Linuxによるセキュリティ入門（3）
― iptablesを用いたパケットフィルタリング ―

西　村　竜　一

Ⅰ．未知の危険に備えて

セキュリティホールを利用したクラッキングに対する最も大事な対策は，日頃からセキュリティパッチのシステムへの適用を忘れないことです。みなさん`apt-get upgrade`を忘れずやってますか？使わないサーバプログラムは動かさないことも大変重要な鉄則です。必要としないパッケージをインストールしてはいけません。サーバプログラムを無闇に立ち上げてると，そのプログラムのセキュリティホールが発見されたときに大変危険です。Debianでは，この２点の対処だけでとりあえずのセキュリティ対策は十分だと言えます。

しかし，だからと言って油断するのも良くありません。敵（クラッカー）はさまざまな新しい方法を考えて攻撃をしてきます。その手段はだんだんと悪質になってきていると言えるでしょう。広く一般には知られていない未知のセキュリティホールを利用したクラッキングを仕掛けてくるかもしれません。それでは，さらなるセキュリティ対策として我々は何ができるのでしょうか。

さらなるセキュリティ対策と言っても，本連載のポリシーとして，高価な機器やコンサルタント契約が必要なものはあまり導入したくありません（お金持ちな人は，セキュリティ対策にお金をかけるのも一つの選択枝だとは思います)。また，セキュリティ対策のために専任の管理者が必要なものも除外して考えます。安く，かつ，毎日の少しの努力だけでセキュリティレベルを確保する方法を考えてみましょう。例えば，以下の基本的な対策を挙げることができます。

安価なPCルータもしくは家庭用ルータによるファイヤウォールの導入
パケットフィルタの導入
詳細な通信ログの記録

最近，最もよく使われるセキュリティ確保の手段は，家庭用ルータの導入だと思います。これらのルータには簡易のファイヤウォール機能が実装されているので，LAN内の計算機のセキュリティを確保することが可能です。Linuxの入ったPCをルータにするのが数年前に流行しましたが，現在では簡単に設定ができる家庭用ルータが１万円程度で売られていますので，それらを導入するのが良いでしょう。家庭用に販売されてますが，研究室単位のような小規模なLANとインターネットの接続にも十分に使えると思います。ただし，`192.168.`ではじまるようなプライベートIPのLANをNATによりインターネット接続することを想定して設計されていることが多いので，グローバルIPによるLANを運用する場合には，ルータに必要な機能が揃っているか確認してください。

ファイヤウォールの導入は，たしかに外部からの攻撃に対しては有効な防御手段です。しかし，LAN内部からの攻撃に対しては効果はありません。例えば，最近流行っているコンピュータウィルスのほとんどはワームと呼ばれるものです。ワームがメールなどを経由してLAN内部の計算機に感染した場合，その計算機はLAN上の他の計算機への攻撃を試みます。こうなるとファイヤウォールは意味をなさず，LAN上の個々の計算機においてセキュリティ対策がなされていないと攻撃を防ぐことはできません。また，同じノートPCを外部と内部の両方で持ち運んで利用しているとそのノートPCを経由してワームが侵入する可能性もあります。ファイヤウォールはどんな方法であれ，いったん破られてしまったら役には立ちません。

あなたはファイヤウォールを過信していませんか？その過信がセキュリティに対する意識を低くしている原因にはなっていませんか？やはり個々の計算機においていままで述べたセキュリティアップデートの適用などの対策が必要不可欠なのです。あなたはLANの管理者がセキュリティを確保してくれると勝手に思い込んでいませんか？何度も言いますが，すべてのインターネット利用者がセキュリティに対して常に高い意識を持っている必要があるのです。このようにファイヤウォールは，とりあえずのセキュリティの確保には便利だが決して頼りきってはいけないのだと理解してください。

