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7回 「ファイアウオール(前編)」

今回はLinuxサーバーのセットアップとファイアウオールの構築を目的とした，TCP/IPの基礎知識を解説します。最近は個人ユーザーでもインターネットに常時接続しやすくなりましたが，セキュアなサーバーを構築するためにはTCP/IPの仕組みを理解することが不可欠です。

会社を設立しました。「株式会社ケイ・ラボラトリー」*1という名称で，次世代携帯電話のコンテンツ関連技術に特化した研究開発型企業です。今後，Javaを搭載した携帯電話が増えていくと予想されますが，携帯電話のさまざまなリソース上の制約により，Java搭載型携帯電話上で動くプログラムを開発することは，一般のプログラマにとってはあまり容易ではありません。開発ツールの提供や技術コンサルティングなどを通して，携帯電話コンテンツの発展のお手伝いをできたらと考えています。

2000年7月下旬に設立発表のプレスリリースを出したこともあり，既に各方面から多くの引き合いがありました。そのため数多くのプロジェクトが同時進行中です。せめてプロジェクト数に見合う程度には人を増やすべく，現在人材急募中です*2。

ディストリビューションを使用

今回ケイ・ラボラトリーのWebサーバーを立ち上げるに当たって，本業の合間にファイアウオールの構築とサーバーのセットアップを行いました*3。あまり時間をかけられなかったので，既存のディストリビューションに手を加えることにしました*4。

実を言うと1994年にSlackwareをインストールして以来，ディストリビューションを使うのは実に6年ぶりのことです。また，Red Hat系ディストリビューションを使うのはこれが初めてです。放っておけば勝手にインストールしてくれるのは手間が省けて良いのですが，何をするにもGUI管理ツール*5が必要で，フラストレーションがたまってしまいました。どうしてこんなにいろんなパッケージをてんこ盛りにするのでしょうか。困ったものです*6。

まずは，勝手にインストールされてしまったWebサーバー，メール・サーバー，ネーム・サーバーなどを削除することから始めました。ディストリビューションに含まれているこれらのサーバー・プログラムは，ディレクトリ構成などに手が加えられています。トラブル発生時に迅速に対応できるようにするため，対外サービスを行うプログラムは，すべてソース・プログラムを取り寄せてインスト

---

*1 「ケイ・ラボラトリー」の「ケイ」はアルファベットの「K」です。「K-Java」の「K」，すなわち「キロ・バイト」の「K」に由来します。「軽自動車」(ケー・カー)などの「軽い」という意味の「ケー」や，「携帯電話」(ケータイ)の「ケー」である，という説もあります。「Java」の「J」の次を目指す，という意味もあるのかも知れませんが，決して「3K」(きつい・きたない・きけん)の「K」ではありません。「きつい」かどうかはさておき，真新しいオフィス(神谷町森ビル)なので，「きたない」と「きけん」はあてはまりません。

*2 興味のある方は，http://www.klab.org/を参照して応募してください。

*3 正確に言えば，セットアップしたのは今年3月です。

*4 何がインストールしてあるか分からない既成のディストリビューションは大嫌いなので，私の自宅のサイトであるGCD(gcd.org)では，1994年のSlackwareをベースに，オリジナルの痕跡を全くとどめないほど手を加えたサーバーを使っています。

