Linuxによるセキュリティ入門（7）
── 最終回：基本に戻り小ネタ集 ──

西 村 竜 一

Ⅰ．はじめに

この原稿を書く少し前に，Windows XPのサービスパック２がリリースされました。主な変更点としてセキュリティの強化がなされたようですので，わたしも導入してみました。なにか新しいサービスを利用しようとすると画面上に警告が表示されます。警告の内容をいちいち確認しないといけないので，たしかに手間が少し増えますが，この程度の手間でセキュリティが強化されるなら好ましいとわたしは感じました（このサービスパックのせいでWindowsの新しいバグが増えていないことを望みます）。

企業では，サービスパックの適用に動作確認を要するため，その導入に慎重にならざるを得ないようです。確認の過程では，時間と労働力，つまり，お金でコストを払う必要が生じます。同様に大企業ではセキュリティ対策もお金で解決することが多いでしょう。お金をかけることで解決できる問題であり，それが完璧になされるならば，それはそれで良いと思います。一方，大学においても，研究や業務に必要な時間の確保を理由に，お金でセキュリティの問題を解決しようという考えを持った人が以前より多くなった気がします。ウィルス対策ソフトウェアを買ったり，場合によっては業者に外注したり…。しかし，大企業とは違い，予算の使途が非常に限られることが多い大学では，セキュリティはお金で解決できる問題だとはわたしは思いません。セキュリティ対策は継続的にメンテナンスされなければなりませんが，大学では予算が継続的に使えることが少ない現状もその理由の一つです。まったく対策されないよりはましですが，一度だけお金を使って対策したからといって油断するのはあきらかに間違っています。

また，大学である以上，多くのみなさんが研究に関わっているのであり，研究の道具であるコンピュータの中身は多少なりとも理解しながら使ってほしいとわたしは望んでいます。何度も書くようでくどいですが，こまめなメンテナンスが今のコンピュータには必要不可欠です。面倒くさがらず，少しの愛情みたいなものをコンピュータにかけてほしいと思います。みなさんお忙しいかもしれませんが，道具は常日頃使うものだからこそ大切に扱いましょう。他人に任せてしまったり，お金で解決しようなんて思うのは悲しいかな，と。

さて，だらだらと続けてきましたこの連載も今回で最終回です。これまでDebian GNU/Linuxシステムにおいて，セキュリティレベルを保ちながら，管理コストを少しだけ下げる方法をいくつか紹介しました。愛情をかけながら，楽をする方法をとりあげたつもりです。最終回の今回は，あえて基本に戻り，簡単に実行できる小ネタを集めてみました。なにかの作業の合間に，ここで

紹介するコマンドを実行するだけで，みなさんのLinuxシステムのセキュリティを（ある程度ですが）確認することができます。それでは始めましょう。

Ⅱ．だれがログインしているか

不正アクセスの発見にはあなたのシステムにだれがログインしているか把握する必要があります。だれが，いつ，どこからログインしたのかの調査は怠らないようにしましょう。現在ログインしているユーザの確認にはいくつかの方法がありますが，代表的なもの，who，w，fingerを紹介します。

2.1 who

whoは，現在ログインしているユーザのログイン名，使っている仮想端末，ログインした時刻，リモートのホスト名を表示するコマンドです。実行すると，以下のような結果が得られます。

```
% who
chikako  :0        Jul 13 23:29
chikako  ttyp1     Jul 13 23:29 (:0.0)
chikako  ttyp2     Jul 13 23:29 (:0.0)
tsuyoki  pts/0     Aug 29 20:28 (pc5.test.nisinisi.jp)
tsuyoki  pts/1     Aug 29 20:28 (pc5.test.nisinisi.jp)
nisimura pts/3     Sep  1 02:17 (sh.nisinisi.jp)
```

chikako，tsuyoki，nisimuraの３人がログインしているのがわかると思います。chikakoは，:0.0からログインしていることになっていますが，これはコンソールからのログインをあらわしています。