つぎに，今回説明をするパケットフィルタの導入はセキュリティ確保には非常に強力な手段です。パケットとはネットワーク上に流れるデータの集合のことです。このパケットをフィルタリングすることにより，ネットワークから計算機に入ってくるデータや出ていくデータを遮断します。前述のように計算機上にインストールするプログラムを必要最低限なものに限定するとセキュリティ的にはより強固な計算機になります。パケットフィルタでは，それをさらに押し進めて，通信の入出力の流れを，使用する必要最低限のものに限定することができます。

例えば，この連載でも紹介したsshはネットワーク上の計算機に遠隔ログインするためのプログラムです。sshは，それ単体でも非常にセキュリティレベルが高いプログラムなのですが，残念ながら今でも時々セキュリティホールが発見されます。もし，そのセキュリティホールの対策がなされるまえに攻撃を受けてしまったら，sshを使っていることでクラッカーによる侵入を許してしまうことになります。これを防ぐためには，パケットフィルタによってsshでの接続元を特定の計算機のみに制限するのが効果絶大です。

パケットフィルタを利用しなくとも，sshやApacheのように高機能なサーバプログラムは，送信元のIPアドレス等によってアクセスを制限する機能を持っています。しかし，それらの機能は，サーバプログラムがパケットを受信した後，そのパケットは受け取るべきか，捨てるべきかを判断します。このため，この判断処理をする前の段階のセキュリティホールが存在する場合，アクセス制御が破られてしまう可能性があります（有名なプログラムでは，そのようなセキュリティホールはすでに調べ尽くされていることがほとんどだと思いますが…）。そこで，今回は，サーバプログラムごとの個別のアクセス制御方法ではなくLinuxカーネルによるパケットフィルタリングを`iptables`コマンドを用いて設定する方法を紹介します。

最後の対策は，詳細なログの記録です。通信内容の記録であるログを収集しても，それ自体は

攻撃に対する予防にはなりません。しかし，攻撃があったことを知ることができれば，今後の対策の参考にできます。ログは，サーバプログラムがそれぞれ出力し，Debianでは/var/log/ディレクトリ下のファイルに保存されます。また，Linuxでは，さらに詳細なログの収集や，ポートスキャンや攻撃などの異常をログに発見した場合のメール通知を利用することができます。本連載の次回以降では，これらログの読み方や詳細なログ記録の設定方法について説明したいと思います。

それではLinuxカーネルによるパケットフィルタリングについて話を進めましょう。

Ⅱ．インターネットプロトコルとポート

まず，インターネット上に使われているプロトコルの基本について簡単におさらいします。プロトコルとは計算機同士が通信するときに使用する決まりのことです。詳細は他の解説本に任せるとして，インターネットを使って通信するときに基本となるプロトコルはTCP/IPです。このTCP/IPとは，TCPとIPという２つのプロトコルの組合わせを意味します（図１）。

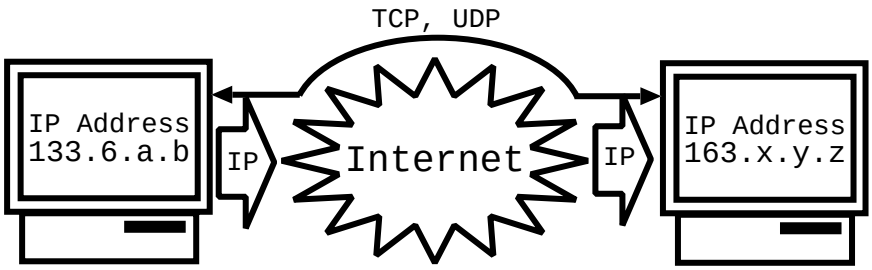

図１　送信元と送信先の計算機の間はIPアドレスを手がかりにIPでパケットを伝えます。IPの上ではTCPやUDPによって通信をします。

IPアドレスという単語で有名なIP（Internet Protocol）は，インターネットに接続された計算機のアドレスを指定し，そこまでパケットを送るためのプロトコルです。このIPの働きによって，パケットは途中さまざまなルータなどを経由して通信先の計算機に届きます。