*5 初心者にはとっつき易いのでしょうが，少しでも慣れてくればGUIは百害あって一利無しですね。

*6 暇を見つけては不要なパッケージを片っ端から削除しました。削ったパッケージは，100個以上にのぼります。

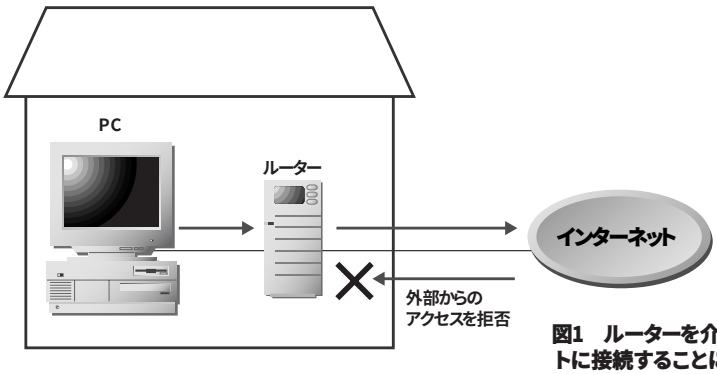

図1　ルーターを介してインターネットに接続することにより外部からのアクセスを拒否できる

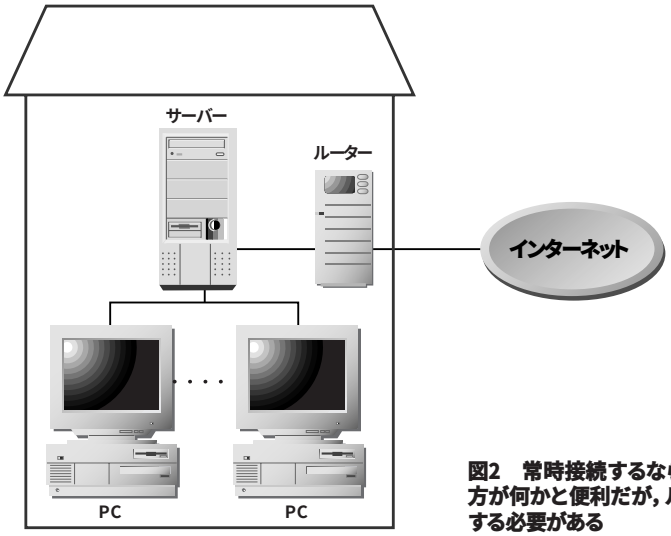

図2　常時接続するならサーバーを設置した方が何かと便利だが，ルーターの設定に注意する必要がある

ールし直すべきでしょう。

次に，inetdの削除です。inetdが面倒を見るサービスのほとんどは今となっては使われないものばかり*7で，なぜ多くのディストリビューションがいまだにinetdをデフォルトで走らせているのか理解に苦しみます。inetdは前世紀の遺物ですから，さっさと捨てましょう。

最後に，ipchainsを使ってIPフィルタを設定します。多くの解説書には，「サーバー・マシンでは不要なサーバー・プログラムを停止せよ」と書かれていますが，これは間違いです。停止させただけでは何かのはずみで再び動き出すかも知れません*8。

1024番以上のポートであれば，一般ユーザーが勝手に変なサーバー・プログラム*9を走らせる危険もあります。仮にサーバーが動き出しても外部からアクセスされないように IPフィルタで外部との通信を完全に遮断するべきでしょう。

来月号で，ルーターとサーバー・マシン上のIPフィルタを使ってファイアウオールを構築する方法を解説する予定です。

今回はそのための準備として，TCP/IPの基礎を解説します。

ダイヤルアップ接続

NTT東日本とNTT西日本のフレッツ・ISDN やケーブルテレビ・インターネットなどのサービスにより，手軽な定額料金でインターネットへつなぎっぱなしにできるようになってきました*10。つながっている時間が長くなれば長くなるほど，インターネット経由で攻撃を受ける可能性が高くなるわけです。まさかダイヤルアップ接続で外部から侵入されることはないだろうなどとタカをくくっていると，ある日突然ハード・ディスク内に見慣れないファイルができていたり，あるいは重要なファイルが無くなっていたりしてあわてることになります。

被害が目に見える形で現れていればまだ良いのですが，本当に恐いのはPC内の重要な情報が知らないうちに盗み読みされている場合です。どこから情報がもれたか分からないままに会社が大損害を受けるかも知れません*11。

よく知られていることですが，Windows95/98などの現在主に流通している一般向けOSや，Linuxであっても特にセキュリティを意識していないディストリビューションは，外部からの攻撃に対して非常にぜい弱です。このようなPCに，仕事上の機密情報が入っている場合は，たとえ数秒のダイヤルアップ接続でも，盗み読みされない保証は全くありません。重要な情報はMOなどのリムーバル・ディスクに保存しておいて，ダイヤルアップする前に必ずリムーバル・ディスクをイジェクトしておくなどの用心が必要でしょう*12。

インターネットにPCを直接つなげることは，本質的に極めて危険な行為なの

ですが，Windows95/98や，最近のLinuxのディストリビューションは，危険性を意識することなく手軽に直接インターネットにダイヤルアップ接続できてしまうので，困ったものです。

ダイヤルアップ・ルーター

インターネットに直接接続するのは危険すぎるので，ダイヤルアップ・ルーターを介して接続することにします（**図1**）。最近のダイヤルアップ・ルーターは，IPフィルタリングのデフォルトが安全側，つまり外部からのアクセスを拒否する設定になっている＊13ので，とりあえずダイヤルアップ・ルーターをつなぐだけでもPCからインターネットに直接接続する場合よりはかなり安全になります。

LinuxでPPP接続に関する質問はFAQリストの筆頭ですが，PPPの設定ができない人がまともなセキュリティ対策を行えるはずはなく，そのような場合は迷わずダイヤルアップ・ルーターを使うべきでしょう。初心者のPPP接続に関する質問に対して親切に教えてあげたつもりが，攻撃の犠牲者を増やす結果につながりかねないので注意が必要です。

