実践で学ぶ

一歩進んだサーバー構築・運用術

written by 仙石 浩明

第8回 「ファイアウォール(後編)」

前回紹介したTCP/IPの基礎知識を踏まえて，セキュアなサーバーを構築するための具体的な方法を解説します。Linuxカーネルに実装されているパケット・フィルタリングの条件を適切に設定して，外部の攻撃から内部LANを防御しましょう。

前回までで会社設立の話が済んでしまったので，今回は趣向を変えて*1，会社の1日について書きましょう。

私は典型的な夜型です。低血圧なためか寝起きが極めて悪く，8:30ごろに起きると午前中はもうろうとしています。たまに朝イチに社外の人との打ち合わせが入ってしまったりするのですが，目を開けているのが精一杯で，頭がまるで回らないので困ったものです。

普段は午前9:00過ぎに起きます。前の日遅かったりすると10:00ごろになることもあります。勤務時間は10:00～18:00ですが，フレックスタイム制なので遅刻にはなりません。これだけ遅いと，首都圏で有数の混雑路線と言われる東急田園都市線でさえも新聞が読めるほどすいています。

銀座線に乗り継いでトータルで会社まで1時間近くかかるので，日によっては昼食を済ませてから出勤することもあります。

午後は打ち合わせが入ることも多いので，じっくり研究開発に取り組むのは夕食後ということもしばしばです。で，調子に乗ると一気に深夜0:00になってしまいます。0:15までに会社を出ないと終電に乗れないので，続きは帰宅してから*2ということになります。こういうとき自宅が常時接続だと便利です。編集中のファイルを閉じることなく*3退社して，帰宅してから会社へログインして編集を継続できます。

パケット・フィルタリング

パケット・フィルタリングとは，特定の条件を満たすパケットを捨てることです。インターネットと内部LANをつなぐルーターにおいてフィルタリングを行えば，パケットはルーターを通過できず，内部LANを守ることができます。

「フィルタリング」と言うと，ルーターやNIC*4を2枚以上差したマシンで行うものと思い込んでいる方も多いのですが，NICを1枚しか持っていないマシンにおいてもフィルタリングは行うべきです。防御は何段階にもわたって行うべきで，フィルタリングは「最初のとりで」*5なのですから。

サーバー・マシンでは不必要なサービスを立ち上げてはいけない，とはよく言われることですが，立ち上げないだけでは不十分で，使用しないポートはフィルタリングでふさぐべきです。そう

＊1　本業については，ここに書いてしまっては大変です。
＊2　こんな夜更かししてるから次の日起きられなくなるのですけど。
＊3　もちろんトラブルに備えてファイルの保存はしますが，編集・参照中のファイルそれぞれのカーソル位置などの状態に至るまで，そのまま自宅で継続できます。
＊4　ネットワーク・インタフェース・カードの略です。イーサネット・アダプタなど，マシンをネットワークにつなぐための周辺機器です。
＊5　1999年11月号特集2「Linuxサーバーのセキュリティーを高める」のpp.103～110「フリーの定番ソフトを利用した4段階Linuxサーバー防御法 ---4つのとりでででクラッカーを撃退せよ---」を参照。「第2のとりで」はサーバー・プログラムの認証，「第3のとりで」は被害の限定，「最後のとりで」は侵入の検知です。

すれば何かのはずみ[*6]でサーバー・プログラムが動き出したとしても，外部から接続されずに済みます。

TCPパケットのフィルタリング

先月号で解説したように，TCPパケットのヘッダーには次の情報が含まれます。それぞれの項目が特定の条件を満たしたとき，そのパケットを捨てることによってフィルタリングを行います。この「特定の条件」を定めることがフィルタリングの設定です。

(1)送信元アドレス
(2)送信元ポート番号
(3)あて先アドレス
(4)あて先ポート番号
(5)シーケンス番号
(6)確認応答番号
(7)SYNフラグ
(8)ACKフラグ
(9)FINフラグ

NICを複数枚差しているマシン（デュアルホーム・ホスト），あるいはルーターの場合，上記情報に加えて，パケットがどのインタフェース(NIC)に届いたかを示した

(10)到着インタフェース

をフィルタリングの条件として用いることができます。

もし送信元アドレスが内部LANに属するものであるにもかかわらず，インターネット側のインタフェースに届いたら，そのパケットは送信元アドレスが偽造されています(IPアドレス・スプーフィング)。そのようなパケットを内部LANに入れては大変です。送信元アドレスと到着インタフェースが矛盾するパケットは必ず捨てる設定にしましょう(**図1**)。

このフィルタリングが正しく行われていれば，内部LAN上のホストは，送信元アドレスを見ることにより，届いたパケットがインターネットからのものか，それとも同じLAN上のホストからのものか判断することができます。