IPはどの計算機にパケットを届けるかまでは担ってくれますが，そのパケットがエラーがなく正しく伝わったかどうかまでは関与してくれません。それらはIP上で動作するプロトコルであるTCP（Transmission Control Protocol）が処理します。TCPでは，パケットの並びかえや途中でパケットがロストした時の再送処理を行ない，確実性の高い通信を実現します。ただし，TCPは，その処理のため負荷が高くなるため大量のビデオデータの送受信には向かないことがあります。また，通信内容によっては少しのパケットロスは構わないがリアルタイム性の高い通信を必要とする場合があります。TCPでは，通信に遅延が生じることがあります。このような目的には，TCPの代わりに，よりシンプルなUDP（User Datagram Protocol）というプロトコルが使用されます。

送信先の計算機に届いたパケットは，サーバプログラムに引き渡されます。このデータを引き

渡すときの窓口のことをポートと言います。httpdやssh, sendmailなどのすべてのサーバプログラムは，必ずポートを経由して送られてきたデータを受け取ります。各計算機には，１～65535のポートがあり（本当は０も含みますが，特別なポートなので除外します），１つのポートに対しては１つのサーバプログラムのみ動かすことができます。つまり，ポート番号さえ知っていればパケットを受け渡す相手のプログラムを指定できることになります（図２）。

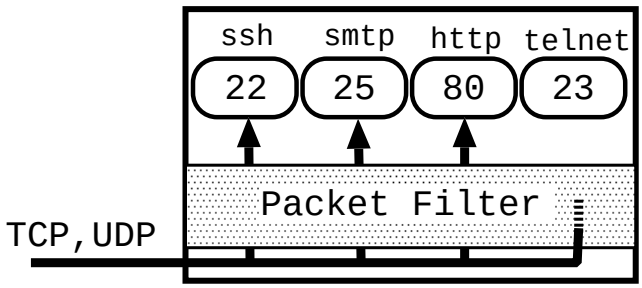

図２　smtpやhttpなどのサーバプログラムはポートを窓口にしてデータを受け渡しします。パケットフィルタによって遮断されてしまったポートはデータを受け取ることができません。

我々はどこかのWebサーバにアクセスする時，そのWebサーバのhttpdが開いているポートの番号を知っている必要があります。Webサーバの管理者は，自分の好きなポート番号でhttpdを立ち上げることができますが，あまり自分勝手な番号を使用すると複雑になってしまいます。そこで良く使うサーバプログラムに対しては，できるだけ同じポート番号を使うことになっています。これらポート番号は，`/etc/services`ファイルのリストに記述されています。実際にファイルの中身をのぞいてみましょう。行がたくさんありますが，その中で以下の３行を見てください。

```
smtp          25/tcp        mail
www           80/tcp        http          # WorldWideWeb HTTP
www           80/udp                      # HyperText Transfer protocol
```

行の先頭がサービスの名前であり，smtpはメール，wwwはWWW（http）プロトコルを示しています。２列目の“`25/tcp`”，“`80/tcp`”がポート番号です。このようにメールは25番，WWW（http）は80番のポートを使うよう定義されています。また，ポート番号は，TCPとUDPでは別々に独立に利用できるので，WWW（http）に対して“`80/tcp`”と“`80/udp`”のTCPとUDPおのおののポート番号が定義されています（ただし，現在のhttpはTCPを使いますのでUDPのポートは使用しません）。

パケットフィルタリングでは，これらポートへのアクセスを遮断します。サーバプログラムに受け渡される前にパケットを遮断しますので，もしサーバプログラムにセキュリティホールが存在しても攻撃を防ぐことができます。また，外からのポートへのアクセスを制限するだけでなく，