サーバーの立ち上げ

Webサイトの閲覧など，インターネットへアクセスするだけなら，ダイヤルアップ・ルーターとWebブラウザが走るPCだけで良いのですが，常時接続しているのならサーバーを立ち上げたいところです（**図2**）。

ところが，サーバーを立ち上げるには，インターネットからサーバー・マシンへのアクセスを，ルーターの設定で許可しなければなりません。つまりサービスに必要なパケットだけを通し，その他のパケットはすべて排除しなければなりません。

適切な設定をするには，TCP/IPなどのプロトコルに関する知識が必要です。今まで見よう見まねで設定をしていた方は，ぜひこの機会にTCP/IPの仕組みを理解して，ルーターの設定を見直してください。

TCP/IPの基礎

昔からパソコン通信をやってた人は実感している＊14と思いますが，データ通信にはエラーがつきものです。IP（Internet Protocol）においても，通信経路上の通信エラー，輻輳（ふくそう）などの原因によってパケットは壊れたり捨てられたりします。

UDP（User Datagram Protocol）＊15のように，失われたパケット対策＊16をアプリケーションが面倒を見るプロトコルもあります。しかし，多くのアプリケーションは，パケットが失われたか否かを監視するのが面倒なので，データが失われずに届き，かつ送信ホストが送り出した順番で受信ホストに届くことを保証したプロトコルであるTCP（Transmission Control Protocol）を利用しています。

TCPについてより詳しく知りたい場合は，RFC761を参照してください。ここではファイアウオールに関係してくる部分だけを簡単に解説します。

TCPのパケットは，ヘッダーとデータからなり，ヘッダーには**図3**のような情報が含まれます。

ただし図3の(1)，および(3)はTCPに限らず，すべてのIPパケットに共通して含まれます。インターネットにおいてホストを識別するためのアドレス，すなわちIPアドレスです。(2)と(4)はUDPにもありますが，その他はTCPに特有で，

内容	bit長
(1) 送信元アドレス	32bit
(2) 送信元ポート番号	16bit
(3) あて先アドレス	32bit
(4) あて先ポート番号	16bit
(5) シーケンス番号(SEQ番号)	32bit
(6) 確認応答番号(ACK番号)	32bit
(7) SYN フラグ	1bit
(8) ACK フラグ	1bit
(9) FIN フラグ	1bit

図3　TCPパケットのヘッダーの内容

＊7　うそだと思うなら，/etc/inetd.confを確認してみましょう。ほとんどすべての行がコメントアウトされているはずです。コメントアウトしてないものがもしあれば，そのサービスが本当に必要か考えてみてください。
＊8　近ごろのディストリビューションは非常に込み入った設定ファイルを使用していますから油断できません。使わないサーバー・プログラムは消してしまうに限ります。
＊9　故意または過失でバックドアを開けてしまうというのはよくある話です。
＊10　私はOCNエコノミーを使って常時接続しているのですが，つなぎ放題にする手段が増えてきたせいか，OCNエコノミーの混雑具合が最近かなり改善されてきたような気がします。OCN エコノミーからフレッツ・ISDNやケーブルテレビ・インターネットなどへ乗り換えた人が結構いるのではないでしょうか?。
＊11　例えば執筆中の発明が競合会社に盗み読みされ，先に特許を取られてしまうかも知れません。こうなると特許が取れなくなってしまうばかりか，将来ばく大な特許使用料を請求されかねません。しかも，なぜ同時期に競合会社が似たような発明ができたのかが，分からずじまいになってしまうかも知れません。
＊12　これでも安全ではありませんが，PCに重要情報を入れっぱなしにしておくよりは数段マシです。
＊13　反面，昔ながらのダイヤルアップ・ルーターに慣れている人にとっては勝手が違ってハマります。最近の大抵のダイヤルアップ・ルーターは，外部からLAN上のネーム・サーバーやメール・サーバーをアクセスする必要はないものと思い込んでいますので，アクセスできるように設定を変更するのが結構大変です。マニュアルを隅から隅まで読まないと，設定変更方法が分からなかったりします。
＊14　MNP（Microcom Networking Protocol）などのエラー訂正プロトコルが普及する以前は，文字化けが頻発していました。
＊15　詳しくはRFC768を参照してください。
＊16　大きく分けて，送信元にパケットが失われた旨通知して再送してもらう方法と，単に失われたパケットを無視する方法があります。後者は動画や音声などの，多少ならパケットが失われても問題ないアプリケーションで用いられます。