2.2 w

つぎに，wです。wは，ログインユーザの表示の他，ログインユーザが何をしているかを調査することができます。また，システムの状態も表示してくれます。実行してみましょう。

```
% w
 02:30:20 up 55 days, 3:12, 7 users, load average: 0.01, 0.01, 0.00
USER     TTY     FROM              LOGIN@   IDLE   JCPU   PCPU    WHAT
chikako  :0      -                 13Jul04  ?xdm?  0.00s  ?       -
chikako  ttyp1   :0.0              13Jul04  2:05m  3:54   3:54    w
chikako  ttyp2   :0.0              13Jul04  2:12   13.44s 12.71s  emacs -nw main.
tsuyoki  pts/0   pc5.test.nisinis  29Aug04  7days  0.40s  0.05s   ssh -C sh.naist
tsuyoki  pts/1   pc5.test.nisinis  29Aug04  7days  0.10s  0.10s   -csh
nisimura pts/3   sh.nisinisi.jp    02:17    0.00s  0.05s  0.01s   w
```

結果の最初の行はシステムの状態を示しており，現在の時刻，システムが稼働している期間，現在ログインしているユーザ数，過去１，15，15分でのシステムの平均負荷（load average）が表示されています。特に心当たりがないのに（処理に時間が要するプログラムを動かしていないのに），平均負荷が高い状態であるときは，なにか不正なプロセスが動いている可能性もあるので

要注意です。

３行目以降は，ユーザごとの情報になります。ユーザ名，端末名，リモートホスト名，ログイン時刻，アイドル時刻，JCPU，PCPU，そしてユーザが実行しているプロセスのコマンドラインが表示されています。JCPUは，その端末（tty）から実行されている全プロセスが要した時間，PCPUは現在のプロセスが使った時間です。ここで注目したいのは，各行の最後“what”で示される実行中プロセスのコマンドラインです。この情報でユーザが何をしているかのだいたいがわかります。

上記の例では一行が長すぎて“what”の情報が途中で切れてしまっています。この場合，-sオプションをつけると，ログイン時刻，JCPU，PCPUを表示しなくなるので，より長く“what”を表示できるようになります。また，wコマンドのオプションにはユーザ名を指定することができ，該当するユーザの情報のみを表示できます。例えば，-sオプションと組合わせると，以下のように実行します。

```
% w -s nisimura
```

2.3　ちょっと危険な？ユーザ情報の表示（finger）

３つめのコマンドはfingerです。fingerは，詳細なユーザ情報を表示するためのプログラムです。なにもオプションをつけずに実行すると以下のような結果を得られます。

```
% finger
Login     Name              Tty    Idle  Login  Time   Office    Office Phone
nisimura  Nisimura Ryuichi  pts/3         Sep  7 02:17  (sh.nisinisi.jp)
tsuyoki   Tsuyoki Nishikawa pts/0    7d  Aug 29 20:28  (pc5.test.nisinisi.jp)
tsuyoki   Tsuyoki Nishikawa pts/1    7d  Aug 29 20:28  (pc5.test.nisinisi.jp)
```

なお，Debianではfingerコマンドはfingerパッケージで提供されているので，必要ならいつものようにapt-getでインストールしてください。例えば以下のようになります。

```
% sudo apt-get install finger
```

fingerコマンドのオプションにユーザ名を指定する，もしくは-lオプションを指定すると，より詳細な情報を表示できます。

```
% finger nisimura
Login: nisimura                         Name: Nisimura Ryuichi
Directory: /home/nisimura               Shell: /bin/csh
On since Tue Sep 1 02:17 (JST) on pts/3 from sh.nisinisi.jp
Mail forwarded to nisimura@linux.or.jp
No mail.
No Plan.
```

