policy file: /etc/tripwire/tw.pol
        Generating the database... *** Processing Unix File System ***
```

以上のコマンドを実行すると、/var/lib/tripwire/.twdというファイル名で
データベースファイルが作成されます。ホストのスペックによりますが、
この作業には数分から数十分の時間がかかります。

ファイルの改ざんが行われていないかどうかの検証（整合性チェック）は、次のコマンドを
実行します。

```
# /usr/sbin/tripwire --check Parsing policy file: /etc/tripwire/tw.pol *** Processing Unix
        File System *** Performing integrity check...
```

整合性チェックにもデータベース作成時と同様に時間がかかります。変更、追加、削除された
ファイルが見つかった場合、その結果が表示されます。

Tripwireをインストールした段階で、/etc/cron.daily/tripwire-checkファイルがインストールされ、
毎日Tripwireが自動的に実行されて整合性チェックが行われます。もし、自動実行が不必要な
場合には、tripwire-checkを削除します。

整合性チェックの結果は標準出力のほかにレポートファイルにも出力されます。この内容を確認する
には、次のコマンドを実行します。は実行した時刻をもとにという名前がついています。

```
# /usr/sbin/tripwire -m r --twrfile /var/lib/tripwire/report/.twr
```

Tripwire－改ざん検知(4)

Tripwireデータベースの表示をするには次のコマンドを実行します。

```
# /usr/sbin/twprint -m d --print-dbfile
```

正常な変更が以後違反として報告されないようにするために、Tripwireデータベースを更新するには、次のコマンドを実行します。

```
# /usr/sbin/tripwire --update --twrfile /var/lib/tripwire/report/.twr
```

上記のコマンドが正常動作しない場合には次のようにします。

```
# /usr/sbin/tripwire --update -r /var/lib/tripwire/report/.twr --accept-all
```

Tripwire－改ざん検知(5)

●設定ファイルの編集

Tripwireのデフォルトの設定ファイルはtw.cfgで、暗号署名されたファイルです。
実際の設定変更には、クリアテキストのtwcfg.txtを使います。
twcfgファイルの設定例については省略します。
クリアテキスト（twcfg.txt）による設定を行ったら、次のコマンドを実行して、このファイルを基に暗号署名したファイル（tw.cfg）を生成します。

```
# /usr/sbin/twadmin --create-cfgfile -c tw.cfg -S site.key /etc/tripwire/twcfg.txt
```

●ポリシーファイルの編集

ポリシーファイルはtw.polで、暗号署名されたファイルです。実際の設定変更には、
クリアテキストのtwpol.txtを使います。
確認の対象から外したいファイルが、行の先頭に"#"をつけてコメントアウトします。
クリアテキスト（twpol.txt）による設定を行ったら、次のコマンドを実行して、このファイルを基に暗号署名したファイル（tw.pol）を生成します。

```
# /usr/sbin/twadmin --create-polfile -S site.key /etc/tripwire/twpol.txt
```

Tripwire－改ざん検知(6)

●確認のメールを送る

整合性チェックで異常があった場合には、その旨をメールで通知することができます。
メール送信はどのグループに所属するファイルに対して行うかを設定します。
例えば、“Critical configuration files”グループに含まれるファイルが変更されたときにメールを送るようにするには、次のように設定します。

▼/etc/tripwire/twpol.txt

```
...
(
rulename = "Critical configuration files",
severity = $(SIG_HI),
emailto = adminuser@sensaba.net
)
...
```

次のコマンドで、テストメッセージを送信することができます。

```
# /usr/sbin/tripwire --test --email adminuser@sensaba.net
Sending a test message to: adminuser@sensaba.net
```

●セキュリティの向上

設定ファイルとポリシーファイルのクリアテキストファイルは、もし盗聴されるとどのファイルの改ざんを監視しているか知られてしまうので、削除することをお勧めします。

```
# rm /etc/tripwire/twcfg.txt # rm /etc/tripwire/twpol.txt
```

rootkitの検出（1）

攻撃者がコンピュータに侵入した後によく利用するツール類をまとめたrootkitと呼ばれるツールキットがあります。rootkitは攻撃者の足跡を消し去り、バックドアを設置します。また、rootkit自体の存在を隠すために、psコマンドやlsコマンドなどのシステム自体を改ざんします。

rootkitには様々種類が存在するために、それらを手動で検出することは非常に困難ですが、chkrootkit を使うことにより自動的な検出ができます。現在のところ、chkrootkitでは60種類以上のrootkitを検出することができます。

詳細は以下のサイトを参照してください。

chkrootkit -- locally checks for signs of a rootkit → http://www.chkrootkit.org/

1. インストール

上記のサイトより最新のソースをダウンロードしてインストールします。

（一般ユーザ権限でもインストール可能です。）

```
$ tar zxvf chkrootkit.tar.gz
$ cd chkrootkit-0.46a
$ su
# make sense
# cp ./chkrootkit /usr/local/bin/
```

chkrootkitプログラムが作業ディレクトリ内に出来上がります。コマンドから実行しやすいように、/usr/local/binディレクトリにコピーしておくとよいでしょう。

rootkitの検出（2）

2. chkrootkitの実行
root権限で実行します。