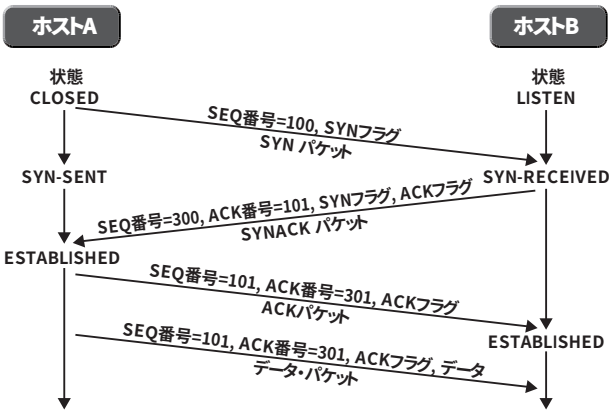

図4　3-ウエイ・ハンドシェイク
シーケンス番号を同期させるために，3つのパケットをやりとりする。

TCPの信頼性を高めたデータ通信を支えるためのものです。

TCPでは送信するデータの1バイトずつにシーケンス番号が割り振られます。最初の1バイトの番号が101ならば次のバイトは102，1000バイトのデータを送った後ではその次のデータは1101といった具合です。図3の(5)のシーケンス番号は，これから相手ホストへ送る最初のデータのシーケンス番号（以下，SEQ番号）を表わします。

また(6)確認応答番号（以下，ACK番号）は，相手ホストから正常に受け取ったデータのシーケンス番号の次の番号を表わします。相手ホストが送信した最初の1バイトの番号が301で，1000バイトを正常に受け取ったのならば，ACK番号は1301になります。つまりシーケンス番号が「ACK番号」であるデータを送ってほしい，という要求であると考えると理解しやすいかも知れません。

ホストAからホストBへTCP/IPで接続する場合を例に，TCPの仕掛けを説明しましょう。**図4**に接続が完了する（ESTABLISHED状態）までのハンドシェイクの様子を示します。

●接続要求（SYN）パケットをホストAが送信

最初，ホストAの状態はCLOSED，つまり接続されていない状態，ホストBはLISTEN，つまり接続受付可能状態です。ホストAは，ホストBに対して接続要求パケットを送信します。「SEQ番号」はホストAが適当に決める数値です。また「SYNフラグ」は，TCP接続において最初に送られるパケットにのみ[*17]フラグが立ちます（すなわち値が1）。

この最初の接続要求パケットのことを，SYNパケットと呼びます。送信後，ホストAの状態は「SYN-SENT」，つまり「SYNパケットを送信した」状態へ移行します。

●接続要求（SYN）パケットをホストBが受信

接続要求パケットがホストBに届くと，ホストBの状態は「SYN-RECEIVED」，つまり「SYNパケットを受信した」状態に移行します。

●最初の確認応答（SYNACK）パケットをホストBが送信

そしてホストBは，SYNパケットを受け取ったことをホストAに伝えるべく，「ACK番号」にSYNパケットのSEQ番号に1を加えた値，つまり101を設定した確認応答パケットを送信します。この101が，ホストAが送るデータの最初の1バイトのシーケンス番号となります。

確認応答パケットの場合，「ACKフラグ」が立ちます。TCP接続の場合，SYNパケット以外のパケットはすべて直前に接続相手から送られたパケットの確認応答という性格を帯びますので，すべてACKフラグが立っています。「SEQ番号」はホストBが適当に決める数値です。このパケットはホストBからホストAに送られる「最初の」パケットですから「SYNフラグ」が立ちます。

この最初の確認応答パケットのことをSYNACKパケットと呼びます。

●最初の確認応答（SYNACK）パケットをホストAが受信

SYNACKパケットがホストAに届くと，ホストAとホストBの双方が，お互いのデータの最初の1バイトのシーケンス番号を知ったことを，ホストAは知ります。

なお，この段階ではまだホストBは，ホストBのデータの最初の1バイトのシーケンス番号がホストAに伝わったかどうかを知ることができないことに注意してください。

「双方がお互いのシーケンス番号を知ったこと」を知る状態を「ESTABLISHED」，つまり「接続完了」状態と呼びます。ホストAは「ESTABLISHED」状態に移行しますが，この段階では，まだホストBは「ESTABLISHED」状態に移行できません。

●確認応答（ACK）パケットをホストAが送信

ホストAは，ホストBが送ったSYNACKパケットを受け取ったことをホストBに伝えるべく，「ACK番号」にSYNACKパケットの「SEQ番号」に1を加えた値，301を設定した確認応答パケット（以下，ACKパケットと略記）を送信します。この301が，ホストBが送るデータの最初の1バイトのシーケンス番号となります。

