=== ch06.docx

[chap]6章　一般的な下層プロトコル

レイテンシの問題をトラブルシューティングする場合やうまく機能しないアプリケーションを識別するとき、あるいは異常なトラフィックを見つけるためにセキュリティ脅威に集中する場合でも、まず正常なトラフィックの理解が不可欠です。次の数章では、正常なネットワークトラフィックについてパケットレベルで見ていきます。TCP、UDP、IPといったもっともよく使われるプロトコル、またHTTP、DHCP、DNSなどのアプリケーション層プロトコルを見てみましょう。それぞれのプロトコルの項には関連するキャプチャファイルがあるので、ダウンロードして直接操作することができます。本章では特に、OSI参照モデルのレイヤ1から4に関連する下層プロトコルに焦点を当てていきます。

この数章は間違いなく、本書においてもっとも重要な章です。ここを読み飛ばすのは、コーンブレッドのない日曜の夕食を作るようなものです。各プロトコルの機能についてすでによく理解しているとしても、最低でもさっと目を通して、それぞれの構造を見直しておきましょう。

ARP

ネットワーク通信では、論理および物理アドレスの両方が使われます。論理アドレスの使用によって、複数ネットワークと直接接続されていないコンピュータ間との通信が可能になります。物理アドレスは、スイッチを通じて直接つながっている機器の単一のネットワークセグメント上での通信を促します。通信を行うには、この両方のアドレスが必要となる場合が多いです。

ネットワーク上のある機器と通信したいとします。この機器はサーバかも知れませんし、ファイルを共有したいワークステーションかもしれません。通信を始めるために使っているアプリケーションはすでにリモートホストのIPアドレスをDNSによって認識しています（これについては7章で説明）。つまり送信したいパケットのレイヤ3からレイヤ7の情報の構築に必要なものを、すべて持っていなければならないということです。この時点で必要な情報は、ターゲットホストのMACアドレス（物理アドレス）を含んだレイヤ2データリンクのデータのみです。

スイッチは、そのポートにプラグインしている機器のMACアドレスが載っているCAM（Content Addressable Memory）テーブルを使っています。スイッチは特定のMACアドレスに向けられたトラフィックを受信すると、このテーブルを使ってどのポートにトラフィックを送るかを判断するのです。宛先のMACアドレスが不明だと、まずキャッシュで調べます。それでも分からない場合は、さらに通信を行って調べることになります。

TCP/IPネットワーク（IPv4）において、IPアドレスからMACアドレスを求めるのに使われるプロトコルをARP（Address Resolution Protocol）と呼び、これはRFC 826で定義されています。ARPは、ARPリクエストとARPレスポンスの2種類のパケットしか使いません（図6-1）。

注　RFC（Request for Comments）とは、プロトコルの標準的な実装について定義している正式なドキュメントのことです。RFCのWebサイト<http://www.rfc-editor.org>で、RFCを検索することができます。

送信元のコンピュータが、「みなさんこんにちは、私のIPアドレスはXX.XX.XX.XXで、MACアドレスはXX:XX:XX:XX:XX:XXです。IPアドレスがXX.XX.XX.XXの人に送信するものがあるのですが、MACアドレスが分かりません。このIPアドレスを持っている方、MACアドレスと合わせて返信してくれませんか？」というARPリクエストを送ったとします。

このパケットはネットワークセグメント上のすべての機器に送られますが、このIPアドレスを持たない機器はパケットを廃棄します。IPアドレスを持った機器だけが、「送信元さん、あなたが探しているIPアドレスXX.XX.XX.XXを持っているのは私です。私のMACアドレスはXX:XX:XX:XX:XX:XXです」という答えを含んだARPレスポンスを送ります。

