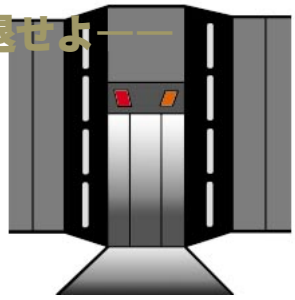
本記事でとり上げたソフトの一部が付録CD-ROMに収録してあります。

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つのとりででクラッカーを撃退せよ――



サーバーが侵入され他のサイトへの攻撃の踏み台として悪用されれば，賠償責任が生じるかもしれない。たとえ守る価値のないサーバーでもインターネットにつなぐ以上，セキュリティを高める努力をしなければならないのである。

4つのとりでを構築せよ

セキュリティの基本は，侵入される可能性を少しでも排除し，万一侵入されたとしても被害を最小限にとどめることである。今まで侵入されたことがないからこのままでよいはずだと考えていると，ある日侵入に気付いて慌てることになる。

侵入の可能性を減らすには，防御の仕掛けを何重にも構築することが重要である。単一の防御では，その部分にバグなどの原因でセキュリティ・ホールがあれば，おしまいだからである。

ここでは，防御の仕掛けを４段階に分けて説明する。それぞれ**「最初のとりで（砦）」「第２のとりで」「第３のとりで」「最後のとりで」**と呼ぶことにする。このうち第２のとりでがサーバー・プログラム自身による防御である。

もしサーバー・プログラムにバグが一切ないのであれば，他のとりでは本来不要であるが，バグがない枯れたプログラムだけでは必要なサービスが提供できないし，もちろん枯れたと思われているプログラムでもバグがないことを保証することは不可能である。他のとりでではサーバー・プログラムの防御を補う役割を果たすことになる。

では，最初のとりでから解説しよう。

最初のとりで

パケット・フィルタリングで守れ

- ルーターのフィルタリング機能で守れ
- カーネルのフィルタリング機能で守れ
  ipfwadm か ipchains を使え
- フィルタリング用ツールで守れ
  tcpwrapper + inetd よりも
  tcpserver を使え
  sendmailからqmailへの移行も勧める

Web など一部のサービスを除けば，不特定多数からのアクセスを許可する必要があるサービスはほとんどない。例えば不特定多数からの telnet login を許可する必要はまずないだろう。ほとんどの場合，特定少数がアクセスしたいだけであり，したがって全世界からのパケットを受け付ける必要はまったくない。telnet の場合であれば login を許可している人達が普段利用しているプロバイダからのアクセスだけを許可すれば十分である。

そこで最初のとりでの役割は，ソースIPアドレス（アクセス元，telnet の場合であれば telnet クライアント・プログラムを実行しているマシンのIPアドレス）やポート番号があらかじめ登録した範囲のものでないパケットを廃棄することである。登録していない範囲のIPアドレスからのアクセスは，サーバー・プログラムへ伝わらずに排除される。

これだけの対策で，世界中にいるクラッカー予備軍からの攻撃のほとんどを未然に防ぐことができるのであるから，多少の不便さ（例えば，ふらり立ち寄ったインターネット・カフェから telnet できないなど）は我慢すべきだろう。ただし，ソースIPアドレスは偽装可能だから，このとりでは万全ではない。どちらかと言えば，レベルの低いクラッカーからの煩わしい攻撃を無視するためのとりでと言える。

ではあまり意味がないとりでかというとそうではない。ほとんどすべての攻撃は幼稚なものであるから，この最初のとりでだけで撃退できるのである。いわば雑魚を無視するためのとりでである。このとりでを破ったクラッカーは，それなりに高度な技術力を持っている可能性があるから，それなりの敬意を持ってじっくり監視すべきである。

最初のとりでを構成する方法は次の３つに分類できる。

(1) ルーターのフィルタリング機能で守れ
(2) カーネルのフィルタリング機能で守れ
(3) フィルタリング用ツールで守れ

(1)ルーターのフィルタリング機能で守れ

ほとんどすべてのルーターにはIPフィルタリングの機能が搭載されている。したがって，不要なパケットをすべてルーターで遮断してしまえば LAN 内へ入ってくることさえないので，LAN に接続したマシンにそのパケットが届くことはない。最も安全確実な方法と言えるが，安価なルーターの場合機能が最小限であることも多いので注意が必要である。ルーターによっては，フィルタリング条件の柔軟性に欠けていたり，設定変更が面倒だったり，フィルタリングしたパケットのログを残すことができなかったりする。

高機能高価なルーターを使っている場合は，ルーターだけで最初のとりでを構成しても良いのだが，そうでなければ次に説明するカーネルのIPフィルタリング機能を併用する方が良いだろう。そしてルーターでのフィルタリングは基本的なものだけに限定する。