●確認応答（ACK）パケットをホストBが受信

このパケットを受け取って初めて，ホストBは，ホストBのデータの最初の1バイトのシーケンス番号がホストAに伝わったことを確認できます。つまりホストBも「ESTABLISHED」状態に移行します。

以上の3つのパケットのやりとりは，「3-ウエイ・ハンドシェイク」と呼ばれます。この3-ウエイ・ハンドシェイクによって，双方のホストがお互いのシーケンス番号を同期させると，後はどちらのホストからでも，データを送信することができます。

データを受信した側は，適当なタイミングで正しく受信できた最後のデータのシーケンス番号の次の番号を「ACK番号」にセットして，ACKフラグを立てたACKパケットを送ります。もしデータが通信エラーなどで受信側に届かなかった場合，送信側が送るパケットのSEQ番号より，受信側が送り返すACKパケットのACK番号の方が小さくなります。その場合，送信側はシーケンス番号が「ACK番号」であるデータから送信し直します。

例えば，n_0バイト，n_1バイト，n_2バイトの3つのパケット（**図5**）を**図6**のように送ったとします。もし3つのパケットとも正常に受信側で受信できたとすると，受信側は「ACK番号」に$x+n_0+n_1+n_2$をセットして，ACKパケットを返します。

もし最後のパケットが何らかの原因で受信側で受信できなかったとすると，正常に受信できた最後のデータのシーケンス番号は$x+n_0+n_1-1$ですから，受信側は「ACK番号」に$x+n_0+n_1$をセットして，ACKパケットを返します。すると，送信側はシーケンス番号$x+n_0+n_1$のデータから再送しなければなりません。つまり，**図7**のパケットを再送します。

TCP/IPに対する攻撃

TCPでは盗聴をはじめとするさまざまな攻撃があり得るので注意が必要です。

盗聴

TCPのパケットで送られるデータは，アプリケーションプログラムが送信したデータそのままです。

しかも，「SEQ番号」を見ればそれがデータの何バイト目であるか分かりますから，通信路の途中でパケットを盗み読むことができれば，双方のホストが送信したデータの流れを再構成することは極めて簡単です。暗号化していないデータは盗み読みされるものと思っていた方が無難です。

セッションのハイジャック

使い捨てパスワードを使っていれば，

n_0バイトのデータ	シーケンス番号	x	から $x + n_0 - 1$	まで
n_1バイトのデータ	シーケンス番号	$x + n_0$	から $x + n_0 + n_1 - 1$	まで
n_2バイトのデータ	シーケンス番号	$x + n_0 + n_1$	から $x + n_0 + n_1 + n_2 - 1$	まで

図5　送信した3つのパケット

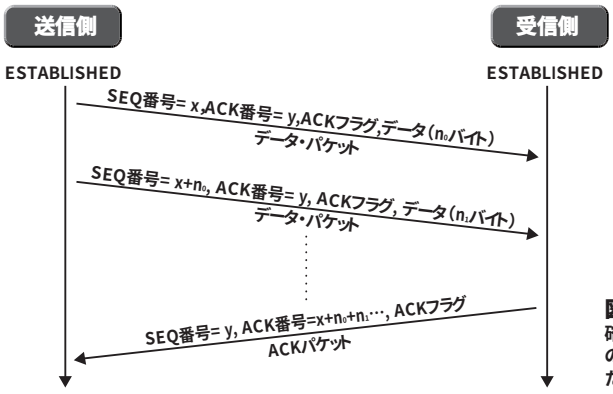

図6　データの送信
確認応答番号によって，何バイト目のデータまで正しく受信側に届いたかを，送信側は認識できる。

n_2'バイトのデータ	シーケンス番号	$x + n_0 + n_1$	から $x + n_0 + n_1 + n_2' - 1$	まで

図7　再送されるパケット

＊17　通信エラーなどで，最初のパケットが届かないことが起り得ます。この時はホストBから何の返答もないので，一定時間待った後，ホストAは再びSYNパケットを送信します。