ご覧のように，ログイン名やログイン時刻などとともに，ユーザのフルネームやホームディレクトリのパス，ログインシェルなどが表示されます。これらは/etc/passwdファイルに記録されている情報であり，chfnコマンドなどを利用することで変更できます。

ところで，fingerコマンドはネットワーク経由で他の計算機のユーザ情報を見る機能も持っています。例えば，pc1.test.nisinisi.jp上のnisimuraというユーザの情報を見るには，

```
% finger nisimura@pc1.test.nisinisi.jp
```

とします。また，ユーザ名を指定せず，“ @ホストネーム ”だけをオプションに与えると，その計算機にログインしているユーザのリストを得ることができます。

この機能は，一見すると便利ですが，外部の第三者にユーザ情報を与えてしまうことになるため危険性を含んでいます。これらユーザの個人性に関わる情報は公開すべきではありません。実は，ここで述べたネットワーク経由での情報公開を許すには，その公開する計算機上でfingerd（fingerのデーモンプログラム）が動いていなければなりませんので，公開したくなければfingerdをインストールしなければ良いだけなのです。Debianではfingerdは，cfingerdやefingerdといった名前のパッケージで提供されていますが，特別な理由がある場合を除きインストールしない方が無難でしょう。

自分の計算機で，情報公開を許しているかどうかを確認する方法としては，localhostに対してfingerすることが挙げられます。以下のコマンドを試してください。

```
% finger @localhost
```

fingerdが動いてなければ，

```
[localhost]
finger: connect: Connection refused
```

といった感じに接続が拒否されるはずです。もし，情報が表示されてしまったらfingerdを停止することをお勧めします。特にインターネットに直接繋がっているような計算機では公開は避けるのが得策です。どうすれば良いでしょう？

一番単純な方法は，fingerdを消してしまうことです。Debianの場合，dpkg --get-selectionsでインストールされているfingerdを確認して，そのパッケージを消してしまいましょう。

```
% dpkg --get-selections |grep fingerd
cfingerd                        install
% sudo apt-get remove cfingerd
```

パッケージを使っていない場合，fingerdはinetdという名のデーモンプログラムから呼び出され，動作するようになっていますので，inetdの設定を変更してしまうことで止めることがで

きます。inetdの設定ファイルは/etc/inetd.confです。このファイルの中から先頭が“ finger ”ではじまる行を削除します。わたしの/etc/inetd.confには，以下の行がありましたので，先頭に‘ # ’をつけてコメントアウトしました。

```
# finger   stream tcp  nowait root  /usr/sbin/tcpd /usr/sbin/cfingerd
```

続いて，inetdデーモンを立上げ直し，設定変更を有効にしてください。一例としてkillallコマンドを使ってみると，

```
% sudo killall -HUP inetd
```

といった感じです。できましたでしょうか？localhostにfingerして動かないことを確認してみましょう。

2.4 ログイン履歴を見る（last）

ここまで紹介した３つのコマンドは現在ログインしているユーザを確認するためのものです。過去のログイン履歴を確認するにはlastを使います。引数なしにlastを実行すると，

```
% last
nisimura pts/0       sh.nisinisi.jp   Tue Sep 2 11:13  still logged in
reboot   system boot 2.4.26           Tue Sep 2 11:13         (00:00)
nisimura pts/1       sh.nisinisi.jp   Tue Sep 2 11:12 - down  (00:00)
chikako  pts/2       pc1.test.nisinis Tue Sep 2 11:08 - 11:12 (00:03)
nisimura pts/0       sh.nisinisi.jp   Tue Sep 2 02:14 - down  (08:58)

wtmp begins Tue Sep 2 00:58:29 2004
```

といった感じの出力が得られ，ユーザ名，端末名，リモートホスト名，ログインしていた時刻を示しています。また，“ Tue Sep 2 11:13 ”にシステムがリブートしたことがわかります。このときログインしていたユーザは，強制的にログアウトされるため，ログアウト時間の部分が“ down ”と表示されています。また，“ still logged in ”と表示されているのは，現在ログイン中のユーザです。