例えば，LAN 内のIPアドレスをソースIPアドレスとするパケットが LAN の外から来るはずはない。もし来たらソースIPアドレスを偽装したパケットである可能性が高い。LAN に接続されたマシンは，同じ LAN から来た（ように見える）パケットは信頼するだろうから，こんなパケットを LAN 内に入れては大変である。したがってルーターのIPフィルタリング機能を使ってこのようなパケットを排除すべきである（図1）。

蛇足だが，ルーターのパスワードは忘れずに設定してほしい。せっかくフィルタリングの設定をしていても，パスワードを設定していなければ，攻撃者によって真っ先にフィルタリングの設定を解除されてしまうだろう。

多くのルーターの設定の際の認証方法は，再生攻撃可能（「第２のとりで」参照）で信用できないので，ルーター自体へあてたインターネットからのパケットはフィルタリングすべきである。当然，外部から直接ルーターへアクセスして設定を変更することは出来なくなるが，一度 LAN 内の（信頼できる）マシンへ login した後そこからルーターへアクセスすれば変更できるので問題ない。

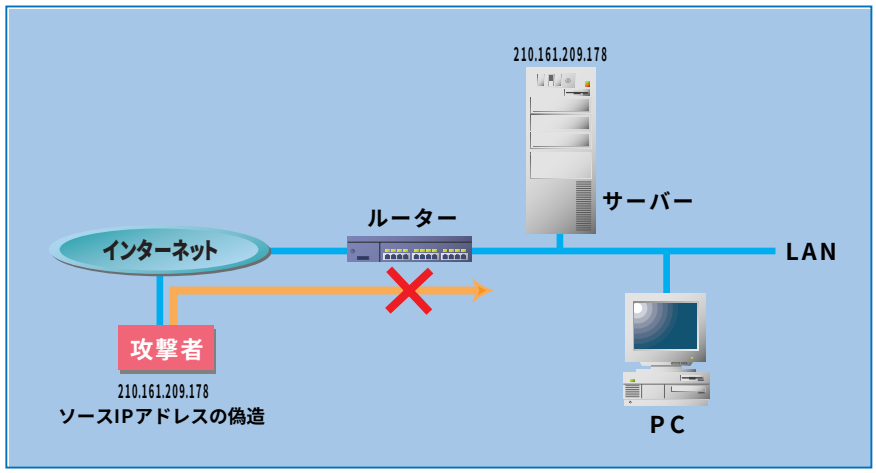


図1　ルーターのIPフィルタリング機能で偽装されたIPアドレスを持つパケットを排除

(2)カーネルのフィルタリング機能で守れ

カーネルにIPフィルタリング機能が実装されている場合は，この機能を使って最初のとりでを構成できる。もちろん，Linux カーネルにはIPフィルタリング機能がある。安物ルーターよりは柔軟な設定ができるし，Linux のように世界中の人によって使い込まれているカーネルならばバグが少ないことが期待できるのでお勧めである。

Linux 2.0.37 など，2.1.102 より前のバージョンでは **ipfwadm**＊（**http://www.xos.nl/linux/ipfwadm/ 参照**），Linux 2.2.12 など 2.1.102 以降のバージョンでは **ipchains**＊（**http://www.rustcorp.com/linux/ipchains/参照**）を使って，フィルタリングの設定を行なう。

NIC（Network Interface Card。 例えば Ethernet アダプタ等）を２枚以上装備しているマシンでは，インタフェースごとに送受信および転送するパケットの条件を指定することができる。

(3) フィルタリング用ツールで守れ

DoS 攻撃に弱い inetd + tcpwrapper

フィルタリング用ツールとして有名なものに tcpwrapper がある。inetd と組み合わせて使ったり，あるいはライブラリとしてサーバー・プログラムにリンクして使う。アプリケーション・レベルのフィルタリングである

＊「ipfwadm」と「ipchain」は付録のCD-ROMに収録しました。

から柔軟性はかなり高いが，反面 DoS（Denial of Service）攻撃に弱く，ルーターやカーネル・レベルのフィルタリングに比べれば，バグが残っている危険性も高い。

パケットが未知のマシンから届いたら任意のコマンドを実行する機能があるので，例えば finger コマンドで送信元の素性を探る等の芸当ができるが，この機能がどれだけ役に立つかは疑問である。実際，finger コマンドで攻撃者に関する有益な情報が得られるようでは，極めて幼稚な攻撃者と言えるだろう。