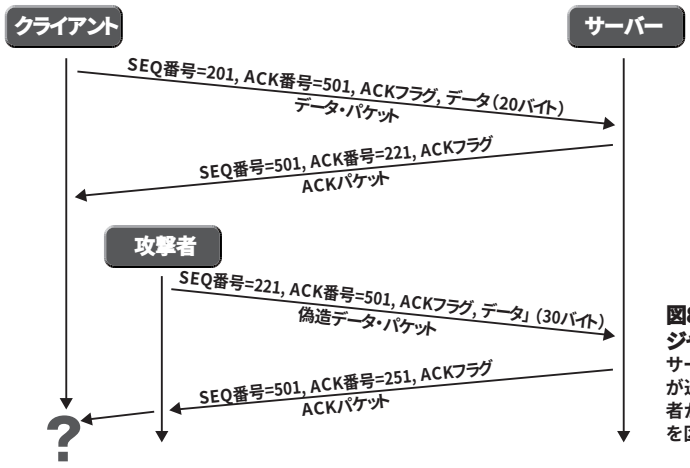

図8　セッションのハイジャック
サーバーは，正規ユーザーが送信したパケットと攻撃者が送信した偽造パケットを区別できない。

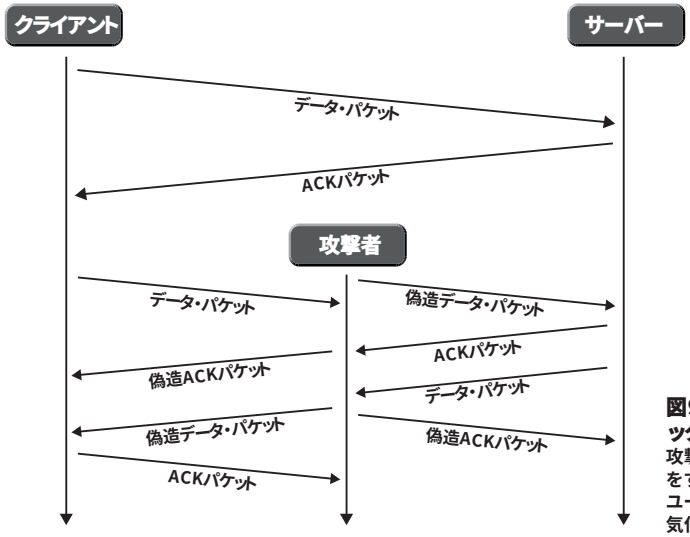

図9　二重のハイジャック
攻撃者は，サーバーのふりをすることによって，正規ユーザーがハイジャックに気付かないようにできる。

たとえ盗み読みされても，盗聴者には次のセッションで必要なパスワードを知ることができないから安全，と思っている人はいませんか?。残念ながら盗聴できる場合はハイジャックも可能である場合が多いので，使い捨てパスワードだから安心とは限らないのです。

例えば，正規ユーザーが正しい使い捨てパスワードを入力してサーバーへログインしたとします。この段階で盗聴者はこのセッションをハイジャックしてしまうことが可能です。なぜなら盗聴者はサーバーが送信したACKパケットの「SEQ番号」と「ACK番号」を知っていますから，正規ユーザーの代わりに偽造パケットをサーバーへ送りつけることが可能です(**図8**)。

サーバーは，正規ユーザーが送ったデータと攻撃者(ハイジャッカ)が送ったデータを区別することはできませんから，攻撃者が送ったコマンドをそのまま実行してしまうことでしょう。

図8のように，サーバーが攻撃者に対して送ったACKパケットが正規ユーザーのマシンにも届いた場合は，何もデータを送っていないのに「ACK番号」が突然増えたことから，ハイジャックされたことを検知できますが，攻撃者が何らかの方法によって*18，サーバーが送ったパケットが正規ユーザーに届かないようにしてしまえば，正規ユーザーはハイジャックされたことに気付くのが難しくなります。

さらに，攻撃者がクライアント側(正規ユーザー)に対しても，偽造データ・パケットを送って，サーバーの応答の真似をすれば，正規ユーザーにハイジャックされたことを全く気付かせることなく，サーバーを自由に操作することも不可能ではありません(**図9**)。

ハイジャックを防ぐには，クライアント・プログラム，サーバー・プログラムの両アプリケーション間で相互に認証を行い，かつ相互にやりとりするデータのすべてを暗号化する必要があります。つまり TCP/IP接続を含め，両ホストの間にあるものは何であれ信用してはいけない*19ということです。

盗聴やハイジャックは，攻撃者が通信線路上のどこかに細工する必要があります。可能性のあるのは次の3カ所のうちのいずれかでしょう。

(1)サーバーあるいはクライアントが属するLAN
(2)アクセス回線(電話加入者回線など)
(3)プロバイダ内部
(4)専用線(電話局間回線など)

(4)での細工は，普通の攻撃者にとっては不可能でしょう。(2)と(3)は場合によっては可能かも知れませんが，普通は困難でしょう。したがってほとんどの

場合，盗聴やハイジャックは(1)で行われると考えて良いでしょう。つまり攻撃者は攻撃対象のすぐ近くにアクセス＊20できる必要があります。攻撃者にとって攻撃可能対象はかなり限定されます。

