

実践 パケット解析

Wiresharkを使ったトラブルシューティング

Chris Sanders 著

園田 道夫 監訳

一瀬 小夜 訳

 O'REILLY®
オライリー・ジャパン

本書で使用するシステム名、製品名は、それぞれ各社の商標、または登録商標です。
本文中では™、®、©マークは省略しています。

PRACTICAL PACKET ANALYSIS

**Using Wireshark to Solve
Real-World Network
Problems**

by Chris Sanders



**no starch
press**

San Francisco

Copyright ©2007 by Chris Sanders.

Title of English-language original: Practical Packet Analysis, ISBN 978-1-59327-149-7.

Japanese-language edition copyright ©2008 by O'Reilly Japan, Inc. All rights reserved.

Japanese translation rights arranged with No Starch Press, Inc., San Francisco, California through Tuttle-Mori Agency, Inc., Tokyo

本書は、株式会社オライリー・ジャパンがNo Starch Press, Inc.の許諾に基づき翻訳したものです。日本語版についての権利は、株式会社オライリー・ジャパンが保有します。

日本語版の内容について、株式会社オライリー・ジャパンは、最大限の努力をもって正確を期していますが、本書の内容に基づく運用結果については責任を負いかねますので、ご了承ください。

本書を愛する両親と神にささげたい。彼らの協力なくしては本書は日の目を見なかっただろう。

「徹底的に神様に頼りなさい。

絶対に自分を頼ってはダメです。

何をするにも、神様を第一にきなさい。

神様がどうすればよいか教えてください。」

——箴言 3:5,6 (<http://www.ibs.org/bibles/japanese/pdf/ot/proverbs.pdf>から引用)

監訳者まえがき

仕事柄出張が多いんですが、最近の宿は無料でLAN使い放題というところが多くなっています。しかし、そのLAN、使うの不安じゃありませんか？

そういうとき、私は必ずパケットキャプチャして怪しい通信パケットが届くかどうか確かめます。パケットを見ていると「怪しい」ネットワークというのはなんとなく分かりますね。

基本にあるのはパケットをキャプチャしてみる、ということと、それを読み解く、ということですね。ちょっとした知識があれば解析などは簡単にできますし、探求心があればその中身を深掘りして怪しさの根源を突き止めることも可能です。いろいろなネットワークでパケットを見ていると、そのうち「怪しい」という感覚がついてくるものですが、本書ではそのとっかかりにある考え方を、Wiresharkというすばらしいソフトウェアを通して学んでいきます。

このソフトウェアは、これがまったくのフリー（無料）で配布されているというのが信じがたいほどよくできています。何しろ、古今東西のパケット好きが文字どおりよってたかって改良してきたので、良いものにならないはずがないのです。その多彩な機能に触れて、現場で使えば使うほど良さが分かってくることでしょう。

ネットワークの勉強なんて、机上でやっているだけだとちっとも面白くないものなのですが、パケットを直に見てその動きを追いかけていると、不思議に楽しくなってくるものです。まあとにかく試してみてください。きっと病みつきになりますよ。

ひとつだけご注意。他人のプライバシーは尊重しましょう。

2007年12月4日

名古屋にてホテルLANのパケットを覗きながら

（あ、もちろん、自分のところに来ているものだけです）

園田 道夫

まえがき

筆者が最初にコンピュータを手にしたのは9歳のときです。コンピュータは1年くらいで壊れてしまいました。もともと家族はコンピュータが買えるほど裕福ではなく、お金を出して修理することは経済的に不可能でした。そのため、筆者は勉強してコンピュータを自力で修理するようになりました。これが、技術に対する興味を持つきっかけになったのです。

高校と大学でその興味は熱意に変わり、その熱意が育つにつれ能力も育ち、自然とネットワークやコンピュータの問題について勉強するようになりました。そして、Wiresharkプロジェクト（当時はEtherealと呼ばれていた）と出会ったのです。このソフトウェアは筆者にまったく新しい世界を見せてくれました。問題を解決する新しい方法が得られ、プロトコルを読み解く技術を身につけたことで、コンピュータやネットワークのトラブルシューティング能力は驚くほど上がりました。

パケット解析のすばらしいところは、それがポピュラーな方法になりつつあり、ネットワークをさらに学ぶことができるということです。本書の知識が仕事をする上で必須となっているのは、ユーザーグループ、Wiki、ブログのおかげでもあります。パケット解析は今日のネットワークを管理するために必要な知識であり、本書はそれがどのように機能しているのかを学ぶ足がかりとなるでしょう。

なぜこの本なのか？

なぜ、パケット解析を解説しているほかの本ではなく、本書を買うべきなのかと疑問に思うでしょう。答えはタイトルにあります。「実践パケット解析」（原著名は“*Practical Packet Analysis*”）。つまり、本書では現実世界で対面する問題とよく似たシナリオを通して、実践的なパケット解析を学ぶことができます。本書の前半では、パケット解析とWiresharkを理解するための前提知識が得られます。後半では、日々のネットワーク管理で遭遇する問題を実践的なシナリオに沿って解決していきます。

ネットワーク技術者、ネットワーク管理者、CIO、コンピュータの技術者、デスクワークする人、いずれであっても本書から多くの知識とパケット解析に関するテクニックを学ぶことができます。

コンセプトとアプローチ

筆者はのんびりした人間なので、コンセプトを語るときもやっぱりのんびり語ることになるでしょう。それは本書の口調にも現れています。技術的なコンセプトを語るときはどうしても技術用語が多くなり、のんびりした口調にはなりません、それでもできる限りカジュアルな説明になるよう努力したつもりです。明確で要領を得て、過不足なく記述しています。

本当にパケット解析を学びたいと思っているのなら、本書の前半で紹介されているコンセプトを理解する必要があります。後半部分を理解するために不可欠だからです。あなたの職場で起こる問題とまったく同じシナリオを読むことはできませんが、本書でコンセプトを学んでおかなければ実際の問題に遭遇したときに解決することは難しいでしょう。

以下は本書の各章の簡単な説明です。

1章 パケット解析とネットワークの基礎

パケット解析とはなんでしょう？ どのように機能するのでしょうか？ どうやってやるのでしょうか？ この章ではネットワークの通信とパケット解析の非常に基礎的な部分を学びます。

2章 ケーブルにもぐりこむ

この章では、ネットワーク上にどうスニッファを配置するのかについて述べています。

3章 Wireshark概要

Wiresharkはどこで入手可能か、どうやって使うのか、何ができるのか、なぜすぐれているのかなどWiresharkの基本について述べています。

4章 Wiresharkでのパケットキャプチャのテクニック

Wiresharkを使ったパケットキャプチャの基礎を学びます。

5章 Wiresharkの高度な機能

基本を身につけたら、高度な機能も使ってみましょう。この章では高度な機能を掘り下げ、普段は目立たない機能に焦点を当てます。

6章 一般的なプロトコル

この章では、もっともよく使われるプロトコルが、パケットレベルではどう見えるかを紹介します。各プロトコルで起こる問題を理解するために、それらがどう機能するかをここで学んでおきましょう。

7章 ケーススタディ（基礎編）

この章から、現実世界で起こる問題をもとにしたシナリオが出てきます。各シナリオは読み進めやすい形式で書いてあり、そのシナリオの問題、問題の解析、そして解決策が盛り込まれています。これらの基本的なシナリオでは数台のコンピュータしか扱っていないので、パケット解析を始めるのにちょうどよい難易度になっています。

8章 ケーススタディ（ネットワークの遅延と戦う）

ネットワークの問題でもっとも多いのは、ネットワークの遅延です。この章ではネットワークの遅延に関する問題に焦点を当てます。

9章 ケーススタディ (セキュリティ解析)

ネットワーク管理者にとって、ネットワークセキュリティはもっとも関心の高い話題でしょう。この章では、パケット解析によってセキュリティ問題を解決する方法を学びます。

10章 無線LANのスニффイング

実践についての最後の章は、無線LANでのスニッフイングの入門です。この章では、有線LANと無線LANのパケット解析の違いについて理解した上、簡単なシナリオを通して、それまでに学んだことを無線LANでどう使うのか学びます。

11章 推薦文献

最後の章には、本書を読んだ後もパケット解析の勉強を続けるために便利なツールやWebサイトを集めました。

付録 Winnyやボットのパケット解析

巻末付録は、監訳者による日本語版オリジナルの書き下ろしです。本書の原著では扱われていないWinnyやボット関連のトラブルシューティングについて学びます。

本書の使い方

本書には2つの使い方があると筆者は思っています。1つは、1つ1つの章を読んでゆき、パケット解析の理解を深めるためのテキストとして使う方法。現実世界の問題に沿ったシナリオが掲載されている後半部分に特に重点を置きます。もう1つの使い道は、本書をリファレンスとして使う方法。頻繁に使用するわけではないWiresharkの機能をいちいち覚えておく必要はありません。そういった機能を使うときのリファレンスとして本書が本棚に置いてあれば役に立つことでしょう。

サンプルファイルについて

本書で使われているキャプチャファイルはすべて、本書日本語版のWebページ (<http://www.oreilly.co.jp/books/9784873113517/>) から入手可能です。本書を最大限利用するために、これらのサンプルファイルをダウンロードしてそれと一緒に本書を読むことをお勧めします。

いくつかのサンプルファイルはPacket Analysis InstituteやWireshark Universityに参加しているLaura Chappellの手で作成された物です。サンプルファイルのリストは以下のとおりです[†]。

5章

- filedownload.dmp
- ftp-netbios3.pcap
- suspectemployeechat.dmp

[†] 監訳注：日本語版オリジナルの付録では、サンプルのパケットキャプチャファイルを用意していません。というのも、Winny通信の場合は通信相手が個人ですし、ボット通信の場合はボットやボットがダウンロードしてきたモジュール本体が含まれてしまうためです。

6章

- arp.pcap
- dhcp.pcap
- dns.pcap
- ftp.pcap
- http.pcap
- icmp.pcap
- msnms.pcap
- telnet.pcap

7章

- barryscomputer.pcap
- bethscomputer.pcap
- destunreachable.pcap
- evilprogram.pcap
- ftpclientdenied.pcap
- ftpserverdenied.pcap
- hauntedbrowser.pcap
- http-fault-post.pcap
- ipfragments.pcap
- slowdownload.pcap
- tcp-con-lost.pcap

8章

- double-vision.pcap
- email-troubles.pcap
- gnutella.pcap
- http-client-refuse.pcap
- icmp-tracert-slow.pcap
- slowdownload.pcap
- torrential-slowness.pcap
- ftp-uploadfailed.pcap

9章

- blaster.pcap
- covertinfo.pcap
- ftp-crack.pcap
- hackersview.pcap
- osfingerprinting.pcap
- portscan.pcap
- printerproblem.pcap

- dosattack.pcap

10章

- 80211traffic.pcap
- FailedWepAuth.pcap
- SuccessfulWepAuth.pcap

本書の表記

本書では、以下の表記を使用しています。

太字 (Bold)

重要な用語を示します。

等幅 (Constant Width)

サンプルコード、コマンド、変数、属性、関数、クラス、名前空間、メソッド、モジュール、値、ファイルの内容、コマンドの出力などを示します。

等幅の太字 (Constant Width Bold)

コードの重要な部分と、そのとおりに打ち込まなければならないコマンドやテキストを示します。



このアイコンとともに記載されている内容は、ヒント、提案、または一般的な注意事項を表します。

意見と質問

本書（日本語翻訳版）の内容については、最大限の努力をもって検証および確認していますが、誤りや不正確な点、誤解や混乱を招くような表現、単純な誤植に気づかれることもあるでしょう。本書を読んで気づいたことは、今後の版で改善できるように知らせていただければ幸いです。将来の改訂に関する提案なども歓迎します。

株式会社オライリー・ジャパン

〒160-0002 東京都新宿区坂町26番地27 インテリジェントプラザビル1F

電話 03-3356-5227

FAX 03-3356-5261

電子メール japan@oreilly.co.jp

本書に関する技術的な質問や意見については、次の宛先に電子メール（英文）を送ってください。

info@nostarch.com

本書のWebページには、正誤表、サンプルコード、追加情報が掲載されています。以下のアドレスでアクセスできます。

<http://www.oreilly.co.jp/books/9784873113517/>

<http://www.oreilly.com/catalog/9781593271497/> (原書)

<http://www.nostarch.com/packet.htm> (原書)

<http://www.chrissanders.org/PPA/> (原著者)

オライリーに関するその他の情報については、次のオライリーのWebサイトを参照してください。

<http://www.oreilly.co.jp>

謝辞

何よりもまず、このプロジェクトを終了させるための力と忍耐を与えてくれた神に感謝したい。彼はTodoリストが果てしなく長くなり、ストレスで押しつぶされそうになったときにいつも私を助けてくれました。

Bill、Tyler、Christina、そのほかの「No Starch Press」のチームメンバー、私にこの本を書く機会を与え、かつそれを私のやり方で自由にやらせてくれてありがとう。Gerald CombsにはWiresharkプログラムのメンテナンスを続けるその熱意と、技術的な編集をしてくれたことに感謝しています。特にお礼を言いたいのはLaura Chappellで、彼女はパケット解析のトレーニング用のデータを提供してくれました。この本でもいくつかのキャプチャファイルを使っています。

個人的なところでは、Tina Nance、Eddy Wright、Paul Flethcerに感謝したい。彼らは本書を出版するという、私のキャリアの中でもっとも大きな偉業を成し遂げる手助けをしてくれました。彼らは信頼の置ける相談相手であり、偉大な友人です。同じく、本書の執筆に力を貸してくれた我慢強い友人がいます。Mandy、Barry、Beth、Chad、Jeff、Sarah、そしてBrandon、本当にありがとう。あなたたちがいなかったら本書を完成させることはできなかったでしょう。

でも、私の感謝のほとんどは愛する両親、KennethとJudy Sandersにささげたい。お父さん、あなたはコンピュータを触ったことは一度もないけど、私が何かできるのもすべてお父さんのおかげです。あなたの私を誇りに思うという言葉以上に、私にやる気を起こさせるものはありません。お母さん、この本を書く5年前に私たちの前から去ってしまい、この本を見ることはできないけど、あなたはいつも私の心の中にいて私を励ましてくれました。あなたが見せてくれた生きることに対する情熱は、私の行動すべてに息づいています。この本は、あなたが作ってくれたも同然です。

目次

監訳者まえがき	vii
まえがき	ix
1 章 パケット解析とネットワークの基礎	1
1.1 パケット解析とは?	1
1.2 パケットスニッファの評価	2
1.2.1 サポートされているプロトコル	2
1.2.2 ユーザーフレンドリかどうか	2
1.2.3 コスト	2
1.2.4 スニッファのサポート体制	3
1.2.5 OSのサポート	3
1.3 パケットスニッファの仕組み	3
1.3.1 収集	3
1.3.2 変換	3
1.3.3 解析	3
1.4 コンピュータはどのように通信するのか	4
1.4.1 ネットワークプロトコル	4
1.4.2 OSI参照モデル	5
1.4.3 プロトコルの相互作用	7
1.4.4 データのカプセル化	8
1.4.5 プロトコルデータユニット	8
1.4.6 ネットワークハードウェア	9
1.4.7 トラフィックの分類	14

2章 ケーブルにもぐりこむ	17
2.1 プロミスクラスモードの使用	18
2.2 ハブで構成されたネットワークでのスニффイング	18
2.3 スイッチで構成されたネットワークでのスニффイング	20
2.3.1 ポートミラーリング	20
2.3.2 ハブの使用	22
2.3.3 ARPキャッシュポイズニング	23
2.3.4 Cain & Abelの使用	24
2.4 ルータで構成されたネットワークでのスニффイング	27
2.5 ネットワーク図	28
3章 Wireshark 概要	29
3.1 Wiresharkの歴史	29
3.2 Wiresharkの利点	29
3.2.1 サポートされているプロトコル	29
3.2.2 ユーザーフレンドリかどうか	30
3.2.3 コスト	30
3.2.4 スニッファのサポート体制	30
3.2.5 OSのサポート	30
3.3 Wiresharkのインストール	30
3.3.1 システム要件	31
3.3.2 Windowsでのインストール	31
3.3.3 Linuxでのインストール	33
3.4 Wiresharkの基本	33
3.4.1 最初のパケットキャプチャ	34
3.4.2 メインウィンドウ	35
3.4.3 設定画面	36
3.4.4 パケットの色分け	37
4章 Wiresharkでのパケットキャプチャのテクニック	41
4.1 パケットの検索とマーキング	41
4.1.1 パケットの検索	41
4.1.2 パケットのマーキング	42
4.2 キャプチャファイルの保存とエクスポート	43
4.2.1 キャプチャファイルの保存	43
4.2.2 キャプチャデータのエクスポート	43
4.3 キャプチャファイルのマージ	44
4.4 パケットの印刷	45

4.5	時間の表示フォーマットと相対時間表示	45
4.5.1	時間の表示フォーマット	45
4.5.2	相対時間表示	46
4.6	キャプチャフィルタとディスプレイフィルタ	47
4.6.1	キャプチャフィルタ	47
4.6.2	ディスプレイフィルタ	47
4.6.3	[Filter Expression] ダイアログ	48
4.6.4	フィルタを自力で作る	49
4.6.5	フィルタの保存	51
5章	Wiresharkの高度な機能	53
5.1	名前解決	53
5.1.1	Wiresharkの名前解決ツール	53
5.1.2	名前解決を有効にする	54
5.1.3	名前解決の欠点	54
5.2	プロトコルの分析	55
5.3	TCPストリームの表示	56
5.4	[Protocol Hierarchy Statistics] ウィンドウ	58
5.5	エンドポイントを見る	58
5.6	ネットワーク上の「対話」	59
5.7	[IO Graphs] ウィンドウ	61
6章	一般的なプロトコル	63
6.1	ARP	63
6.2	DHCP	64
6.3	TCP/IPとHTTP	65
6.3.1	TCP/IP	66
6.3.2	セッションの確立	66
6.3.3	データ送信の開始	67
6.3.4	HTTPの通信	68
6.3.5	セッションの終了	68
6.4	DNS	70
6.5	FTP	71
6.5.1	CWDコマンド	72
6.5.2	SIZEコマンド	72
6.5.3	RETRコマンド	72
6.6	TELNET	73
6.7	MSNメッセージャーサービス	74

6.8	ICMP	76
6.9	まとめ	78
7章	ケーススタディ(基礎編)	79
7.1	TCPの通信障害	79
7.2	届かないパケットとICMPコード	81
7.2.1	宛先到達不可能	81
7.2.2	ポート到達不能	82
7.3	IPフラグメンテーション	82
7.3.1	IPフラグメンテーションを実行するかどうか	83
7.3.2	順番に組み立てる	84
7.4	接続不能	85
7.4.1	分かっていること	85
7.4.2	パケット解析開始	85
7.4.3	解析	86
7.4.4	まとめ	87
7.5	Internet Explorerの悪魔	87
7.5.1	分かっていること	88
7.5.2	パケット解析開始	88
7.5.3	解析	88
7.5.4	まとめ	89
7.6	FTPサーバとの通信	89
7.6.1	分かっていること	90
7.6.2	パケット解析開始	90
7.6.3	解析	90
7.6.4	まとめ	91
7.7	私のせいじゃない!	92
7.7.1	分かっていること	92
7.7.2	パケット解析開始	92
7.7.3	解析	92
7.7.4	まとめ	93
7.8	悪魔のプログラム	94
7.8.1	分かっていること	94
7.8.2	解析開始	94
7.8.3	解析	94
7.8.4	まとめ	99
7.9	考察	99

8章 ケーススタディ(ネットワークの遅延と戦う)	101
8.1 ダウンロードの遅延の原因	101
8.2 ルーティングの不具合	104
8.2.1 分かっていること	104
8.2.2 パケット解析開始	105
8.2.3 解析	106
8.2.4 まとめ	108
8.3 二重に見える	108
8.3.1 分かっていること	108
8.3.2 パケット解析開始	109
8.3.3 解析	109
8.3.4 まとめ	110
8.4 サーバが私を拒否してる?	111
8.4.1 分かっていること	111
8.4.2 パケット解析開始	111
8.4.3 解析	111
8.4.4 まとめ	112
8.5 BitTorrentの大雨	113
8.5.1 分かっていること	113
8.5.2 パケット解析開始	113
8.5.3 解析	113
8.5.4 まとめ	115
8.6 メールサーバに流れ込むPOP	115
8.6.1 分かっていること	116
8.6.2 パケット解析開始	116
8.6.3 解析	116
8.6.4 まとめ	117
8.7 Gnutellaも大雨	117
8.7.1 分かっていること	117
8.7.2 パケット解析開始	118
8.7.3 解析	118
8.7.4 まとめ	121
8.8 考察	121
9章 ケーススタディ(セキュリティ解析)	123
9.1 OSのフィンガープリント	123
9.2 ポートスキャン	124
9.3 プリンタの氾濫	125

9.3.1	分かっていること	125
9.3.2	パケット解析開始	125
9.3.3	解析	125
9.3.4	まとめ	126
9.4	FTPサーバへの侵入	126
9.4.1	分かっていること	126
9.4.2	パケット解析開始	127
9.4.3	解析	127
9.4.4	まとめ	129
9.5	Blasterワーム	129
9.5.1	分かっていること	129
9.5.2	パケット解析開始	129
9.5.3	解析	129
9.5.4	まとめ	131
9.6	隠された情報	131
9.6.1	分かっていること	131
9.6.2	パケット解析開始	131
9.6.3	解析	131
9.6.4	まとめ	132
9.7	ハッカーの視点	132
9.7.1	分かっていること	133
9.7.2	パケット解析開始	133
9.7.3	解析	133
9.7.4	まとめ	135

10章	無線LANのスニффイング	137
10.1	1つのチャンネルをスニッフイング	137
10.2	無線LANのインターフェース	138
10.3	無線LANカードのモード	138
10.4	Windows上での無線LANのスニッフイング	140
10.4.1	AirPcapの設定	140
10.4.2	AirPcapを使ったパケットキャプチャ	141
10.5	Linux上での無線LANのスニッフイング	143
10.6	802.11のパケット	144
10.6.1	802.11のフラグ	145
10.6.2	ビーコンフレーム	146
10.7	無線LAN特有の情報	147
10.8	無線LAN特有のフィルタ	148

10.8.1	特定のBSSIDでフィルタリング	148
10.8.2	無線LANのタイプでフィルタリング	148
10.8.3	特定のデータタイプでフィルタリング	148
10.9	無線LANに接続できない	150
10.9.1	分かっていること	150
10.9.2	パケット解析開始	150
10.9.3	解析	150
10.9.4	まとめ	153
10.10	考察	153
11章 推薦文献		155
あとがき		158
付録 Winnyやボットのパケット解析		159
A.1	その1：Winnyパケットを追え	159
A.1.1	P2P通信Winny	160
A.1.2	Winny通信の解析	161
A.1.3	まとめ	163
A.2	その2：ウイルス、ワーム、ボットの追跡	164
A.2.1	ウイルス、ワームの変質	164
A.2.2	ボット（踏み台）の特徴	165
A.2.3	「怪しい通信」の解析	166
索引		172