したがってこのような設定は意味がない。むしろ第３者のマシンを攻撃する道具として悪用されるのがオチだからやめたほうがましである。結局，柔軟性が高いことによるメリットは何もないので，tcpwrapper はお勧めではない。可能な限り(2)のカーネル・レベルのフィルタリングか，次に説明する**ucspi-tcp** を利用すべきだろう。

ucspi-tcp に含まれる tcpserver を使え

ucspi は UNIX Client-Server Program Interface の略であり，「うくすぱい」と発音する。ucspi-tcp（**http://pobox.com/~djb/ucspi-tcp.html 参照**）は UNIX 上で TCP クライアント/サーバー・プログラムを構成するためのツール群である。そしてこれに含まれる**tcpserver**は，inetd＋tcpwrapperの代わりに使うツールである。inetd＋tcpwrapper が持つ無用な機能を廃し，徹底してプログラムの構造をシンプルにすることによってセキュリティ・ホールが生じる可能性を防いでいる。inetd がクラッカーが存在しなかった牧歌的時代にデザインされたのに対し，tcpserver は DoS 攻撃を受けることを前提に設計されている。したがって，特に理由がない限り inetd は走らせるべきではなく，代わりに tcpserver を使うべきである。

```
/bin/tcpserver -x/etc/tcprules.dat -u0 -g2107 -pS 0 pop \
    /bin/qmail-popup toyokawa.gcd.org \
    /bin/checkpassword \
    /bin/qmail-pop3d Maildir &
```

（注：行末の\は継続行を意味する）

図2　tcpserverを使ってPOPサービスを行う例

tcpserver はソースIPアドレスの許可/不許可の設定にハッシュ・テーブルを用いるため設定するアドレスの数が大量にあっても性能劣化の問題はない。サイトごとにサービス提供の許可不許可をきめ細かく設定したいときなどに適している。

tcpserver を使って POP サービスを行う例を図2に示す。

/etc/tcprules.dat は，ソースIPアドレスによってアクセスの許可/不許可を指定するハッシュ・ファイルである。tcpserver はポート110番（POP）にアクセスがあると /bin/qmail-popup を呼び出す。qmail-popup は POP サービスのうち PASS コマンドでパスワードを受け取るところまでを担当し，次の /bin/checkpassword を呼び出す。checkpassword はその名の通り qmail-popup から引き継いだパスワードを検証するプログラムである。パスワードが正しければ，処理が qmail-pop3d に引き継がれる。メールの一覧や転送はこの qmail-pop3d が担当する。

このように複数のプログラムを組み合わせて POP サーバーの機能を実現するため，一つひとつのプログラムの機能は極めて単純になっている。このためセキュリティ・ホールが生じる可能性を最小限に抑えられる。また，サーバーの機能のうち危険な部分のみ犠牲パーティション（第３のとりでで説明する）で走らせる，という設定も可能になる。

ちなみに，上記 qmail-pop3d は **qmail（http://www.jp.qmail.org/参照）**用なので，MTA として sendmail 等を使っている場合は適用することができない。qmail が前述した POP サーバー・プログラムと同様，単機能の小さいプログラムを組み合わせてメール・サーバーの機能を実現するのに対し，sendmail は一つの大きく複雑なサーバー・プログラムを用いる。したがって，sendmail はセキュリティ的にはあまりお勧めではない。inetd から ucspi-tcp へ移行する機会に sendmail をやめて qmail へ乗り換えてはどうだろうか。

第２のとりで

サーバー・プログラムによる認証で守れ

再生攻撃

再生攻撃が出来る認証は論外である。再生攻撃とは，正規のユーザーがサーバ

ーへ送信したデータを記録しておいて，ちょうどテープ・レコーダを再生するがごとく同じデータをサーバーへ送り付けることにより正規ユーザーになりすます攻撃である。

例えば telnet プロトコルの場合，ユーザーが login 時に送るデータはユーザーIDとパスワードの組であり，毎回同じであるから，正規ユーザーが login するときに打ち込んだ文字列をそのまま「再生」すれば login できてしまう。

読者の中にはどこで盗聴されるんだ，盗聴なんて普通は有り得ないんじゃないか，という人もいるかも知れない。たしかに現在のインターネットは1次プロバイダの回線をバックボーンとして構成されているから，1次プロバイダ内で盗聴されるのでない限り盗聴される危険は現実にはかなり低い。

だから十分信頼できる技術力に定評のあるプロバイダだけを常に利用するのであれば，盗聴を心配する必要はあまりない。しかし考えてみてほしい。常に信頼のできる端末だけを使っていると断言できるだろうか。例えば私は出張先で端末を借りてインターネット経由で自分のサーバー・マシンへアクセスすることがある。出張先のLAN でパケットを監視している人がいないとは断言できない。特に大学の LAN などでは，学生が軽いイタズラ心で telnet プロトコルを記録しパスワード等を抽出していることも有り得る。内部に悪い人がいなくても，LAN 内のいずれかのマシンが侵入されていて盗聴のためのソフトウエアを仕掛けられている場合もある。