このプロセスが完了すると、送信元はこの機器のMACおよびIPアドレスの情報を含めてキャッシュを更新、データ送信を開始します。

図6-1　ARPプロセス

ARPリクエスト　ARPレスポンス

送信元IP　送信元MAC　宛先IP　宛先MAC

注　コマンドプロンプトでarp –aと入力すると、WindowsホストのARPテーブルを見ることができます。

このプロセスが実際に行われているところを見れば、どう機能するかが理解できるでしょう。例を見る前に、ARPパケットヘッダについて説明します。

ARPヘッダ

図6-2で示したように、ARPヘッダには以下のフィールドが含まれています。

ハードウェアタイプ　レイヤ2タイプを使用。イーサネットである場合が多い（タイプ1）

プロトコルタイプ　ARPリクエストが使われるより高いレイヤのプロトコル。

ハードウェアアドレス長　使用されるハードウェアアドレスの長さ（オクテット/バイト）。イーサネットの場合は6。

プロトコルアドレス長　特定のプロトコルタイプの論理アドレスの長さ（オクテット/バイト）。

オペレーション　ARPパケットの機能。リクエストの場合は1、レスポンスの場合は2。

送信元ハードウェアアドレス　送信元のハードウェアアドレス。

送信元プロトコルアドレス　送信元のより高いレイヤのプロトコルアドレス。

宛先ハードウェアアドレス　宛先のハードウェアアドレス（ARPリクエストに集中）。

宛先プロトコルアドレス　宛先のより高いレイヤのプロトコルアドレス。

図6-2　ARPパケット構造

アドレス解決プロトコル

ビットオフセット

ハードウェアタイプ

プロトコルタイプ

ハードウェアアドレス長

プロトコルアドレス長

オペレーション

送信元ハードウェアアドレス（1番目16ビット）

送信元ハードウェアアドレス（2番目16ビット）

送信元ハードウェアアドレス（3番目16ビット）

送信元プロトコルアドレス（1番目16ビット）

送信元プロトコルアドレス（2番目16ビット）

宛先ハードウェアアドレス（1番目16ビット）

宛先ハードウェアアドレス（2番目16ビット）

宛先ハードウェアアドレス（3番目16ビット）

宛先プロトコルアドレス（1番目16ビット）

宛先プロトコルアドレス（2番目16ビット）

arp\_resolution.pcapファイルを開き、解決プロセスを見てみましょう。個々のパケットをひとつずつ見ていきます。

パケット1：ARPリクエスト

図6-3に示したように、最初のパケットはARPリクエストです。パケット詳細ペインのイーサネットヘッダを調べれば、このパケットが本物のブロードキャストパケットであることを確認できます。宛先のアドレスはff:ff:ff:ff:ff:ffです①。これはイーサネットブロードキャストアドレスなので、この宛先に送信されたものは、現在のネットワークセグメント上にあるすべての機器にブロードキャストされます。イーサネットヘッダの中にある送信元アドレスはMACアドレスとしてリストアップされます②。

図6-3　ARPリクエストパケット

構造から見て、これは確かにIPを使っているイーサネットネットワーク上のARPリクエストであることがわかります。送信元のIPアドレス（192.168.0.114）とMACアドレス（00:16:ce:6e:8b:24）③と、宛先のIPアドレス（192.168.0.1）が表示されています④。宛先のMACアドレス（入手しようとしている情報）が不明なので、標的のMACアドレスは00:00:00:00:00:00と表示されています⑤。

パケット2：ARPレスポンス

最初のパケットからMACアドレスを受け取ったので、最初のリクエストに対するレスポンス（図6-4）のイーサネットヘッダに、宛先のアドレスが含まれました。このARPレスポンスのヘッダはARPリクエストのヘッダとよく似ていますが、何カ所か変更されています。

・パケットのオペレーションコード（opcode）が0x0002となり①、リクエストではなくレスポンスであることを示しています。

・アドレス情報が変わっています。送信元MACアドレスとIPアドレスが、宛先のMACアドレスとIPアドレスになっています②。

・何よりも重要なのは、IPアドレス192.168.0.1のホストのMACアドレス（00:13:46:0b:22:ba）③を取得したことです。

図6-4　ARPレスポンスパケット

無償ARP

何かが「無償で」行われるというのは、否定的な意味合いを持つことがよくあります。しかしながら無償ARPはよいことです。

IPアドレスは変更される場合がよくあります。IPアドレスが変わると、キャッシュに記憶されたIPアドレスとMACアドレス間のマッピング情報も無効となります。このことによる通信エラーを防ぐため、無償ARPパケットが送信され、それを受け取った機器のキャッシュを新しいマッピング情報で強制更新します（図6-5）。

図6-5　無償ARPプロセス

送信元IP　192.168.0.101

送信元MAC　f2:f2:f2:f2:f2:f2

宛先IP　192.168.0.101

宛先MAC　00:00:00:00:00:00

無償ARPパケットが使用される場面はいくつかあります。もっとも一般的なのは、IPアドレスの変更です。キャプチャファイルarp\_gratuitous.pcapを開き、実際に見てみてください。無償ARPに関わっているパケットは1つだけなので、ファイルに含まれているパケットも1つです（図6-6）。

図6-6　無償ARPパケット

イーサネットヘッダを調べると、このパケットはブロードキャストとして送信されているため、すべてのホストが受信していることがわかります①。無償ARPのヘッダはARPリクエストのヘッダと似ていますが、無償ARPでは送信元IPアドレス②と宛先IPアドレス③が同一です。他のホストが無償ARPパケットを受け取ると、ARPテーブルが新しいものへと更新されます。このARPパケットは未承諾パケットですが、受けとった結果APRキャッシュが更新されるため、無償とみなされているのです。

無償ARPパケットが使われる場面は他にもあります。機器のIPアドレスの変更によって生成されるのに加え、一部のOSは起動時に無償ARPを生成します。またトラフィックのロードバランスのために無償ARPパケットが使われる場合があります。

IP

OSI参照モデルにおけるレイヤ3のプロトコルの主な目的は、ネットワーク間で通信を行うことです。MACアドレスはレイヤ2の単一ネットワーク上の通信に利用されます。同様に、レイヤ3はネットワーク間の通信処理を担当しています。この通信を可能にするプロトコルはいくつかありますが、もっとも一般的なのはIP（Internet Protocol）でしょう。ここではRFC 791で定義されたIPv4を見ていきます。

IPv4の働きを理解するには、ネットワーク間のトラフィックの流れを知る必要があります。IPv4は通信プロセスに貢献し、エンドポイントの場所を問わず、その間のデータ通信を管理しています。

ハブまたはスイッチですべての機器がつながっているシンプルなネットワークをローカルエリアネットワーク（LAN）と呼びます。2つのLANを接続する場合はルータでつなぎます。複雑なネットワークは、世界中に分布する数万台のルータを経由する数万ものLANで構成されていることがあります。インターネット自体も、数百万ものLANとルータの集合といえます。

IPアドレス

IPアドレスは32ビットのアドレスで、ネットワークに接続された機器に割り振られた識別番号です。32文字もの長さの1と0が連続したものを覚えるのは至難の業なので、IPアドレスはドット区切り10進数表記で記述されます。