自分から他への計算機へのアクセスもパケットフィルタリングによって同様に制限することもできます。

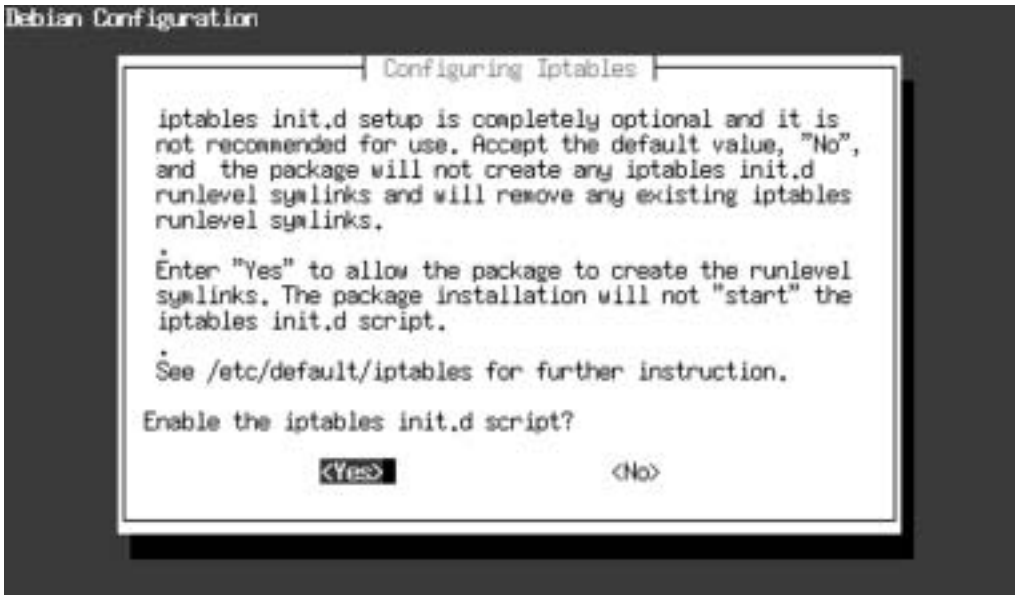

図３　apt-getを用いたiptablesのインストール

Ⅲ．前準備とカーネルの再構築

今回，紹介するパケットフィルタリングは，Linuxカーネルに含まれる機能ですので，カーネル内に必要な機能が組み込まれている必要があります。そこで，前準備としてカーネルの再設定が必要です。また，パケットフィルタのフィルタリングルールなどの設定には`iptables`というコマンドを用います。なお，この`iptables`コマンドは，Linuxのカーネルバージョン2.4から使われるようになりました。それ以前のカーネルでは，バージョン2.2で`ipchains`，バージョン2.0で`ipfwadm`が使われてきました。現在の2.4でも`ipchains`，`ipfwadm`コマンドは，過去との互換性のために使えるようになっていますが，今後はサポートされなくなる予定のため今回は使いません。

さて，それでは`iptables`コマンドをインストールしましょう。前回に引き続きroot権限への切り換えには`sudo`コマンドを使います。

```
% sudo apt-get install iptables
```

続いて，図３の画面が表示されます。これは「保存されている設定内容をシステムの起動時に有効にするか」という設問です。ここでは“ Yes ”と答えてください。

つぎに，カーネルの設定を変更します。カーネル再構築の方法については，本連載の前回

「Linuxカーネルの構築」で説明しましたのでそちらも参考にしてください。なお，以下の例では，前回に続き，~/kernel/linux-2.4.20ディレクトリにカーネルのソースが展開されているものとして説明をします。

カーネルの設定変更には，make menuconfigを使います。カーネルを構築した後に設定を再び変更する際は，一度，make-kpkg cleanを実行して中間生成ファイルの削除を行なってください。

```
% cd ~/kernel/linux-2.4.20
% make-kpkg clean
% make menuconfig
```