1章

パケット解析と ネットワークの基礎

コンピュータネットワーク上では、毎日100万もの問題——単なるスパイウェアの影響から複雑なルータの設定エラーまで——が発生しています。そしてすべての問題を迅速に解決することは不可能です。知識とツールをしっかりと用意することが、最高の準備になるはずですが。すべてのネットワークの問題はパケットレベルまで掘り下げることができます。そこでは見た目がかわいいアプリケーションもその醜い実装をさらけ出し、信用できるように見えるプロトコルが悪意あるものでありうることを図らずも証明してしまうのです。ネットワークの問題をより理解し解決するためには、すべてをさらけ出しているパケットを見る必要があります。パケットはアプリケーションにありがちなメニューの見間違いや目を引くグラフィックがなく、信頼できない従業員によってごまかされることもありません。パケットにはなんの秘密もなく、パケットレベルでできることが増えれば、ますますネットワークを制御し問題を解決することができるようになります。

これがパケット解析の世界です。本書はパケット解析の世界に頭から飛び込んでいます。本書を通じて、ネットワークの通信を調べる前にパケット解析とは何かを学び、異なるシナリオを調査するために必要な、基礎的な背景知識を得ることができます。また、パケット解析ツールWiresharkの機能の使い方や、ネットワークの遅延の解決、ボトルネックになっているアプリケーションの特定、実際のシナリオを通してのハッカーの追跡術を学ぶことができるでしょう。本書を読み終えるころには、高度なパケット解析の技術を体得しているはずです。その技術を用いれば、ネットワーク上で起こる多くの困難な問題を解決することができるでしょう。

1.1 パケット解析とは？

パケットスニффイングやプロトコル解析と呼ばれることもあるパケット解析とは、ネットワーク上で起こっていることをより理解しやすくするために、データをキャプチャして解析することを意味します。パケット解析は通常、スニッフアを使って行います。スニッフアとは、ケーブルを通っている生のネットワークデータをキャプチャ

するツールです。パケット解析はネットワークの特性を理解し、ネットワーク上に誰がいるのか、何が使用可能な帯域を使っているのか、ネットワークの使用がピークになる時間はいつか、攻撃や悪意ある行為がなされていないか、安全でなく負荷が高いアプリケーションは何かを知るための手助けをしてくれます。

スニッファプログラムには、フリーと商用どちらもたくさんの種類があります。各プログラムはそれぞれ違う目的のために設計されています。もっとも有名なパケット解析プログラムとして、tcpdump (コマンドラインのプログラム)、OmniPeek、そしてWiresharkがあります。OmniPeekとWiresharkはGUIベースのスニッファです。

1.2 パケットスニッファの評価

パケットスニッファにはさまざまな種類があります。どれを使うか決めるには、以下の点を考える必要があります。

- サポートされているプロトコル
- ユーザーフレンドリかどうか
- コスト
- OSのサポート
- スニッファのサポート体制

1.2.1 サポートされているプロトコル

パケットスニッファはさまざまなプロトコルを解釈することができます。ほとんどのスニッファは、DHCP、IP、ARPのような一般的なプロトコルを解釈できますが、中には新しいプロトコルを解釈できないものもあります。スニッファを選ぶときは、使用する予定のプロトコルがサポートされているかを確認しましょう。

1.2.2 ユーザーフレンドリかどうか

パケットスニッファプログラムのレイアウト、インストールのしやすさ、通常の操作の流れを検討しましょう。もしパケット解析の経験がほとんどないのであれば、tcpdumpのような高度なコマンドラインのパケットスニッファは避けるべきでしょう。逆に経験豊富なら、高度なプログラムのほうがよいかもしれません。

1.2.3 コスト

パケットスニッファのすごいところは、商用の製品に匹敵するフリーの製品が数多く存在することです。お金を払わなくても、よいパケットスニッファが入手できます。

1.2.4 スニッファのサポート体制

たとえスニッファプログラムの基本をマスターしても、新たな問題を解決するため

のサポートが必要になるときもあるでしょう。サポートを評価する際には、開発者のドキュメント、公開されているフォーラムやメーリングリストを探してみてください。Wiresharkのようなフリーのパケットスニッファでは開発者のサポートはあまりないかもしれませんが、ユーザーのコミュニティがそれを補っている場合がよくあります。これらのユーザーや貢献者のコミュニティではディスカッションのための掲示板、Wikiやブログを提供しており、パケットスニッファについてより多くのことを知る手助けをしてくれます。

1.2.5 OSのサポート

残念ながら、すべてのパケットスニッファがどんなOSでも使えるわけではありません。スニッファが使えるのかどうかをOSごとに確認しましょう。

1.3 パケットスニッファの仕組み

パケットスニッファの処理は、以下の3つのステップに分けることができます。すなわち、収集、変換、そして解析です。

1.3.1 収集

最初のステップでは、パケットスニッファはスニッフィングするネットワークに接続されているインターフェースをプロミスキャスモードに切り替えます。このモードでは、ネットワークカードは特定のネットワークセグメント上を流れるすべてのネットワークトラフィックを監視することができます。

1.3.2 変換

次のステップでは、キャプチャされたバイナリデータを解読可能な形式に変換します。多くの高度なコマンドラインベースのスニッファは、ここまでしかやりません。このステップではネットワークのデータは非常に基礎的なレベルでの解釈しか行われず、解析のほとんどはエンドユーザーの手にゆだねられます。

1.3.3 解析

第3のそして最後のステップでは、キャプチャし変換されたデータを実際に解析します。このステップでは、スニッファを用いてネットワークデータをキャプチャし、抜き出したデータを元にプロトコルを特定し、そしてプロトコルの特徴を解析します。

さらに、解析は複数のパケットやネットワークのほかの多くの要素を比較することで行われます。

1.4 コンピュータはどのように通信するのか

パケット解析を完全に理解するためには、コンピュータ同士がどうやって通信しているのかを理解する必要があります。この節ではOSI参照モデル、ネットワークのデータフレーム、それらをサポートするハードウェアといったネットワークプロトコルの基礎を勉強します。

1.4.1 ネットワークプロトコル

現在のネットワークは、各種プラットフォームとシステムの非常に多くの組み合わせで成り立っています。この通信を支援するために、ネットワーク通信を統治するネットワークプロトコルと呼ばれる共通の言語を使用しています。一般的なネットワークプロトコルとして、TCP、IP、ARP、DHCPがあります。プロトコルスタックとは、プロトコルを論理的にグループ化したものです。

ネットワークプロトコルは、その機能によってシンプルにも複雑にもなります。さまざまなネットワークプロトコルは、それぞれ性質が大きく異なる場合があります。異なる点は主に以下の機能です。

フロー制御

受信側のシステムが、送信側のシステムのデータ転送の速度を上げたり下げたりするためのメッセージを生成すること

パケットの応答確認

受信側のシステムから、データを受信したことを知らせるために送信側のシステムに返信されるメッセージの転送のこと

誤り検出

送信側のシステムが、送信されたデータが転送中に破損していないかを検証するために使用するコードのこと

エラー訂正

最初の転送の間に破損や消失したデータを再転送すること

分割

長いデータストリームを、効率的に転送するために小さいものに分割すること

データの暗号化

ネットワーク上を転送されるデータを保護するために暗号鍵を用いる機能

データの圧縮

ネットワーク上を転送されるデータのサイズを減らすために、余分な情報を削除する機能

1.4.2 OSI参照モデル

プロトコルは、**OSI参照モデル**と呼ばれる業界標準の参照モデルを元に、その機能によって分けられています。このモデルは1983年にISO（International Organization for Standardization：国際標準化機構）によって、ISO7498として公開されました。

OSI参照モデルは、ネットワーク通信のプロセスを以下の7つの階層に分けています。

- アプリケーション層（レイヤ7）
- プレゼンテーション層（レイヤ6）
- セッション層（レイヤ5）
- トランスポート層（レイヤ4）
- ネットワーク層（レイヤ3）
- データリンク層（レイヤ2）
- 物理層（レイヤ1）

この7階層から成るOSI参照モデル（図1-1）が存在するおかげで、ネットワーク通信というものが理解しやすくなっています。最上層のアプリケーション層はネットワークリソースにアクセスするための実際のプログラムを表しています。最下層は物理層で、実際のネットワークデータの転送を行う層です。各層のプロトコルは、データを上位層のためにパッケージするため、連携して機能します。



図1-1 OSI参照モデルの7つの階層



OSI 参照モデルは業界が推奨しているスタンダード以上の何ものでもありません。プロトコルの開発者は、このモデルに正確に準拠する必要はありません。実際のところ、現存するネットワークモデルは OSI 参照モデルだけではありません。たとえば、DoD (Department of Defense : 米国国防総省) モデルを好む人もいます。本書は OSI 参照モデルの概念にのっとって書かれているため、DoD モデルについては言及しません。

OSI 参照モデルの各階層の機能と、各層で使用されているプロトコルの例をいろいろと見ていきましょう。

1.4.2.1 アプリケーション層

OSI 参照モデルの最上層であるアプリケーション層は、ネットワークリソースにアクセスするための手段をユーザーに提供します。これは通常エンドユーザーが見ることができる唯一の層であり、ネットワーク上のすべての作業の基点となるインターフェースを提供します。

1.4.2.2 プレゼンテーション層

プレゼンテーション層は、受信するデータをアプリケーション層が読むことができる形式に変換します。この層でデータをどうエンコードまたはデコードするかは、送信するデータのアプリケーション層のプロトコルに依存します。この層ではデータの安全のための暗号化や復号のさまざまな形式も操作します。

1.4.2.3 セッション層

セッション層は2台のコンピュータ間の対話や、通信デバイス間の接続の確立、管理、終了といったセッションを管理します。またセッション層は、通信が全二重なのか半二重なのかを明確にし、通信を唐突に切断するのではなくきれいに終了させるという機能もあります。

1.4.2.4 トランスポート層

トランスポート層の主要な目的として、下位層に信頼できるデータを渡すということがあげられます。フロー制御、データの分割と再構築、誤り制御といった機能のおかげで、トランスポート層は2点間のデータのやり取りをエラーなしで行えるわけです。信頼性の高いデータ転送を保証することは極めて難しいため、OSI 参照モデルではすべての層が信頼性の保証のために機能します。トランスポート層は接続確立ありのサービスと接続確立なしのサービス両方を提供します。ファイアウォールとプロキシサーバはこの層で動作します。

1.4.2.5 ネットワーク層

ネットワーク層は物理的なネットワーク間でのデータのルーティングを提供している、OSI参照モデルの中でもっとも複雑な層です。ネットワークホストの論理的なアドレス（たとえばIPアドレス）の管理を担当します。また、パケットの分割、プロトコルの特定、場合によっては誤り検出も行います。ルータはこの層で動作します。

1.4.2.6 データリンク層

データリンク層は物理層を通してデータを転送する手段を提供します。この層の主な目的は、物理的なデバイスを特定するためのアドレスを管理し、データの正当性を保証するためのエラーチェックを提供することです。ブリッジとスイッチはこの層で動作する物理的なデバイスです。

1.4.2.7 物理層

物理層はOSI参照モデルの最下層であり、ネットワークでのデータ転送における物理的な媒体です。この層では、電圧、ハブ、ネットワークアダプタ、リピータ、ケーブルなど、使用されるすべてのハードウェアの物理的かつ電氣的なものを扱います。物理層は接続を確立および終了させ、通信リソースを共有する手段を提供し、信号をデジタルからアナログ、またはその逆に変換します。

表1-1では、OSI参照モデルの各層における、もっとも一般的に使用されているプロトコルを一覧にしています。

表1-1 OSI参照モデルの各層で使用される代表的なプロトコル

層	プロトコル
アプリケーション層	HTTP、SMTP、FTP、TELNET
プレゼンテーション層	ASCII、MPEG、JPEG、MIDI
セッション層	NetBIOS、SAP、SDP、NWLink
トランスポート層	TCP、UDP、SPX
ネットワーク層	IP、ICMP、ARP、RIP、IPX
データリンク層	Ethernet、Token Ring、FDDI、AppleTalk

1.4.3 プロトコルの相互作用

OSI参照モデルでは、データはどのように上位層または下位層に流れているのでしょうか？ ネットワーク上を転送される最初のデータは、それを送信するシステムのアプリケーション層から始まります。データはそれぞれの方法でOSI参照モデルの7つの階層を送信側のシステムの物理層まで下っていき、受信側のシステムに送られます。受信側のシステムは物理層でデータを受信し、データは最上層のアプリケーション層まで受信側のシステムの各層を上がっていきます。

OSI参照モデルの各層におけるさまざまなプロトコルによって提供されるサービスは、重複することはありません。たとえば、もしある層でプロトコルが特定のサービスを提供すれば、他の層でのプロトコルは同じサービスを提供することはありません。

送信側と受信側のコンピュータでは、同一層のプロトコルは一致します。送信側のコンピュータのレイヤ7のプロトコルが転送されるデータを暗号化する場合、受信側のコンピュータのレイヤ7のプロトコルはデータを復号することを求められます。図1-2は通信中の2つのクライアントにおけるOSI参照モデルの図です。片方のクライアントの最上層から最下層を通り、もう片方のクライアントに到達した後その逆をたどることで通信が成り立ちます。

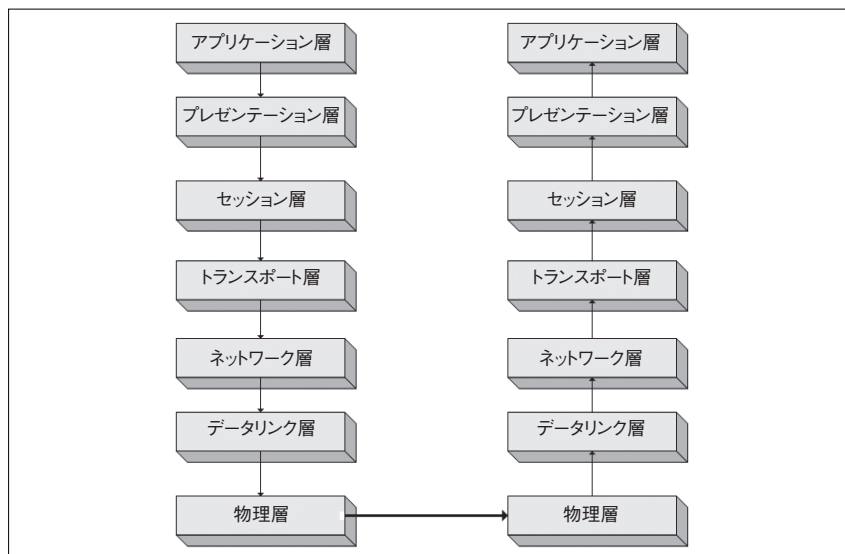


図1-2 送信側と受信側のシステムにおいて同じ層で機能するプロトコル

OSI参照モデルの各層は、その直上または直下の層としか通信できません。たとえば、レイヤ2はレイヤ1およびレイヤ3のデータしか送受信できません。

1.4.4 データのカプセル化

各層同士が通信するためには、データをカプセル化する必要があります。**カプセル化**とは、ヘッダやフッタを追加することです。たとえばトランスポート層がセッション層からデータを受信した場合、トランスポート層は次の層にデータを渡す前にヘッダ情報を追加します。

1.4.5 プロトコルデータユニット

カプセル化とは、PDU (Protocol Data Unit : プロトコルデータユニット) を生成することをいいます。PDUとは、送信されるデータと追加されたヘッダおよびフッタ情報すべてを含みます。

データがOSI参照モデルに従って階層を降りていくとき、PDUはさまざまなプロトコルが追加していくヘッダ情報とフッタ情報によって大きくなっていきます。PDU

は物理層に到達すると最終的な形式となり、受信側のコンピュータに送られます。受信側のコンピュータは、データがOSI参照モデルの階層を上がっていくにつれ、プロトコルのヘッダおよびフッタをPDUから取り除いていきます。PDUがOSI参照モデルの最上層に到達するときには、もともとのデータしか残っていません。



パケットという単語はPDUを連想させます。本書でパケットという単語を使うときは、OSI参照モデルのすべての層が追加するヘッダとフッタを含んだPDUのことを指しています。

1.4.6 ネットワークハードウェア

それではこれからネットワークハードウェアを見ていきましょう。ここではハブ、スイッチ、ルータといった一般的なネットワークハードウェアに焦点を当てましょう。

1.4.6.1 ハブ[†]

ハブとは通常、図1-3のNetgearのハブのような、RJ-45のポートを複数持つただの箱にすぎません。ハブは4ポートという非常に小型なものから、企業向けにラックマウント用に設計された48ポートの大型のものまであります。ハブは通信のためにネットワークデバイスに接続するように設計されています。

ハブはOSI参照モデルの物理層で動作する、データの中継を行うデバイスです。こ



図1-3 典型的な4ポートのイーサネットハブ

のデバイスは、デバイス上のすべてのポートに送信されたパケットを伝送（中継）します。たとえば、コンピュータが4ポートハブのポート1に接続されていて、ポート2に接続されているコンピュータにデータを送信する場合、ハブはパケットをポート1、2、3、4のすべてに送信します。ポート3とポート4に接続されているクライアントは、彼らのためのデータではないのでこのデータを無視し、破棄します。このため、不要なネットワークトラフィックが多く発生します。

ある企業の社員にメールを送る場合を想像してください。メールの題名には「マーケティング部の皆さまへ」とありますが、メールはマーケティング部で働いている人のみに送信されるのではなく、その企業の社員全員に送信されます。マーケティング

[†] 監訳注：ここでいうハブは、シェアードハブ、リピータハブのことです。

部の社員はメールが自分宛であることが分かりますから、そのメールを開封します。しかしながら他の社員はメールが自分宛でないことが分かれば、それを破棄するでしょう。多くの不要な通信と無駄な時間がなぜ発生するかこれで分かるでしょう。これがハブの機能です。

図1-4はハブの使用によって何が起るかを図で表しています。この図では、コンピュータAがコンピュータBにデータを送信しています。しかしながら、コンピュータAがデータを送信したとき、ハブに接続されているすべてのコンピュータがそれを受信します。コンピュータBのみがデータを実際に受信し、他のコンピュータはそれを破棄します。

ハブについての最後の注意は、ハブは半二重モード（送受信を同時にできない）で

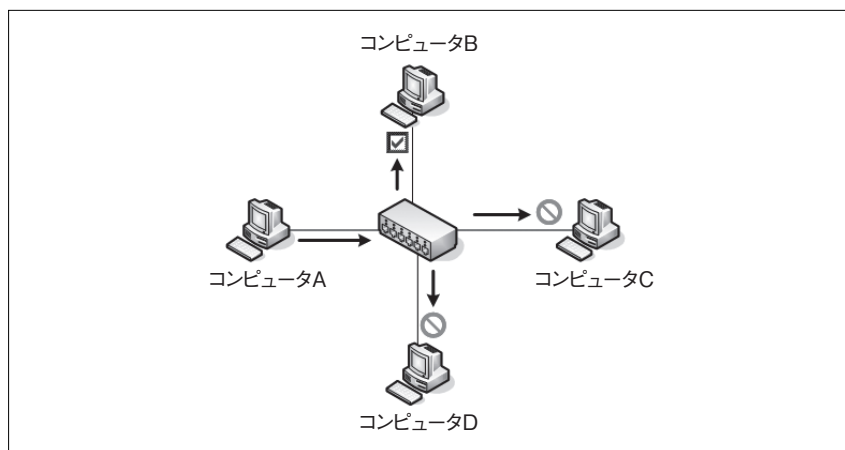


図1-4 コンピュータAがコンピュータBにハブを通してデータを送信するときのトラフィックの流れ

のみ動作するということです。これがスイッチと違うところで、スイッチは送受信が同時にできる全二重モードで使用できます。

現在の高密度なネットワークの多くは、後述する理由でハブではなくスイッチが使われていますが、ハブの機能を理解することは、パケット解析を行う上で非常に重要です。

1.4.6.2 スイッチ[†]

高密度なネットワークにおいてハブに替わる最良の機器として、スイッチがあげられます。ハブと同じく、スイッチはパケットを中継するよう設計されていますが、大きな違いがあります。ハブと同じくスイッチは通信経路をデバイスに提供しますが、スイッチを使った通信はとても効率的です。スイッチはすべてのポートにデータを送

[†] 監訳注：ここでのいうスイッチは、マネージメントスイッチもしくはスイッチングハブのことです。

信するのでなく、送信したいコンピュータにのみデータを送信します。見た目は、スイッチはハブとよく似ています。実際には表面にそれと示すものを書いていなければ、どちらなのか分からないでしょう (図 1-5)。

大きなスイッチは、ベンダ固有のソフトウェアや Web 上のインターフェースを通

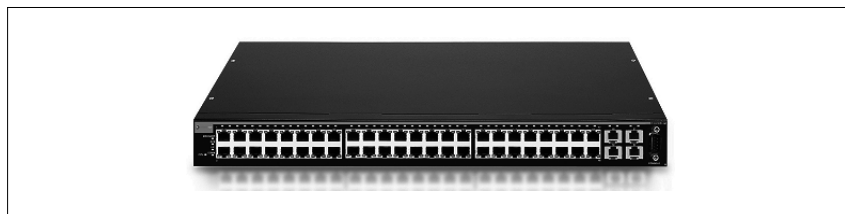


図 1-5 ラックマウント型の24ポートイーサネットスイッチ

して管理することができます。これらのスイッチはマネージメントスイッチと呼ばれ、ネットワークを管理する際に便利なさまざまな機能を持っています。特定のポートを有効または無効にしたり、ポートの詳細を表示したり、設定を変更したり、リモートからスイッチを再起動したりできます。

スイッチはパケット送信を操作するための高度な機能を持っています。特定のデバイスと直接通信できるようにするため、スイッチはデバイスをアドレスで管理します。つまり、スイッチはOSI参照モデルのデータリンク層で動作するということです。

スイッチは、接続されているすべてのデバイスのレイヤ2のアドレスを、トラフィックの見張り番のような働きをする**CAM テーブル**に記録しています。パケットが送信されると、スイッチはパケット内にあるレイヤ2のヘッダ情報を読み、CAM テーブルを参照してどのポートにパケットを送信するか決定します。スイッチは特定のポートにしかパケットを送信しないため、ネットワークトラフィックを劇的に減らすことができます。

図 1-6 はスイッチを通したトラフィックの流れを図で示しています。この図でも、コンピュータ A がコンピュータ B にデータを送信しています。この例では、コンピュータはスイッチに接続されており、他のコンピュータが通信に気づくことなく、コンピュータ A はコンピュータ B に直接データを送っています。さらに、同時に複数通信することができます。