ドット区切り10進数表記では、IPアドレスを構成する1と0の数字を4分割し、10進数へと転換して、A.B.C.Dの形式で0から255までの数字を使って表現します（図6-7）。1100000010101000 00000000 00000001というIPアドレスがあるとします。この値はまず覚えられません。しかしドット区切り10進数表記を使えば、192.168.0.1と表記することができます。

IPアドレスを4分割するのには理由があります。IPアドレスは2つの部分、つまりネットワークアドレスとホストアドレスから成り立っています。ネットワークアドレスは機器が接続しているLANを識別するもので、ホストアドレスは機器そのものを識別するためのものです。IPアドレスのどの部分がネットワークアドレスで、どの部分がホストアドレスとなるかは常に同じというわけではありません。これを決めるのは、ネットワークマスク（ネットマスク）と呼ばれるもうひとつのアドレス情報で、サブネットマスクと呼ばれることもあります。

図6-7　ドット区切り10進数表記によるIPv4アドレス表記

ネットマスクは、IPアドレスのどの部分がネットワークアドレスで、どの部分がホストアドレスかを識別します。ネットマスクの数値も32ビット長であり、1にセットされた各ビットが、ネットワークアドレスのために取り置かれたIPアドレスの部分を識別します。残ったビットは0にセットされ、ホストアドレスを識別します。

たとえば、バイナリでは00001010 00001010 00000001 00010110となるIPアドレス10.10.1.22があるとします。IPアドレスの各セクションの配置を決めるのに、ネットマスクを適用します。この場合のネットマスクは11111111 11111111 00000000 00000000です。つまり図6-8に示したように、ネットワークアドレス（10.10または00001010 00001010）のために取り置きされたIPアドレス部分は前半分で、後半分は個々のホスト（.1.22、または00000001 00010110）だということになります。

図6-8　IPアドレスのビットの配置はネットマスクで決まる

ネットワーク　ホスト　ネットワーク　ホスト

ネットマスクもドット区切り10進数表記で記述することができます。たとえばネットマスク11111111 11111111 00000000 00000000は、255.255.0.0と表記できます。

一般にIPアドレスとネットマスクは、CIDR（Classless Inter-Domain Routing）表記で記述されます。この形式では、IPアドレスはそのままで、そのあとにフォワードスラッシュと、IPアドレスのネットワーク部分を表すビットが続きます。たとえばIPアドレス10.10.1.22、ネットマスク255.255.0.0をCIDR表記で記述すると、10.10.1.22/16となります。

IPv4ヘッダ

送信元と宛先のIPアドレスは、IPv4パケットヘッダの重要な構成部分ですが、パケット内のIP情報はこれだけではありません。IPヘッダは、先に見たARPパケットのヘッダと比べるとかなり複雑です。IPが正しく働くよう、たくさんの機能が盛り込まれているのです。

図6-9に示したように、IPv4ヘッダには以下のフィールドがあります。

バージョン　使用されているIPのバージョン

ヘッダ長　IPヘッダの長さ

サービス種別　トラフィックに優先順位をつけるために、ルータが使用する優先フラグとサービス種別フラグ

全長　IPヘッダとパケットに含まれるデータの長さ

識別子　パケットまたは断片化したパケットの配列を識別するのに用いられる識別番号

フラグ　パケットが断片化されたパケットの一部なのかどうかを識別するのに用いられる

断片位置　パケットが断片化されている場合、このフィールドの値を使ってパケットを正しい順序で並べて復元する

生存時間　パケットの余命を定義する値。ルータ経由でホップ毎秒で測られる

プロトコル　次に来るパケットのタイプを識別するのに用いられる

ヘッダチェックサム　IPヘッダに誤りがないかどうかを検証するのに用いられるエラー検出機構

送信元IPアドレス　パケットを送信したホストのIPアドレス

宛先IPアドレス　パケットの宛先のIPアドレス

拡張情報　追加のIPオプションのために取り置かれている。ソースルーティングやタイムスタンプのオプション情報も埋め込まれる。

データ　IPアドレスとともに送信される実際のデータ

図6-9　IPv4パケット構造

インターネットプロトコル

ビットオフセット

バージョン　ヘッダ長　サービス種別　全長　識別子　フラグ　断片位置　生存時間　プロトコル　ヘッダチェックサム　送信元IPアドレス　宛先IPアドレス　拡張情報　データ

生存時間

パケットが破棄される前に経過する時間、あるいはパケットが通過できるルータ数の最高値を定義するのが、生存時間（TTL）です。TTLはパケットが作成されるときに定義され、パケットがルータによって転送されるごとに、1単位ずつ減っていきます。たとえば、あるパケットのTTLが2の場合、最初のルータに到達するとTTLは1となり、2番目のルータへと転送されます。このルータでさらにTTLは0へと減り、パケットの最終的な宛先がそのネットワーク上にない場合、パケットは破棄されます（図6-10）。TTL値は時間ベースであるため、非常に混み合っているルータの場合、TTL値が1以上減ってしまうことがあります。しかし通常は、1台のルータを通過するごとにTTLが1だけ減ると仮定して問題ありません。

図6-10　パケットのTTLはルータを通過するごとに減少する

TTL値はなぜ重要なのでしょうか？パケットの生存時間が気になるのは、送信元から宛先までにかかる時間に関してのみです。しかしながら、何十台ものルータを経由して、ホストまで辿り着かなければならないパケットを考えてみましょう。その途中で設定に誤りがあるルータにぶつかれば、宛先へと到着できなくなる可能性があります。こうしたとき、ルータはいくつかのことを行いますが、その行為のためにパケットがネットワーク上を永遠にさまよい続ける場合があるのです。