同じパスワードを複数のマシンで使っていると，危険はさらに高まる。例えばファイアウオールで守られた LAN に接続したマシンではセキュリティの心配がないのでパスワードを平文のまま流すことが多いが，同僚の中に悪い奴がいてパスワードを盗聴していたら，同じパスワードを使っているすべてのマシンへ侵入されてしまう。

つまり再生攻撃が可能な認証を利用する限り，認証データが安全なネットワークのみを流れているか常に細心の注意を払わなければならず，そんな注意を払うくらいだったら，再生攻撃が不可能な認証を使う方がよっぽど楽である。

再生攻撃が不可能な認証には，OTP，APOP などハッシュ関数を利用した認証や，SSL，ssh など暗号を利用した認証がある。いずれもクライアント・マシンからサーバ・マシンへ認証のために送られるデータは毎回異なるから，攻撃者が盗聴して同じデータをサーバーマシンへ送ったとしても侵入されることはない。

OTP（One Time Password，使い捨てパスワード）

telnet や ftp を使う場合なら，OTP（One Time Password）を使えば良い。これは OPIE（One-time Passwords In Everything）パッケージに含まれる。OPIE をインストールしたマシンへ telnet でlogin したときの例を図3に示す。

「otp-md5 470 as5266 ext」はチャレンジと呼ばれ，login ごとに異なる。チャレンジと，秘密のパスフレーズ（パスワードに相当）を OTP 計算機に入力すると，1回限りのパスワード（レスポンス）「WOK MOP GAY HAM CUP VAN」が出力される。これを Response: プロンプトが出たところで入力すると login できる。

パスフレーズは毎回同じものを入力するが，OTP の計算が端末の中で閉じていてパスフレーズが外部に漏れる心配がないようにしておくのがミソである。間違ってもリモートのマシン上で走る OTP 計算機を使ったり，X端末など盗聴される危険がある端末上でパスフレーズを入力したりしてはいけない。

一番確実なのはネットワークに物理的につながっているマシンにはパスフレーズを打ち込まないことである。すなわち，ネットワークに接続されていないマシン上で OTP 計算機を走らせ，端末に表示されたチャレンジを OTP 計算機へ「手で」打ち込み，計算されたレスポンスを「手で」端末へ入力する。しかしこれはかなり面倒であるので，次善の策として端末上で OTP 計算機を走らせ，チャレンジ＆レスポンスの入力を自動化することになる。

この目的にかなう OTP 計算機はいろいろなものが公開されているが，私は Windows 98/95/NT上で動作する **dotkey95（http://ftp.vector.co.jp/soft/win95/util/se049268.html 参照）** を愛用している **（図3）**。

dotkey95 をタスクトレイへ入れた状態で端末エミュレータ上に表示されたチャレンジの部分をクリップボードへ入れると，自動的にパスフレーズを入力するためのウィンドウがポップアップし**（図3左）**，パスフレーズを入力するとレスポンスを計算して自動的に送信してくれる **（図3右）**。つまり普通の telnet で login するのとほとんど同程度の手間で OTP を使うことができる。

他人の端末を借りて telnet する場合は，いちいち dotkey95 等のソフトウエアをインストールするわけにはいかない。こういうとき便利なのが PDA 上で動く