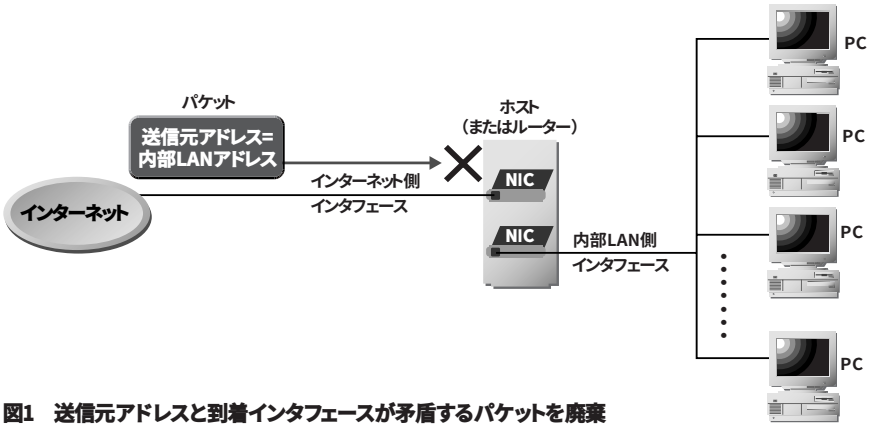

図1　送信元アドレスと到着インタフェースが矛盾するパケットを廃棄

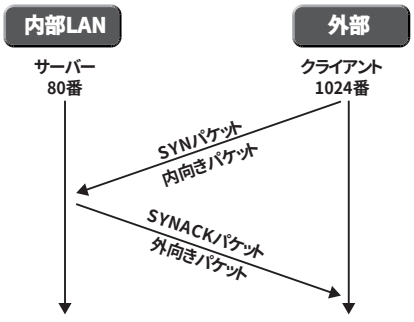

図2　内向き接続

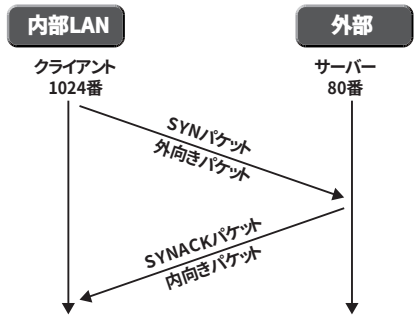

図3　外向き接続

内向きパケット

まとめると，インターネットから届いた内部LAN上のホストあてのパケット(以下，「内向きパケット」と呼びます)は，次のように判断できます。

●デュアルホーム・ホスト(NIC2枚)の場合

パケットを受け取ったインタフェースがインターネット側のインタフェースであれば，内向きのパケット

●通常のホスト(NIC1枚)の場合

パケットの送信元アドレスが，内部LANに属するアドレスでなければ，内向きのパケット

内向き接続と外向き接続

さて，インターネットから内部LAN上のホストあてのTCP接続(**図2**，以下，「内向き接続」と呼びます)を拒否したい場合，内向きパケットをすべて捨てればOKかというと，そうではありません。こ

のような設定だと内部LANから外部のサイトのサーバーに対するTCP接続（以下，「外向き接続」と呼びます）までできなくなってしまいます。これは，外向き接続における外部のサーバーからのACKパケットや応答データも，外部から入ってくる内向きパケットであるためです（**図3**参照）。この種の内向きパケットは通してやらないと，外部のサーバー（例えばWebサーバー）へアクセスできなくなってしまいます。

送信元ポートに基づくフィルタリング

多くのTCPのサービスにおいて，サーバーが返すパケットの送信元ポート番号は，サービスを受け付けるポート番号と同一です。例えば，Webサーバーであれば80番，telnetサーバーであれば23番，SMTPサーバーであれば25番，といった具合です。

よく使われるサービスのポート番号は1023番以下であり，かつクライアントが送信するパケットの送信元ポート番号は1024番以上ですから，送信元ポートが1023番以下の内向きパケットのみ通せば，外向き接続を許可し，内向き接続は禁止することができそうです[*7]。

実際，1023番以下のポートは，特権ポートとも呼ばれ，root権限がないとパケットを送信できないポートです。つまり送信元のマシンの管理者が正規にインストールしたサーバーのみが使用できるポートですから，送信元ポートが1023番以下のパケットは受け入れても問題なさそうに見えます[*7]。

しかし，これが成り立ったのはインターネットが普及する以前，今から10年以上も昔の話[*8]です。現在では，だれでも自分のマシンにLinuxなどをインストールすればrootになることができ，インターネットにつなぐのも簡単です。特権ポートと言ってもだれでも得られる特権では何の意味もありません。そもそもUNIX互換なOSでなければ（例えばWindows），特権ですらありません。したがって，送信元ポートを用いたフィルタリングを行ってはいけません。