プログラミングの経験がまったくなくても、無限ループがあらゆる問題を引き起こし、プログラムあるいはOS全体をクラッシュさせてしまう結果につながるのはお分かりでしょう。同じことがパケットでも起こるのです。ルータ間を延々と行き来するパケットの数が増えれば、DoS状態に陥るまで、ネットワークの使用可能帯域は消耗します。こうした問題を回避するために、IPヘッダのTTLフィールドが作られたのです。

Wiresharkで一例を見てみましょう。ip\_ttl\_source.pcapファイルには、２つのICMPパケットが含まれています。ICMP（本章でのちほど説明します）がIPを使ってパケットを送っていることは、パケット詳細のペインのIPヘッダセクションを開けば分かります（図6-11）。

図6-11　パケットのIPヘッダ

IPのバージョンは4①、IPヘッダ長は20バイト②、ヘッダとペイロードの全長は60バイト③、TTLフィールドの値は128④であることが分かります。

ICMP pingの主な目的は、通信機器間の通信テストです。データはあるホストから別のホストへリクエストとして送られ、受信したホストはそのデータを応答として送り返さなければなりません。このファイルでは、アドレス10.10.0.3の機器⑤が、アドレス192.168.0.128の機器⑥へとICMPリクエストを送っています。この最初のキャプチャファイルは、送信元であるホスト10.10.0.3で作成されています。

今度はip\_ttl\_dest.pcapファイルを開きましょう。このファイルでは、データは宛先である192.168.0.128でキャプチャされています。最初のパケットのIPヘッダを開き、TTL値を調べます（図6-12）。

図6-12　IPヘッダからTTLが１減少しているのがわかる

このTTL値が127①で、最初のTTL値128よりも１少ないのがすぐに分かるはずです。ネットワークの構造を知らなくても、この2台の通信機器の間には1台のルータがあり、そのルータを通過するときにTTL値が１減ったと結論づけることができます。

IPフラグメンテーション

パケットの分割は、データストリームをより小さな断片とし、多種多様なネットワーク上でも確実に送信するためのIP機能です。

レイヤ2データリンク層プロトコルのMTU（maximum transmission unit）サイズと、このプロトコルを使っている通信機器の設定をもとに、パケットが分割されます。多くの場合、使われるプロトコルはイーサネットです。イーサネットのデフォルトのMTUは1500、つまりイーサネットネットワーク上で送信できるパケットの最大サイズは1500バイトということになります（イーサネットヘッダ分の14バイトは除く）。

注　MTUには標準設定がありますが、機器のMTUは手動で再設定できます。MTU設定はインターフェースごとに定められており、WindowsおよびLinux搭載システム、またルータのインターフェースでも変更できます。

通信機器はIPパケットの送信準備をするとき、ネットワークインターフェースのMTUサイズとパケットのデータサイズを比較して、パケットを分割すべきかどうか判断します。データサイズがMTUより大きければ、パケットは分割されます。パケットの断片化は次のようなステップで行われます。

1．データ送信を成功させるため、データをいくつかのパケットへと分けます。

2．分割されたそれぞれのデータのサイズに合わせ、IPヘッダの全長フィールドをセットします。

3．最終パケットを除いたすべてのパケットのMF（More Fragments）フラグを1にセットします。

4．断片のIPヘッダの断片位置フィールドをセットします。

5．パケットが送信されます。

アドレス192.168.0.128の通信機器へpingリクエストを送っているアドレス10.10.0.3のコンピュータから取られたのがip\_frag\_source.pcapファイルです。パケット一覧のペインのInfoカラムに分割されたIPパケットが2つあり、その後にICMP（ping）リクエストが続いているのが分かります。

パケット1のIPヘッダを調べることから始めましょう（図6-13）。

MFおよび断片位置フィールドから、このパケットが分割されたものの一部であるのが分かります。分割されたパケットは、断片位置の値がプラスになるか、MFフラグがセットされています。最初のパケットではMFフラグがセットされているので①、受信側はこのあと続いて別のパケットを受信することになります。断片位置が0にセットされている場合②、分割されたパケットの最初のパケットであることを意味しています。

図6-13　MFと断片位置値が断片化パケットであることを示している

2番目のパケットのIPヘッダ（図6-14）でもMFフラグがセットされていますが①、断片位置値は1480です②。これは1500バイトのMTUから、IPヘッダの20バイトを差し引いた数字です。

図6-14　断片位置値はパケットのサイズによって増える

3番目のパケット（図6-15）ではMFフラグはセットされていないので①、これは最終パケットであることを示しており、断片位置値は1480＋（1500‐20）の2960にセットされています②。これらの分割されたパケットは、IPヘッダの識別子フィールドの値が等しいことから③、いずれも同じパケットの一部であると分かります。

図6-15　MFがセットされていないので最終パケットである

TCP

TCP（Transmission Control Protocol）の最終目標は、データを確実に配送することです。RFC 793で定義されているTCPは、OSI参照モデルのレイヤ4で動作します。TCPはデータシーケンスとエラーリカバリを処理し、データが到着すべき場所に確実に届くようにします。よく使われるアプリケーション層プロトコルの多くが、TCPとIPによってパケットを宛先に配送しています。