1.4.6.3 ルータ

ルータはスイッチやハブよりハイレベルな機能を持った高度なネットワークデバイスです。ルータはさまざまな形のものがありますが、多くは前面にインジケータランプ (LED) が付いていて背面にポートがあります。ポートの数はネットワークの大きさに依存します (図 1-7)。ルータはOSI参照モデルのレイヤ3で動作し、2つ以上のネットワーク間でパケットを転送します。ネットワーク間のトラフィックの流れを指示することを、ルーティングといいます。

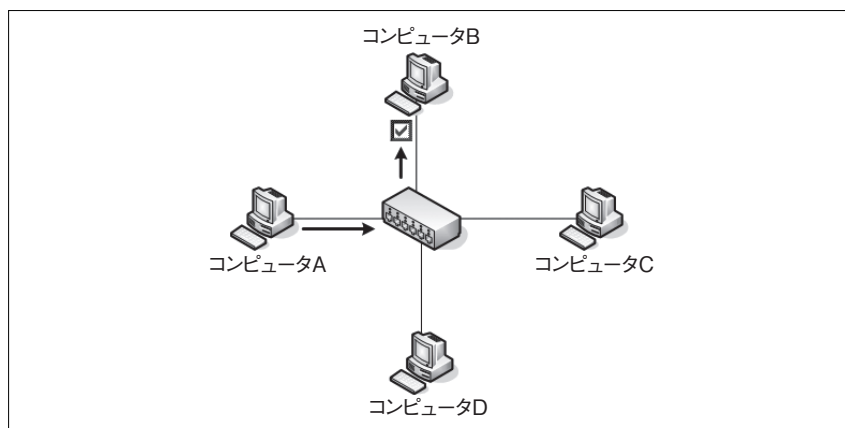


図 1-6 スイッチを通してコンピュータ A がコンピュータ B にデータを送信する際のトラフィックの流れ



図 1-7 小規模ネットワークのための小さなルータ

異なる種類のパケットを、どうやって他のネットワークに転送するかを決定するプロトコルを、ルーティングプロトコルといいます。ルーティングプロトコルにはいくつかの種類があります。ルータは通常、ネットワーク上のデバイスを識別するために、IPアドレスのようなレイヤ3のアドレスを使用します。

ルーティングの概念をイメージする簡単な方法は、通り沿いの近所の家を考えることです。それぞれの通りには家があり、各家には固有の住所があります(図 1-8)。ある通りに住んでいるとすれば、その通り沿いのすべての家の間を行き来することができます。これは、スイッチに接続することによって、ネットワークセグメント上のすべてのコンピュータと通信できるということとよく似ています。しかしながら、他の通りの隣人を訪ねるためには、その人は道路標識に従う必要があります。

通りを横断して通信する例を見てみましょう。図 1-8 のバインストリートの 503 からドッグウッドレーンの 202 に行かなければならないとします。そのためには、オークストリートを通ってドッグウッドレーンに行かなければいけません。これを、ネットワークセグメントを横断する場合で考えてみてください。192.168.0.3 のデバイスが 192.168.0.54 のデバイスと通信する必要がある場合、10.100.1.1 のネットワークに行く

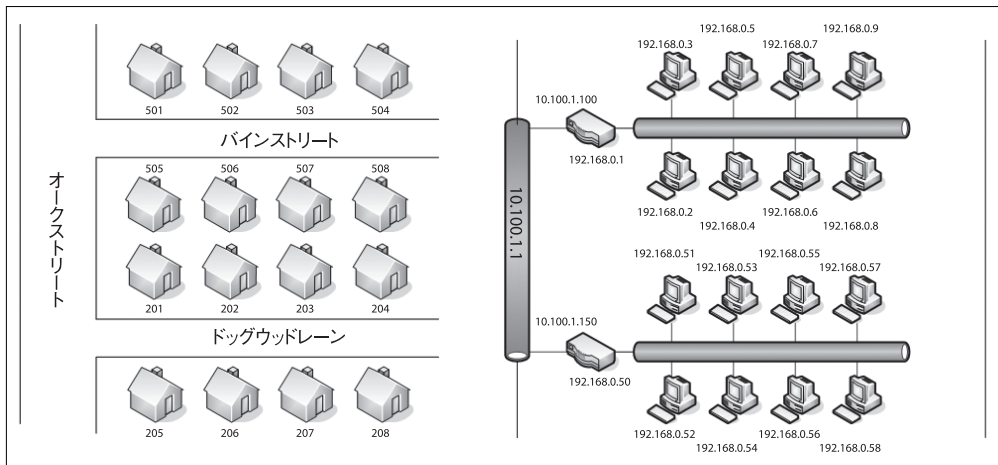


図1-8 ルーティングと近所の通りとの比較

ためには、ルータを通らなければいけません。そして通信するデバイスが存在するネットワークセグメントのルータを通ります。

ネットワーク上のルータの大きさや数は、ネットワークの大きさや機能によって変わります。個人やホームオフィスのネットワークの場合は、ネットワークの中央に置かれたルータのみで構成されているでしょうし、巨大企業のネットワークではいくつものルータがさまざまな部門に置かれ、それらすべては中央の巨大なルータやレイヤ3スイッチに接続されているでしょう。**レイヤ3スイッチ**はスイッチが進歩したもので、ルータのような機能がビルトインされています。

ネットワーク構成図にたくさん触れると、さまざまなポイントを通るデータの流れを理解することができるでしょう。図1-9はルーティングのよくある形を示しています。この例では、2つのネットワークが1つのルータで接続されています。ネットワー

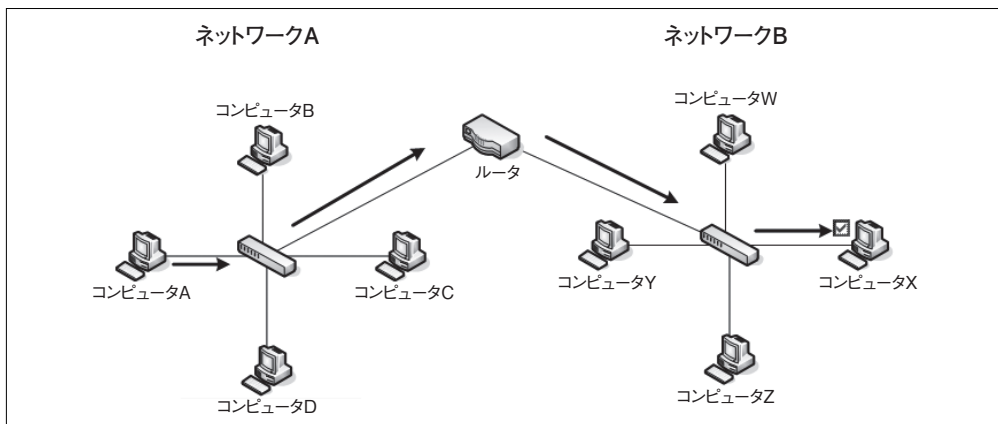


図1-9 コンピュータAがコンピュータXにルータを介してデータを送信したときのトラフィックの流れ

ク A のコンピュータがネットワーク B のコンピュータと通信する場合、送信されるデータは必ずルータを通らなければいけません。

1.4.7 トラフィックの分類

ネットワークトラフィックは、ブロードキャスト、マルチキャスト、ユニキャストの3つに分類することができます。これらはそれぞれ異なる特徴を持っています。それによってネットワーク上のハードウェアがパケットをどのように扱うかが決まります。

1.4.7.1 ブロードキャスト

ブロードキャストパケットは、ネットワークセグメント上のハブ、スイッチ、ルータのすべてのポートに送信されます。「1.4.6.1 ハブ」でも説明しましたが、ハブはブロードキャストしかできません。

1.4.7.2 マルチキャスト

マルチキャストは、1つの送信元から複数の宛先に同時にパケットを送信する手段です。できる限り小さな帯域を使い、できる限りシンプルにこの手段を実現しています。トラフィックは、宛先に到達するために何回データが複製されたかによって、どう最適化されるか決まります。マルチキャストのトラフィックを正確に操作できるかどうかは、個々のプロトコルの実装に大きく依存しています。マルチキャストの主な実装方法は、パケットを受信するシステムをマルチキャストグループとしてグループ化し、そのグループにアドレスを割り振ることです。これがIPマルチキャストの働きです。マルチキャストグループにアドレスを割り振ることによって、パケットを受け取るべきでないコンピュータにパケットを送信しないようにします。

1.4.7.3 ユニキャスト

ユニキャストパケットはコンピュータからコンピュータへ直接送信されます。ユニキャストがどう機能するかは、使用するプロトコルによって決まります。

1.4.7.4 ブロードキャストドメイン

ブロードキャストパケットは特定のセグメント上のすべてのデバイスに送信されるということを思い出してください。異なる媒体を通して接続されている複数のハブやスイッチからなる巨大なネットワーク上では、1個のスイッチから送信されたブロードキャストパケットは、スイッチからスイッチへと中継され、ネットワーク上の他のスイッチに到達します。

ブロードキャストパケットが到達する範囲をブロードキャストドメインといいます。これはルータを通らずにコンピュータからコンピュータに直接到達できるネットワークセグメントを指します。図1-10は小さなネットワーク上の2つのブロードキャスト

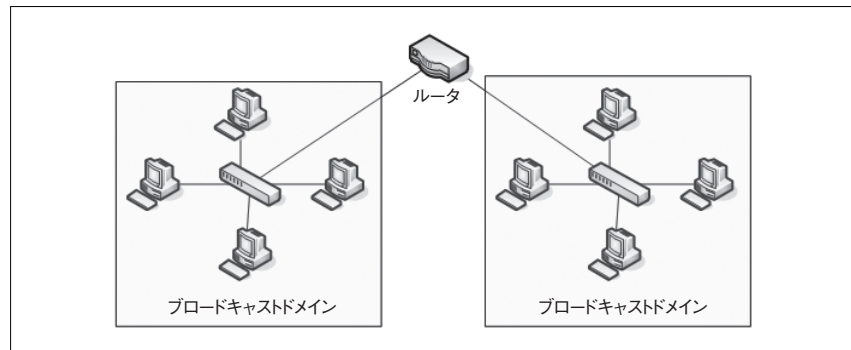


図 1-10 ブロードキャストドメインはルータに到達するまで

ドメインの例を示しています。ルータに到達するまでがブロードキャストドメインなので、ブロードキャストパケットはブロードキャストドメイン内を巡回します。

ルーティングと近隣の家との関係を前に説明しましたが、ブロードキャストドメインの働きについても同じことがいえます。ブロードキャストドメインは近隣の通りだと考えてみてください。もしポーチに立って叫んだら、その通りにいる人たちはそれを聞くことができます。他の通りの人と話したい場合は、ポーチからブロードキャストする（叫ぶ）のではなく、直接その人と話す方法を見つける必要があります。

ここで学んだことは、パケット解析の不変の基礎です。ネットワークのトラブルシューティングの前に、ネットワーク通信で何が起きているかを理解しなければいけません。次の章ではこれらの概念に基づき、ネットワーク通信のより高度な原理について議論していきます。

2章

ケーブルにもぐりこむ

さあ、いよいよパケットキャプチャを始めるための最後のステップに移ります。ここでは、スニッファをネットワーク上のどこで使うかを決定する方法を学びます。

残念ながらスニッフィングは、単純にノートPCをネットワークポートに接続してパケットをキャプチャすればよいというものではありません(図2-1)。実際には、パケットを解析するよりもスニッファをつなぐネットワーク上の位置を決めるほうが難しいこともあります。

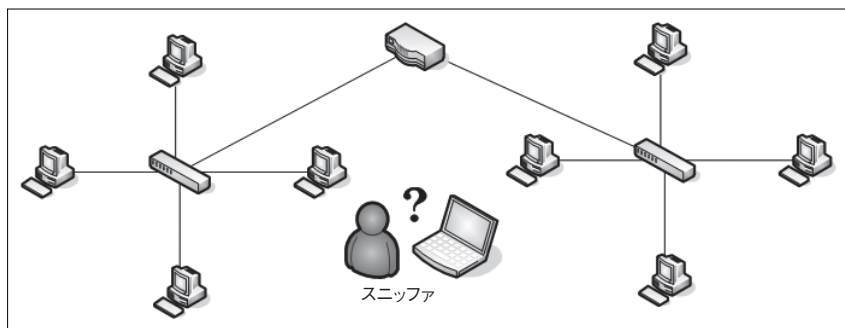


図2-1 スニッファを配置する位置を決めることが、もっとも難しいかもしれない

なぜスニッファの配置が難しいかというと、ネットワーク機器の種類が非常に多いからです。ハブ、スイッチ、ルータといった、現代のネットワークで使われている3つの主なネットワーク機器は、それぞれがまったく違った形で通信を扱うため、スニッファを配置する際にはその点を十分考慮しなければなりません。

この章の目的は、さまざまなネットワークトポロジでのスニッファの配置方法を理解することです。スニッファを正しい場所で使うためには、ハブやスイッチ、ルータそれぞれの環境でパケットをキャプチャするもっともよい方法を学ぶ必要があります。スニッファの配置方法を理解する前に、プロミスキャスモードのNICや、それがどう動作するのか、なぜそれがパケット解析に必要なのか、を学習していきましょう。

2.1 プロミスキヤスモードの使用

ネットワーク上でパケットを監視するには、プロミスキヤスモードをサポートしているNIC (Network Interface Card : ネットワークインターフェースカード) が必要です。

NICがプロミスキヤスモードでない場合、NICは自分宛ではないパケットを受け取っても、それを破棄してしまいます。プロミスキヤスモードのときは、たとえレイヤ2のアドレスが自分宛のものでなくても、すべてのパケットをキャプチャします。スニッファはすべてのパケットをキャプチャし、解析に必要な情報を与えてくれます。

2.2 ハブで構成されたネットワークでのスニッフィング



Windowsを含むほとんどのOSで、プロミスキヤスモードは、高い権限がないと使用できません。プロミスキヤスモードを使用するための権限がないなら、スニッファを使うべきではありません。

ハブで構成されたネットワークでのスニッフィングがもっとも簡単です。すでに学んだとおり、パケットはハブのすべてのポートに流れます。したがって、ハブに接続されているコンピュータの通信を解析するためには、ハブ上の空いているポートにスニッファがインストールされたコンピュータ (以下、スニッファ用コンピュータ) を接続するだけでよいのです。これで図2-2にあるように、スニッファは、ハブに接続されているすべてのコンピュータの通信を見ることができます。

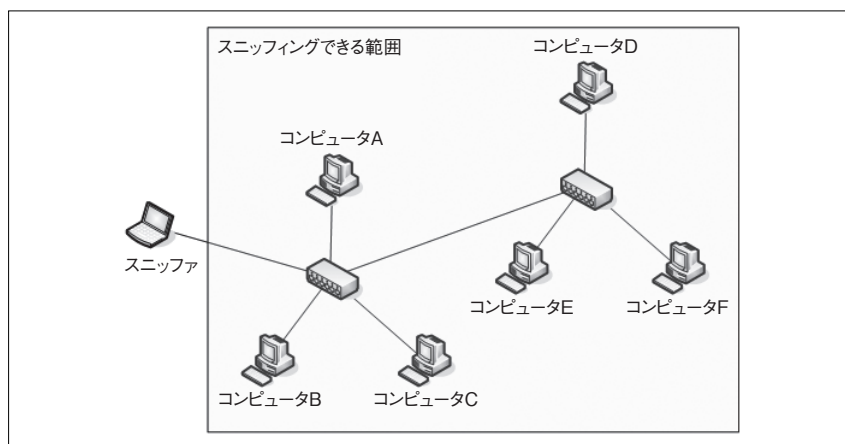


図2-2 ハブで構成されたネットワークでは、すべてをスニッフィングできる



本書ではスニッファがスニッフィングできるコンピュータの範囲を、「スニッフィングできる範囲」という枠で示しています。

残念ながら、ハブで構成されたネットワークは、今ではほとんど使われていません。ハブは1回に1つの通信しか扱えないため、通信の遅延の原因になります。したがって、ハブを通して接続されているコンピュータは、通信しようとしている他のコンピュータと帯域幅を取り合うことになります。2台以上のコンピュータが同時に通信しようとする、図2-3に示すようなパケットの衝突が起きるため、パケットが消失してしまい再送する必要があります。

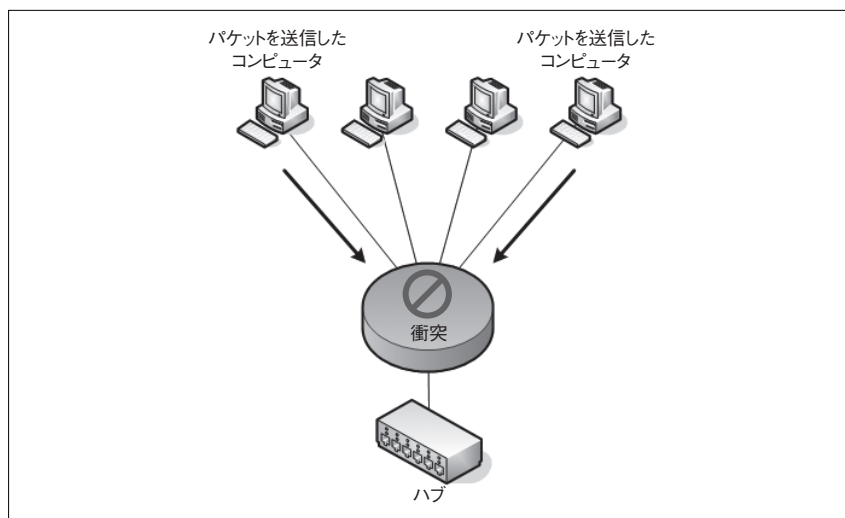


図2-3 2つのコンピュータが同時にパケットを送信すると、パケットの衝突が起こる

衝突が増えると、コンピュータはパケットを3回4回と送信しなければならないため、ネットワークのパフォーマンスが劇的に落ちてしまいます。現在のネットワークがハブでなくスイッチを使用している理由はそこにあります。

ハブで構成されたネットワーク上のコンピュータの通信をスニッフィングするとき、気をつけなければならないのはキャプチャするパケットの量です。プロミスキャスモードのNICは、ハブ上を行き来するすべてのパケットをキャプチャするため、解析するデータの量が膨大になります。次の章ではより効率的にパケットを解析するためのテクニックを紹介しています。

2.3 スイッチで構成されたネットワークでのスニффイング

現在もっとも一般的なのは、スイッチで構成されたネットワークです。スイッチはブロードキャスト、ユニキャスト、マルチキャスト（これら3つの分類の詳細については1章を参照）のデータを効率的に転送するネットワーク機器です。さらに、スイッチは全二重の通信が可能のため、データの送信と受信を同時に行うことができます。しかし、スイッチで構成されたネットワーク上でのスニッフイングは、ハブで構成されたネットワークほど単純ではありません。図2-4が示すとおり、スイッチに接続されたスニッファは、ブロードキャストパケットとスニッファがインストールされているコンピュータ宛のパケットしか見ることができないのです。

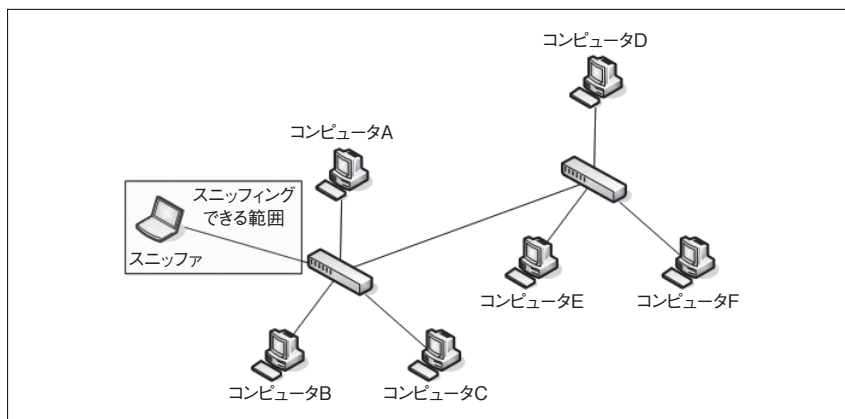


図2-4 スイッチで構成されたネットワークでは、スニッファがインストールされているコンピュータが接続されているポートしかスニッフイングできない

スイッチで構成されたネットワークで、特定のコンピュータの通信をキャプチャする3つの主な方法として、ポートミラーリング、ARP キャッシュポイズニング、ハブを使用するという方法があります。

2.3.1 ポートミラーリング

ポートミラーリングは、スイッチを使ってパケットをキャプチャするもっとも簡単な手段です。ミラーリングを利用すれば、特定のコンピュータが送受信するパケットをキャプチャすることができます。ポートミラーリングを使用するためには、ターゲットとなるマシン（以下、ターゲットマシン）が接続されているスイッチを、コマンドを使って操作する必要があります。加えて、スイッチがポートミラーリングをサポートしていること、そのスイッチに、スニッファ用コンピュータを接続するための空きポートがあることも必要です。

ポートミラーリングを使用するときは、スイッチのコマンドラインインターフェースを使い、特定のポートの通信を他のポートにコピー（ミラーリング）するように、

コマンドを入力する必要があります(図2-5)。たとえば、ポート3のパケットをキャプチャするには、スニッファ用コンピュータをポート4に接続し、ポート3からポート4にミラーリングするように設定すればよいのです。それによって、ポート3のコンピュータの通信を見ることができるようになります。

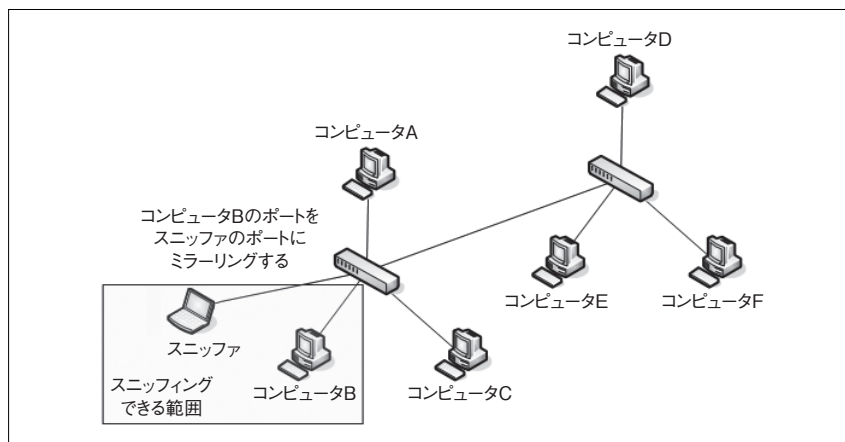


図2-5 ポートミラーリングによって、スニッフィングできる範囲を広げることができる

ポートミラーリングのためのコマンドは、スイッチのメーカーによって異なります。表2-1は、主なメーカーのコマンド一覧です。

表2-1 メーカーごとのポートミラーリングのためのコマンド

メーカー	ポートミラーリングのコマンド
シスコ	<code>set span <ミラーリング元> <ミラーリング先></code>
エンテラス	<code>set port mirroring create <ミラーリング元> <ミラーリング先></code>
ノーテル	<code>port-mirroring mode mirror-port <ミラーリング元> monitor-port <ミラーリング先></code>