最後の行を注目してみてください。“ wtmp begins Tue Sep 2 00:58:29 2004 ”はなにをあらわしているのでしょうか。少し正確にlastの動作を説明すると，lastは/var/log/wtmpファイルに記録されているログインのログを表示するためのコマンドです。先ほどのメッセージは，現在表示している/var/log/wtmp/ファイルが，Tue Sep 2 00:58:29 2004以降に作られたファイルであることを意味しています。Debianでは，/var/log/wtmp/は，他のログファイルと同様に，古くなったものからファイル名を変更し，削除されるようになっています。例えば，一つ過去のwtmpファイル（/var/log/wtmp.1というファイル名になっていることが多い）の中身でログイン履歴を確認したいときは，

```
% last -f /var/log/wtmp.1
```

と実行することになります。

Ⅲ．不審なプロセスの監視（ps）

つぎに，動いているプロセスのチェックをしてみましょう。不正なユーザによるログインがなくても，正規ユーザのアカウントがのっとられ，不正なプログラムが動かされてしまうこともあります。不審なプロセスが動いていないかの確認が肝心です。

前述のwコマンドでも，ユーザが実行しているコマンドラインはわかりますが，バックグランドで動いている不正プロセスは発見できません。ここで登場するのが，おそらくみなさんもご存じのpsコマンドです。さっそく実行してみます。psには指定することができるオプションが多くあり，その役割の解説は省略します。オンラインマニュアルを参照してください。わたしがpsを使うときは，オプションにauxwを使うことが多いので，以下はそのときの出力例です。

```
% ps auxw
USER      PID %CPU %MEM   VSZ  RSS TTY   STAT START TIME COMMAND
root        1  0.0  0.1  1272  484 ?     S    11:13 0:06 init [2]
root        2  0.0  0.0     0    0 ?     SW   11:13 0:00 [keventd]
root        3  0.0  0.0     0    0 ?     SWN  11:13 0:00 [ksoftirqd_CPU0]
root        4  0.0  0.0     0    0 ?     SW   11:13 0:00 [kswapd]
root        5  0.0  0.0     0    0 ?     SW   11:13 0:00 [bdflush]
root        6  0.0  0.0     0    0 ?     SW   11:13 0:00 [kupdated]
root      171  0.0  0.2  1344  596 ?     S    11:13 0:00 /sbin/syslogd
root      174  0.0  0.4  1968 1080 ?     S    11:13 0:00 /sbin/klogd
root      181  0.0  0.1  1292  508 ?     S    11:13 0:00 /usr/sbin/inetd
root      189  0.0  0.4  2756 1244 ?     S    11:13 0:00 /usr/sbin/sshd
root      197  0.0  0.2  1652  684 ?     S    11:13 0:00 /usr/sbin/inetd /tmp/.inetd.conf
root      201  0.0  0.1  1252  468 tty1  S    11:13 0:00 /sbin/getty 38400 tty1
nisimura 4586  1.5  0.6  5688 1788 ?     S    23:05 0:00 /usr/sbin/sshd
nisimura 4587  0.5  0.4  2216 1220 pts/0 S    23:05 0:00 -bash
nisimura 4589  0.0  0.5  3308 1424 pts/0 R    23:05 0:00 ps auxw
```

ここで注目してほしいのは，下から５行目です。root権限でinetdというプロセスが動いているのがわかります。inetdは，fingerdのところでも使ったように他のサービスを起動するための特殊なデーモンであり，動いていること自体は珍しいものではありません。しかし，下から７行目でもinetdが動いており，このシステムでは二つのinetdが動いていることがわかります。さらにです。片方のinetdは，/tmp/.inetd.confという設定ファイルで動いています（inetdは，/etc/inetd.confを設定として使うのが一般的です）。これは不正に動作しているinetdであり，このようなプロセスが動いているということはすでに不正アクセスされてしまった後，となります。設定ファイルが/tmp/.inetd.confとドットからはじまる名前になっているのは，-aオプションをつけないとlsではドットファイルは表示されないことを利用して，発見