TCPヘッダ

TCPのヘッダの複雑さをみれば、どれほど多くの機能を提供しているかがわかります。図6-16に示したように、TCPヘッダのフィールドは次のようになっています。

送信元ポート番号　パケットの送信に用いられるポート。

宛先ポート番号　パケットが送信される先のポート番号。

シーケンス番号　TCPセグメントを識別するための番号。このフィールドはデータストリームの部分の損失を防ぐために使われます。

確認応答番号（ACK番号）　通信の相手側から受け取る次のパケットに含まれているシーケンス番号。

フラグ　送信されたTCPパケットのタイプを識別するための、URG、ACK、PSH、RST、SYN、FINフラグ。

ウィンドウサイズ　受信側のバッファサイズをバイト単位で知らせます。

チェックサム　TCPヘッダとデータの内容が、到着時に正確かどうかを確認するのに用いられます。

緊急ポインタ　URGフラグをセットした場合、パケット内のデータをCPUがどこから読むべきかの指示を知るのに、このフィールドをチェックします。

オプション　TCPパケットで指定できるさまざまなオプションフィールド。

図6-16　TCPヘッダ

TCP

ビットオフセット

送信元ポート番号

宛先ポート番号

シーケンス番号

確認応答番号

データオフセット

予約済み

フラグ

ウィンドウサイズ

チェックサム

緊急ポインタ

オプション

TCPポート

すべてのTCP通信は送信元と宛先のポートを使って行われ、そのことはTCPヘッダから分かります。ポートとは、旧式の配電盤のジャックのようなものです。配電盤のオペレータは、配電盤のライトとプラグを監視しています。ライトが光ると、電話をかけてきた相手とつなぎ、誰に電話をかけたいのかを尋ね、ケーブルのプラグを入れて相手につなぎます。すべての通話には送信元ポート（電話のかけ手）と宛先ポート（受け手）が必要です。TCPポートもほぼ同じように機能します。

リモートサーバまたは通信機器上の特定のアプリケーションにデータを送信するには、TCPパケットは待ち受けしているリモートサービスのポートを知っていなければなりません。設定されたポート以外のポートにアクセスしようとすると、通信が途絶えてしまいます。

送信元ポートはそれほど重要ではなく、ランダムに選ぶことができます。リモートサーバは、送信されるオリジナルのパケットから、送信するポートを決定します（図6-17）。

図6-17　TCPはポートを使ってデータを送信する

クライアント　Webサーバ　80番ポートで待ち受け

送信元1024番ポート／宛先80番ポート

送信元80番ポート／宛先1024番ポート

クライアント　Eメールサーバ　25番ポートで待ち受け

送信元3221番ポート／宛先25番ポート

送信元25番ポート／宛先3221番ポート

TCPとの通信に使えるポートは6万5535個あり、一般に2つのグループに分けることができます。

・1から1023までがスタンダードポートグループです（0は予約済みなので除外します）。特定のサービスが、一般に標準ポートグループに入る標準ポートを使っています。

・1024から65535までがエフェメラルポートグループです（ただしOSによってはこの定義が異なります）。どんなときでも、1個のポートで通信できるのは1つのサービスだけなので、最近のOSは、通信を独自のものにするために、送信元ポートをランダムに選択します。これらの送信元ポートは、エフェメラルポートの範囲に入ります。

tcp\_ports.pcapファイルを開いて、パケットが使っているポート番号を識別してみましょう。このファイルには、2つのWebサイトをブラウズしているクライアントのHTTP通信があります。前述したように、HTTPは通信にTCPを使用するので、これは標準TCPトラフィックのいい例となります。

このファイルの最初のパケットでは（図6-18）、最初の2つの値がパケットの送信元ポートと宛先ポートを表しています。このパケットは、172.16.16.128から212.58.226.142へ送信されるものです。送信元ポートは2826①で、エフェメラルポートです（送信元ポートはランダムな選択からインクリメントしますが、OSがランダムに選択するということを思い出してください）。宛先ポートは標準ポートの80番ポートで②、この標準ポートはHTTPを使用するWebサービスが利用するものです。

図6-18　TCPヘッダに送信元ポートと宛先ポートが含まれている

Wiresharkではこれらのポートにslc-systemlog(2826)とhttp(80)のラベルがつけられており、ポートのリストと一般的な用途が管理されています。これらのポートは主にスタンダードポートですが、エフェメラルポートの多くも関連サービスで使われています。ポートのラベルは非常に分かりにくいため、名前解決を無効にしておきましょう。これを行うには、[Edit]から[Preferences]、さらに[Name Resolusion]を選択し、[Enable Transport Name Resolution]の横のチェックを外します。この機能は有効のままにし、特定ポートを識別する方法だけを変更したい場合は、Wiresharkプログラムディレクトリにある「Services」ファイルを修正します。ディレクトリはIANA（Internet Assigned Numbers Authority）の一般的なポート一覧を元にしています。

2番目のパケットは、212.58.226.142から172.16.16.128へと送り返されたものです（図6-19）。IPアドレス同様、送信元と宛先ポートが入れ替わっています①。

図6-19　返信では送信元と宛先のポート番号が入れ替わる

すべてのTCP通信が、ランダムに選ばれた送信元ポートが既知の宛先ポートと通信するという形で行われます。最初のパケットが送られると、リモートの通信機器が確立されたポートを使って送信元と通信します。

このサンプルのキャプチャファイルには、もうひとつ通信ストリームが含まれています。通信に使用されたポート番号を探してみましょう。

注　本書を読み進めるに従い、一般的なプロトコルとサービスに関するポートについての知識がさらに深まるでしょう。そのうちにポートを見れば、使われているサービスと通信機器が分かるようになります。一般的なポートの完全な一覧については、http://www .iana.org/assignments/port-numbers/を参照してください。