送信元ポートがフィルタリングの条件として使えず，内部のホストの1024番以上のポートでサーバーが立ち上がってしまう可能性を考えると，内向き接続における内向きパケットと，外向き接続における内向きパケットを区別することはできません。

では，どうすれば良いのでしょうか？。

ACKフラグに基づくフィルタリング

TCPの場合，接続するには3-ウエイ・ハンドシェイクが完了することが不可欠であることを思い出してください。3-ウエイ・ハンドシェイクを構成する3つのパケットのうちどれか1つだけでも捨ててしまえば，接続不可能になります。内向き接続における全パケットを捨てる必要はありません。3つのパケットの中でも最初のSYNパケットを捨ててしまえば，最も効率が良いですし，SYNフラッディング攻撃を防ぐこともできます。

SYNパケットはTCPのパケットの中で唯一ACKフラグが立っていないパケットなので，ACKフラグの値を見るだけでSYNパケットか否かを判別できます。したがってフィルタリングの条件は次のような非常に単純なものになります。

ACKフラグが0で，かつ内向きパケットを捨てる

外向き接続の場合，SYNパケットは外向きパケットですからこの条件を満たさず，かつSYNパケット以外のすべてのパケットはACKフラグが1ですから，やはり条件を満たしません。だから外向き接続のためのパケットはいずれも破棄されません。一方，内向き接続の場合，SYNパケットは内向きパケットですから上記条件を満し，破棄されてしまいます。したがってSYNパケットは内部のサーバーに届くことはなく，3-ウエイ・ハンドシェイクが始まりさえしませんから，内向き接続は不可能です。

最近のダイヤルアップ・ルーターはデフォルトがこのような設定になっているものが多いようです（**図4**）。プロバイダへダイヤルアップ接続する人のほとんどすべてにとって，Webとメールの読み書きなど，外向き接続ができれば十分なので，当然と言えるでしょう。

ダイヤルアップ・ルーターではなく，内部LAN上のPCにTAあるいはモデムを接続してダイヤルアップ接続する場合（**図5**），そのPCがダイヤルアップ・ルーターの役割を果たすデュアルホーム・ホストになります。PPP接続の設定だけでなく，フィルタリングの設定にも気を配らなければなりません。

＊6　ユーザーが不用意に実行したトロイの木馬プログラムが，バックドアを提供するサーバー・プログラムだった，というのはよくある話です。
＊7　後述しているように，全くの嘘ですから信じないでください。
＊8　当時は，インターネットに接続されたマシンのrootともなれば絶対的な権力者であり，サイト全体のポスト・マスターともなれば，医者・弁護士に匹敵する社会的信用がありました（かなり大げさです）。

Linuxの場合，フィルタリングの設定を行うのは ipchainsコマンドの役割です。まずデュアルホーム・ホストにおける設定方法を説明します。

ipchains

Linuxカーネルではフィルタリングの条件は，ルールのチェーン(chain，連鎖)として表現します。つまり設定された複数のルールを順番に見ていって，パケットの内容がルールの条件部とマッチする最初のルールが適用されます。

Linuxカーネルには**図6**に示すような3本のチェーンがあります。

入力用チェーンは，ホストに入ってくるパケットに対して適用されるルールのチェーンです。入力用チェーンで廃棄されたパケットは，ホスト上のプログラムに届くことはありません。

出力用チェーンは，ホストから出ていくパケットに対して適用されるルールのチェーンです。ホスト上のプログラムが送信したパケットのうち特定の条件を満たすものを捨てることができます。

中継用チェーンは，ホストが中継するパケットに対して適用されるルールのチェーンです。ホストをルーターとして用いる場合，ホスト上のプログラムとは関係ないパケットをホストが中継することになりますが，この時特定の条件を満すパケットを捨てることができます。図6から明らかなように，あるパケットがこのホストで中継されるためには，中継用チェーンを通過するだけでなく，入力用チェーンと出力用チェーンの両方も通過しなければなりません。

さて，目標は前述したように「ACKフラグが0で，内向きのパケット」の廃棄です。このホストに届くパケットも，このホストで中継されて内部LANに届くパケットも，ACKフラグが0であれば破棄しなければなりません。このホストに届くパケットに対して適用されるチェーンは入力用チェーンだけで，入力用チェーンは中継されるパケットに対しても適用されますから，入力用チェーンを設定すればよいことになります。

入力用チェーンにおいて，ACKフラグが0で，内向きのパケットを廃棄するには，rootになって**図7**のように実行します。「-I input」は，入力用チェーンの先頭にルールを挿入する，という意味です。出力用チェーン，中継用チェーンにルールを設定する場合は，それぞれ「-I output」，「-I forward」を指定します。先頭に挿入するのではなく，末尾に追加する場合は，「-I」の代りに「-A」を指定します。「-j DENY」が，ルールの条件にマッチしたパケットに対して行う動作