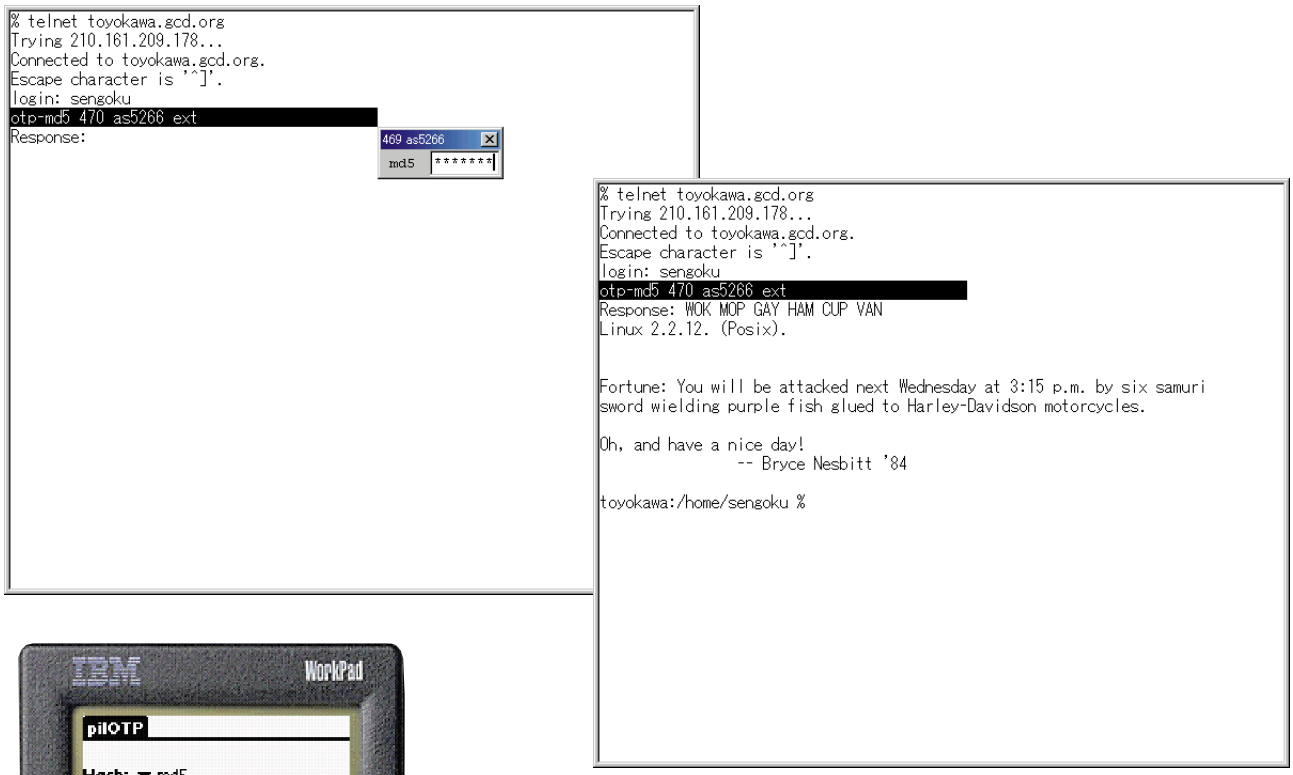


図3　OTPの例とOTP計算機「dotkey95」

図4　文字が化けるのはご愛敬。日本語に対応していないだけで動作に支障はない。

OTP 計算機である。私は常に IBM社製 WorkPad 30J を持ち歩いているので，これに OTP 計算機ソフト **pilOTP**（**http://astro.uchicago.edu/home/web/valdes/pilot/pilOTP/ 参照**）をインストールしている（**図4**）。ネットワークにつながっていない PDA であるから安心してパスフレーズを打ち込むことができる。

さて，OTP を使えば再生攻撃には対抗できるが，telnet セッションの内容はすべて平文でインターネット上を流れるので盗聴される危険がある。だから侵入者の役に立つようなデータを流すことはご法度である。特にユーザー・アカウントの新規作成など管理作業はしてはいけない。

ssh（Secured Shell）による暗号化

telnet の代わりに ssh を使ってセッション全体を暗号化すれば，盗聴を防ぐことができるし，使い勝手の点で telnet の方が勝るということはないのだから，常日頃から（ LAN 内の通信でも!）ssh を使うようにすべきである。telnet + OTP は ssh が何らかの原因で使えないときの非常用の login 手段と思ったほうが良い。例えば他人の端末を借りるときは，たとえその端末に ssh がインストールしてあったとしても自分の秘密かぎをインストールするのは面倒だし危険が伴うので，telnet + OTP の方が適しているといえる。

ssh で使われている暗号化方法は，通常の目的には十分過ぎる強度を持ち，通信路上での盗聴を心配する必要はないが，注意すべきは接続元のマシンのセキュリティである。ssh は通信に先立ち，双方のマシンが持っている秘密かぎを使ってお互いの認証を行なう。秘密かぎ自体は通信路上を流れないので，通信経路上の第３者が秘密かぎを盗むことは不可能であるが，もし攻撃者がどちらかのマシンへの