3ウェイハンドシェイク

TCPを使用する通信は、必ず2つのホストのハンドシェイクで始まります。ハンドシェイクのプロセスには、いくつかの異なる目的があります。

・送信元のホストが、宛先のホストと確実に通信できるようにします。

・送信元のホストが、通信を行おうとしているポートを確認できるようにします。

・送信元のホストが受信者に向けてシーケンス番号の送信を開始し、両ホストがパケットの順序を適切に保てるようにします。

図6-20に示したように、TCPハンドシェイクには3つのステップがあります。最初のステップで、通信を行いたい機器（ホストA）は宛先（ホストB）にTCPパケットを送ります。この最初のパケットには、下層プロトコルヘッダ以外のデータは含まれていません。このパケットのTCPヘッダにはSYNフラグがセットされており、通信プロセスに使われる最初のシーケンス番号とMSS（Maximum Segment Size）が含まれています。ホストBはこのパケットに応答し、SYNフラグとACKフラグがセットされ、最初のシーケンス番号を含んだ同様のパケットを送り返します。最後にホストAは、ACKフラグのみがセットされた最終パケットをホストBに送ります。このプロセスが完了すると、通信開始に必要なすべての情報を、両ホストが持つことになります。

図6-20　 3ウェイハンドシェイク

ホストA　ホストB

注　TCPパケットは、セットされたフラグの名前で呼ばれることがよくあります。たとえばSYNフラグがセットされたパケットなら、SYNパケットと呼びます。つまりTCPハンドシェイクプロセスで使われるパケットは、SYNパケット、SYN/ACKパケット、ACKパケットと呼ばれます。

tcp\_handshake.pcaを開き、このプロセスを実際に見てみましょう。Wiresharkには、TCPパケットのシーケンス番号を、解析しやすい番号に置き換える機能があります。実際のシーケンス番号を見るため、ここではこの機能を無効にしておきます。無効にするには、[Edit]から[Preferences]を選択、[Protocols]を広げ[TCP]を選びます。次に[Relative Sequence Numbers]と[Window Scaling]の横のボックスのチェックを外し、[OK]をクリックします。

最初のパケットはSYNパケットです（図6-21）。このパケットは172.16.16.128の2826番ポートから、212.58.226.142の80番ポートへと送られています。シーケンス番号は3691127924であることが分かります①。

図6-21　最初のSYNパケット

2番目のパケットは、212.58.226.142からの応答となるSYN/ACKパケットです。このパケットには、最初のシーケンス番号（233779340）①と、応答確認番号（3691127925）②が含まれています。応答確認番号は、送信元から送られてきたシーケンス番号に1を加えたものになっています。受信者側が次に受け取るシーケンス番号の指定に使われるのがこのフィールドだからです。

図6-22　SYN/ACK応答パケット

最後のパケットは、172.16.16.128から送られるACKパケットです（図6-23）。このパケットには、先程のパケットの応答確認番号フィールドにあったシーケンス番号3691127925①が含まれています。

図6-23　最終のACKパケット

TCP通信の前には必ずハンドシェイクが起こります。中身のごちゃごちゃしたキャプチャファイルで通信の始まりを検索する場合、SYN-SYN/ACK-ACKのシーケンスが目印となります。

TCPティアダウン

どんな出会いにも別れはつきものですが、TCPハンドシェイクにはティアダウンがあります。TCPティアダウンは、通信を終えた2つの通信機器間の接続を終わらせるのに用いられるものです。このプロセスには4個のパケットが関わり、接続の終了にはFINフラグが使われます。

ティアダウンでは、ホストAはホストBに、FINフラグとACKフラグがセットされたTCPパケットを送り、通信終了を知らせます。ホストBはACKパケットで応答し、自分のFIN/ACKパケットも送ります。これに対しホストAはACKパケットで応答、通信プロセスが終了します。このプロセスを図式化したのが図6-24です。

図6-24　TCPティアダウンのプロセス

ホストA　ホストB

tcp\_teardown.pcapファイルを開いて、このプロセスを見てみましょう。67.228.110.120のホストが、FIN/ACKフラグがセットされたパケット①を送って、ティアダウンのプロセスを開始しているのがわかります（図6-25）。

図6-25　FIN/ACKがティアダウンのプロセスを開始

このパケットを受け取ると、172.16.16.128は受信したことを知らせるACKパケットで応答し、FIN/ACKパケットを送ります。67.228.110.120が最後のACKを送信すると、プロセスが完了します。ここで2つのホスト間の通信は終了するので、通信を再開するには新たにTCPハンドシェイクを行わなければなりません。

TCPリセット

すべての接続がTCPティアダウンで終了するというのが理想です。しかし実際には、接続の突然の終了が頻繁に起こります。たとえば、攻撃者によるポートスキャンの実行や、単なるホストの設定ミスからも起こりえます。こうした場合、RSTフラグをセットしたTCPパケットが使われます。RSTフラグは接続の突然の終了や、接続の試みが拒否されたことを示すためのものです。

tcp\_refuseconnection.pcapファイルは、RSTパケットを含んだネットワークトラフィックの一例です。最初のパケットは、192.168.100.1と80番ポートで通信しようとしている、192.168.100.138のホストから送られたものです。しかしこのホストは、80番ポートが待ち受け状態になっていないのを知りません。Ciscoルータの場合、Webインターフェースが設定されていないと、80番ポートで接続を待ち受けているサービスは存在しないからです。この通信の試みに対し、192.168.100.1は192.168.100.138にパケットを送り、80番ポートでは通信が行えないと知らせます。図6-26は、通信の試みが突如終了したことを示す、2番目のパケットのTCPヘッダです。このRSTパケットにはRSTフラグとACKフラグ以外には何も含まれておらず①、この後の通信はありません。