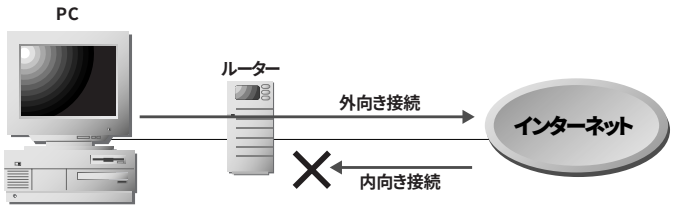

図4　ダイヤルアップ・ルーターによるフィルタリング

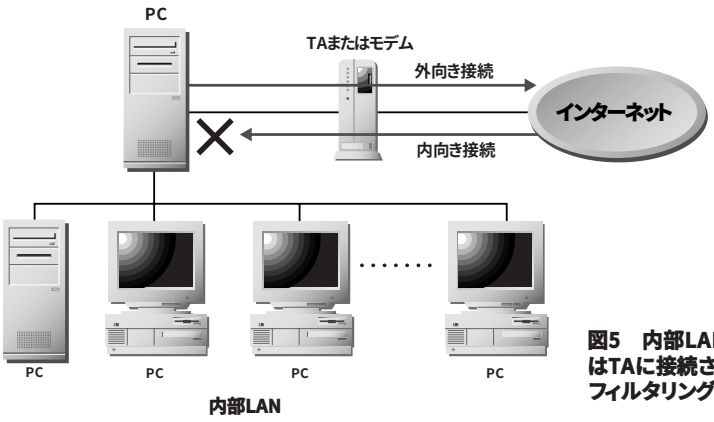

図5　内部LANとモデムあるいはTAに接続されたPCにおけるフィルタリング

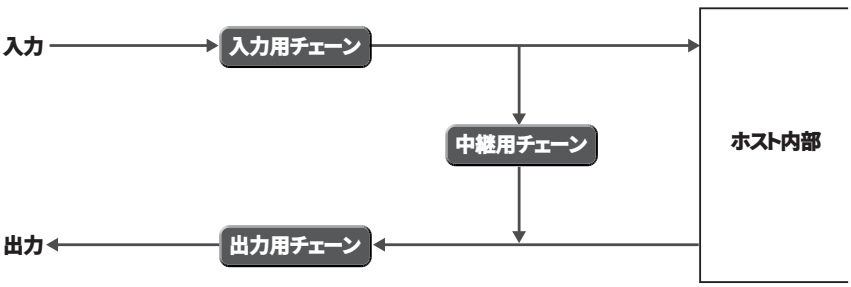

図6　フィルタリング・チェーン

```
# ipchains -I input -j DENY -i ppp0 -p tcp -y
```

図7　入力用チェーンでACKフラグが0の内向きパケットを廃棄する

で，「DENY」はパケットの通過を拒否する，つまりパケットを廃棄するという意味です＊9。通過を許可する場合は，「-j ACCEPT」を指定します。

残りのパラメータ「-i ppp0 -p tcp -y」がルールの条件部です。「-i ppp0」はパケットが入ってきたインタフェースがppp0である，という条件です。ppp0はダイヤルアップ接続で用いられるインタフェースですが，ppp0が使用中（あるいは使用不可）の場合，ppp1以降が用いられるので注意が必要です。

ダイヤルアップ接続が完了したとき，/etc/ppp/ip-up（通常はシェル・スクリプト）が呼ばれますが，この時1番目のパラメータにインタフェース名が渡される＊10ので，/etc/ppp/ip-up内でチェーンの設定を行うのが良いでしょう。

入力インタフェースがイーサネット・アダプタである場合は，eth0などのインタフェース名を指定することになります。「-p tcp」はパケットのプロトコルがTCPである，という条件です。「-y」はパケットがSYNパケット，すなわちACKフラグが0であるという条件です。

以上の条件がすべて成立したとき，「-j DENY」が実行されます。すなわち上記条件のすべてを満すパケットは廃棄されます。

チェーンの内容を確認するには，**図8**のように実行します。「-L input」は入力用チェーンのルールをすべて表示する，という意味です。「-v」は詳細表示を行うためのオプションで，これを指定しないと入力インタフェースの条件などが表示されません。

チェーンのルールを削除するには，「-I」の代りに「-D」を指定します。例えば，上記で入力用チェーンに設定したルールを削除するには，**図9**のように実行します。

チェーンのルールをすべて削除したい場合は**図10**のように実行します。/etc/rc.localなどのシェル・スクリプトにおいてipchainsコマンドを実行してチェーンを設定する場合は，最初に図10のコマンドを実行してチェーンを初期化するようにすると良いでしょう＊11。