ポートミラーリングを使用するときは、ミラーリングしているポートのスループットに注意してください。スイッチの中には、2台以上のコンピュータの通信を同時に解析できるようにするため、複数のポートを1つのポートにミラーリングできるものがあります。しかしながら、たとえば24ポートのスイッチで、100Mbps、全二重で通信する23ポートの通信を1つのポートにミラーリングした場合を考えてみてください。4,600Mbpsものパケットが1つのポートに流れることになります。そうなればかなりの確率でポートを通るパケットの量が物理的な限界を超えることになり、パケットの損失やネットワークの遅延を引き起こすことになります。スイッチは通信を抑制するために、処理しきれないパケットをすべて破棄したり、バックプレーンを停止させたりします。パケットをキャプチャする際には、このような状況にならないように気をつけてください。

2.3.2 ハブの使用

スイッチで構成されたネットワーク上でパケットをキャプチャするもう1つの方法は、ハブを使用することです。パケットをキャプチャしたコンピュータとスニッファ用コンピュータを、ハブに接続することで同じネットワークセグメント上に置いてしまうのです。

多くの人々は、そのようにハブを使用することは不正行為だと思っていますが、ポートミラーリングが使えない環境で、かつキャプチャしたいコンピュータが接続されているスイッチに物理的に触ることが可能ならば、ハブの使用はスニффイングを実現する完璧な方法といえます。

スイッチで構成されたネットワーク上で、ハブを使用してコンピュータの通信をスニッフイングするために必要なのは、ハブと数本のネットワークケーブルだけです。スニッファ用コンピュータを持ってスイッチのある場所に行き、ターゲットマシンのケーブルを抜きます。そして抜いたケーブルとハブを接続し、スニッファ用コンピュータもハブに接続します。次に、ハブとスイッチを接続します。これで、スニッファ用コンピュータとターゲットマシンが同じブロードキャストドメイン上に存在することになります。これでターゲットマシンが送受信するパケットは、ハブに接続されているすべてのコンピュータにブロードキャストされることになり、スニッファがパケットをキャプチャすることができるようになります(図2-6)。

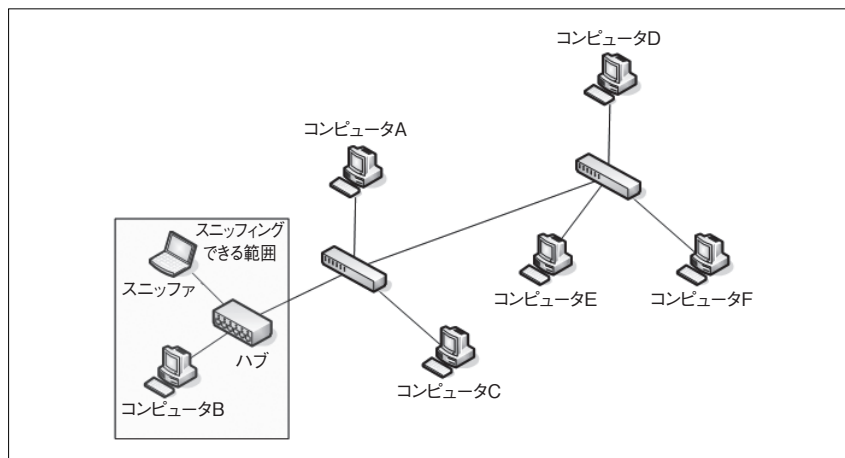


図2-6 ハブを使うことで、ターゲットマシンとスニッファ用コンピュータを同じブロードキャストドメインに置くことができる

ほとんどの場合、ハブを使えば全二重の通信が半二重に半減されることになります。ハブの使用は最良の方法とはいえませんが、ポートミラーリングをサポートしていない場合は、この方法を使うしかありません。



ついでに言っておくと、CEOが使っているコンピュータのケーブルを抜くのは楽しい作業になるでしょう。

ハブを使用する際、それがスイッチではなく本当にハブなのかを確認してください。ネットワークハードウェアのベンダの中には、低機能なスイッチをハブとして販売しているところがあります。ハブを使わなければ、スニッファは自身が送受信するパケットをキャプチャするだけになってしまいます。ハブであるかどうかは、2台のコンピュータをハブに接続して、一方が他方のパケットをキャプチャできるかどうかを確認すれば分かります。もしそれができれば、それは本当のハブということになります。

2.3.3 ARP キャッシュポイズニング

1章では、パケットのアドレスにはレイヤ3のものとレイヤ2のものの2種類があるということを学びました。レイヤ2のアドレス (MAC アドレス) は、レイヤ3のアドレスに連動して使用されます。本書では (そして業界の標準では)、レイヤ3のアドレスをIPアドレスと呼びます。

レイヤ3を使用するネットワーク機器はすべて、IPアドレスを使います。スイッチはOSI参照モデルのレイヤ2で動作するため、コンピュータにパケットを転送するためには、MACアドレスからIPアドレスへの変換、またはその逆の変換が必要になります。ARP (Address Resolution Protocol) と呼ばれるレイヤ3のプロトコルを通して、この変換のプロセスを説明していきます。

コンピュータがあるIPアドレスに向けて通信したいとき、そのIPアドレスを持つマシンのMACアドレスを知るためにARPリクエストを発信します。該当するIPアドレスを持つマシン、あるいはそのIPアドレスを持つマシンがどこにあるかを知っている機器はそのリクエストに応答し、最終的な、もしくは一次的な宛先であるMACアドレスを知らせます。送信側のコンピュータは、そのMACアドレスの情報とIPアドレスの情報 (持ち主がそれぞれ一致していなくても構いません) を元に、データを送信することができるわけです。経路の情報はスイッチのARPキャッシュに記録されるため、コンピュータがデータを送信するたびにARPリクエストのブロードキャストを送信する必要はありません。

ARPキャッシュポイズニング (ARPスプーフィングとも呼ばれる) は本来、通信への割り込みやDoS攻撃 (Denial of Service attack : サービス不能攻撃) を仕掛けるために、偽装したアドレスを持ったパケットをコンピュータやネットワーク機器に送信することです。しかしながら、スイッチで構成されたネットワーク上で、特定のコンピュータのパケットをキャプチャする手段としても使うことができます。

ARPキャッシュポイズニングを使えば、偽のMACアドレスを含むARPパケットをスイッチやルータに送信し、ネットワーク機器を騙すことができるのです (図2-7)。

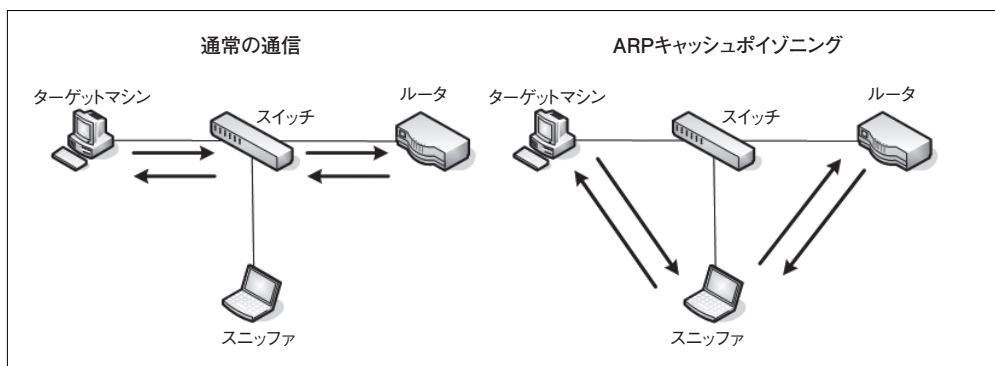


図2-7 ARP キャッシュポイズニングによって、ターゲットマシンの通信に割り込む

2.3.4 Cain & Abelの使用

ARP キャッシュポイズニングを使用するには、まずツールをインストールして必要な情報を集める必要があります。ここでは、有名なセキュリティツールである、Oxid.it (<http://www.oxid.it>) のCain & Abelを使います。それではインストールしてみてください。

Cain & Abelをインストールしたら、スニッファ用コンピュータのIPアドレス、ターゲットマシン、そのコンピュータが接続されているルータの情報などを集める必要があります。

Cain & Abelを起動すると、ウィンドウのトップにいくつかのタブが見えるはずですが (ARP キャッシュポイズニングは、Cain & Abelの多くの機能の1つにすぎません)。ARP キャッシュポイズニングは、[Sniffer] タブで利用することができます。[Sniffer] タブをクリックすると、空の表が表示されるはずです (図2-8)。

以下の手順に沿って、Cain & Abelの「スニッファ」機能 (パケットをキャプチャする、いわゆるスニффイング機能ではなく、ARP スプーフィングの機能) を使って、ネットワーク上のコンピュータをスキャンしてください。

1. ツールバーの左から2番目、NICのアイコンに似たアイコンをクリックします。初めてこのアイコンをクリックすると、スニッフイングに利用するNICを聞かれます。ARP キャッシュポイズニングを利用したいネットワークに接続しているNICを選択してください。
2. NICを選択したら、[OK] をクリックしてください。Cain & Abelのスニッファ機能が有効になります。
3. [+] アイコンをクリックし、[OK] をクリックしてください。ネットワーク上のコンピュータのリストが作成されます (図2-9)。

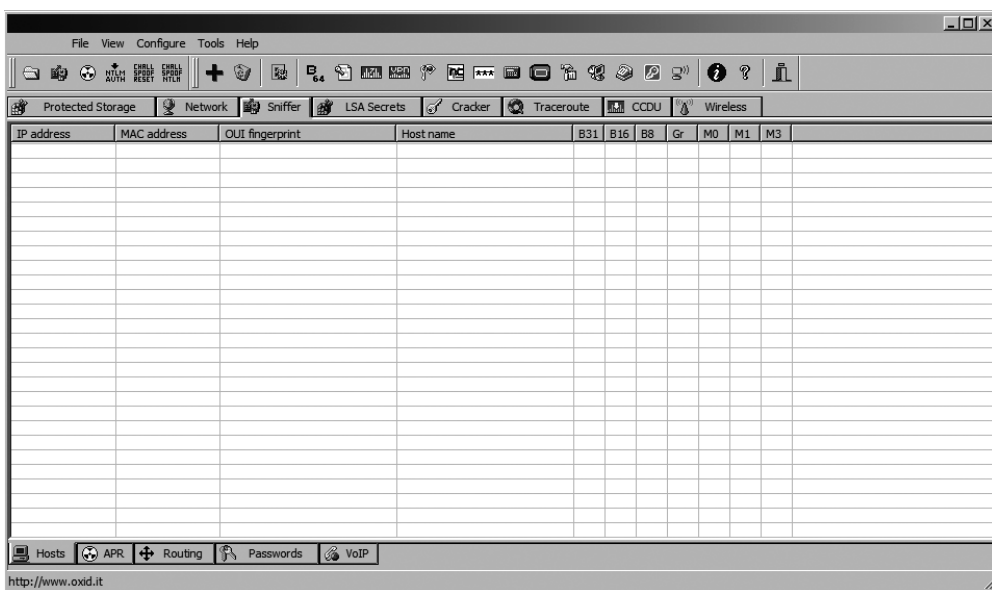


図2-8 Cain & Abelの [Sniffer] タブ

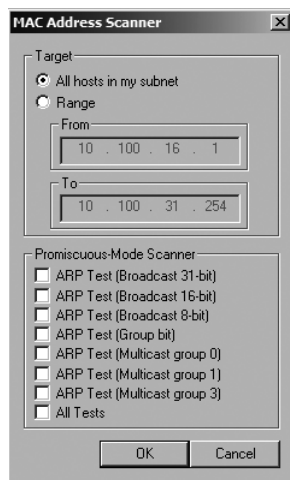


図2-9 Cain & Abelのネットワーク検出ツール

これで、空だった表にネットワークに接続しているコンピュータのMACアドレス、IPアドレス、ベンダ特有の情報などの一覧が表示されます。これらの情報を利用して、ARPキャッシュポイズニングを使用します。

ウィンドウの下部には、[Sniffer] タブ上で使用できる機能のタブが表示されているはずです。コンピュータのリストを作成したら、[ARP] タブをクリックします。

[ARP] タブをクリックすると、上下2つに分かれた空の表が表示されます。

ARP キャッシュポイズニングの設定手順は以下のとおりです。

1. ツールバー上にある [+] アイコンをクリックします。左右2つに分かれたウィンドウが表示されます。
2. 左側の表で、ARP キャッシュポイズニングによる通信の割り込みが可能なコンピュータのリストが表示されます。ターゲットマシンのIPアドレスをクリックすると、右側の表には残りのコンピュータのリストが表示されます。
3. 右側の表で、ターゲットマシンが接続されているルータのIPアドレスをクリックし、[OK] をクリックします (図2-10)。これで、標的のコンピュータとルータのIPアドレスが、メインのウィンドウの上部の表に表示されます。

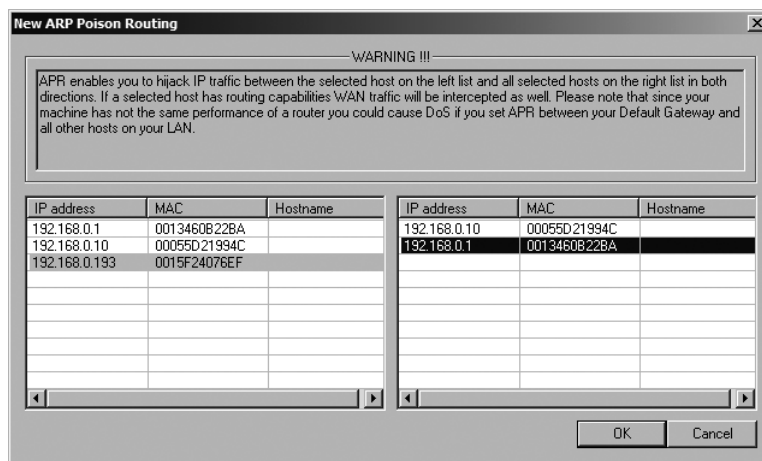


図2-10 ARP キャッシュポイズニングを利用してターゲットマシンを選択する

4. 最後に、ツールバー上にある黄色と黒の放射能のマークをクリックします。これで、Cain & AbelのARPキャッシュポイズニング機能が有効になり、ターゲットマシンとルータの通信の間に割り込むことができるようになります。

ARP キャッシュポイズニングを実行すると、上部の表にはARPキャッシュポイズニングを実行しているコンピュータの情報が、下部の表にはそのコンピュータを介して通信しているすべてのコンピュータの情報が表示されます。

これで、スニッファを使って通信を解析できるようになりました。パケットのキャプチャが終了したら、黄色と黒の放射能マークをクリックすれば、ARP キャッシュポイズニングを停止することができます。



ARP キャッシュポイズニングは、ネットワークの構成に注意して使用してください。たとえば、1Gbps でファイルサーバを利用していて、かつパケットキャプチャしているコンピュータが100Mbpsの場合など、大量のパケットが発生する場所ではこの方法を使うべきではありません。そんな状況でARP キャッシュポイズニングを利用すれば、大量のパケットが通信に割り込んでいるコンピュータに流れ込みますから、そのコンピュータがボトルネックになってしまいます。これではあなたのコンピュータがDoS 攻撃を受けているのと同じことになり、ネットワークのパフォーマンスを下げ、解析もうまくいかないでしょう[†]。

2.4 ルータで構成されたネットワークでのスニッフイング

スイッチで構成されたネットワークでのパケットキャプチャの方法は、そのままルータで構成されたネットワークでも使えます。ルータを含むネットワークでは、複数のネットワークセグメントにまたがる問題を解決しようとしたときにスニッファをどこに設置するか、ということを考えなければいけません。

すでに学んだとおり、ブロードキャストドメインはルータで途切れてしまいます。パケットがルータを通過すると、送信側のコンピュータは応答確認のパケットが受信側のコンピュータから返ってくるまで通信することはできません。ルータをまたがる環境では、ルータに接続されているすべてのネットワークセグメントを監視する必要があります。

いくつかのネットワークセグメントが複数のルータに接続されている場合を考えてみましょう (図2-11)。ネットワークDは、ネットワークBを介してネットワークAのコンピュータと通信しています。ネットワークDのコンピュータが、ネットワークAのコンピュータと通信できないという問題が発生したとしましょう。

まずネットワークDにスニッファを設置してみましょう。するとネットワークAへ送信されるパケットを見ることはできますが、ネットワークAから返ってくるはずの応答確認のパケットが見当たりません。そこでネットワークBにスニッファを設置してみると、ネットワークBのルータがパケットを破棄していることが分かりました。これで、問題の原因がネットワークBのルータの設定にあることが分かります。ルータの設定を直せば問題は解決です。複数のセグメント上にスニッファを設置しなければならない理由が、この例から分かると思います。

[†] 監訳注：選択的に特定のコンピュータの通信だけをキャプチャする場合は、それほどボトルネックにならずに済むでしょう。

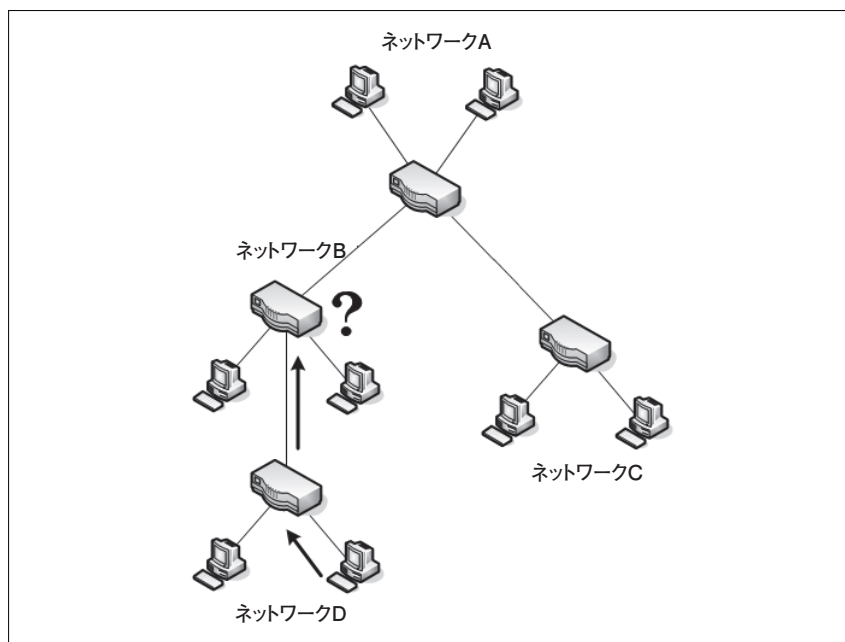


図2-11 ネットワークD上のコンピュータが、ネットワークAと通信できない

2.5 ネットワーク図

今までネットワークについての説明の中で、いくつかのネットワーク図を見てきました。ネットワーク図（またはネットワーク構成図）には、ネットワーク上のコンピュータやネットワーク機器がどのように接続されているかが描かれています。

スニッファの設置場所を決定するには、ネットワーク図を明確にすることが一番です。ネットワーク図はトラブルシューティングや解析に非常に役に立つので、可能なら常に手元に置いておきましょう。もっと詳細なネットワーク図が欲しいとさえ思うかもしれません。トラブルシューティングにおける最大の難関は、往々にして問題がどこにあるかを特定することにあります。

3章

Wireshark 概要

パケット解析に使うスニッファにはさまざまな種類がありますが、本書では Wireshark を取り上げています。この章では Wireshark の歴史、特徴、インストールと基本的な使用方法を学びます。

3.1 Wireshark の歴史

Wireshark には長い歴史があります。Wireshark はカンザスシティにあるミズーリ大学でコンピュータサイエンスを学んだジェラルド・コームズ (Gerald Combs) が作ったものです。1988 年に GPL (GNU Public License) により公開されました (当時の名前は Ethereal)。

Ethereal が公開された 8 年後、コームズは新たなキャリアを求めてそれまで勤めてきた企業を退職しました。残念ながら退職した企業が Ethereal の著作権を持っていたため、コームズは Ethereal をそれ以上開発することができなくなっていました。かわりに、コームズと Ethereal の開発チームは、2006 年半ばに Wireshark という新たな商標を取得しました。

Wireshark は非常に人気があり、劇的に成長してきました。開発には 500 人もの人がかかわっています。Ethereal という名のプログラムはもう開発されていません。

3.2 Wireshark の利点

Wireshark には、日々パケット解析を行う人にとって便利なさまざまな特徴があります。1 章で述べたスニッファの評価項目に従って Wireshark を評価してみましょう。

3.2.1 サポートされているプロトコル

現在、Wireshark は IP や DHCP のような一般的なものから、AppleTalk や BitTorrent のような特定のメーカーやソフトウェアでしか使われないちょっと珍しいものまで、850 ものプロトコルをサポートしています。Wireshark はオープンソース

モデルとして開発されており、Wireshark のアップデートごとに新しいプロトコルが追加されています。Wireshark がサポートしていないプロトコルがあれば、自分でそのプロトコルをサポートするコードを書いて Wireshark の開発者に提供することもできます。とはいっても、Wireshark がサポートしていないプロトコルはほとんどありませんが³。

3.2.2 ユーザーフレンドリかどうか

Wireshark には他のスニッファに比べ非常に分かりやすいインターフェースが備わっています。Wireshark は見やすいレイアウトの GUI ベースのアプリケーションです。たとえばデータはプロトコルごとに色分けされているなど、高いユーザビリティを持つ設計になっています。tcpdump のような難解なコマンドラインインターフェースのアプリケーションと違い、Wireshark はパケット解析を始めようという人にとって使いやすいツールといえます。

3.2.3 コスト

Wireshark はオープンソースで、GPL ライセンスのもと無償で入手することができます。個人利用でも商用利用でも、誰でも Wireshark をダウンロードして使うことができます。

3.2.4 スニッファのサポート体制

ソフトウェアの善し悪しはそのサポートによって決まるといっても過言ではありません。Wireshark のようなフリーで公開されているソフトウェアには、公式サポートというものが存在しません。オープンソースのソフトウェアのサポートは、ユーザーに頼っている部分があります。幸運なことに、Wireshark のユーザーコミュニティはオープンソースプロジェクトの中でも非常に活発です。Wireshark の Web ページには、オンラインドキュメント、開発者のための Wiki、FAQ、開発者も参加しているメーリングリストに登録するための方法が載っています。Wireshark の開発者は、質問を放置しておくようなことはありません。

3.2.5 OS のサポート

Wireshark は、Windows、Mac OS X、Linux など、現代の主要な OS のほとんどをサポートしています。Wireshark の Web ページで、サポートしている OS の一覧を見ることができます。

3.3 Wireshark のインストール

Wireshark のインストールは驚くほど簡単です。この節では、Wireshark のシステム要件を確認し、Windows と Linux それぞれで Wireshark をインストールする方法を