IPアドレス・スプーフィング

一方，どこのサイトのサーバーに対しても行える攻撃が，IPアドレス・スプーフィング（spoofing，だますこと）と次で説明するSYNフラッディング（flooding，氾濫させること）です。逆に言うと世界中のどこからでもこの攻撃を受ける可能性があるわけで，より注意深く警戒する必要があります。

図4の3-ウエイ・ハンドシェイクにおいて，接続元のホストAが送るパケットの「送信元アドレス」を，ホストAのIPアドレスではなく，全く関係のないホストCのIPアドレスに設定して，接続要求パケットを送信したらどうなるでしょうか？。

接続先のホストBに届く前に，通信路上のルーターなどの設定によっては，「送信元アドレス」が異常ということで排除されてしまうこともあるのですが，ここでは間にそのような設定のルーターがなく，ホストBに届いたと仮定します。

するとホストBは，届いたパケットの「送信元アドレス」すなわちホストCに対して，SYNACKパケットを送信します。ホストCがホストAと全く関係ないサイトにあるマシンであれば，ホストAでこのACKパケットを受信することはできません。

したがってホストAは，ACKパケットに設定されている「SEQ番号」を知ることができず，3-ウエイ・ハンドシェイクのACKパケットの「ACK番号」に何を設定したら良いか分からないのでハンドシェイクを完結することができない，だから安全だ，というのがTCP/IPが作られた当時の考え方でした。

ところが，ホストBのシーケンス番号を予測することは不可能ではありません。ずばり番号を言い当てることは難しいにしても，ある程度の範囲内で予測できれば，試してみればいいだけです。「ACK番号」に予測した値を入れたパケットを入れたACKパケット送り，続いてデータを送ります。

運良く「ACK番号」がホストBのシーケンス番号に一致すれば，ホストB は「ESTABLISHED」状態になり，続いて送られてきたデータをホストCから送られたデータと思い込ませることができます。

これが，IPアドレス・スプーフィングです。サーバーが，特定のIPアドレス（上記の例で言えばホストC）からの接続を盲目的に信頼する設定になっていると，この種の攻撃のえじきになります。

例えばrsh（リモート・シェル）は，~/.rhostsファイルに特定のホスト名（あるいは，IPアドレス）を書いておくことにより，そのホストからパスワードなしでアクセスできますが，攻撃者が~/.rhostsに書いてあるホスト名を何らかの方法で知れば，そのホストからの接続と見せかけて任意のコマンド＊21を実行することができます。

この種の攻撃を防ぐには，送信元アドレスだけの認証を行わず，必ず他の認証方法と組み合わせることが重要です。

SYNフラッディング

3-ウエイ・ハンドシェイクにおいて，サーバー側（ホストB）がSYNACKパケットを送信した後，クライアント側（ホストA）がACKパケットを送らないとどうなるでしょうか?。サーバーは一定時間待ってもACKパケットが帰ってこないと，SYNACKパケットが途中で失われたと判断してSYNACKパケットを再送します。しかし，そのためにはクライアントから送られてきたSYNパケットに含まれる，送信元アドレス，ポート番号，「SEQ番号」，そしてSYNACKパケットで送った自分のシーケンス番号を記憶し続けなければなりません。

もちろん，一定回数再送を試みてもACKパケットが帰ってこなければ，あきらめて記憶内容を破棄するのですが，その前にクライアントが繰り返しSYNパケットを送りつけると，そのたびにサーバーはSYNパケットに含まれる情報とSYNACKパケットで送った自分のシーケンス番号を記憶しなければならず，ついには記憶領域を食いつぶして，他のTCP/IP接続を受け付けられなくなってしまいます。この攻撃は「SYNフラッデ

＊18 通信線路の途中にルーターを挟んで，目的のパケットを横取りしてしまえば可能です。
＊19　クライアント側のLANとサーバー側のLAN同士がVPN（仮想プライベート・ネットワーク）で接続されていたとしても，攻撃者がLAN内に潜んでいれば意味ありません。
＊20　物理的なアクセスだけでなく，同じLAN上にroot権限を奪ったマシンがあれば，攻撃可能です。
＊21　攻撃元のホスト（もちろん攻撃者の所有するマシンなどではなく，侵入して踏み台として利用しているマシンでしょう）名を~/.rhostsに書込めば，次回からはスプーフィングせずに堂々と侵入できます。