を困難にすることを狙ったものです。このような不正なinetdによる不正アクセスの窓口は，バックドアと呼ばれる常套手段の一種です。

Ⅳ．不審な通信を探す（netstatとnmap）

さらに不審な通信がないかのチェックも必要です。まず，覚えなければいけないコマンドはnetstatです。以下の例のように-anオプションをつけて実行してください（ここでもオプションの詳細は省略します）。

```
% netstat -an
Active Internet connections (servers and established)
Proto  Recv-Q  Send-Q  Local Address       Foreign Address       State
tcp         0       0  0.0.0.0:37          0.0.0.0:*             LISTEN
tcp         0       0  0.0.0.0:9           0.0.0.0:*             LISTEN
tcp         0       0  0.0.0.0:13          0.0.0.0:*             LISTEN
tcp         0       0  0.0.0.0:22          0.0.0.0:*             LISTEN
tcp         0       0  0.0.0.0:23          0.0.0.0:*             LISTEN
tcp         0     272  192.168.24.106:22   192.168.24.10:50171   ESTABLISHED
udp         0       0  0.0.0.0:9           0.0.0.0:*
udp         0       0  192.168.24.106:123  0.0.0.0:*
udp         0       0  127.0.0.1:123       0.0.0.0:*
udp         0       0  0.0.0.0:123         0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags     Type     State       I-Node Path
unix  5      [ ]       DGRAM                325    /dev/log
unix  2      [ ACC ]   STREAM   LISTENING   79     /var/run/pump.sock
unix  3      [ ]       STREAM   CONNECTED   3795
unix  3      [ ]       STREAM   CONNECTED   3794
unix  2      [ ]       DGRAM                3283
unix  2      [ ]       DGRAM                422
unix  2      [ ]       DGRAM                354
```

netstatの出力は，インターネットドメインソケットとUNIXドメインソケットに関する部分の二つから構成されています。今回は，前半のTCP/IP通信に関係するインターネットドメインソケットの方に注目します（“ ActiveUNIX domain sockets ”より前）。各行の最後の“ State ”がESTABLISHEDになっているものは通信が確立しているものです。この例ではESTABLISHEDは一つですが，192.168.24.10:50171（こちらはネットワーク経由の別の計算機“ Foreign Address ”）から192.168.24.106:22（こちら側がnetstatを実行しているローカル“ LocalAddress ”）に通信していることがわかります。192.168.24.106:22の192.168.24.106はIP，22はポート番号です。行の先頭“ Proto ”からこれはTCPの通信であることがわかります。TCPの22番はsshが受け口として用いるポートです。192.168.24.106はnetstatを実行している方の計算機ですが，この通信は192.168.24.10からのsshを使ったログインが成立していることをあらわします。このように，不正な通信，不審なIPからの通信，心当たりのないポートへのアクセスがないかチェックをすることができます。

“ State ”がLISTENになっているのは，ネットワークの接続をポートを開けた状態で待っていることをあらわします。つまり，TCPでは9，13，22，23，37のポートが外部ネットワークに対して開放されていることがわかります。もし，ここで必要ではないポートが開いているようだったら，ポートを閉じる等の対策が必要です。22は先ほども述べたようにsshのものです。さて，それ以外のポートはどのサービスで利用されているものでしょうか。ここでは`nmap`を使って調べる方法を紹介します。