学びましょう。

3.3.1 システム要件

Wiresharkをインストールする前に、システムが以下の要件を満たすかどうか確認してください。

- CPU 400MHz 以上
- 60MB 以上の空き容量
- プロミスキヤスモードをサポートしている NIC
- WinPcap パケットドライバ (Windows のみ)

WinPcap パケットドライバは、Pcap パケットドライバの Windows API です。このドライバによって生のパケットデータをキャプチャしたり、それをフィルタリングしたり、NIC をプロミスキヤスモードに切り替えたりできるようになります。WinPcap パケットドライバは、<http://www.winpcap.org> から入手できます。



WinPcap を個別でダウンロードすることもできますが、Wireshark には WinPcap が同梱されているのでその必要ありません。Wireshark に同梱されている WinPcap は Wireshark での動作が確認されたバージョンですので、WinPcap は個別にインストールせず、Wireshark のパッケージからインストールすることをお勧めします。

3.3.2 Windowsでのインストール

まず、Wireshark の Web ページ (<http://www.wireshark.org>) から、最新版の Wireshark をダウンロードしてください。Web サイトの [Get It] をクリックし、ミラーサイトを選択してください。パッケージをダウンロードしたら、以下の手順でインストールしてください。

1. .exe ファイルをダブルクリックし、表示されたダイアログ上で [Next] ボタンをクリックします。
2. 使用承諾書を読み、同意するなら [I Agree] ボタンをクリックします。
3. インストールするコンポーネントを選択します。ここでは何も変更せず [Next] をクリックします (図 3-1)。
4. [Next] をクリックします。
5. Wireshark のインストール先を入力し、[Next] ボタンをクリックします。
6. WinPcap をインストールするかどうかのダイアログが表示されるので、[Install WinPcap] チェックボックスがオンになっていることを確認し、[Install] ボタンをクリックします (図 3-2)。インストールが始まります。
7. Wireshark のインストールの途中で、WinPcap パケットドライバのインストール

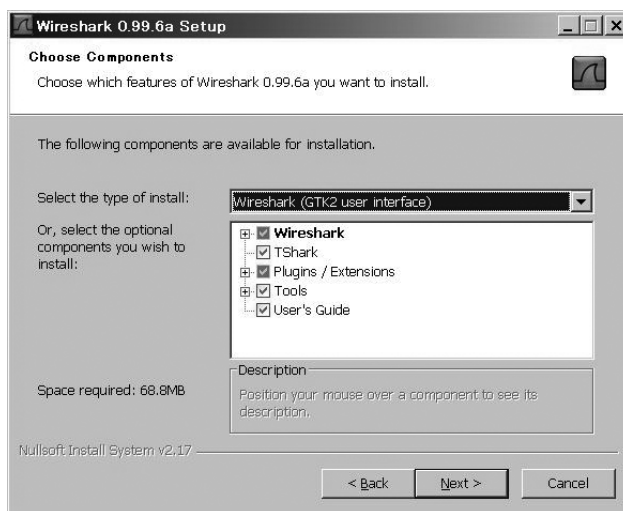


図 3-1 インストールするコンポーネントを選択

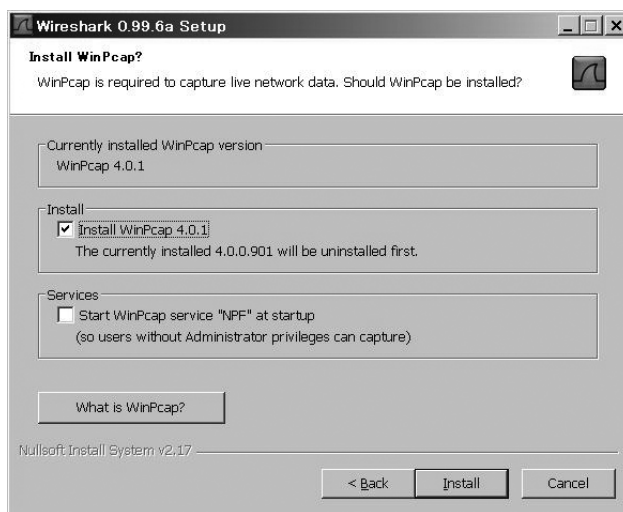


図 3-2 WinPcap パケットドライバをインストールする

が始まります。ダイアログが表示されるので [Next] ボタンをクリックし、使用承諾書を読んでから [I Agree] ボタンをクリックしてください。

8. WinPcap パケットドライバがインストールされます。終了したら [Finish] ボタンをクリックします。
9. Wireshark のインストールが続きます。終了したら [Next] ボタンをクリックします。
10. インストールの終了を示すダイアログが表示されるので、[Finish] ボタンをクリックします。

3.3.3 Linuxでのインストール

まず、インストールパッケージを入手します。ただし、すべてのLinuxディストリビューションがパッケージを提供しているわけではありません。

3.3.3.1 RPMベースのシステム

RedHatのようなRPMベースのディストリビューションにWiresharkをインストールする場合は、以下の手順に従ってください。

1. WiresharkのWebサイトから、適切なインストールパッケージをダウンロードしてください。
2. コンソールを開き、`rpm -ivh wireshark-0.99.3.i386.rpm`を実行してください。ファイル名はダウンロードしたパッケージのものに変えてください。
3. 依存するパッケージがインストールされていない場合は、それらをインストールし再度「2.」のコマンドを実行してください。

3.3.3.2 DEBベースのシステム

DebianやUbuntuのようなDEBベースのディストリビューションにWiresharkをインストールする場合は、以下の手順に従ってください。

1. WiresharkのWebサイトから、適切なインストールパッケージをダウンロードしてください。
2. コンソールを開き、`apt-get install wireshark`を実行してください[†]。

3.4 Wiresharkの基本

Wiresharkを首尾よくインストールできれば、すぐにそれを使うことができます。ついにあなたはスニッファを手にしたのです！しかし…何も表示されません！

実際のところ、Wiresharkは起動しただけでは特に興味を引くものは表示されません。面白いものを見るには何かデータが必要です。

[†] 監訳注：Fedora CoreやCentOSなどのシステムではyumを使うこともできます。インターネットに接続している環境でコンソールを開き、

```
yum install wireshark-gnome
```

を実行してください。GPGキーのエラーが出しまったときは、

```
rpm --import /usr/share/doc/fedora-release-5/RPM-GPG-KEY-fedora
```

を実行し、キーをインポートしてください (Fedora Core 5の場合)。Fedora Core 5以外でインポートするには、数字の5を該当するディストリビューションに合わせて訂正してください。

3.4.1 最初のパケットキャプチャ

Wiresharkでパケットを解析するには、まずパケットをキャプチャしなければいけません。「ネットワークに障害がないのにどうやってパケットをキャプチャするのだろうか？」と疑問に思うかもしれません。その考えには2つの間違いがあります。1つはネットワークには常に障害が存在するという事です。疑うなら全従業員にメールを送信して、すべてが完璧に届くかどうか確認してごらん下さい。

もう1つの間違いは、パケット解析は、障害があるときだけやるものではないということです。実際のところ、ネットワーク管理者はトラブルシューティングより障害のないネットワークの解析に時間を割いています。ネットワークのトラブルシューティングを効果的に行うためには、ネットワークが正常な状態にあるときの情報が必要なのです。たとえば、DHCPの障害を解決しようとしたときには、DHCPのトラフィックがどういうものなのかを理解しておく必要があります。つまりネットワークの異常を見つけるためには、正常な状態も知っておかなければならないということです。

これで基本は終了です。さっそくパケットをキャプチャしてみましょう！

1. Wiresharkを起動します。
2. メインメニューの [Capture] を選択し、[Interfaces] をクリックします。パケットをキャプチャできるNICの一覧が、IPアドレスとともにダイアログ上に表示される (図3-3) ので、キャプチャしたいNICの [Start] ボタンをクリックします。

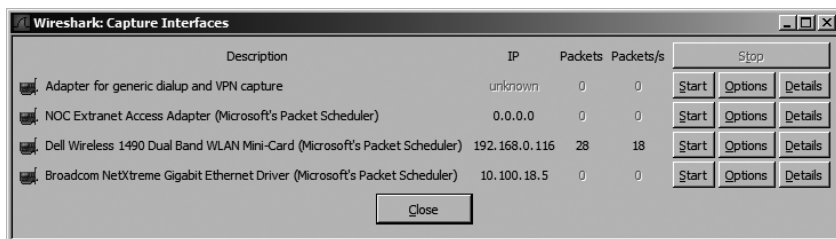


図3-3 パケットをキャプチャするNICを選択する

3. パケットのキャプチャが始まり、Wiresharkのウィンドウにキャプチャされたパケットが表示されます。このウィンドウにはパケットについてのサマリも表示されます (図3-4)†。
4. 数分待つて十分にパケットをキャプチャできたら、[Stop] ボタンをクリックします。

以上の手順でパケットキャプチャを終了すると、Wiresharkのメインウィンドウにデータが表示されます。大量のデータに圧倒されるかもしれませんが、メインウィンドウの機能を理解すれば、それほど難しいことはありません。

† 監訳注：現行バージョンはデフォルトがhideなのでパケットのサマリが表示されません。

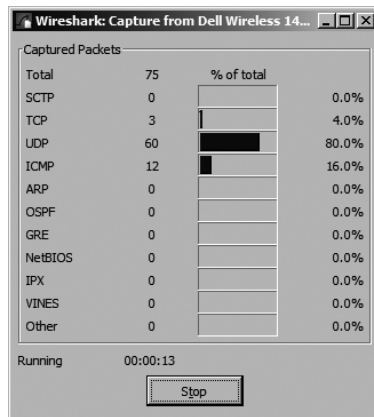


図3-4 パケットのサマリが表示される

3.4.2 メインウィンドウ

パケット解析中に一番よく見るのが、このメインウィンドウでしょう。ここにはキャプチャされたすべてのパケットが、見やすい形にフォーマットされて表示されています。たった今キャプチャしたパケットを使って、Wiresharkのメインウィンドウの見方を見てみましょう(図3-5)。メインウィンドウには、3つのペインがあります。

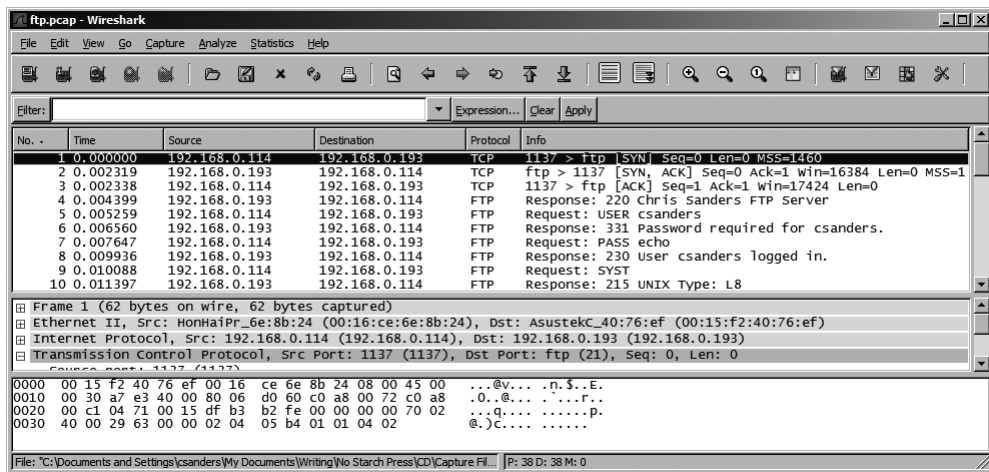


図3-5 3つのペインのメインウィンドウ

メインウィンドウの3つのペインの表示はお互いに依存しています。パケット一覧のペイン(上段)からパケットをクリックすると、パケット詳細のペイン(中段)にそのパケットの詳細が表示されます。また、バイナリのペイン(下段)でそのパケットのバイナリデータを見ることができます。

3.4.2.1 パケット一覧のペイン

上段のペインには、キャプチャファイルに存在するパケットの一覧が、パケット番号、キャプチャされた時間、パケットの送信元と送信先、パケットのプロトコル、その他パケットに含まれる情報とともに表示されています。

3.4.2.2 パケット詳細のペイン

中段のペインには、パケットの詳細がツリー状に表示されています。ツリーは最初折りたたまれています。展開することですべての情報を見ることができます。

3.4.2.3 バイナリのペイン

下段のペインには、フォーマットされる前の生のパケットが表示されています。つまり、ケーブルを通るパケットの本来の形です。このままでは解析が非常に困難だということが分かると思います。



これら3つのペインの違いをしっかりと理解しておいてください。パケット解析では、メインウィンドウを一番よく使います。

3.4.3 設定画面

Wiresharkはさまざまなカスタマイズが可能です。重要なものをいくつか見てみましょう。

Wiresharkの設定画面は、メインメニューの [Edit] から [Preferences] をクリックすると表示されます (図3-6)。

Wiresharkの設定画面は、[User Interface]、[Capture]、[Printing]、[Name Resolution]、[Protocol] の5つのセクションに分かれています。

3.4.3.1 [User Interface] セクション

Wireshark上でのデータの表示方法を設定できます。ここでは、ウィンドウの場所を記憶するかどうか、ペインのレイアウト、スクロールバーの位置、パケット一覧のペインの位置、データを表示するときのフォント、ウィンドウのカラーなどが設定できます。

3.4.3.2 [Capture] セクション

デフォルトのNIC、プロミスキヤスモードをデフォルトで使用するか、パケット一覧のペインにリアルタイムにキャプチャしたパケットを表示するかなど、キャプチャの方法について設定できます。

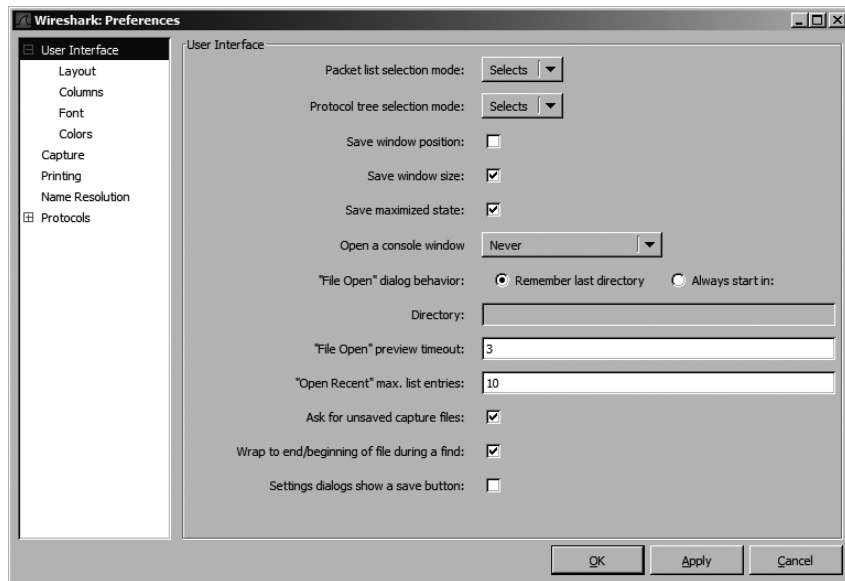


図 3-6 設定画面から Wireshark をカスタマイズする

3.4.3.3 [Printing] セクション

データをプリントアウトする際のオプションを設定できます。

3.4.3.4 [Name Resolution] セクション

MAC アドレス、コンピュータ名、ポート番号の名前解決をするかどうかを設定できます。また、名前解決のリクエストの最大数も設定できます。

3.4.3.5 [Protocols] セクション

プロトコルの表示方法についてのオプションを設定できます。すべてのプロトコルを設定できるわけではありませんが、さまざまな変更が可能なプロトコルもあります。しかしながら、これらのオプションは特に理由がなければ変更しないほうがよいでしょう。

3.4.4 パケットの色分け

Wireshark でパケットをキャプチャすると、パケット一覧のペイン上でパケットがさまざまな色に色分けされることに気づくでしょう (図 3-7)。これらの色はランダムに決められているように見えますが、そうではありません。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.100.17.47	10.100.16.15	DCERPC	Request: call_id: 95 opnum: 69 ctx_id: 0
2	0.000361	10.100.16.15	10.100.17.47	DCERPC	Response: call_id: 95 ctx_id: 0
3	0.001946	10.100.17.47	10.100.16.15	DCERPC	Request: call_id: 96 opnum: 26 ctx_id: 0
4	0.002034	10.100.16.15	10.100.17.47	DCERPC	Response: call_id: 96 ctx_id: 0

図 3-7 プロトコルごとに色分けされ、見やすくなっている



本書中のトラフィックという単語は、パケット一覧のペインに表示されている一連のパケットのことだと思ってください。たとえばDNSのトラフィックと言ったときには、パケット一覧のペインに表示されているすべてのDNSのパケットのことです。

各パケットはプロトコルごとに色分けされています。たとえば、DNSのトラフィックは青、HTTPのトラフィックは緑といった具合です。プロトコルごとに色分けされているおかげで、パケット一覧のペインに表示されているすべてのパケットを1つ1つ確認する必要がありません。巨大なキャプチャファイルを扱うときに、この色分けの機能のおかげで解析が大幅にスピードアップすることを実感することでしょう。

色分けのルールは、[Coloring Rules] ダイアログで設定が可能です。設定は以下の手順で行います。

1. Wireshark を起動します。
2. メインメニューの [View] をクリックします。
3. [Coloring Rules] をクリックし、[Coloring Rules] ダイアログを表示します (図 3-8)。色分けのルールの一覧が表示されるので、好みの色に変更します。

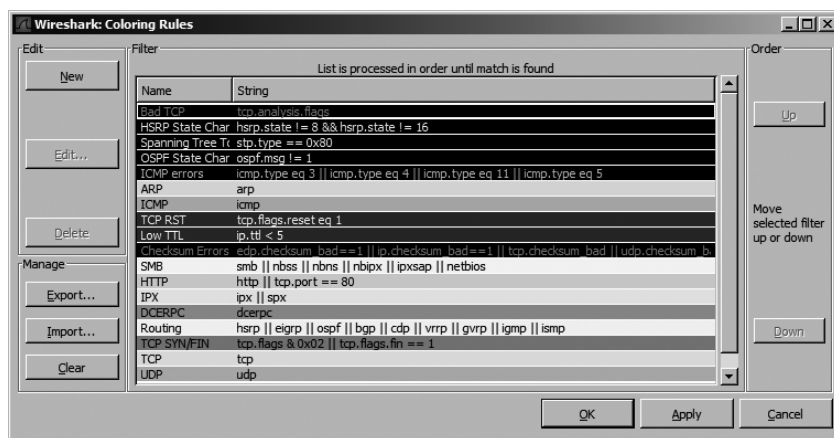


図 3-8 [Coloring Rules] ダイアログで、パケットの色分けルールを設定する

たとえば、HTTPのトラフィックの背景色をデフォルトの緑からラベンダーに変える手順は、以下のとおりです。

1. Wireshark を起動し、メインメニューの [View] から [Coloring Rules] をクリックして、[Coloring Rules] ダイアログを表示します。
2. 一覧からHTTPの色分けルールをクリックします。
3. [Edit] ボタンをクリックします (図 3-9)。

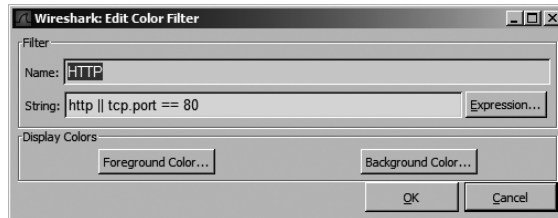


図 3-9 [Coloring Rules] ダイアログでは、文字色と背景色を設定できる

4. [Background Color] ボタンをクリックします。
5. 色を選択し、[OK] ボタンをクリックします。
6. [OK] ボタンを2回押してメインウィンドウに戻ります。
7. 該当するパケットが、設定した色に変更されています。

Wiresharkを使ってパケット解析をしていると、あるプロトコルが他のプロトコルより多く表示されていることに気づくでしょう。色分けされていることでそれがより分かりやすくなるのです。たとえばDHCPサーバに障害が起こりIPアドレスの割り当てがうまくいかなかった場合、DHCPのトラフィックを黄色に色分けしていれば、メインウィンドウが黄色に染まります。こうして、DHCPのトラフィックの選別が容易になり、より効率よくパケット解析ができるようになります。

4 章

Wiresharkでの パケットキャプチャのテクニック

前章で初めてのパケットキャプチャはうまくいったので、キャプチャしたパケットの基本的な操作方法についてお話しします。パケットのマーキング、キャプチャファイルの保存、キャプチャファイルのマージ、プリントアウト、時間の表示フォーマットの変更を学びます。

4.1 パケットの検索とマーキング

パケット解析を始めると、膨大な量のパケットに遭遇することになります。数千、数百万とパケットの数が膨れ上がってくると、よほど効率的に解析しないと対応しきれなくなるでしょう。このためWiresharkでは、ある法則に従い、パケットをマーキングすることができるようになっています。

4.1.1 パケットの検索

特定のパケットを見つけるには、[Find Packet] ダイアログを使います(図4-1)。メインメニューの[Edit]から[Find Packet]をクリックします。また、Ctrl-Fでもダイアログを表示することができます。

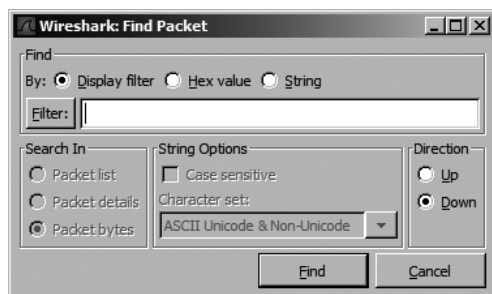


図4-1 Wiresharkで特定のパケットを見つける

パケットの検索には、[Display filter]、[Hex value]、[String] の3つのオプションがあります。[Display filter] オプションでは、検索するフィルタの項目を入力します (フィルタについては後述)。[Hex value] および [String] では、検索するパケットを16進数または文字列で指定します。表4-1にそれぞれの例があります。他のオプションとして、[Search in] (検索するペイン)、[String Options] (使用するキャラクターセットの設定)、[Direction] (検索する方向) があります。

表4-1 パケット検索の例

検索のタイプ	例
Display filter	not ip, ip address==192.168.0.1, arp
Hex value	00:ff, ff:ff, 00:AB:B1:f0
String	ワークステーション 1、ユーザー B、ドメイン名

オプションを選択し、テキストボックスに検索する文字列を入力して、[Find] ボタンをクリックすれば、条件に一致するパケットが表示されます。次候補を検索する場合にはCtrl-Nを、前候補を検索する場合にはCtrl-Bを押してください。

4.1.2 パケットのマーキング

検索したパケットに、マーキングしておくことができます。マーキングされたパケットは、図4-2のように黒地に白文字になり目立つようになります。また、キャプチャされたパケットをファイルに保存するときに、マーキングしたパケットのみを保存することも可能です。あるパケットを分けて保存しておきたい場合や、色を付けてあとから簡単に見つけられるようにしておきたい場合など、マーキングの使い道はいろいろあります。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.100.17.47	10.100.16.15	DCERPC	Request: call_id: 95 opnum: 69 ctx_id: 0
2	0.000361	10.100.16.15	10.100.17.47	DCERPC	Response: call_id: 95 ctx_id: 0