```
# chkrootkit
ROOTDIR is `/'
Checking `amd'... not infected
Checking `basename'... not infected
Checking `biff'... not found
Checking `chfn'... not infected
Checking `chsh'... not infected
Checking `cron'... not infected
Checking `date'... not infected
（省略）
….
```

rootkitが検出された場合は、「INFECTED」と表示されます。

rootkitの検出（3）

3. cronで実行
chkrootkitをcronに登録して定期的に実行させます。
また、実行結果を記録するとともに、rootあてにメールを送信します。
/etc/cron.dailyディレクトリにchkrootkitを実行するシェルスクリプト
/etc/cron.daily/chkrootkitを作成します。
▼/etc/cron.daily/chkrootkit

```
#!/bin/sh
/usr/local/bin/chkrootkit > /var/log/chkrootkit.log
grep "INFECTED" /var/log/chkrootkit.log | mail -s "chkrootkit log" root
chmod 600 /var/log/chkrootkit.log
```

スクリプトに実行権を付加します。

```
# chmod 755 chkrootkit
```

注1：“INFECTED”と出力されたら改ざんされている可能性が高いですが、誤って検知される事も多いようですので、必ずしも改ざんされているとは限りません。また、まだ一般的に知られていない新しい rootkit も検知される事はありませんので、検出されなかったら安全というわけではありません。

注2：攻撃者にrootkitそのものを改ざんされてしまったら意味がありません。chkrootkitをCD-ROMなどの書き込みできないメディアに移しておくとよいでしょう。
また、chkrootkitが呼び出すコマンドは、awk、cut、egrep、find、head、id、ls、netstat、ps、strings、sed、unameですが、これらのコマンドが汚染されている場合もありますので、汚染されていないと想定できるシステム（KNOPPIXなど、CDブートできるディストリビューション）で一時的にブートし、そのシステムのコマンドを使ってchkrootkitを実行するのがより望ましいでしょう。

RPMパッケージの改ざん検出(1)

コンピュータが不正アクセスの被害を受け、システムが改ざんされてしまった場合に、
改ざんされたファイルを調査する方法として、RPMパッケージの改ざん検出をする方法があります。
RPMパッケージをインストールする時には、ファイルすべてのMD5チェックサムを保存しています。
また、ファイルのサイズ、ユーザー、グループ、モード、更新時刻などについても保存しています。
これらの情報を用いて、変更が加えられていないか調べることができます。
rpmコマンドで-Vオプションを使用して調査します。
ここでは試しにWEBサーバ(httpd)やメールサーバ(sendmail)など、いくつかのパッケージについて調べてみます。

```
# rpm -V httpd
S.5....T c /etc/httpd/conf/httpd.conf
.M...UG. /var/log/httpd

# rpm -V sendmail
SM5....T c /etc/aliases
S.5....T c /etc/mail/access
S.5....T c /etc/mail/local-host-names
S.5....T c /etc/mail/sendmail.cf
S.5....T c /etc/mail/sendmail.mc
S.5....T c /etc/mail/statistics
SM5....T c /etc/mail/submit.cf
```

RPMパッケージの改ざん検出(2)

インストール時から変更されているファイルについて、8個の文字と属性マークの形式で変更されている内容が出力されます。

文字の意味

- S ファイルのサイズ (Size) が異なる
- M モード (Mode; 許可属性とファイルの種類) が異なる
- 5 MD5 チェックサムが異なる
- D デバイス (Device) のメジャー/マイナー番号が一致しない
- L readLink(2) したパスが一致しない
- U 所有者 (User) が異なる
- G グループ (Group) が異なる
- T 修正時刻 (mTime) が異なる

属性マークの意味

- c %config 設定ファイル。
- d %doc 文書ファイル。
- g %ghost ファイル(すなわち、パッケージの内容物としては含まれていないファイル)。
- l %license ライセンスファイル。
- r %readme readme ファイル。

RPMパッケージの改ざん検出(3)

文字の部分で、“.”(ピリオド) はテストを通過した こと意味し、“?” (クエスチョンマーク) はテストが実施されなかった (すなわち、ファイルパーミッションにより読み込めなかった) ことを意味しています。
オプションをつけることによって検証結果を絞り込むことができます。設定ファイルなどのように管理者によって変更されたファイルを除外することができます。次はそれらのオプションの一部です。

- --nouser
- --nogroup
- --nomtime
- --nomode

```
# rpm -V --nouser --nogroup --nomtime --nomode httpd
S.5..... c /etc/httpd/conf/httpd.conf
```

すべてのパッケージについて検証を行う場合は次のようにします。

```
# rpm -Va
```

この出力結果をファイルにリダイレクトして記録しておくとよいでしょう。

WEBセキュリティ

- WEBのセキュリティ項目
  - SSL
  - ログイン認証
  - メタ文字無効化
  - Apacheのセキュリティ設定
  - WEBアプリケーションのセキュリティ

メールセキュリティ

- メールのセキュリティ項目
  - 送信ドメイン認証
  - POP before SMTP
  - APOP
  - SMTP Auth
  - POP3 over SSL / IMAP4 over SSL
  - SMTP over TLS
  - アンチウィルス
  - アンチスパム