`nmap`はTCP/IPのポートをスキャンすることで開いているポートを調べるためのツールです。実は，`netstat`を使わずとも`nmap`だけで開いているポートを確認することができます（ESTABLISHED等の確認には不可）。通常はネットワーク上の別の計算機のポートスキャンに利用することが多いのですが，今回は自分自身（`localhost`）に対して実行してみましょう。Debianで`nmap`は`nmap`パッケージで提供されているので，`apt-get`でインストールしてから実行します。

```
% sudo apt-get install nmap
% nmap localhost

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on debian (127.0.0.1):
(The 1549 ports scanned but not shown below are in state: closed)
Port       Stat     Service
9/tcp      open     discard
13/tcp     open     daytime
22/tcp     open     ssh
23/tcp     open     telnet
37/tcp     open     time
```

ご覧のように，開いているポートを，そのポートに対応するサービスの名前とともにリストにします。（一般的に良く使われるポートとして定義されていない）well-knownではないサービスに関しては，“ Service ”がunknownと表示されます。この結果，23はtelnetのポートであることがわかりました。discardやdaytime，timeはinetdが内蔵のルーチンとして提供しているものであり，受け取ったパケットそのまま破棄（discard）や時間情報を接続相手に返す（daytime，time）簡単なサービスです。これらを使うことはほとんどないので，ポートを閉じてしまうのが賢明です。`inetd`が提供しているサービスなので，`/etc/inetd.conf`から該当するエントリを削除（コメントアウト）し，`killall -HUP inetd`してポートを塞いでください。

`nmap`を使う上で注意事項を一点補足しておきます。`nmap`はネットワーク上の他の計算機に対してポートスキャンを実行できます。例えば，

```
% nmap www.example.ac.jp
```

で，`www.example.ac.jp`をポートスキャン（すべてのポートが開いているかチェック）します。

しかし，ポートスキャンはそれ自信が不正アクセス（攻撃）と勘違いされる可能性があるので，むやみに実行するのは好ましくありません。nmapは，みなさんが自分自身で管理されている計算機に対してのみ行うようにしてください。

Ⅴ．setuidされたファイルを探そう

sudoはこの連載で頻繁に登場したコマンドであり，ユーザに一時的にroot権限を与えることができる便利なものです。ところで，なぜsudoは，root権限を扱うことができるのでしょうか。sudoが持つ属性をlsで確認してみましょう。

```
% ls -l /usr/bin/sudo
-rwsr-xr-x   1 root   root     80584 Apr 27 2002 /usr/bin/sudo
```

“ -rwsr-xr-x ”は，プログラムのアクセス権限を示しています。この中で“ rws ”の“ s ”は，このプログラムが実行されたとき，このプログラムの所有者の権限の下で実行されることを示しているシンボルです。sudoの所有者はrootですので，rootの権限の下で実行されます。このためroot権限を一般ユーザに与えることもできるのです。このようにroot権限を付与されたものをsetuidされたプログラムと言います。もしも不正な処理をするプログラムがrootにsetuidされたら，root権限を奪われてしまい，大変危険です。Debianをはじめとするプログラム群の中には，このようにrootの権限が動作上どうしても必要であり，そのため，setuidされたプログラムがいくつか存在します。setuidはchmodコマンドで設定することができますが，root権限が不必要なプログラムに権限を付与してはいけません。と，言うことは，rootの権利を狙ってシステムを攻撃するクラッキング行為では，このsetuidがしばしば悪用されます。

したがって，どのプログラムがsetuidされているか。それをあなたが把握しておくのがシステムの安全を保つ上で必要になってきます。システム上のいろいろなところにちらばっているsetuidプログラムを一覧にしてみましょう。findコマンドを用います。

```
% sudo find / -type f -perm -u+s -ls
 1480  20 -rwsr-xr-x  1 root  root  19048 Apr  8 2002 /usr/bin/newgrp
 1548  28 -rwsr-xr-x  1 root  root  25864 Apr  8 2002 /usr/bin/chfn
 1549  24 -rwsr-xr-x  1 root  root  23944 Apr  8 2002 /usr/bin/chsh
 1551  36 -rwsr-xr-x  1 root  root  33064 Apr  8 2002 /usr/bin/gpasswd
 1552  28 -rwsr-xr-x  1 root  root  24680 Apr  8 2002 /usr/bin/passwd
  132  84 -rwsr-xr-x  1 root  root  80584 Apr 27 2002 /usr/bin/sudo
  502  24 -rwsr-xr-x  1 root  root  22460 Oct  1 2001 /usr/bin/crontab
```