図4-2 マーキングされたパケットとされていないパケットとの比較。普通とは違う色で表示される。この図では1番目のパケットがマーキングされている

パケットをマーキングするには、パケット一覧のペインでパケットを右クリックし、ポップアップメニューから [Mark Packet] を選んでください。また、パケットをクリックし、Ctrl-Mを押すことでもマーキングできます。Ctrl-Mをもう一度押せばマーキングを解除することができます。複数のパケットをマーキングした場合、Shift-Ctrl-NまたはShift-Ctrl-Bでマーキングされたパケットにジャンプすることができます。

4.2 キャプチャファイルの保存とエクスポート

パケット解析は、パケットキャプチャと同時にできるわけではありません。通常、パケットをキャプチャして、保存して、それから解析を開始します。そのため、Wiresharkにはキャプチャしたパケットをキャプチャファイルとして保存する機能が付いています。

4.2.1 キャプチャファイルの保存

キャプチャしたパケットを保存するには、メインメニューの [File] をクリックし、[Save As] を選択します。または、キーボードから Shift-Ctrl-S でも保存が可能です。[Save file as] ダイアログが表示されるので (図4-3)、キャプチャファイルを保存する場所とファイル形式を選択します。何も指定しない場合は、.pcap ファイル形式で保存します。

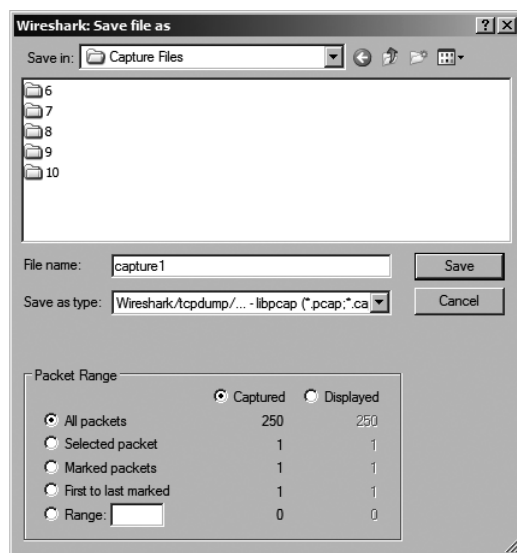


図4-3 [Save file as] ダイアログからキャプチャファイルを保存

Wiresharkでは、ある範囲のパケット番号を持つパケット、マーキングされたパケット、ディスプレイフィルタによって表示されたパケットなど、特定のパケットを保存することができます。これにより、キャプチャファイルのサイズをかなり小さくすることが可能です。

4.2.2 キャプチャデータのエクスポート

Wiresharkでは、テキスト、ポストスクリプト、CSV、XMLなど、他のパケット解析ツールのキャプチャファイルの保存形式にキャプチャデータをエクスポートする

ことができます。エクスポートするには、メインメニューの [File] から [Export] をクリックし、保存形式を選択してください。[Save As] から保存するときにも、[ファイルの種類] から保存形式を選択することができます。

4.3 キャプチャファイルのマージ

パケット解析をしていると、複数のキャプチャファイルをマージしたくなることがあります。Wireshark には、キャプチャファイルをマージする方法が2つあります。

キャプチャファイルをマージするには、以下の手順に従ってください。

1. マージしたいキャプチャファイルを開きます。
2. メインメニューの [File] から [Merge] を選択し、[Merge with Capture File] ダイアログを開きます (図 4-4)。

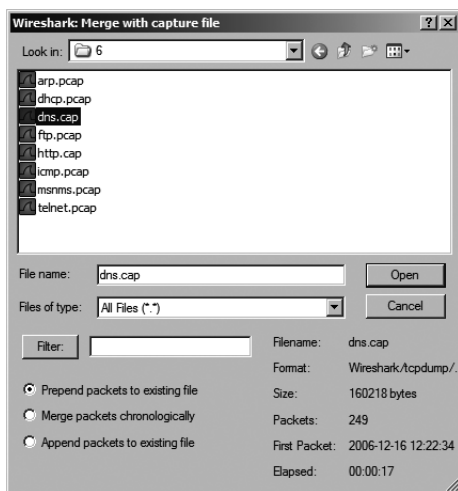


図 4-4 [Merge with capture file] から2つのファイルをマージする

3. マージしたいファイルを選択し、どのようにマージするかを選択します。マージする方法には、[Prepend packets to existing file] (現在表示されているパケットの前にマージするキャプチャファイルのパケットを追加する)、[Merge packet chronologically] (タイムスタンプに沿って時系列に追加する)、[Append packets to existing file] (現在表示されているパケットの後にマージするキャプチャファイルのパケットを追加する) の3つがあります。

マージしたいキャプチャファイルをエクスプローラから Wireshark にドラッグ&ドロップすることで、複数のファイルをマージし時系列に表示することができます。

4.4 パケットの印刷

実際の解析は画面上で行われることがほとんどですが、データを印刷する必要があるかもしれません。キャプチャしたパケットを印刷するには、メインメニューの [File] から [Print] を選択します (図 4-5)。

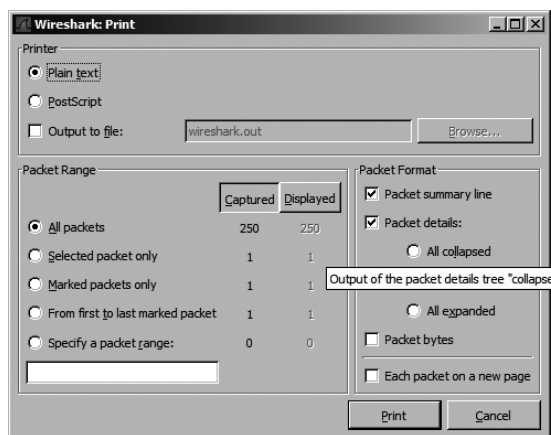


図 4-5 [Print] ダイアログからパケットの印刷ができる

[Print] ダイアログから、選択したデータをテキストまたはポストスクリプトとして印刷するか、ファイルとして出力することができます。[Save file as] ダイアログと同じように、ある範囲のパケット番号を持つパケット、マーキングされたパケット、ディスプレイフィルタによって表示されたパケットなど、特定のパケットのみを印刷することができます。また、3つのペインのうちどれを印刷するかを選択することも可能です。オプションを選択したら [Print] をクリックしてください。

4.5 時間の表示フォーマットと相対時間表示

パケット解析において、時間は重要な要素です。解析するには、通信にかかる時間とその傾向を調べる必要があります。Wireshark は時間の重要性を認識して、いくつかのオプションが提供されています。ここでは、時間の表示フォーマットと相対時間表示を見ていきましょう。

4.5.1 時間の表示フォーマット

Wireshark では、各パケットにはシステム時刻を元にタイムスタンプが記録されています。パケットがキャプチャされたときのシステム時刻や、最初にキャプチャされたパケットからの相対的な時間を表示することもできます。

時間の表示に関するオプションは、メインメニューの [View] の [Time Display Format]

から設定できます (図 4-6)。時間の表示フォーマットのほか、時間の精度についても選択が可能です。秒、ミリ秒、マイクロ秒などを指定できます。本書では頻繁にこれらのオプションを変更するので、今のうちに慣れておいてください。

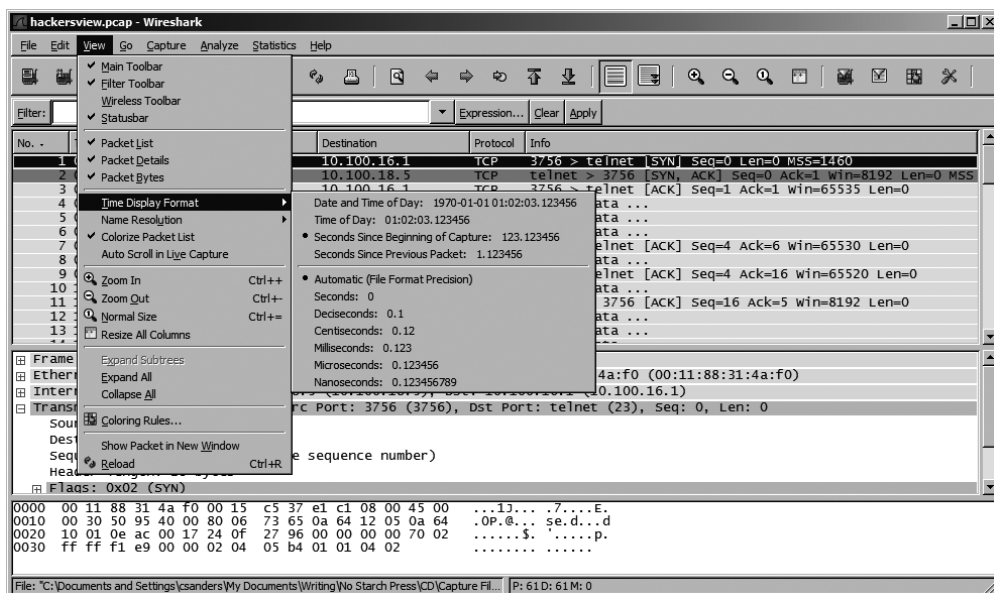


図 4-6 時間のオプションは頻繁に変えることになる

4.5.2 相対時間表示

Wireshark では、あるパケットがキャプチャされた時間からの相対的な時間を表示することができます。この機能を使えば、キャプチャファイルでデータの要求 (リクエスト) が複数行われているとき、そのリクエストが終了するまでにどのくらいの時間がかかるかが簡単に分かります。

相対的な時間を表示するには、パケット一覧のペインから基準となるパケットをクリックし、メインメニューの [Edit] から [Set Time Reference] を選択します。または、時間の基準となるパケットをクリックし、Ctrl-T を押してください。もう一度同じ動作を繰り返せば、基準を解除することができます。

相対的な時間を表示するよう設定すると、時間の基準となったパケットの時間の部分に *REF* と表示されます (図 4-7)。

No. -	Time	Source	Destination	Protocol	Info
3	0.001263	10.100.18.5	10.100.16.1	TCP	3756 > telnet [ACK] Seq=1 Ack=1 win=65535 Len=0
4	0.000058	10.100.18.5	10.100.16.1	TELNET	telnet [REQ] DATA ...
5	0.000058	10.100.18.5	10.100.16.1	TELNET	telnet Data ...
6	0.001018	10.100.16.1	10.100.18.5	TELNET	telnet Data ...

図 4-7 時間の基準になったパケット



相対的な時間の表示は、時間の表示フォーマットを [Seconds Since Beginning of Capture] (キャプチャを開始してからの経過時間) にしておかないと意味がありません。ほかのフォーマットでは混乱を招くでしょう。

4.6 キャプチャフィルタとディスプレイフィルタ

4.2.1 節で、フィルタを使ったパケットの保存について触れました。フィルタを使うことによって、特定のパケットのみを表示させることができます。フィルタは文字列で表され、パケットの表示/非表示を Wireshark に指示します。

Wireshark には、キャプチャフィルタとディスプレイフィルタの2種類があります。

4.6.1 キャプチャフィルタ

キャプチャフィルタは、パケットをキャプチャしている最中に使用されるもので、WinPcap により適用されます。キャプチャフィルタの構文は他のパケット解析ツールでも使用できる場合があります。[Capture Options] ダイアログから、キャプチャするトラフィックを指定することができます。

キャプチャフィルタは、複数のサービスを提供しているサーバに関するトラフィックを解析したいときに役に立ちます。たとえば、262 番ポートを使用するサービスを提供しているサーバのトラブルシューティングを考えてみましょう。サーバがさまざまなポートでサービスを提供しているのならば、262 番ポートのトラフィックのみを見るのは大変ですが、フィルタを使えばそれが可能です。以下の手順でフィルタを作成してください。

1. [Capture Options] ダイアログを開き (図 4-8)、キャプチャする NIC を選択します。
2. [Capture Filter] ボタンの横のテキストボックスにフィルタを記述します。または [Capture Filter] ボタンを押して、フィルタの作成を支援するダイアログを開き、フィルタを作成します。今回は、262 番ポートを通るトラフィックだけをキャプチャしたいので、テキストボックスに「port 262」と記述します (図 4-8)。
3. フィルタを作成したら、[Start] ボタンをクリックしてキャプチャを始めてください。これで、262 番ポートを通るトラフィックのみがキャプチャされます。

4.6.2 ディスプレイフィルタ

ディスプレイフィルタは、作成されたキャプチャファイルに適用されるフィルタです。フィルタに一致するパケットのみを表示させます。パケット一覧のペインの上部にあるテキストボックスにフィルタを記述します。

ディスプレイフィルタはキャプチャフィルタより使う機会が多いでしょう。実際の

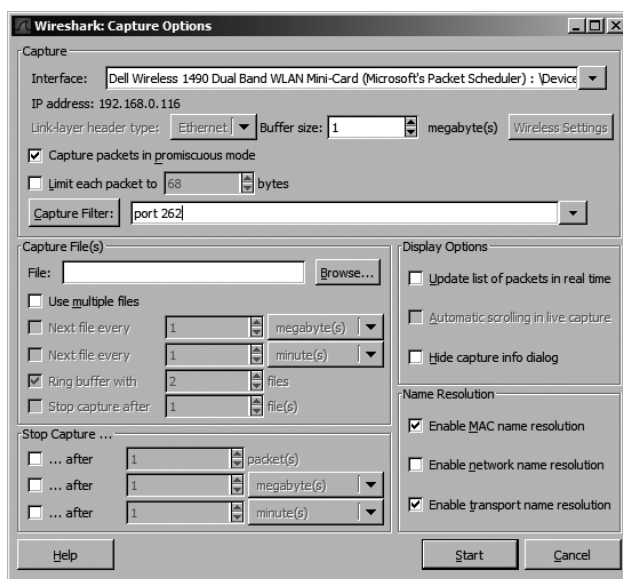


図4-8 [Capture Options] ダイアログでキャプチャフィルタを作成する

キャプチャファイルのデータを変更することなく、特定のパケットだけを表示することができからです。もともとのキャプチャファイルのパケットを表示させたいなら、テキストボックスに記述したフィルタを消せばよいのです。

フィルタは、キャプチャファイルから解析に無関係のパケット（ARPブロードキャストパケットなど）を一時的に消去する役にも立ちます。しかしながら、ARPブロードキャストパケットは後で解析に必要な場合があるので、キャプチャフィルタを使うよりも、ディスプレイフィルタで一時的に表示させないようにするほうが便利なのです。

ARPパケットを表示させないようにするには、以下の手順に従ってください。

1. パケット一覧のペインの上部にある、[Filter] テキストボックスに移動します。
2. `!arp`と入力してEnterキーを押します（図4-9）。フィルタを削除するには、テキストボックスの中身を消去しEnterキーを押します。



図4-9 [Filter] テキストボックスでディスプレイフィルタを作成する

4.6.3 [Filter Expression] ダイアログ

[Filter Expression] ダイアログ（図4-10）は、Wireshark 初心者がキャプチャフィルタやディスプレイフィルタを作成する支援をしてくれる機能です。ダイアログを表示するには、メインメニューの[Analyze]から[Display Filter]を選択し、[Display Filter]

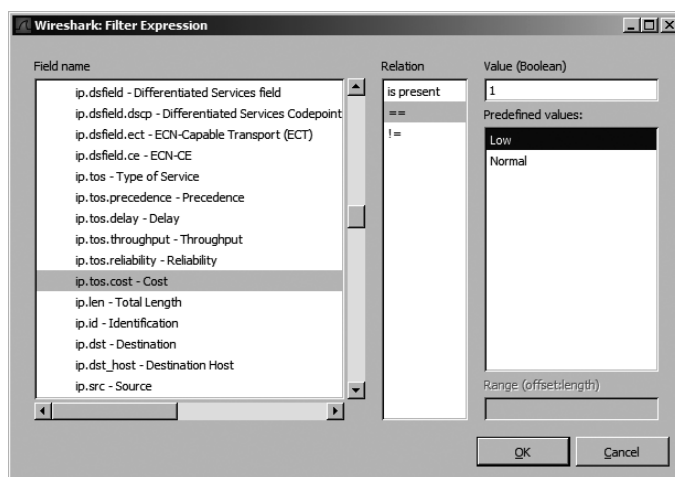


図 4-10 [Filter Expression] ダイアログを使えば、より簡単にフィルタを作成できる

ダイアログの [Expression] をクリックします。

[Filter Expression] ダイアログの左側には、使用可能なプロトコルの一覧が表示されており、各プロトコルで利用可能なフィルタ要素を指定できます。フィルタを作成するには、以下の手順に従ってください。

1. プロトコル名の左にある [+] をクリックすれば、各プロトコルで利用可能なフィルタ要素を見ることができます。利用したいフィルタ要素をクリックしてください。
2. 選択したフィルタ要素と、その評価値の評価方法を指定してください。評価方法は、イコール (=)、大なり (>)、小なり (<) などの演算子です。
3. 評価値を指定して、フィルタを作成します。Wireshark が提供する評価値を選択するか、自身で値を指定してください。
4. フィルタを作成したら、[OK] ボタンをクリックしてください。作成したフィルタがテキストで表示されます。

4.6.4 フィルタを自力で作る

[Filter Expression] ダイアログは初心者には非常に便利な機能ですが、フィルタの使用方法が理解できれば、手でフィルタを作成するほうが効率がよいでしょう。

ディスプレイフィルタは非常に強力ですが、構文は簡単です。この構文は Wireshark 独自のものです。フィルタの構文の種類といくつかの例を見ていきましょう。

4.6.4.1 特定のプロトコルをフィルタリングする

キャプチャフィルタやディスプレイフィルタは、特定のプロトコルを選別するときに使うことが多いでしょう。たとえば TCP のトラブルシューティングの場合は、TCP

のトラフィック以外は必要ないので、TCP以外のものをフィルタリングしてしましましょう。

トラブルシューティングのためにpingを多用して、ICMPのトラフィックが大量に発生した場合を考えてみましょう。`!icmp`というフィルタを使えば、ICMPのトラフィックを削除することができます。

4.6.4.2 比較演算子

比較演算子を使えば、パケットを比較することができます。たとえば特定のIPアドレスを含むパケットを見たい場合、「`==`」という比較演算子を使って、`ip.addr==192.168.0.1`というフィルタを入力すれば、192.168.0.1というIPアドレスを含むパケットのみを表示することができます。

また、長さが128バイト以下のパケットのみを表示するというような高度な使い方もできます。この場合は、「`<=`」という比較演算子を使って、`frame.pkt_len<=128`というフィルタを作ればよいのです。

Wiresharkで使用可能な比較演算子は表4-2のとおりです。

表4-2 Wiresharkのフィルタとして使用できる比較演算子

演算子	説明
<code>==</code>	等しい
<code>!=</code>	等しくない
<code>></code>	大なり
<code><</code>	小なり
<code>>=</code>	以上
<code><=</code>	以下

4.6.4.3 論理演算子

論理演算子を使えば、複数のフィルタを1つの表現として使用することができます。論理演算子を使いこなすことができれば、使用できるフィルタが飛躍的に増えます。

たとえば、前述の例では特定のIPアドレスを含むパケットのみを表示しています。今度は192.168.0.1または192.168.0.2というIPアドレスを含むパケットを見たい場合を考えてください。その場合、`ip.addr==192.168.0.1 or ip.addr==192.168.0.2`というフィルタを使えばよいのです。Wiresharkで使用可能な論理演算子は表4-3のとおりです。

表4-3 Wiresharkのフィルタとして使用できる論理演算子

演算子	概要
<code>and</code>	論理積
<code>or</code>	論理和
<code>xor</code>	排他的論理和
<code>not</code>	否定

4.6.4.4 フィルタのサンプル

フィルタの概念は単純ですが、実際にフィルタを作成するときには、どんなキーワードや演算子を使ったらよいか悩むところでしょう。本書ではすべてのキーワードや演算子を紹介はしませんので、WiresharkのWebサイトを参照してください。表4-4にフィルタのいくつかのサンプルを載せておきます。

表4-4 キャプチャフィルタとディスプレイフィルタのサンプル

フィルタ	説明
host www.example.com	www.example.comのトラフィックを表示
host www.example.com and not (port 80)	www.example.comのHTTP (80番ポート) 以外のトラフィックを表示
!dns	DNSのトラフィック以外を表示
not broadcast and not multicast	ユニキャストのトラフィックを表示
ip.dst==192.168.0.1	宛先が192.168.0.1のトラフィックを表示

4.6.5 フィルタの保存

フィルタを使用していると、特定のフィルタを頻繁に使うことがあります。同じフィルタを何度も作成する必要はありません。Wiresharkには、フィルタを保存する機能が付いているのです。

フィルタを保存するには、以下の手順に従ってください。

1. メインメニューの[Analyze]から[Display Filter]を選択し、[Display Filter]ダイアログを表示します(図4-11)。

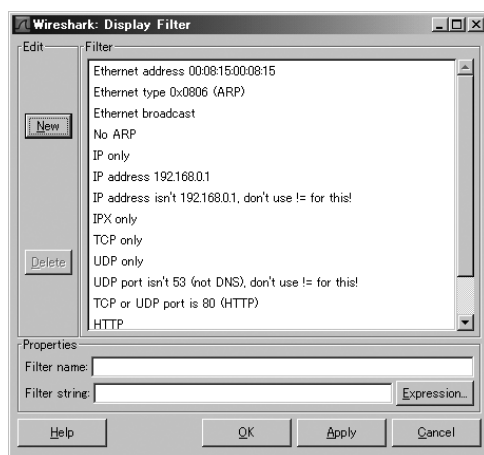


図4-11 [Display Filter] ダイアログからフィルタを保存できる

2. [New] ボタンをクリックして、新しいフィルタを作成します。
3. [Filter name] テキストボックスにフィルタの名前を入力します。
4. [Filter string] テキストボックスにフィルタを入力します。

5. フィルタを入力したら、[Save] ボタンをクリックしてフィルタを保存します。

Wireshark には、ビルトインのフィルタがいくつかありますが、これらはフィルタがどのようなものかを示す例にすぎません。しかし、それを元に実用的なフィルタを作成することができるでしょう。

5 章

Wiresharkの高度な機能

Wiresharkの基本的な使い方をマスターしたら、より高度な機能を使いたくなるでしょう。この章では、Wiresharkの名前解決、プロトコル解析、パケットのリアセンブルなどの高度な機能について説明します。

5.1 名前解決

ネットワーク上のデータは00:16:CE:6E:8B:24というような、覚えにくい英数字のアドレスを使って転送されています。名前解決とは、各プロトコルが使用するアドレスを別のものに変換するプロセスのことです。たとえば00:16:CE:6E:8B:24というMACアドレスは、DNSとARPによってMarketing-2という名前に変換されます。暗号のようなアドレスを読みやすいアドレスに変換することによって、コンピュータを識別しやすいようにしているのです。