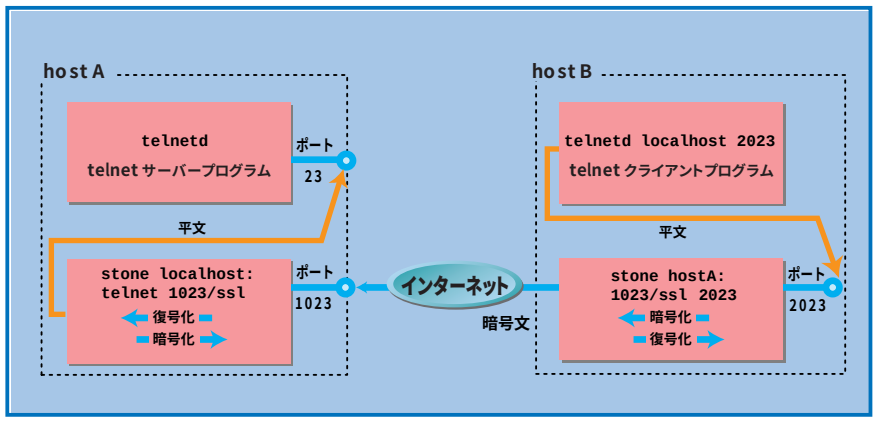


図5　stoneで暗号化する
（注）図のstoneコマンドは実際には1行です。
`stone localhost:telnet 1023/ssl`
`stone hostA:1023/ssl 2023`

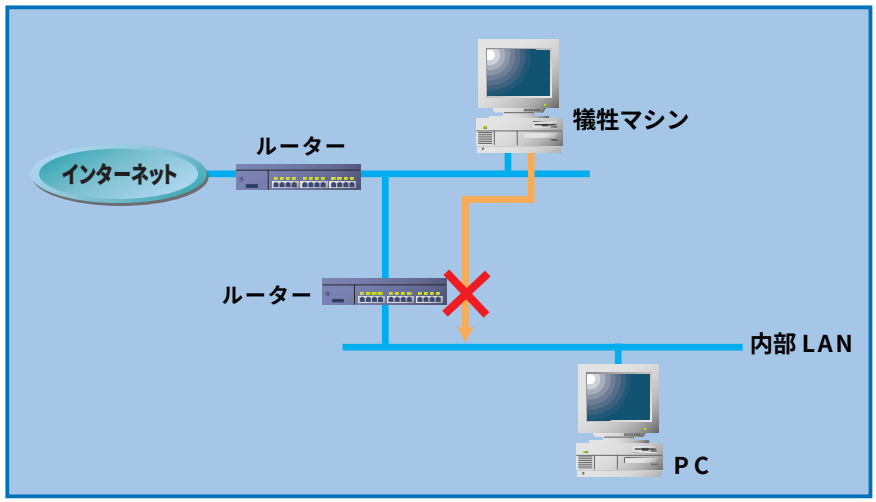


図6　犠牲マシンの設置

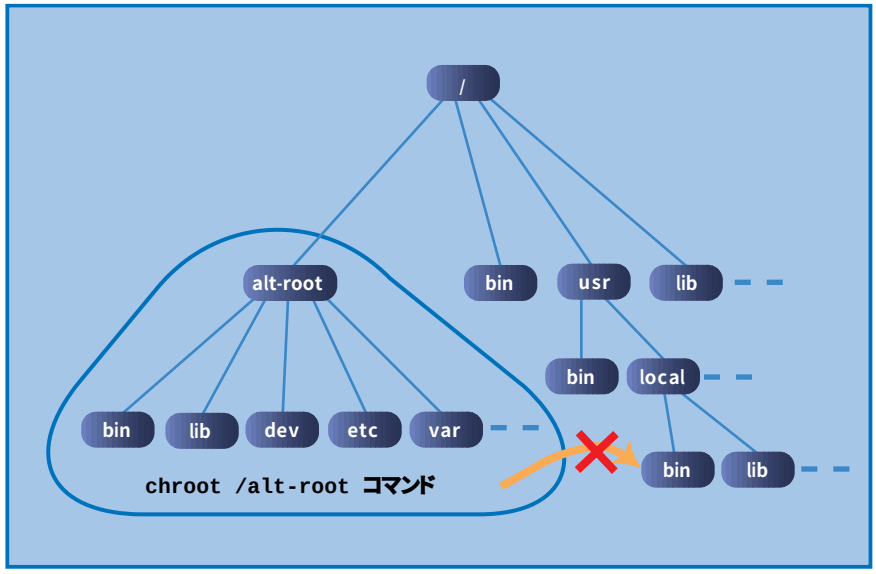


図7　犠牲パーティションのディレクトリ階層

侵入に成功したらどうなるだろう。そのマシンが持つ秘密かぎを盗んで，そのマシンになりすまし，相手マシンへ侵入することまで可能になってしまう。つまりsshでお互いを信頼し合うマシン群は一連托生（いちれんたくしょう）ということである。

もちろん，sshには秘密かぎをパスワードで保護して，たとえマシンが侵入されても秘密かぎが盗まれないようにする仕掛けがあるが，ユーザーによるパスワードの管理がずさんであれば役に立たない。sshを使ったloginが許可されているユーザーに対し，秘密かぎの管理を慎重に行なうよう徹底すべきである。特に，インターネットに常につながっているマシンに秘密かぎを保存する場合は，そのマシンのセキュリティが十分なレベルにあるか確認すべきであり，もし侵入される恐れがあるならば，そのマシンからのloginを禁止すべきである。