RSTパケットは、この例のように通信の試みの最初、または通信の途中で送られ、通信を終了します。

図2-26　RSTフラグとACKフラグが通信終了を知らせる

UDP

UDP（User Datagram Protocol）は、ネットワークでよく使われるもう1つのレイヤ4プロトコルです。TCPがエラー検出機能により信頼性の高いデータ転送を目的に設計されているのに対し、UDPは高速な転送を目的としています。このためUDPは、「コネクションレスプロトコル」と呼ばれます。コネクションレスプロトコルは、TCPのハンドシェイクやティアダウンのように、正式な接続の確立や終了を行いません。

信頼性の高いデータ送信を提供しないUDPの場合、誤りがあって当然のように思えます。確かにそうなのですが、実際はUDPを使うプロトコルは、信頼性を保証するための独自のサービスを内蔵していたり、信頼性の高い接続を実現するためにICMP機能を使ったりしています。たとえばパケットの高速送信を重視する、アプリケーション層プロトコルのDNSとDHCPなどは、UDPをトランスポート層プロトコルとして利用しますが、エラー検出と再送信タイマーは自分のものを使っています。

UDPヘッダ

UDPヘッダは、TCPヘッダよりもはるかに小さく、シンプルです。図6-27に示したように、UDPヘッダのフィールドは次のようになっています。

送信元ポート　パケット送信に使われるポート

宛先ポート　パケットが送信されるポート

パケット長　パケットの長さをバイトで示したもの

チェックサム　受信時のUDPヘッダの内容確認に使われる

図6-27　UDPヘッダ

UDP

ビットオフセット

送信元ポート　宛先ポート

パケット長　チェックサム

udp\_dnsrequest.pcapファイル内のパケットは1つです。このパケットはUDPを使ったDNSリクエストを表しています。UDPヘッダを開くと、4つのフィールドがあるのがわかります（図6-28）。

図6-28　UDPパケットの中身は非常にシンプル

UDPはデータ配信の信頼性を保証しないことを覚えておいてください。したがってUDPを使うアプリケーションは、信頼性が必要な場合、別の手段を取る必要があります。

ICMP

ICMP（Internet Control Message Protocol）は、TCP/IPが動作するための補助的な役割を果たすプロトコルで、TCP/IPネットワーク上の通信機器やサービスなどが使用可能かどうかに関する情報を提供します。ネットワークのトラブルシューティングのほとんどが、ICMPメッセージタイプを使っています。ICMPはRFC 792で定義されています。

ICMPヘッダ

ICMPはIPの一部であり、メッセージの送信にIPを使っています。ICMPヘッダは比較的小さく、目的によって変わります。図6-29に示したように、ICMPヘッダのフィールドは次の通りです。

タイプ　RFC仕様に基づく、ICMPメッセージのタイプまたは分類

コード　RFC仕様に基づく、ICMPメッセージの下位分類

チェックサム　受信時にICMPヘッダとデータの内容を確認するためのもの

データ　タイプとコードのフィールドによって変わる部分

図6-29　ICMPヘッダ

ICMP

ビットオフセット

タイプ

コード

チェックサム

データ

ICMPタイプとメッセージ

先述したように、ICMPパケット構造はその目的によって変わり、「タイプ」と「コード」フィールドの値によって決まります。

タイプをパケットの分類、コードを下位分類と考えるといいかも知れません。たとえばタイプの値が3なら、「宛先到達不可能」を意味します。この情報だけでは問題解決には不十分ですが、コードの値が3、つまり「ポート到達不能」と分かれば、通信を試みているポートに問題があると結論づけることができます。

注　ICMPのタイプとコードの完全なリストについては、<http://www.iana.org/assignments/icmp-parameters>を参照してください。

エコー要求とエコー応答

ICMPでもっともよく知られているのが、接続確認に使われるpingコマンドです。情報技術の専門家なら、通常pingに馴染んでいるはずです。

Pingを使うには、コマンドプロンプトでping<相手のIPアドレス>と入力します。相手のコンピュータがオンになっていて、自分のコンピュータと接続しており、ファイアウォールでブロックされていなければ、pingコマンドに対する応答が見られるはずです。

図6-30では、応答を4回受け取ることに成功しており、そのサイズとRTT、TTLが表示されています。Windowsのユーティリティも、パケットがいくつ送信、受信、消失したかの詳細を提供しています。通信に失敗した場合には、その理由を説明したメッセージが表示されます。

図6-30　接続確認に用いられるpingコマンド

基本的には図6-31のように、pingコマンドは相手先にパケットを送って応答を待ち、接続しているかどうかを判断します。

図6-31　pingコマンドは2ステップのみ

エコー／ping要求　エコー／ping応答

注　pingは長い間ITの必需品となっていますが、ホストベースのファイアウォールの配備により、その結果は少々あてにならなくなっています。今日のファイアウォールの多くが、ICMPパケットへの応答を制限しているからです。攻撃者がpingを使って、ホストへのアクセスを確認する行為が阻止されるため、セキュリティの観点からいえば素晴らしいのですが、同時にトラブルシューティングをも難しくしています。相手と通信ができるとわかっているのに、接続確認のpingを送っても応答が得られないというのは、イライラさせられるものです。

シンプルなICMP通信の一例として、実際にpingコマンドがどう機能するかを見てみましょう。icmp\_echo.pcapファイルのパケットは、pingを実行すると何が起こるかを示しています。