名前解決を使って、キャプチャファイルをより読みやすくすれば、解析の時間を節約することができます。たとえばDNSを使えば、あるパケットの送信元のコンピュータを容易に識別することができます。

5.1.1 Wiresharkの名前解決ツール

Wiresharkには、MACアドレス、IPアドレス、ポート番号の3つの名前解決ツールがあります。

5.1.1.1 MACアドレスの名前解決

ARPを使って、00:09:5B:01:02:03というようなMACアドレスを10.100.12.1というようなIPアドレスに変換します。IPアドレスに変換できない場合は、MACアドレスの先頭3バイトをNetgear_01:02:03というようにIEEEが定めたメーカー名に変換します。

5.1.1.2 IPアドレスの名前解決

192.168.1.50というようなIPアドレスを、DNSを使用してMarketingPC1というような読みやすい名前に変換します。

5.1.1.3 ポート番号の名前解決

ポート番号を名前に変換します。たとえば80番ポートをhttpに変換します。

5.1.2 名前解決を有効にする

メインメニューの[Capture]から[Option]を選択するか、Ctrl-Kを押して[Capture Options]ダイアログを表示してください(図5-1)。そこから名前解決を有効にすることができます。

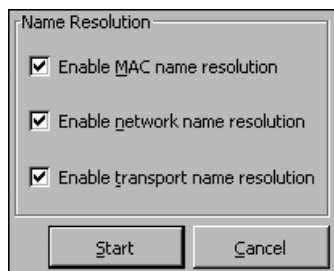


図5-1 [Capture Options] ダイアログから名前解決を有効にする

5.1.3 名前解決の欠点

名前解決はいいことづくしのように見えますが、以下のような欠点があります。

- ネームサーバがリクエストされた名前を解決できなければ、名前解決はできません。
- 名前の情報はキャプチャファイルに保存されないため、キャプチャファイルを開くたびに名前解決を行う必要があります。そのためキャプチャファイルを開いたときにネームサーバに接続できなければ名前解決できません。
- DNSのパケットがキャプチャファイルに追加されるため、キャプチャファイルが見にくくなるかもしれません。
- 名前解決のための処理時間が必要になります。巨大なキャプチャファイルを扱っていてメモリが不足しているときには、名前解決は行わないほうがよいかもしれません。

5.2 プロトコルの分析

ftp-netbios3.pcap

Wiresharkは各プロトコルごとの分析機能を持っており、その機能を使ってプロトコルをさまざまな要素に分解することで、解析がしやすいうにしています。たとえば生のパケットをICMPの分析機能を使って、ICMP特有の各ヘッダやデータに分けて表示するという具合です。分析機能は、生のパケットをプロトコルとしてWiresharkに表示させる翻訳機のようなものです。Wiresharkがあるプロトコルの分析機能を持っていれば、そのプロトコルをサポートしているといえます。

Wiresharkは各パケットを解釈するのに複数のプロトコルの分析機能を使用します。どの分析機能を使用するかは、プログラム化された論理から推測します。

残念ながら、Wiresharkがいつでも正しい分析機能を選択するとは限りません。デフォルトのポートを使用しないプロトコルは特に間違えやすいです。そういうときには、使用する分析機能を変更することができます。

たとえば、ftp-netbios3.pcapを開いてみてください。このファイルにはNetBIOSを使った通信が記録されています。しかしながら、パケットをクリックしてバイナリのペインを見てみれば、明らかにNetBIOSではないパケットが含まれていることに気づくでしょう。実際、パケット5と8では、ユーザー名とパスワードが送信されていることが分かります。

少し調べてみれば、このファイルに記録されているのはFTPの通信であることが分かるでしょう。図5-2には「FTP Server」という単語が記録されています。Wiresharkは、このFTPの通信がNetBIOSのデフォルトのポート137番で行われているために、NetBIOSの通信として分析してしまったのです。

0000	00 0b 97 2c 9d 66 00 0c	29 8e a7 f2 08 00 45 10	...,f..).....E.
0010	00 64 8f d9 40 00 40 06	27 4f c0 a8 01 08 c0 a8	.d.@.@.'O.....
0020	01 03 00 89 06 5c 38 db	a7 d8 e3 85 50 c1 50 18\8.P.P.
0030	16 d0 6b 56 00 00 32 32	30 20 6c 6f 63 61 6c 68	..kv..22 0 localh
0040	6f 73 74 2e 6c 6f 63 61	6c 64 6f 6d 61 69 6e 20	ost.local ldomain
0050	46 54 50 20 73 65 72 76	65 72 20 28 56 65 72 73	FTP serv er (vers
0060	69 6f 6e 20 35 2e 36 30	29 20 72 65 61 64 79 2e	ion 5.60) ready.
0070	0d 0a		..

図5-2 これはNetBIOSのトラフィックでなくFTPのトラフィック

この問題を解決するためには、FTPの分析機能を使うようにWiresharkに指示しなければなりません。以下の手順に従って設定してください。

1. パケットを右クリックして「Decode As」を選択します。どのプロトコルの分析機能を使うかを選択するダイアログが表示されます(図5-3)。
2. 「Transport」タブでドロップダウンメニューから「source (137)」を選択し、「FTP」をクリックします。これで、137番ポートのトラフィックでFTPの分析機能が利用されます。
3. 「OK」ボタンをクリックすれば、キャプチャファイルに即座に変更が適用されま

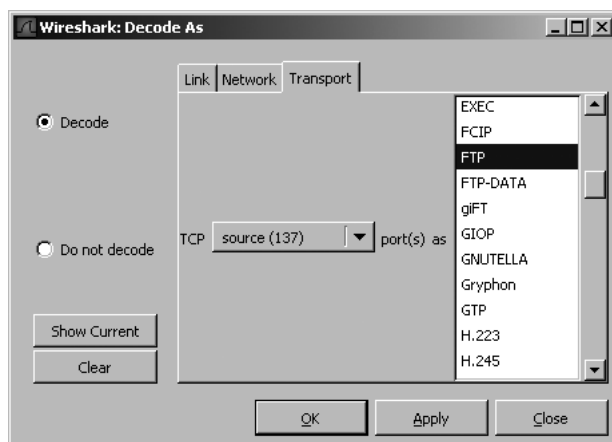


図5-3 [Decode As] ダイアログで分析機能を指定する

す。パケット一覧のペインから、正しく分析されているかどうかを確認してください。



分析機能の変更はキャプチャファイルには保存されません。キャプチャファイルを開くたびに、分析機能を変更する必要があります。

1つのキャプチャファイル上でなら、同じ変更を繰り返し行うことができます。複数のキャプチャファイルを扱っているときに分析機能の変更をいちいち覚えていられないでしょうから、Wiresharkが代わりに覚えておいてくれます。[Decode As] ダイアログの [Show Current] ボタンをクリックすれば、今までに行った変更の一覧を見ることができます(図5-4)。[Clear] ボタンを押すことで、それらの変更をクリアすることも可能です。

5.3 TCP ストリームの表示

suspectemployeechat.dmp

Wireshark のもっとも役に立つ機能の1つが、TCP ストリームの表示です。この機能はパケットを結合し、エンドユーザーが使用しているアプリケーションが受け取るデータの形にして表示します。クライアントからサーバに送信される細かいデータの破片を見るよりも、TCP ストリームとしてまとめられたデータを見るほうが簡単です。

たとえば、新人のIT技術者が管理するサーバをハッキング(クラッキング)した疑いがある社員のインスタントメッセージャー(Instant Messenger: IM)の通信を解析しようとするときに、この機能は役立ちます。サンプルファイルの suspectemployeechat.dmp を開いてみてください。このキャプチャファイルの中には、IM クライアントの MSN Messenger による通信が記録されています。パケット一覧のペインに表示

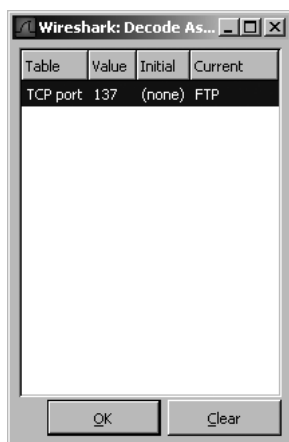


図5-4 [Show Current] ボタンを押すと今までに行った変更の一覧が表示される

されている「MSNMS」はMSN Messengerのトラフィックであることを意味しています。

1つ1つのパケットを見ると、それぞれに少しずつテキストが含まれていることが分かるでしょう。チャットで何を話しているのかを知るためには、何時間もかけて各パケットに含まれるテキストをつなげていかなければいけません。TCPストリームを使えば、もっと簡単に何を話しているのかが分かります。

TCPストリームを表示するには、パケットを右クリックして[Follow TCP Stream]を選択します。TCPストリームのウィンドウが表示され、疑わしい人物のチャットの内容を見ることができます(図5-5)。

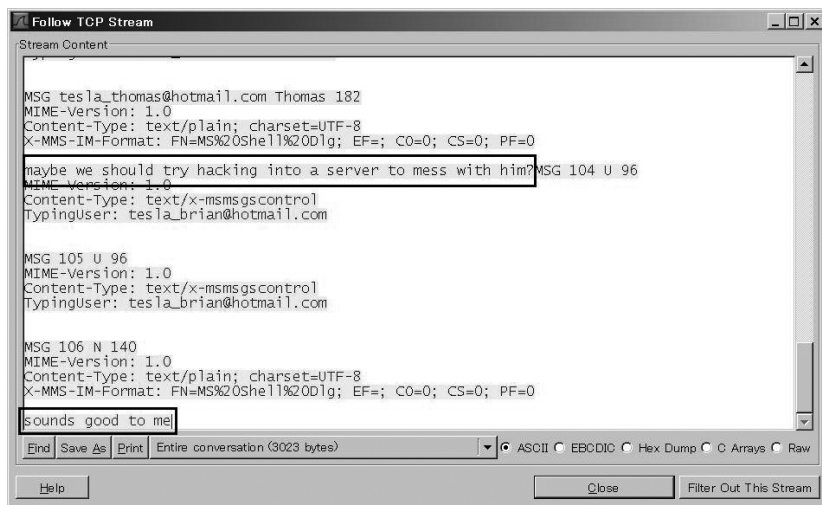


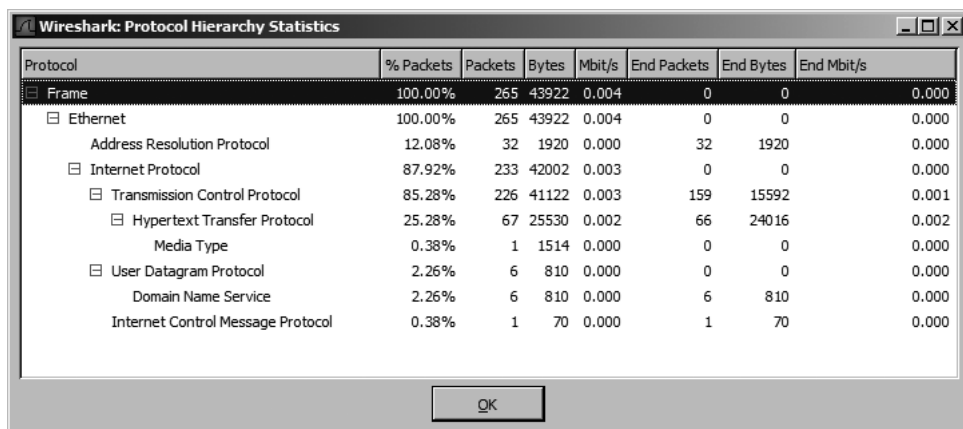
図5-5 TCPストリームなら一目瞭然

TCPストリームは、テキストファイルとして保存または印刷することができます。また、文字列をASCII、EBCDIC、16進数、C言語の配列、生のデータに変換することも可能です。

5.4 [Protocol Hierarchy Statistics]ウィンドウ

巨大なキャプチャファイルを解析するとき、たとえばキャプチャしたパケットのうち何パーセントがDHCPかなど、各プロトコルがどのような配分になっているか知る必要がある場合があります。その際、パケットを1つ1つ数える必要はありません。[Protocol Hierarchy Statistics] ウィンドウを見ればよいのです。このウィンドウはネットワークのベンチマークを知るのに役立ちます。たとえば普段はARPのトラフィックが全体の10%なのに、それが50%になっていたら、何か問題が起きていると予測できます。

[Protocol Hierarchy Statistics] ウィンドウを開くには、メインメニューの[Statistics]から[Protocol Hierarchy]を選択します(図5-6)。



Protocol	% Packets	Packets	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00%	265	43922	0.004	0	0	0.000
Ethernet	100.00%	265	43922	0.004	0	0	0.000
Address Resolution Protocol	12.08%	32	1920	0.000	32	1920	0.000
Internet Protocol	87.92%	233	42002	0.003	0	0	0.000
Transmission Control Protocol	85.28%	226	41122	0.003	159	15592	0.001
Hypertext Transfer Protocol	25.28%	67	25530	0.002	66	24016	0.002
Media Type	0.38%	1	1514	0.000	0	0	0.000
User Datagram Protocol	2.26%	6	810	0.000	0	0	0.000
Domain Name Service	2.26%	6	810	0.000	6	810	0.000
Internet Control Message Protocol	0.38%	1	70	0.000	1	70	0.000

図5-6 [Protocol Hierarchy Statistics] ウィンドウでは各プロトコルの割合が表示される

合計が100%にならない場合があります。キャプチャ量が大きくなると、最小単位以下の割合になってしまうプロトコルが出てきてしまっただけでカウントされなくなるので、パケットの総数とプロトコルの割合が一致なくなってしまうからです。それでも、プロトコルの配分はかなり正確といえます。

5.5 エンドポイントを見る

エンドポイントとは、通信の送信元や送信先のことです。たとえば、TCP/IPの通信では、2つのエンドポイントがあります。送信元の192.168.1.5と送信先の192.168.0.8

という具合です。レイヤ2では、01:00:5e:00:00:16や01:00:5e:01:01:06といったMACアドレスがエンドポイントになります。図5-7にエンドポイントの例があります[†]。

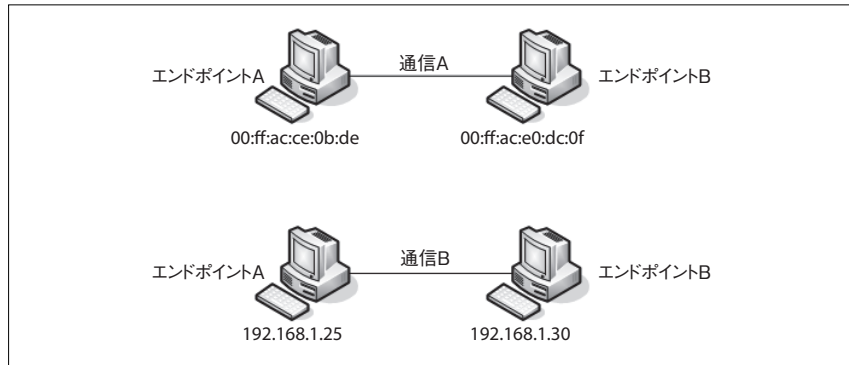


図5-7 ネットワークのエンドポイント

あるエンドポイントに焦点を絞ってパケット解析することもできます。メインメニューの [Statistics] から [Endpoints] を選択すると [Endpoints] ダイアログが表示され、そこから各エンドポイントのアドレスや、送受信したパケット数、バイト数などを見ることができます (図5-8)。ダイアログのトップにあるタブで、エンドポイントをプロトコルごとに表示することができます。[Name resolution] チェックボックスをオンにすると、名前解決を有効にすることができます。

[Endpoints] ダイアログは、パケット一覧のペインから特定のパケットを消去するフィルタとして使用することもできます。エンドポイントを右クリックすると、いくつかのオプションが表示されます。そこから、選択したエンドポイントを含む、または除くトラフィックを表示するフィルタを作成することができます。また、選択したエンドポイントを色分けすることも可能です。

5.6 ネットワーク上の「対話」

ネットワーク上の「対話」は、人の会話のように2つのホスト（エンドポイント）の間で行われます。たとえば、「やあ、元気?」「元気よ。あなたは?」「この上なく元気だよ!」というジムとサリーの会話を、192.168.1.5のコンピュータと192.168.0.8のコンピュータの会話に置き換えると、「SYN」「SYN/ACK」「ACK」となります (TCP/IPの通信についての詳細は6章で学びます)。

メインメニューの [Statistics] から [Conversations] を選択すると、[Conversations] ダイアログが表示されます (図5-9)。[Conversations] ダイアログには、2つのエンド

[†] 監訳注: Wiresharkが認識するエンドポイントにはほかにも、USBのポート、TCPやUDPのポート番号、FDDIやトークンリングなどがあります。

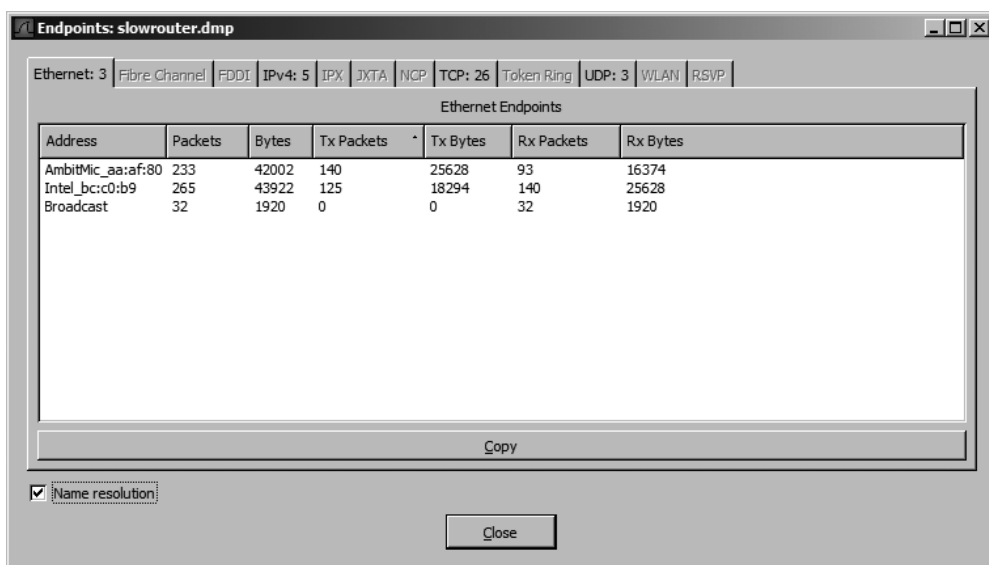


図 5-8 [Endpoints] ダイアログから各エンドポイントを表示する

ポイントをアドレス A、アドレス B とし、それぞれが送受信したパケット数やバイト数が表示されます。

[Conversations] ダイアログの通信の一覧は、ダイアログのトップにあるプロトコルをクリックすると、エンドポイントがそのプロトコル特有のものに切り替わります。

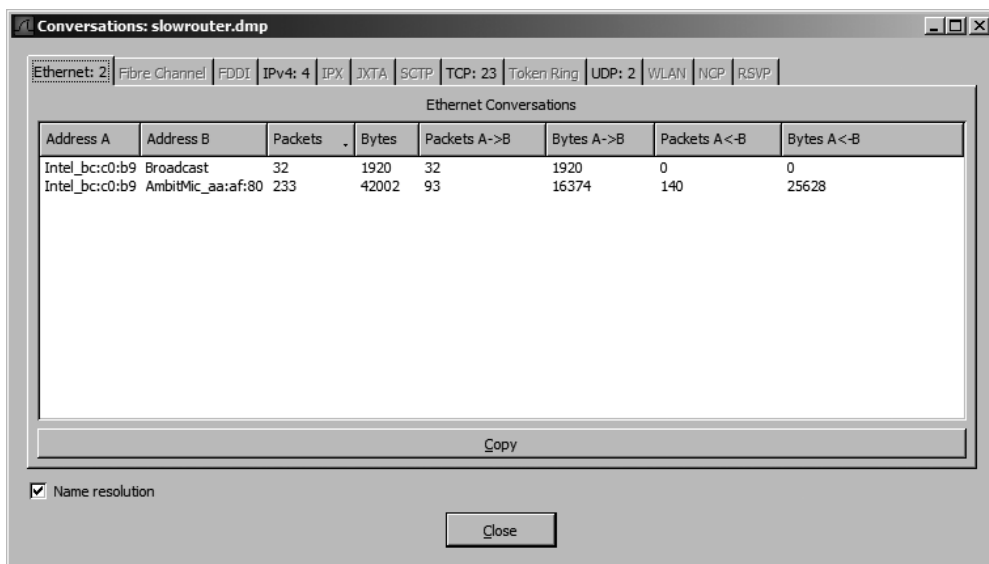


図 5-9 [Conversations] ダイアログには各エンドポイントの通信が表示される

ある行を右クリックすると、エンドポイント A からのトラフィックのみを表示、エンドポイント B から受信したトラフィックのみを表示、エンドポイント A とエンドポイント B の通信のみを表示、といったフィルタを作成することができます。

5.7 [IO Graphs] ウィンドウ

filedownload.dmp

トラフィックの傾向を見極めるもっともよい方法は、それをグラフにしてみることです。Wireshark では、[IO Graphs] ウィンドウで送受信されているデータのグラフを見ることができます。このグラフで各プロトコルのスループットを見て、ネットワークの動静を確認することができます。

インターネットからファイルをダウンロードしたときの IO グラフを見てみましょう。サンプルファイルの filedownload.dmp を開いてください。メインメニューの [Statistics] から [IO Graphs] を選択します。これで、IO グラフを見ることができます。最初のほうでは送受信されるバイト数が小さかったのが、ダウンロードが始まると跳ね上がっていることが分かります (図 5-10)。

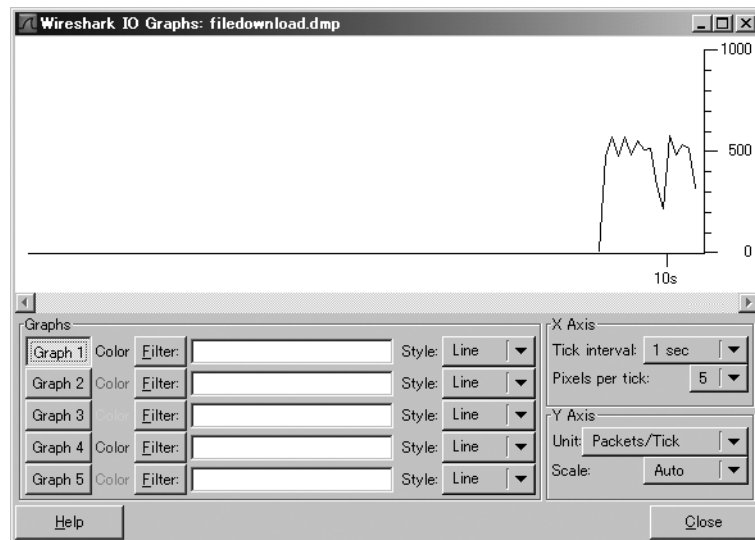


図 5-10 IO グラフから、トラフィックの傾向を知ることができる

このグラフはいろいろなカスタマイズが可能です。もっとも重要なのは X 軸と Y 軸の設定でしょう。グラフの目盛りの縮尺や間隔を変更できます。

このグラフでは、フィルタを作成することが可能です。最大 5 個のフィルタ (構文はディスプレイフィルタやキャプチャフィルタと同じ) を作成して、フィルタごとに色分けすることができます。たとえば ARP を赤、DHCP を青で表示するフィルタを作成すれば、スループットの傾向を簡単に解析することができます。