パケットがチェーン内のどのルールともマッチしなかった場合のデフォルトの動作は，ACCEPTすなわちパケットの通過許可です。入力用チェーンのデフォルトをDENYにしたい場合は，「-P」オプションを指定して**図11**のように実行します。ただし，デフォルトをDENYにした場合は，パケットの通過を許可するルールを設定しない限り，TCP以外のパケット（例えばICMP）もすべて捨ててしまいます。TCPに限りデフォルトを廃棄にしたい場合は，**図12**のように実行して入力チェーンの末尾に「TCPパケットならば廃棄」というルールを追加すると良いでしょう。パケット通過のルールはチェーンの先頭から挿入するようにすれば，どのルールにもマッチしなかった

＊9　DENYの代りにREJECTを指定することもできます。DENYはパケットをただ捨てるだけですが，REJECTは捨てたことをICMPで送信元へ知らせます。
＊10　詳しくはpppdのマニュアルを参照してください。
＊11　初期化を行わずに挿入だけを行うシェル・スクリプトだと，そのシェル・スクリプトを実行するたびに同じルールが挿入されてしまい，無駄になります。
＊12　分かったからといっても，既にパケットは破棄してしまっているのだし，特に対策すべきことは何もないのですが，どんなポート・スキャンがはやっているかを把握することは何かの役に立つかも知れません。

```
# ipchains -L input -v
Chain input (policy ACCEPT: 1031475 packets, 160506770 bytes):
 pkts bytes target     prot opt    tosa tosx  ifname     mark     outsize  source   destination    ports
    0     0 DENY       tcp  -y---- 0xFF 0x00  ppp0                        anywhere      anywhere    any ->   any
```

図8　チェーンの内容を確認する

```
# ipchains -D input -j DENY -i ppp0 -p tcp -y
```

図9　入力用チェーンに設定したルールを削除する

```
# ipchains -F input
```

図10　チェーンのルールをすべて削除する

```
# ipchains -P input DENY
```

図11　入力用チェーンのデフォルトをDENYにする

```
# ipchains -A input -j DENY -p tcp -l
```

図12　TCPに限り廃棄をデフォルトにする

```
Sep 10 13:42:43 asao kernel: Packet log: input DENY eth0 PROTO=6 211.120.2.190:4944 210.145.125.162:143 L=60 S=0x00 I=17146 F=0x4000 T=48
SYN (#21)
```

図13　syslogにおけるパケットの記録

TCPパケットは最後のこのルールによって廃棄されます。

「-l」オプションは，このルールにマッチするパケットの記録をsyslogへ送ります。例えば私のマシンの場合，**図13**のように記録されます。これはホスト「211.120.2.190」の4944番ポートから，ホスト「210.145.125.162」（これは私のマシン）の143番ポート（IMAPサービス）あてのTCP（PROTO=6）パケットを「input」チェーンでDENYしたことを意味します。

記録しておくと，ポート・スキャンを受けたときすぐ分かります*12。ただし，すべてのポート・スキャンをいちいち監視していては時間がいくらあっても足りないので，ログ・ファイルから珍しいポートへのアクセスのログのみをピックアップして管理者へメールを送るように設定しておくと良いでしょう。

送信元アドレスに基づくフィルタリング

前述したように，デュアルホーム・ホストにおいては，送信元アドレスと到着インタフェースが矛盾するパケットを廃棄しなければなりません（図1参照）。そのためには入力用チェーンの先頭に

送信元アドレスが内部LANのアドレスである内向きパケットは捨てる

というルールを挿入します。例えば，内部LANのアドレスが210.145.125.160～210.145.125.175の範囲である場合，**図14**のように実行します。「-s 210.145.125.160/28」が，パケットの送信元アドレスに対する条件です。送信元アドレスの先頭28ビットの値が210.145.125.160であり，かつインターネット側のインタフェース（ppp0）に届いたパケットを廃棄します。

以上はデュアルホーム・ホストにおけるフィルタリングですが，NICが1枚だけのホストの場合は，到着インタフェースの条件（-i ppp0 など）の代わりに，送信元アドレスに基づく条件を用いればOKです。例えば，上記内部LAN上のホストにおいて，ACKフラグが0で，内向きのパケットを廃棄するには，**図15**のように実行します。「-s ! 210.145.125.160/28」は，送信元アドレスの先頭28ビットの値が210.145.125.160でない（「!」は否定），という条件です。

あるいは，**図16**のように3つのルールで実現することもできます。この3つのコマンドを順に実行すると，入力用チェーンは図17のようになります（「-A」はチェーンの末尾に追加し，「-I」は先頭に追加するため，順序が逆になる点に注意してください）。