手軽な暗号化の方法「stone」

Webの暗号化方式として有名なSSL（Secure Socket Layer）であるが，拙作「**stone**」＊（**http://www.gcd.org/sengoku/stone/Welcome.ja.html 参照**）を使うと，任意のプロトコルをSSLで暗号化することができる（**図5**）。

再生攻撃可能なプロトコルも，SSLで暗号化すれば，流れるデータは毎回異なったものになり，セキュリティ・レベルを向上することができる。

＊「stone」は付録のCD-ROMに収録しました。

第3のとりで

被害の限定

- 犠牲マシンを設置せよ
- chroot で犠牲パーティションを作れ

犠牲マシン

マシンがたくさん設置できる場合は，侵入されても構わない犠牲マシンを置き，対外サービスは主に犠牲マシン上で提供するようにする。犠牲マシンは踏み台として利用されないようにするため，犠牲マシンから他のマシンへ絶対に侵入出来ないようにしておく。間にルーターを置いて犠牲マシンからのパケットをすべて遮断するのが安全確実である（図6）。

こうすることにより，万一侵入されても被害を犠牲マシンだけに限定することが可能になり，復旧作業が容易になる。犠牲マシンのハード・ディスクの内容をどこか安全な場所へ保存しておけば，侵入経路をふさぎ次第即座に元の状態へ戻すことも可能である。

犠牲パーティション

犠牲マシンをサービスごとに設置できればそれが理想なのであるが，個人でインターネットに接続する場合，何台もマシンを動かしておくわけにはいかない。第一，電気代がばかにならないし，狭い家だとマシンを設置している部屋で寝ることになるわけで，騒音も無視できない。1台のマシンで被害を限定するためのツールが **chroot** である。

chroot を使うと，任意のディレクトリをルート・ディレクトリに設定してサーバー・プログラムを実行できる。こうするとサーバー・プログラムは設定されたルート・ディレクトリの外にあるディレクトリにアクセスすることができない。もちろんルート・ディレクトリを元のルートに戻すことは不可能である（カーネルにバグがなければ。（図7））。

したがってこのサーバー・プログラムが侵入に利用されたとしても，侵入者が自由にアクセスできるのはルート・ディレクトリとして設定されたディレクトリの中だけであり，もしこのディレクトリが，sh や perl など侵入者が喜びそうなものは一切置かないパーティションであれば被害をこのパーティション内に限定することができる。

例えば，不特定多数にサービスを行なうための犠牲パーティション /alt-root を作り，そこにインストールした http サーバー・プログラム（図8）を立ち上げるには，次のように実行すればよい。

```
chroot /alt-root /bin/httpd
```

犠牲「パーティション」と書いたが，/alt-root はパーティションでなく普通のディレクトリであってもよい。ただしその場合，侵入者が /alt-root 下に大量のファイルを置いて，/alt-root を含むパーティション（この場合はルート・パーティション）をあふれさせ，マシンの他の機能に