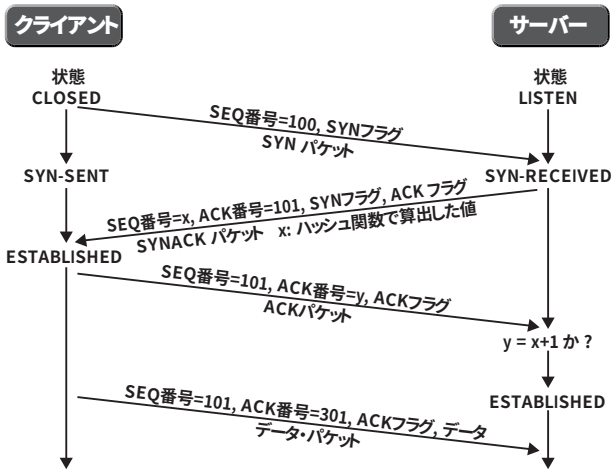

図10 SYNクッキー方式
SYNACKパケットの「SEQ番号」に設定する値を，相手のIPアドレスなどに基づいてハッシュ関数で算出し，その値を保持しない。

```
echo 1 >/proc/sys/net/ipv4/tcp_syncookies
```

図11 SYNクッキーを有効にするための手順

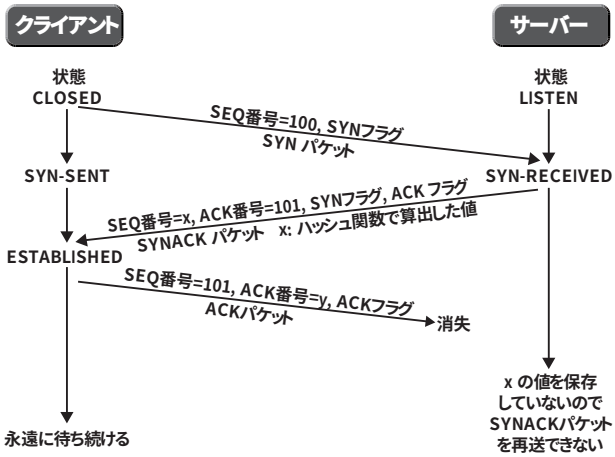

図12 SYNクッキーの問題点
SYNACKパケットに対するACKパケットが途中で失われると，クライアントは「ESTABLISHED」状態のまま永遠に待ち続ける羽目になる。

ィング」と呼ばれます。

盗聴や侵入などの直接的な被害を与えるのではなく，他のユーザーに対するサービスを妨害する攻撃を，一般にDoS（Denial of Service，サービス妨害）攻撃と呼びますが，SYNフラッディングは他のDoS攻撃と比べて，手軽に攻撃できてしまうという（攻撃される側にとっては困った）特徴があります。

SYNフラッディングを回避するために，最近のLinuxには「SYNクッキー」と呼ばれる仕掛けが組み込まれています。これはSYNACKパケットで送った自分のシーケンス番号を記憶する代わりに，一方向ハッシュ関数を用いてSYNパケットからシーケンス番号を算出する*22方法です。算出した番号（これをクッキーと呼びます）を「SEQ番号」に設定してSYNACKパケットを送信し，ACKパケットが送られてきたら，再び同様の方法でクッキーを算出して，その値に1を加えた値がACKパケットの「ACK番号」に一致していたらハンドシェイク成功とみなします（**図10**）。

SYNクッキーを有効にするには，Linuxカーネル構築時に「CONFIG_SYN_COOKIES」項目に対して「Y」と答え，さらに**図11**を実行する*23必要があります。

ただし，SYNクッキーは根本的な解決策ではありません。SYNクッキーのアイデア自体は前述した通り簡単なものですが，実装方法は複雑です。詳しくは，http://cr.yp.to/syncookies.htmlを参照していただくとして，ここではSYNクッキーの問題点を簡単に説明します。

例えば，SYNACKパケットを送信後，クライアントが送信したACKパケットが通信エラーなどの原因で途中で失われてしまった場合を考えます（**図12**）。この場合，再送の責任はサーバー側にあるのですが，サーバーはSYNACKパケットの内容を記憶していないので，SYNACKパケットを再送することができません。哀れなクライアントは，「ESTABLISHED」状態のまま永遠に待ち続けることになります。

このような問題点を回避するため，LinuxにおけるSYNクッキーの実装では，攻撃を受けていないときは，SYNACKパケットの内容を記憶して再送に備えるようになっています。そしてSYNフラッディングを検出した時に限ってSYNACKパケットを送出後に，速やかにその内容を忘れてしまいます。この場合，ACKパケットが運良く失われずに戻ってくれば接続することができます。

*22 SYNパケットの送信元のIPアドレス，ポート番号，シーケンス番号，および分単位で増加する数値などに基づいて，一方向ハッシュ関数であるMD5を用いて算出します。
*23 rc.localなどに記述しておくと良いでしょう。