指定したオプションの / は，ルートディレクトリ（システムの最上位ディレクトリ）から探索を開始することを指定したものです。-type fはファイルタイプのファイルのみを探索，そして-perm -u+sでsetuidされたものを探索することを指定しています。最後の-lsはファイルの詳細を表示するオプションです。この例の出力のようにsetuidされたプログラムの一覧を得ること

ができ，把握することができると思います。

しかし，このリストをこまめにチェックして，不審なプログラムがsetuidされていないか監視するのは少々やっかいですよね。Debianをつかっていると，これを楽にすることができます。/usr/sbin/checksecurityスクリプトがそれであり，標準設定でcronから毎日実行されるように設定されています。/usr/sbin/checksecurityは，毎日のsetuidの変化を/var/log/setuid.changesの中に記録します。/var/log/setuid.changesを確認し，不審なプログラムがsetuidされないか監視すれば良いわけです。中身を覗いてみると，以下のようになっているはずです。この例のように/devの下のptmxやttyのタイムスタンプ（日時情報）が変化することは正常ですので，とりあえず無視できます。しかし，/devの下に不正プログラムがコピーされることもあるので注意は怠らないようにしてください。

```
% sudo less /var/log/setuid.changes
debian changes to setuid programs and devices:
--- setuid.today     Mon Sep 6 06:25:31 2004
+++ /var/log/setuid.new.tmp     Tue Sep 7 06:25:04 2004
@@ -2867 +2867 @@
-   33399 666 1 root   tty     0 Sun Sep  5 23:18:56 2004 /dev/ptmx
+   33399 666 1 root   tty     0 Tue Sep  7 03:42:27 2004 /dev/ptmx
```

もっと楽をしたい方向けに，/etc/checksecurity.confの中のCHECKSECURITY_EMAILをrootに書き換えれば，setuidの変化情報をroot宛にメールで送信してくれます。標準設定ではメール送信はしてくれませんので，設定することをお勧めします。

Ⅵ．おわりに

さて，そろそろ大詰めです。今回は，wやps，nmap等を用いてシステムの不審箇所を探すことは可能であると述べました。しかし，不正ログインを許してしまったシステムでは，クラッカーによってこれら基本コマンドは改竄したものに置き換えられてしまっている可能性があるのも覚えておきましょう。改竄されてしまったプログラムでは正しい状態が出力される保証は全くありません。root権限が奪われてしまった場合は，おとなしくシステム全体を空の状態から再インストールするのがベストです。

「じゃあ，なんだ，今回の内容は無駄じゃないか。」と思う方もいらっしゃるかもしれません。しかし，何度も言いますが，常日頃のこまめなチェックが不正アクセスを防ぐための最も効果的な手段です。これらの基本を知っているのと知らないのでは，あなたのスキルは違ってくるはずです。ぜひ覚えておいてください。

Ⅶ．ありがとうございました

連載は以上で終わりです。私事ですが，この４月からは和歌山大学に勤務し，日々の雑務に追われております。わたしはまだ恵まれている方で，忙しい毎日の中で巨大ネットワークの維持に

日々努力されている先生方を拝見すると頭が下がる想いです。名古屋から離れて久しく，少しさびしいですが，これからも母校名古屋大学の発展を願っております。これまでお付き合いいただき，ありがとうございました。

（にしむら　りゅういち：和歌山大学システム工学部）
（nisimura@linux.or.jp）