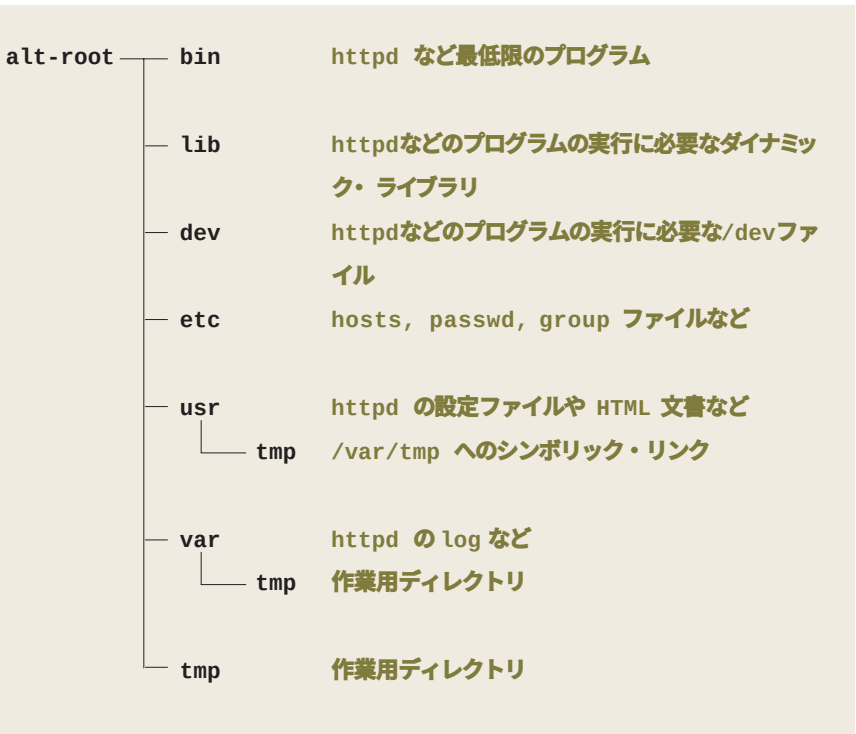


図8　犠牲パーティションには，httpdの実行に必要な最低限のファイル／ディレクトリだけを整えておく

影響を及ぼすことが可能になってしまう。であるから，chroot する先はあふれてもあまり困らないパーティションにすべきである。

さて，犠牲パーティションである /alt-root には，httpd の実行に必要な最低限のファイル/ディレクトリだけを整えておく。もし CGI の実行にどうしても perl が必要なら /alt-root/bin にインストールしても良いが，perl のような汎用インタープリタは侵入者にとっても非常に便利なツールである，ということを忘れないでほしい。もし CGI のプログラムをすべてC言語等で書くことが可能ならそうすべきである。

/alt-root/etc/passwd 等は，/etc/passwd 等と同じにする必要はない。httpd の実行に必要な内容に限定すべきであり，もちろんパスワード・フィールドは「*」などで埋めておく。

犠牲パーティションを作ることは難しいことではないが，１つ注意すべき点がある。それは syslog の扱いである。多くのサーバー・プログラムは /dev/log へ log を出力する。/dev/log は unix-domain socket と呼ばれる特殊ファイルで，サーバー・プログラムが書き出した log データを syslogd が /dev/log から読み込み，/etc/syslog.conf の設定にしたがって /var/adm/syslog 等の log ファイルへ出力する。

当然，/alt-root へ chroot してから実行したサーバー・プログラムは /alt-root/dev/log に log を出力しようとするので，syslogd は /dev/log と /alt-root/dev/log の両方から log を読み込む必要がある。犠牲パーティションが複数ある場合は，それぞれのパーティションの /dev/log を読まなければならない。sysklogd-1.3-31 に含まれている syslogd は -a オプションで読み込む unix-domain socket を19個まで追加できる。例えば次のように syslogd を実行すれば良い。

```
syslogd -a /alt-root/dev/log
```

古いディストリビューションを使っている場合，syslogd が -a オプションをサポートしていないかも知れない。実は，私が犠牲パーティションを使ったサーバーを立ち上げた時は sysklogd-1.3-25 が最新バージョンで，この時点では -a オプションがサポートされておらず，syslogd が読み込める unix-domain socket は1つだけだった。仕方ないので syslogd.c に手を加えて使っていたのだが，1998年10月に named を犠牲パーティションで走らせるために -a オプションがサポートされた。

最後のとりで

侵入を検出せよ

●tripwireでファイル/ディレクトリを検査せよ

侵入されても，すぐ気付けばいい。ハード・ディスクのバックアップをマメに取っていれば，ファイルを破壊されてもすぐ元に戻せるだろう。最悪なのは侵入に気付かず，クラッカーたちの集会場兼，違法ソフトウエアの闇取り引き市場兼，他のサイトへ攻撃を仕掛ける前線基地となってしまうことである。

集会場くらいならマシンが少々重くなるくらいの被害で済むかもしれないが，闇取り引きなどに使われてしまうと共犯者として逮捕されかねないし，逮捕されなかったとしてもマシンを証拠品として押収されてしまう。前線基地として使われたら被攻撃サイトから抗議メールが殺到してしまうだろう。場合によっては莫大な損害賠償を求められることがあるかも知れない。

tripwire

ファイルが侵入者によって書き換えられていないか調べるためのツールが **tripwire** である。検査対象のファイル/ディレクトリごとに，検査方法を指定できる。例えば，重要で書き換えが起こるはずのないファイルは，ファイルのハッシュ値を記録しておき，ハッシュ値が変化したら警告を出す，また log のように常にサイズが増え続けるファイルは，サイズが減った時のみ警告を出すなどの設定が可能である。

毎日 tripwire で検査した結果を管理者あてにメールで送るようにしておくと良いだろう。何人かで共同管理しているマシンでは，他の管理者が何か変更を行なったか知るための手段としても使える。

言うまでもないことだがセキュリティに完全はなく，tripwire もまた万全ではない。侵入者が tripwire のことを熟知している場合，侵入の痕跡を消されてしまう可能性は残る。特定のツールを過信せず，総合的に監視することが重要である。

（仙石 浩明）