つまり，SYNパケットでなければ最初のルール（! -y）が適用されてACCEPT，送信元アドレスが内部のものであれば2番目のルール（-s 210.145.125.160/28）が適用されてACCEPT，どちらの条件も成立しなければ最後のルールが適用されてDENYになります。

どちらのフィルタリング設定方法でも構わないのですが，後者の設定法のように，まずチェーンの最後尾にデフォルトでDENYするルールを設定し，その前にACCEPTするルールを必要に応じて追加しておく方が，フィルタリング漏れが防げるので好ましいと思います。いずれの場合も，ローカル・インタフェース（lo）に届いたパケット（つまり自ホストが自身に対して送信したパケット）は無条件に受け入れる設定にしておく必要があります。**図18**のように実行します。こ

```
# ipchains -I input -j DENY -i ppp0 -s 210.145.125.160/28
```

図14　送信元アドレスが内部LANのアドレスである内向きパケットは捨てる

```
# ipchains -I input -j DENY -s ! 210.145.125.160/28 -p tcp -y
```

図15　ACKフラグが0で内向きのパケットを廃棄する

```
# ipchains -A input -j DENY -p tcp -l
# ipchains -I input -j ACCEPT -s 210.145.125.160/28
# ipchains -I input -j ACCEPT -p tcp ! -y
```

図16　図15の条件を3つのルールで実現する

```
# ipchains -L input
Chain input (policy ACCEPT):
target     prot opt     source               destination          ports
ACCEPT     tcp  !y----  anywhere             anywhere             any ->   any
ACCEPT     all  ------  210.145.125.160/28   anywhere             n/a
DENY       tcp  ----l-  anywhere             anywhere             any ->   any
```

図17　図16を実行後の入力用チェーン

```
# ipchains -I input -j ACCEPT -i lo
```

図18　自ホストが自身に送信したパケットは無条件に受け入れる

のルールは，チェーン内において，他のDENYルールよりも前にくる必要があります。

あて先ポートに基づくフィルタリング

以上で，内向き接続を禁止するフィルタリング設定が完了しました。デュアルホーム・ホストあるいは内部LAN上のホストで対外サービスを行うサーバーを立ち上げる場合は，サービスごとにポートに対する内向き接続のみを許可するルールを挿入していけばOKです。

例えば，SMTPサーバー，ネーム・サーバー，Webサーバーを外部へ公開する場合は，**図19**を実行します。「--destination-port」は，あて先ポートの条件を指定するオプションです。

telnetサーバー，POPサーバー，NNTPサーバーなど，よく使われるプロトコルのほとんどはポート番号が違うだけで同様に設定できます。ところが一筋縄で行かないのがFTPサーバーです。

FTPサービス

通常のTCPサービスが，クライアントからサーバーへのTCP接続だけで済むのに対し，FTPは，もう1つ，サーバーからクライアントへのTCP接続を必要とします。**図20**のように，FTPクライアントはサーバーにTCP接続した後，接続を受け付けるためにポートを開き，そのポート番号をサーバーに伝えます。するとサーバーはそのポートに対してTCP接続します。前者の接続はFTPのコマンドをサーバーに伝えるために使われ，後者の接続はデータの転送に使われます。

サーバーからクライアントへ接続する，しかもクライアントが開けるポートはその都度変わる，このプロトコルはファイアウオール管理者にとって頭痛の種と言えるでしょう。サーバーからクライアントへ接続するのはいくらなんでも論外だ，ということで，第2の接続の方向をクライアントからサーバーへ変更したパッシブ・モードが考案されました。

パッシブ・モードでは，**図21**のようにクライアントがPASVコマンドをサーバーへ送ると，サーバーは第2の接続のポート番号をクライアントへ返します。こ

```
# ipchains -I input -j ACCEPT -p tcp --destination-port smtp
# ipchains -I input -j ACCEPT -p tcp --destination-port domain
# ipchains -I input -j ACCEPT -p tcp --destination-port http
```