設定メニューが開いたら，“Networking options ---> ”を選択して階層を移動します。カーソルを移動して，リターンキーを押します（図４）。

つぎに“Network packet filtering (replaces ipchains)”を組み込みます。この項目の先頭が[ ]になっている時は，ｙキーを押して組み込んでください。先頭が，[*]になれば，組み込みができていることをあらわしています（図５）。

Network packet filteringが組み込まれると，同じ階層に“IP: Netfilter Configuration ---> ”という項目が新たに追加されます。先ほどの“Network packet filtering

図４　Networking options階層へ移動

図５　Network packet filteringの組み込み

図６　IP: Netfilter Configuration階層へ移動

図７　IP: Netfilter Configuration階層（初期画面）

(replaces ipchains)”の項目の少し下のほうにありますので，カーソルを移動して選択してください（図６）。１つ下の階層に移動して図７に示すIP: Netfilter Configurationが表示されます。

この中で必要な機能を組み込む必要がありますが，項目が多く複雑で必要な機能のみ正しく選択するのは難しいのが現状です。iptablesコマンドは，これら項目をモジュール化しておけば，必要なときに読み込んで使うことができます。ということで，すべてモジュール化してしまいましょう。ただし，過去との互換のために提供されているipchainsとipfwadmは使いませんから，最後の２項目（“ipchains (2.2-style) support (NEW)”と“ipfwadm (2.0-style) support (NEW)”）に関しては指定する必要はありません。モジュール化するには，カーソルを項目のところに移動させてmキーを入力します。項目の先頭に，<M>と表示されれば，その項目はモジュールとして選択されていることになります（図８）。一部，モジュール不可能な項目もありますが，その場合はカーネルに組み込んでしまって構いません。

図８　IP: Netfilter Configuration階層（すべてモジュール化）

カーネルの設定が終了したら，何度かESCキーを押してmake menuconfigを終了させます。続いて，コンパイルを行ない，生成されたカーネルパッケージのインストールをします。pcmciaやalsaのように外部モジュールが必要なものも再コンパイルしてインストールします（詳細は前回参照）。

```
% fakeroot make-kpkg --revision 20030401 kernel_image modules_image
% cd ..
% sudo mv /lib/modules/2.4.20 /lib/modules/2.4.20.old
% sudo dpkg -i kernel-image-2.4.20_20030401_i386.deb
```

つぎに，コネクション追跡モジュールというカーネルモジュールを起動時に自動的にロードするように設定します。エディタなどを用いて以下の２行を/etc/modulesファイルに追加してください。

```
ip_conntrack
ip_conntrack_ftp
```

最後に新しいカーネルでリブートします。

```
% sudo reboot
```

Ⅳ．フィルタリングルールの作成

さて，いよいよ本題のパケットフィルタリングの設定です。でも，まだその前に少しやることがあります。どのパケットをとおして，どのパケットを遮断するかのフィルタリングルールを考えなければなりません。

この時，基本となる考えは，Debianパッケージのインストールの時と同じで，必要十分な最低限度のパケットのみをとおすべきであるということです。とりあえず全ポートへのアクセスを遮断した後，必要なプロトコルのみを選択して，そのプロトコルに対してのみパケットフィルタに穴を開けましょう。不必要なプロトコルを削除していくのではなく，すべてカラの状態から必要なものを選ぶようにしてください。

ここで，すこしだけ面倒ですが，気をつけないといけない点に，本当にすべての外からのパケットを遮断してはならないという問題があります。ついさっきすべて遮断せよと言っておきながら，なんだと言われそうですが，完全にすべて遮断すると接続先の計算機から送られてくる返信データを受け取れなくなってしまい，双方向の通信が成立しません。また，ftpは，クライアントからサーバにアクセスした時，サーバからクライアントにも，もう１つ別のコネクションを張るプロトコルになっています。このように，すでに確立しているコネクションとの依存関係を見ながら，必要ならば外からのパケットを受け取るようにフィルタリングしなければなりません。すばらしいことにiptablesでは，これらのコネクションの依存関係を考慮したフィルタリングルールを簡単に作成できる仕組みが備わっています。これら依存関係を自動的に解決してくれるの