最初のパケット（図6-32）は、ホスト192.168.100.138が、192.168.100.1にパケットを送信していることを示しています①。このパケットのICMP部分を開き、タイプとコードのフィールドを見れば、このパケットの種類が分かります。この例ではパケットのタイプは8②、コードは0③なので、エコー要求であることを示しています（Wiresharkではタイプとコードについての説明が表示されるはずです）。エコー要求（ping）は前半部分です。これはIPを使うシンプルなICMPパケットで、少しのデータしか含んでいません。ICMPパケットには、タイプ、コード、チェックサムに加え、要求と応答を一組にするために使われるシーケンス番号と、データ部分にはランダムなテキストストリングが含まれています。

注　エコー、またpingという用語はしばしばほぼ同じ意味で使われますが、pingはツールの名称であることを覚えておきましょう。PingツールはICMPエコー要求パケットを送るために使われます。

図6-32　ICMPエコー要求パケット

2番目のパケットは、要求に対する応答です（図6-33）。このパケットのICMP部分のタイプは0①、コードは0②で、これがエコー応答であることを示しています。シーケンス番号が最初のパケットと一致しているので③、このエコー応答が先のパケットのエコー要求と一致していることがわかります。またこの応答パケットには、要求と一緒に送られたものと同じ32バイトのストリングが含まれています④。192.168.100.138が2番目のパケットを受け取ると、pingがそれを報告します（図6-30）。

図6-33　ICMPエコー応答パケット

Pingのデータ部分を使ってデータのサイズを増やし、パケットを分割させて、ネットワークのさまざまなトラブルシューティングに利用できることを覚えておきましょう。断片の小ささが求められるトラブルシューティングの際には、この作業が必要となります。

注　ICMPエコー要求に使われるランダムなテキストは、攻撃者に狙われる可能性があります。攻撃者はこの情報を使って、標的とするシステムの搭載OSを突き止めることができるからです。また攻撃者が秘密通信の手法として、このフィールドに小さなデータを埋め込むことも可能です。

トレースルート

トレースルートは、あるホストから別のホストまでの経路を調べるのに利用されます。単純なネットワークであれば、経路上にはルータが1個のみ、あるいはひとつもないかも知れません。しかし複雑なネットワークになると、パケットは最終目的地へ到達するまでに、何十個ものルータを通過しなければならないので、問題が発生した場合にそれを解決できるよう、パケットの経路を正確に追跡することが重要となります。

ICMPを使い（少しだけIPの助けを借りて）、トレースルートはパケットがたどる経路をマッピングします。たとえばicmp\_traceroute.pcapファイルの最初のパケットは、先ほど見たエコー要求とかなりよく似ています（図6-34）。

図6-34　TTL値が1のICMPエコー要求パケット

まずこのパケットは、192.168.100.138から4.2.2.1②への、単純なエコー要求のように見えます①。またパケットのICMP部分は、エコー要求パケットの形式とまったく同じです。しかしながら、このパケットのIPヘッダを開くと、ひとつ値があることに気付くはずです。パケットのTTLは1③、つまり最初のルータに届いたときに、パケットが消失したことを意味しています。宛先の4.2.2.1はインターネットアドレスなので、送信元と宛先の間には最低1台のルータが存在するはずであり、つまりこのパケットは宛先へ届かないことになります。このパケットは最初のルータまでしか到達しないとこのトレースルートは伝えているので、今回の場合は届かなくても問題ありません。

2番目のパケットは予想通り、最初のルータからの応答です（図6-35）。パケットは192.168.100.1のホストに到達し、TTLは0に減っているので、パケットはこれ以上転送されないことになり、ルータはICMPレスポンスを返します。このパケットのタイプは11①、コードは0②で、転送の途中でTTLが時間切れになってしまったために、宛先に到達しなかったことを意味しています。

図6-35　経路の最初のルータからのICMPレスポンス

このICMPパケットは、ICMP部分の最後にIPヘッダのコピー③と、最初のエコー要求で送られたICMPデータ④を含んでいるため、「ダブルヘッドパケット」と呼ばれる場合があります。この情報はトラブルシューティングに非常に役立ちます。

TTL値を増やしてパケットを送信するこのプロセスは、7番目のパケットが到達するまであと2回繰り返されます。ここでも最初のパケット同様のことが行われますが、今回はIPヘッダのTTL値が2にセットされているので、パケットは2番目のルータまで到達することになります。予想通り、12.180.241.1のルータから、同じように宛先到達不可能のメッセージとTTL切れのメッセージが送られてきます。最終的な宛先である4.2.2.1に到達するまで、TTL値が１ずつ増えるプロセスが続きます。

まとめると、トレースルートプロセスとは、経路にある各ルータとやり取りし、宛先までの経路図を作成するものです。図6-36がこの経路図です。

注　ここではICMPのみを使うWindowsのトレースルートについて説明しています。Linuxのトレースルートユーティリティにはもう少し汎用性があり、経路情報の取得に他のプロトコルを利用することができます。

図6-36　トレースルートユーティリティからのサンプル出力

本書を通じて見ていきますが、ICMPには多種多様な機能があります。解析にはICMPをさらに頻繁に用いることになります。

本章では、パケット解析のプロセスで研究するもっとも重要なプロトコルの一部を紹介しました。IP、TCP、UDP、ICMPはネットワーク通信の基礎であり、日々の作業において必須のものです。次章では、アプリケーション層の一般的なプロトコルについて見ていきます。