図19　SMTPサーバーとネーム・サーバーとWebサーバーを外部へ公開する

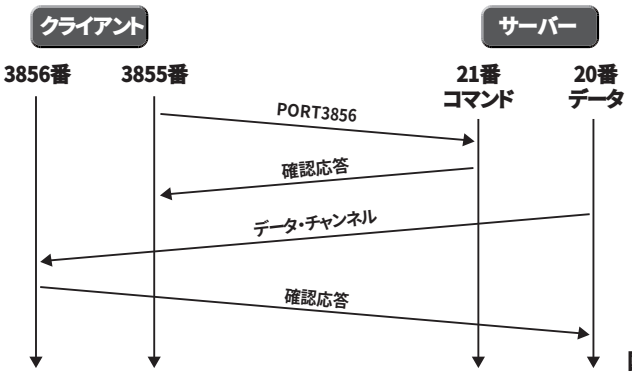

図20　FTPでの接続

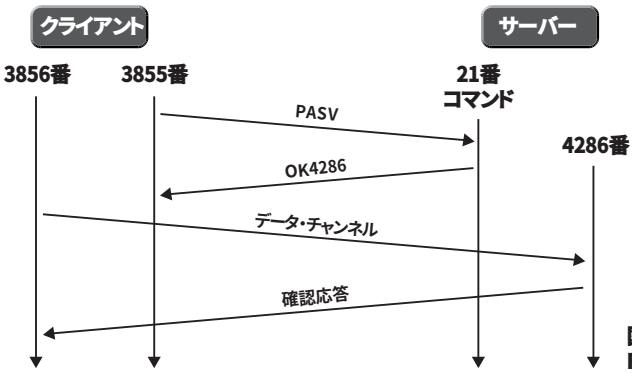

図21　パッシブ・モードのFTP

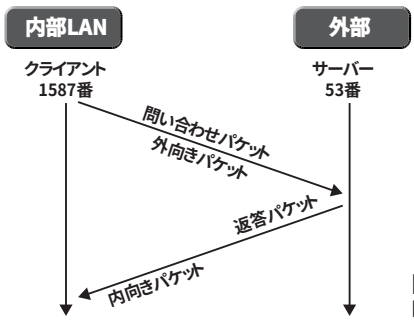

図22　外向きUDP問い合わせ
問い合わせ時の送信元ポートへ返答パケットが届く。

れを受けてクライアントは教えられたポートに対して接続します。

パッシブ・モードだと，クライアント側のファイアウオールは外向き接続のみを通せば良いので，FTPに関して特別扱いをせずに済みます。現在主流のWebブラウザの多くがFTPクライアントの機能を持っていますが，FTPクライアントとして動作するときはパッシブ・モードを選択するようです。

パッシブ・モードによりクライアント側の問題は解決されたのですが，問題がサーバー側に押し付けられただけ，と見ることもできます。つまりサーバー側から見ると，FTPサーバーが適当に選んだポートに対する内向き接続を受け付けなければなりません。このポートは毎回異なるので，それに追随して内向き接続を許可するポートを変更できるファイアウオールが必要になってしまいます。

仕組みが複雑になればそれだけセキュリティ・ホールの危険が高くなるわけで，パケット・フィルタリングだけのファイアウオールが一番単純で安全なのですが，FTPサーバーを立ち上げようとする限り，ファイアウオールが複雑化してしまいます。したがって，それだけのコストをかけ，危険をおかしてまで FTPサーバーを立ち上げる必要があるのか，よく考える必要があります。Web登場以前はファイル転送になくてはならないFTPでしたが，不特定多数のユーザーがサーバーからファイルをダウンロードするために用いられることがほとんどであり，この用途であればWebの方がよほど便利です。

特定ユーザー間でのファイル転送であれば，メールに添付して送る＊13，scpコマンドなどで送る，プロバイダなどのホームページ・サービスを利用して，相手にダウンロードしてもらう，などの方法を使うことができます。

FTPに限らず，セキュリティは常に，利便性と，危険性およびコストとのトレードオフの関係にあります。高機能ファイアウオールのコストと危険性＊14を考えれば，大抵の利便性＊15は犠牲にすべきであると思います。

UDPパケットのフィルタリング

UDPの場合，TCPと異なりACKフラグなどはありませんから，内向きパケットが，外部から内部のサーバーへのアクセスなのか，あるいは内部から外部のサーバーへの問い合わせに対する返答なのかを区別することはできません。

返答パケットの場合，あて先ポートが問い合わせに利用した送信元ポートと同じになります（**図22**参照）。したがって，UDPクライアントを利用する場合，そのクライアントが送信元ポートを固定できるか否かが重要です。

固定できるならば，返答パケットのあて先ポートも常に同じ番号になりますから，あて先ポートに基づくフィルタリングを行って，そのポートあてのUDP内向きパケットを許可すれば済みます。ただし，このクライアントは，他のプログラムにそのポートを横取りされないようにするため，常にそのポートを開いている必要があります。つまりこのクライアント・プログラムはデーモンでなければなりません。

ポートが固定できないUDPクライアントが動いている場合は少々やっかいです。そのクライアントを動かすことが絶対必要ならば，そのクライアントが送信元ポートとして利用する可能性があるすべてのポートについて，それらのポートあての内向きパケットを許可してしまう＊16か，あるいはクライアントに追随して内向きパケットを許可するポートを変更できるファイアウオールが必要になってしまいます。

```
# ipchains -I output -j ACCEPT -p udp -l -s 210.145.125.160/28 1024:65535 -d ! 210.145.125.160/28 0:1023
```