が，先ほど読み込んだコネクション追跡モジュールです。

それではフィルタリングルールを決めましょう。とりあえず最初は，以下のような例を考えることにします。

外部からのすべてのポートに対する新しいコネクションを拒否。

内部から外部へのアクセスは制限しない。

外部からsshdへのアクセスは受け付ける。

このルールによってパケットフィルタが設定された計算機には，外からはsshでのみ接続できるようになります。

Ⅴ．パケットフィルタの設定

それではルールにしたがってフィルタを設定しましょう。まず，新しいチェインを作成します。チェインとは，いくつかのルールをまとめて収納しておく入れ物のようなものだと考えてください。チェインの中にこれから設定するフィルタのルールを定義し，そのチェインをカーネルに渡すことによってフィルタを有効にします。今回の例では，`block`という名前のチェインを作成します。なお，`iptables`コマンドはすべてrootで実行します。本当は`sudo`を使ってほしいのですが，コマンドが長くなってしまうので，書面の都合上，毎回の`sudo`を省略して，説明します。

```
% sudo -s
# iptables -N block
```

作成されたか確認するために`-L`オプションでチェインのリストを表示します。

```
# iptables -L
```

さきほど作成したチェインの他に`INPUT`，`FORWARD`，`OUTPUT`という名前のチェインがあることがわかると思います。これらははじめから用意されている重要なチェインです。詳細は後述します。

それでは`block`チェインの中にルールを定義していきましょう。以下の例を見てください。

```
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
```

このルールは，外部からの新しいコネクションを拒否するためのルールです（外からのアクセスはすべて遮断）。“`-A block`”は，`block`チェインに新しいルールを追加（Append）することを示しています。なお，このルールをチェインから削除（Delete）するときは，`-A`を`-D`に変えて，

```
# iptables -D block -m state --state ESTABLISHED,RELATED -j ACCEPT
```

とします。

“ -m state ”は，パケットがルールに定義された条件にあてはまるかどうかのマッチ判定に，コネクション追跡モジュールによって拡張された機能を用いる時のオプションです。そして，“ --state ESTABLISHED,RELATED ”では，その条件を定義しています。ESTABLISHEDは，既存のコネクションに属するパケットを示しています。すでに確立したコネクションの応答のパケットはこれに該当します。RELATEDは，ftpのような既存のコネクションに関係するとわかっている新たなる接続のパケットにマッチします。

そして，最後の-jは，最終的にこのルールの動作をどうするかを定めるターゲットオプションです。このルールでは，上記の条件にマッチしたとき，パケットを受理する必要があるので，-j ACCEPTと指定しています。

つぎに，sshdへのアクセスを受け付けるためのルールをblockチェインに追加します。

```
# iptables -A block -p tcp --dport ssh -j ACCEPT
```

-pは，TCPやUDPのようなトランスポート層（この単語の意味は参考書を参照してください）のプロトコルを指定します。“ tcp ”，“ udp ”または“ icmp ”を指定することができます。sshは，TCPを使いますので，“ -p tcp ”とします。また，TCPでないパケットは“ -p ! tcp ”と書くことができます。ルールの記述の中で!記号は，否定をあらわします。

--dportは，宛先のポートを指定するオプションです。ポートの指定には，ポート番号もしくは/etc/servicesファイルに書かれているサービス名を使うことができます。つまり，“ --dport ssh ”は“ --dport 22 ”と書くこともできます。

やはりこのルールは受理する条件ですので，“ -j ACCEPT ”としています。

最後に，これらルールにあてはまらないパケットは拒否するようにルールを記述します。

```
# iptables -A block -j DROP
```

特にルールは指定していないので，この行より前に指定したルールにあてはまらないすべてのパケットが対象になります。また，-j DROPでパケットを捨てるように指定しています。