図23　UDPクライアントが動いているかどうかを確認する

```
Sep 10 22:53:46 asao kernel: Packet log: output ACCEPT eth0 PROTO=17 210.145.125.162:1587 192.5.5.241:53 L=63 S=0x00
I=5114 F=0x0000 T=64
(#1)
```

図24　syslogの記録

```
  query-source address 210.145.125.162 port 53;
```

図25　問い合わせ時の送信元アドレス（210.145.125.162）と送信元ポート（53番）を設定ファイルのoptionsで指定する

まずはそのようなクライアントが動いていないか確認しましょう。**図23**のようにipchainsコマンドを実行してください。

「210.145.125.160/28」は，内部LANのIPアドレスの範囲で置き換えてください。内部LAN上のホストのポート1024～65535番から，外部のサーバーのポート0～1023番へUDPパケットが送信されると，syslogに記録されます（「-l」オプション）。送信元ポートが固定されていないクライアントが動いていれば，そのクライアントから送信されたパケットが記録されるはずです。

例えば**図24**のようなものです。これを見ると，内部LAN上のマシン「210.145.125.162」のポート1587番からネーム・サーバー「192.5.5.241」に対してUDPによる問い合わせが行われています。幸い，ネーム・サーバーの場合は問い合わせ時の送信ポートを53番に固定することができます。ネーム・サーバー以外のサーバーへの問い合わせがある場合は，その問い合わせが本当に必要か，あるいはネーム・サーバーへの問い合わせのように送信元ポートが固定できないか，考えてみてください。

問い合わせポートを固定

ネーム・サーバーへ問い合わせを行うクライアントが通常利用する送信元ポートは1024番以上のポートから適当に選ばれます。ただし1つだけ例外があり，ネーム・サーバーが他のネーム・サーバーへ問い合わせを行うときは，あらかじめ決められたポートを送信元ポートとして利用します。bind-8.2.2-P5の場合，設定ファイルのoptionsにおいて，**図25**などのように書くことにより，問い合わせ時の送信元アドレス（210.145.125.162）と送信元ポート（53番）を指定できます＊17。

したがって，上記のように設定したネーム・サーバーを内部LAN上に設置し，ネーム・サーバーへの問い合わせはすべてこのネーム・サーバーを経由して行うようにすれば，外部のネーム・サーバーへの問い合わせパケットの送信元ポートをあらかじめ決められたポート（53番）に固定することができます。

あて先ポートに基づくフィルタリング

内部LAN上で動くUDPクライアントの送信元ポートが固定できれば，返答パケットのあて先ポートも固定できて，内向きパケットのあて先ポートに基づくフィルタリングが可能になります。例えば，**図26**のように実行すると良いでしょう。

図26の4つのコマンドを順に実行すると，入力用チェーンは**図27**のようになります。あて先ポートが「domain」か「ntp」であれば，それぞれ1番目（--destination-port domain），2番目のルール（--destination-port ntp）が適用されてACCEPT，送信元アドレスが内部のものであれば3番目のルール（-s 210.145.125.160/28）が適用されてACCEPT，いずれの条件も成立しなければ最後のルールが適用されてDENYになります。

＊13　30Kバイト程度のメールが巨大メールと見なされた大昔ならいざ知らず，現在のメールは直接相手サイトへ届けられるわけで，相手サイトのメール・サーバーが受け入れ可能なら，数Mバイトのメールを送ることも十分許されるでしょう。
＊14　高価なファイアウォールを導入したから安心と思うのが最も危険です。
＊15　そのほとんどは，客観的に見れば，セキュリティに無頓着なユーザーのわがままに過ぎないことが多いようです。
＊16　アクセスしたい外部のUDPサーバーが限定できるならば，内向きパケットの送信元アドレスがそれらのサーバーであるときに限り許可するようにすれば，少しマシになります。
＊17　query-sourceを指定しない場合，送信元ポートは1024番以上のポートからランダムに選ばれます。

```
# ipchains -A input -j DENY -p udp -l
# ipchains -I input -j ACCEPT -s 210.145.125.160/28
# ipchains -I input -j ACCEPT -p udp --destination-port domain
# ipchains -I input -j ACCEPT -p udp --destination-port ntp
```

図26　内部LAN上で動くUDPクライアントの送信元ポートが固定することによりフィルタリングが可能になる

```
# ipchains -L input
Chain input (policy ACCEPT):
target     prot opt     source               destination          ports
ACCEPT     udp  ------  anywhere             anywhere             any ->   ntp
ACCEPT     udp  ------  anywhere             anywhere             any ->   domain
ACCEPT     all  ------  210.145.125.160/28   anywhere             n/a
DENY       udp  ----l-  anywhere             anywhere             any ->   any
```

図27　図26の入力用チェーン