ルールの記述は以上です。正しくルールがチェインに追加したか確認するためにリストを表示しましょう。

```
# iptables -L block
```

以下のようなルールリストが表示されます。

```
Chain block (0 references)
target prot opt source    destination
ACCEPT all  --  anywhere  anywhere     state RELATED,ESTABLISHED
ACCEPT tcp  --  anywhere  anywhere     tcp dpt:ssh
DROP   all  --  anywhere  anywhere
```

ルールの作成はできましたが，このままではフィルタとしては動いてくれません。フィルタを有効にするのに重要なのが，さきほどすこし紹介したINPUT，FORWARD，OUTPUTの３つのチェインです。

INPUTチェインは，計算機に入っているパケットをチェックするためのチェインです。同様にOUTPUTチェインは，自分の計算機を送信元として出ていくパケットをチェックするためのチェインです。最後にFORWARDチェインは，計算機がルータとして動作している時，その計算機を通過していくパケットをチェックするためのチェインです。

よって，今回の例ではフィルタを有効にするために，INPUTチェインが，先ほど作成したblockチェインを参照するようにしなければなりません。この設定は，ターゲットオプション-jを用いて，

```
# iptables -A INPUT -j block
```

とします。

```
# iptables -L INPUT
```

でINPUTチェインの中身をみると，

```
Chain INPUT (policy ACCEPT)
target prot opt source    destination
block  all  --  anywhere  anywhere
```

と表示されて，blockチェインを参照していることが確認できます。

以上で，外の計算機からはsshのみがアクセスできるようになったと思います。正しく動いたでしょうか？この例ではsshでしたが，実際の運用では，外部からのアクセスをゆるしたいプロトコルに対して同様にルールを追加していくことになります。ただし，このままでは計算機内部同士の（localhostに対する）通信もできなくなっていますので，以下のルールも追加しておくと良い

でしょう（必須ではありません）。

```
# iptables -A block -i lo -j ACCEPT
```

このルールでは，計算機内部（localhost）からの通信は，すべて受け取るようにします。計算機内部（localhost）からの通信は，loという名前の仮想的なネットワークインタフェースを通じて送られてくるようになっています。-i loオプションを用いて，このloネットワークインタフェースを指定しています。

少し設定項目が多くなりましたので，ここまでの流れをおさらいします。チェインの作成からルールの設定，フィルタの有効化までのコマンドは以下のようになります。

```
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -p tcp --dport ssh -j ACCEPT
# iptables -A block -i lo -j ACCEPT
# iptables -A block -j DROP
# iptables -A INPUT -j block
```

．設定の保存

せっかく作ったチェインも，そのままシステムをリブートすると消えてしまいます。Debianでは，iptablesのチェインを保存して，システム起動時に読み込む便利なスクリプトが用意されているので利用しましょう。

まず，先ほどまでの例にしたがって，チェインを作成し，ルールを定義してフィルタを有効にします。つぎに以下のコマンドで，現在有効なチェインを保存します。

```
# /etc/init.d/iptables save active
```

保存したルールは，/var/lib/iptables/activeファイルに記録されます。

これだけでシステムを再起動してもパケットフィルタは有効になっていると思います。もし，apt-getでiptablesをインストールした時に，「保存されている設定内容をシステムの起動時に有効にするか」を“No”で答えていたら，

```
% sudo dpkg-reconfigure iptables
```

で設定を変更してください。

．設定の削除

チェインからルールを削除するには，前述したように-Aオプションを付けて実行したiptablesコマンドをそのまま，-Dオプションに変更して実行します。例えば，

```
# iptables -A INPUT -j block
```

を取り消して（INPUTチェインからルールを削除して），パケットフィルタを無効にするには，以下のようにします。

```
# iptables -D INPUT -j block
```

また，チェインの中身をすべて消してチェインを空にするには-Fオプションを使います

```
# iptables -F block
```

さらにチェイン自体を削除するには-Xオプションを用います。ただし，チェインの中身が空である必要があります。また，他のチェインから参照されているチェインを削除することもできません。

```
## INPUTチェインからの参照を削除
# iptables -D INPUT -j block
## blockチェインを空にする
# iptables -F block
## blockチェインを削除
# iptables -X block
```

変更を保存するときは，

```
# /etc/init.d/iptables save active
```

を再実行してください。

．他のフィルタリングルール例

以下ではいくつかの実用的なフィルタリングルールの設定例を紹介します。なお，チェインには，先ほどのblockチェインを用いています。

```
# iptables -A block -p tcp -s 192.168.1.1/32 --dport 23  -j ACCEPT
```

-sオプションで接続元のソースアドレスを指定できます。この例では，IPアドレス192.168.1.1のみからの23番ポート（telnet）への接続を許可しています。-s 192.168.10.0/24のようにネットワークのアドレスを指定することもできます。

```
# iptables -A block -p tcp -i eth1 --dport www  -j ACCEPT
```

ネットワークカードを複数枚持っている計算機の場合，-iオプションでネットワークカードを指定することができます。この例は，eth1のネットワークカードから入ってきたWebへのアクセスを許可する設定です。

```
# iptables -A block -p tcp --dport 6000:6999 -j ACCEPT
```

“ : ”を使ってポートを範囲で指定できます。この例では，6000番から6999番間の全ポートへのアクセスを許可します。また，6000番以上のすべてのポートは“ 6000: ”，6000番以下の全ポートは“ :6000 ”と指定できます。

```
# iptables -A block -m mac --mac-source 00:02:B3:20:0E:49 -j ACCEPT
```

ルールの定義にMAC（Media Access Control）アドレスを用いることもできます。例では“ 00:02:B3:20:0E:49 ”のMACアドレスを持つネットワークカードからの全ポートへのアクセスを許可しています。MACアドレスは，ネットワークカードに固有な物理的アドレスのため，基本的にはIPアドレスよりも高いセキュリティを実現することができます。ただし，ルータを越えた別のLANにつながった計算機のMACアドレスを指定することはできません。Linuxではifconfigコマンドで使っているネットワークカードのMACアドレスを調べることができます。

```
# iptables -A block -p icmp -j ACCEPT
```

ネットワークの診断に使うpingなどに利用されているICMP（Internet Control Message Protocol）を受け取るための設定です。この設定でpingに対して応答を返すことができるようになります。

外に出ていくパケットを制限することもできます。次頁の新たに作成したoutput-blockチェインでは，IPアドレス192.168.1.11を持つ計算機への接続ができなくなります。

```
# iptables -N output-block
# iptables -A output-block -d 192.168.1.11/32 -j DROP
# iptables -A OUTPUT -j output-block
```

．おわりに

これで，iptablesを用いたLinuxカーネルでのパケットフィルタの話は以上です。ルールの作成において，今回は簡単なルールの例しか紹介できませんでしたが，iptablesは他にもルール定義のためのオプションを持っており，複雑なルールの作成が可能です。最後に参考となるドキュメントを紹介します。詳しいiptablesのオプションの説明などはこれらの説明を参照してください。

Linux 2.4 Packet Filtering HOWTO
http://www.linux.or.jp/JF/JFdocs/packet-filtering-HOWTO.html
netfilter/iptables FAQ
http://www.linux.or.jp/JF/JFdocs/netfilter-faq.html
iptablesオンラインマニュアル
http://www.linux.or.jp/JM/html/iptables/man8/iptables.8.html
netfilter/iptablesのWebページ
http://www.netfilter.org/

次回は，ログの記録の仕方と読み方について説明します。

（にしむら　りゅういち：奈良先端科学技術大学院大学情報科学研究科）
（nisimura@linux.or.jp）