これまで説明した機能をいつ利用するか、まだよく分からないかもしれませんが、Wiresharkを使っていくうちにうまく利用できるようになるでしょう。重要なのは、Wiresharkのウィンドウと機能の使い方そのものを知ることです。次の章では、これらの機能をたくさん使います。

6章 一般的なプロトコル

この章では、インターネットでよく使われるプロトコルを紹介します。いくつかのプロトコルのサンプルファイルを見ながら、それぞれがどういった機能を持っているのかについて学びます。この章の目的は、プロトコルを理解しパケット解析する際に必要となる基礎知識を身につけることです。この章では非常に重要なプロトコルを紹介します。この章を読み飛ばすことは、映画のパート1を見ずにパート2を見ることと同じです。この章を読まずして、この後の章を理解することはできないでしょう。



本書では、各プロトコルの設計を詳細に述べることはしません。そのかわりに、各プロトコルのRFCの番号を載せています。RFC (Request For Comments) とは、TCP/IPにおけるプロトコルの実装について定義している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。

6.1 ARP

arp.pcap

ARP (Address Resolution Protocol) の勉強から始めましょう。ARPはパケットを少ししか使わない、シンプルなプロトコルです。ARP (RFC 826) は、レイヤ3のアドレス (IPアドレス) をレイヤ2のアドレス (MACアドレス) に変換するプロトコルで、スイッチやルータのようなネットワーク機器のどのポートにコンピュータが接続されているかを確認するために使用されます。

ARPの興味深いところは、OSI参照モデルのネットワーク層とデータリンク層という2つの層にまたがってサービスが提供されているということです。

あるコンピュータが他のコンピュータと通信をするとき、相手のコンピュータがどこにあるのかをまず知る必要があります。スイッチやルータがARPを使って場所を確認してくれます。

サンプルファイルを見てみましょう (図6-1)。送信元のコンピュータ (MACアドレスは01:16:ce:6e:8b:24) が、「誰が192.168.0.1ですか?」というパケットをffffffffffと

いうアドレスに対して送信しています。

No.	Time	Source	Destination	Protocol	Info
1	0.000000	HonHa1Pr_6e:8b:24	BroadCast	ARP	who has 192.168.0.1? tell 192.168.0.114
2	0.004081	D-LInk_0b:22:ba	HonHa1Pr_6e:8b:24	ARP	192.168.0.1 is at 00:13:46:0b:22:ba

図6-1 ARPは、要求と応答の2つのパケットしか使わない

すでに学んだとおり、スイッチはレイヤ2で動作するのでレイヤ3のアドレスを認識することはできません。ではいったいどうしているのでしょうか？ スミスさんに電話しようとしたときに、彼のファーストネームを知らない場合はどうするか考えてみてください。電話帳に載っているスミスという名前の人に、順に電話すればよいのです！

ARPはブロードキャストアドレスに要求を送信することによって、クライアントのレイヤ3アドレスを知ることができます。レイヤ2アドレスのブロードキャスト「ffffffffffff」にARPリクエストを送信すれば、スイッチのブロードキャストドメインのすべてのコンピュータに要求が送信されることになります。

このパケットは、そのコンピュータのIPアドレスが192.168.0.1かどうかをすべてのコンピュータに聞いています。違うIPアドレスのコンピュータはこのパケットを破棄します。192.168.0.1のコンピュータは、自身のレイヤ2アドレスを返します。

図6-1の2番目のパケットは、送信先のコンピュータのARP応答です。応答は「192.168.0.1のレイヤ2アドレスは00:13:46:0b:22:baです」という非常にシンプルなものです。これで送信元のコンピュータは送信先のコンピュータのレイヤ2アドレスを知ることができ、セッションを確立することができるようになります。

6.2 DHCP

dhcp.pcap

DHCP (Dynamic Host Configuration Protocol) もシンプルなプロトコルです。DHCP (RFC 2131) は、コンピュータ名、NTPサーバのアドレス、IPアドレスなどをコンピュータに提供します。DHCPの通信はクライアントサーバ型の通信で、クライアントがIPアドレスをDHCPサーバに要求し、DHCPサーバがそれを与えるという形になっています。

DHCPの基本的な機能は4つのステップから成り立っています。第1のパケットがクライアントからDHCPDISCOVERパケットをブロードキャストアドレス(255.255.255.255)に送信します(図6-2)。

No.	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x3d1d

図6-2 DHCPの通信はDHCPDISCOVERパケットの送信から始まる

クライアントがIPアドレスを獲得するには、DHCPサーバの場所を知る必要があります。DHCPDISCOVERパケットはネットワーク上のDHCPサーバを探すための

パケットです。DHCPサーバがこのパケットを受け取ると、クライアントにDHCPOFFERパケットを送信します(図6-3)。このパケットにはDHCPサーバがクライアントに提供する予定のIPアドレスやその他のネットワークの情報が含まれています。

No. -	Time	Source	Destination	Protocol	Info
2	0.000295	192.168.0.1	192.168.0.10	DHCP	DHCP Offer - Transaction ID 0x361d

図6-3 DHCPOFFERパケットがサーバからクライアントに送信される

クライアントはDHCPOFFERパケットを受け取ると、クライアントはDHCPREQUESTパケットを送信します。この時点ではまだクライアントにはアドレスが割り振られていないため、DHCPREQUESTパケットはブロードキャスト宛に送信されます。このパケットはDHCPサーバが複数存在する場合に、あるDHCPサーバが提供するIPアドレスを使いたい旨を他のDHCPサーバに伝え、これ以上DHCPOFFERは必要ないということを伝える役割を持っています。DHCPサーバがDHCPREQUESTパケットを受け取ると、DHCPACKパケットをクライアントに送信し、これでIPアドレスの割り当てが終了します(図6-4)。

No. -	Time	Source	Destination	Protocol	Info
4	0.070345	192.168.0.1	192.168.0.10	DHCP	DHCP ACK - Transaction ID 0x361d

図6-4 DHCPACKパケットをクライアントに送信してIPアドレスの割り当てが終了する

パケット一覧のペインでは、DHCPパケットには「Transaction ID」というIDが付けられています。このIDは複数のクライアントのトランザクションを識別するために割り振られています。他のクライアントの通信と間違えないように、パケット解析する際にはこのIDを確認してください。

4種類のDHCPパケットについて説明しましたが、キャプチャファイルには8つの種類のDHCPパケットが記録されています。DHCPの詳細はRFC 2131を参照してください。

6.3 TCP/IPとHTTP

http.pcap

TCP/IPは本書に登場する多くの通信の基本となるものです。もっとも広く使われているネットワークプロトコルですので、詳しく学んでいきましょう。

HTTP (Hypertext Transfer Protocol, RFC 2616) は、Webページを転送するためのサーバ/クライアントシステムのプロトコルです。HTTPの通信はTCP/IPのよい例となります。Googleで調べ物をしたり、天気予報を見たり、お気に入りのスポーツチームをチェックするときなどにTCP/IPのプロトコルの1つであるHTTPを使っています。

6.3.1 TCP/IP

TCP/IPのプロトコルは、OSI参照モデルのレイヤ3およびレイヤ4からなるプロトコル群です。TCP、IP、ARP、DHCP、ICMPなど、多くのプロトコルがあります。

TCP (Transmission Control Protocol, RFC 793) は、レイヤ4のプロトコルです。透過的で信頼性が高く、双方向の通信が効率的にできるプロトコルで、広く使われています。双方向の通信とは、1つのコンピュータがデータの送信と受信を同時に行えるということの意味します。

TCPの機能と利点は、パケットとフラグのタイプによって異なります。それぞれのタイプの機能を見ていきましょう。

IP (Internet Protocol, RFC 791) は、通信を可能にするためのアドレスを提供する、レイヤ3のプロトコルです。IPは通信セッションを確立しないプロトコルですが、これは逆に言えば、IPにバンドルされているTCPのほうに、データ転送の信頼性を確保することが求められる、ということです。

キャプチャファイルにはTCP/IPのセッションの確立から始まり、HTTPデータの要求と転送、そしてセッションの終了までが記録されています。クライアントサーバ間のこの単純な通信を通して、TCPやIPの機能を理解しましょう。

6.3.2 セッションの確立

ほかのコンピュータとセッションを開始するには、**3ウェイハンドシェイク**を成功させる必要があります。3ウェイハンドシェイクは、送信元（この例ではクライアント）と送信先のコンピュータ（サーバ）のセッションを確立する3つのステップです（図6-5）。

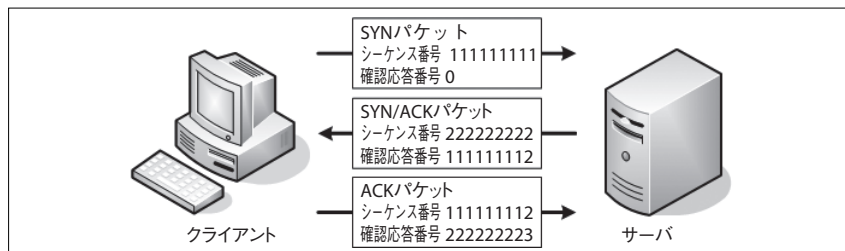


図6-5 3ウェイハンドシェイクの3つのステップ

それでは、クライアントサーバ間のセッションを確立してみましょう。クライアントのIPアドレスは145.254.160.237で、サーバのIPアドレスは65.208.228.223です。

6.3.2.1 SYNパケット

3ウェイハンドシェイクは、クライアントがサーバにSYNパケットを送ることから始まります。このパケットの先頭32ビットにはシーケンス番号が含まれており、クライアントサーバ間でこの番号の同期を取り、正しく通信が行われるようにします。

パケット詳細のペインのTCPの部分を広げると、送信元および送信先が使用しているポート番号、シーケンス番号、TCPのタイプ、その他TCP特有の情報を見ることができます。最初のSYNパケットのシーケンス番号は0になっています(図6-6)。

[-]	Frame 1 (62 bytes on wire, 62 bytes captured)
[-]	Ethernet II, Src: Xerox_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
[-]	Internet Protocol, Src: 145.254.160.237 (145.254.160.237), Dst: 65.208.228.223 (65.208.228.223)
[-]	Transmission Control Protocol, Src Port: 3372 (3372), Dst Port: http (80), Seq: 0, Len: 0
	Source port: 3372 (3372)
	Destination port: http (80)
	Sequence number: 0 (relative sequence number)
	Header length: 28 bytes
[-]	Flags: 0x02 (SYN)
	Window size: 8760
	Checksum: 0xc30c [correct]
[-]	Options: (8 bytes)
	Maximum segment size: 1460 bytes
	NOP
	NOP
	SACK permitted

図6-6 パケット詳細のペインに必要な情報が表示されている



Wiresharkでは、シーケンス番号は「相対的」な番号として扱われています。そのため、本当の値がなんであれ最初の番号は必ず0になります。これで、シーケンス番号がより見やすくなります。

6.3.2.2 SYN/ACKパケット

次はサーバからの応答です。サーバはクライアントからSYNパケットを受け取ると、そのパケットに含まれるシーケンス番号を使うようになります。応答のパケットはSYN/ACKパケットと呼ばれます。サンプルファイルの2番目のパケットがそれです。

パケットのACK部分には、クライアントから送信されてきたシーケンス番号に1を加えたものが応答確認番号として添えられています。こうして、サーバがSYNパケットを受け取ったという旨をクライアントに伝えているのです。SYN/ACKパケットのSYN部分は、クライアントが送ったSYNパケットと同じように、クライアントにシーケンス番号を送信することを目的としています。

6.3.2.3 ACKパケット

最後に、クライアントはサーバに、ACKパケットを送ってSYN/ACKパケットを受け取った旨を知らせます。SYN/ACKパケットと同じように、シーケンス番号に1が加えられたものが応答確認番号として添えられています。サーバがACKパケットを受け取れば、データの送受信を始めることができます。

6.3.3 データ送信の開始

データのやり取りは、3ウェイハンドシェイクによって決められたシーケンス番号を使って行われます。しかしながら、ここからはシーケンス番号に1が加えられるの

でなく、送信されたデータのサイズ分だけ加えられます。TCPについてもっと学びたい場合は、RFC 793を参照してください。

6.3.4 HTTPの通信

以上でセッションが確立しました。次にWebページを表示するときに実際に何が送受信されているかを見てみましょう。HTTPとTCPの両方が使われています。

4番目のパケットが、HTTPの最初のパケットです。Webページをクライアントに送信するように要求しています。パケット詳細のペインで、HTTPの部分を広げてGETリクエストの中身を見てみてください(図6-7)。

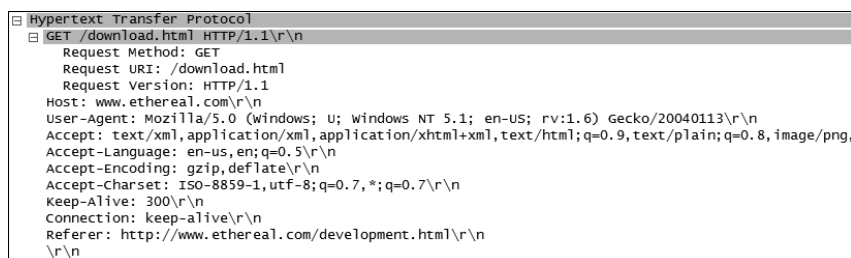


図6-7 パケット詳細のペインから、要求の詳細を見ることができる

このパケットにはGETリクエスト (Request Method: GET) が含まれており、www.ethereal.com の/download.htmlというWebページを取得しようとしています。そのほかにも、文字コードの種類 (Accept-Charset: ISO-8859-1, ...) やリファラ (Referer: http://www.ethereal.com/development.html) などの情報も表示されます。

クライアントがGETリクエストを送信すると、サーバがTCPを使ってデータの転送を始めます。サーバはデータを送る前に、HTTP OKというメッセージを送信してリクエストを正しく受け取ったことをクライアントに知らせています。図6-8 (およびサンプルファイル) の4番目にGETリクエストが、38番目にOKレスポンスが送信されています。

6.3.5 セッションの終了

データの送信が終了したら、3ウェイハンドシェイクと同じような手順でセッションを終了します。セッションの終了は、SYNパケットやACKパケットの代わりに、FINパケットとACKパケットを使用します(図6-9)。

サーバはデータの送信を終えると、FIN/ACKパケットをクライアントに送ります(図6-10)。FINパケットは、通信を正常に終了するために設計されたパケットです。

クライアントはFIN/ACKパケットを受け取ると、ACKパケットをサーバに返します。ACKパケットの応答確認番号は、FIN/ACKパケットのシーケンス番号に1を加えたものになります。これで、サーバからのデータの送信は終了します。この時点

No. -	Time	Source	Destination	Protocol	Info
4	0.911310	145.254.160.237	65.208.228.223	HTTP	GET /download.html HTTP/1.1
<div> <div>Transmission Control Protocol, Src Port: 3372 (3372), Dst Port: http (80), Seq: 0, Ack: 0, Len: 479</div> <div> <div>Source port: 3372 (3372)</div> <div>Destination port: http (80)</div> <div>Sequence number: 0 (relative sequence number)</div> <div>[Next sequence number: 479 (relative sequence number)]</div> <div>Acknowledgement number: 0 (relative ack number)</div> <div>Header length: 20 bytes</div> <div>Flags: 0x18 (PSH, ACK)</div> <div>Window size: 9660</div> <div>checksum: 0xa958 [correct]</div> </div> </div>					
No. -	Time	Source	Destination	Protocol	Info
38	4.846969	65.208.228.223	145.254.160.237	HTTP	HTTP/1.1 200 OK (text/html)
<div> <div>Transmission Control Protocol, Src Port: http (80), Dst Port: 3372 (3372), Seq: 17941, Ack: 480, Len: 424</div> <div> <div>Source port: http (80)</div> <div>Destination port: 3372 (3372)</div> <div>Sequence number: 17941 (relative sequence number)</div> <div>[Next sequence number: 18365 (relative sequence number)]</div> <div>Acknowledgement number: 480 (relative ack number)</div> <div>Header length: 20 bytes</div> <div>Flags: 0x18 (PSH, ACK)</div> <div>Window size: 6432</div> <div>checksum: 0x3d97 [correct]</div> <div>[SEQ/ACK analysis]</div> <div>TCP segment data (424 bytes)</div> <div>[Reassembled TCP Segments (18364 bytes): #6(1380), #8(1380), #10(1380), #11(1380), #14(1380), #16(1380),</div> </div> </div>					

図6-8 4番目のパケットがGETリクエスト、38番目のパケットがOKパケット

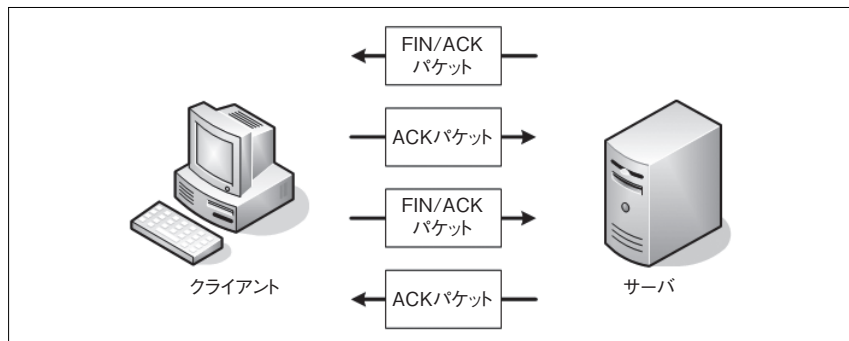


図6-9 FIN/ACKハンドシェイクによる通信の終了†

<div> <div>Transmission Control Protocol, Src Port: http (80), Dst Port: 3372 (3372), Seq: 18365, Ack: 480, Len: 0</div> <div> <div>Source port: http (80)</div> <div>Destination port: 3372 (3372)</div> <div>Sequence number: 18365 (relative sequence number)</div> <div>Acknowledgement number: 480 (relative ack number)</div> <div>Header length: 20 bytes</div> <div>Flags: 0x11 (FIN, ACK)</div> <div>Window size: 6432</div> <div>checksum: 0x3c64 [correct]</div> <div>[SEQ/ACK analysis]</div> </div> </div>					
---	--	--	--	--	--

図6-10 パケット詳細のペイン(40番目のパケットであるFIN/ACKパケットの詳細を表示)

ではサーバはまだデータを受信することができますが、送信することはありません。

セッションを終了するには、クライアントがサーバにFIN/ACKパケットを送り、サーバがACKパケットを返すという逆の処理が必要になります。

† 監訳注：ハンドシェイクといえば、通信セッションを確立するときの3ウェイハンドシェイクがすぐに思い浮かびますが、通信セッションを終了するときのやり取りもハンドシェイクと呼ぶことがあります。

図6-11でいえば、40番目のパケットはサーバがFIN/ACKパケットをクライアントに送信しているところで、41番目のパケットはクライアントがACKパケットを返しているところです。さらに42番目のパケットはクライアントからのFIN/ACKパケットで、43番目のACKパケットでセッションが終了しています。

No. -	Time	Source	Destination	Protocol	Info
40	17.905747	65.208.228.223	145.254.160.237	TCP	http > 3372 [FIN, ACK] Seq=18365 Ack=480 win=6432 Len=0
41	17.905747	145.254.160.237	65.208.228.223	TCP	3372 > http [ACK] Seq=480 Ack=18366 win=9236 Len=0
42	30.063228	145.254.160.237	65.208.228.223	TCP	3372 > http [FIN, ACK] Seq=480 Ack=18366 win=9236 Len=0
43	30.393704	65.208.228.223	145.254.160.237	TCP	http > 3372 [ACK] Seq=18366 Ack=481 win=6432 Len=0

図6-11 パケット一覧のペインから分かるセッション終了の手順

6.4 DNS

dns.pcap

DNS (Domain Name System、RFC 1034) は、www.google.comやMARKETING-PC1のようなドメイン名をIPアドレスに変換する機能を持っています。OSI参照モデルのレイヤ3はIPアドレスでコンピュータを識別するので、そういった名前解決が必要なのです。

DNSの名前解決は非常にシンプルで、たいていの場合2つのパケットで事足ります。最初のパケットは、「www.google.comのIPアドレスはなんですか?」というDNSサーバへの名前解決の要求で、2番目のパケットは「www.google.comのIPアドレスはXX.XX.XX.XXXです」というサーバからの応答です。

DNSパケットの例を見てみましょう(図6-12)。まず192.168.0.114から205.152.37.23にhttp://www.chrissanders.orgの名前解決の要求が送信されています。2番目のパケットでは、要求されたドメインのIPアドレスは208.113.140.24だという返答がなされています。これで、レイヤ3での3ウェイハンドシェイクが可能となり、データを送信できるようになります。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.114	205.152.37.23	DNS	Standard query A chrissanders.org
2	0.112121	205.152.37.23	192.168.0.114	DNS	Standard query response A 208.113.140.24

図6-12 DNSは、要求とその返答という2つのパケットで事足りる



実際にパケットをキャプチャしてみると、複数のDNSパケットが流れていることに気づくでしょう。1つのWebページにほかのサイトからの情報を表示していることがあるので、その名前解決が必要なのです。試しにDNSのトラフィックのみを表示するフィルタを作成し、どれほどのDNSパケットが流れているかを確認してみてください。

6.5 FTP

ftp.pcap

FTP (File Transfer Protocol, RFC 959) は、クライアントサーバ間のデータ転送のためのレイヤ7のプロトコルです。20番ポートと21番ポートを使います。FTPはクライアントサーバ型のプロトコルなので、通信がクライアントとサーバの間を行ったり来たりします。FTPはTCPを利用しているので、通信は3ウェイハンドシェイクから始まります (図6-13)。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.114	192.168.0.193	TCP	1137 > ftp [SYN] Seq=0 Len=0 MSS=1460
2	0.002319	192.168.0.193	192.168.0.114	TCP	ftp > 1137 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1452
3	0.002338	192.168.0.114	192.168.0.193	TCP	1137 > ftp [ACK] Seq=1 Ack=1 Win=17424 Len=0

図6-13 多くのプロトコルの通信は3ウェイハンドシェイクで始まる

3ウェイハンドシェイクが終わると、サーバはクライアントにウェルカムメッセージを送ります。サーバはこのメッセージで自分がFTPサーバであることを告げ、ログイン処理を求めます (図6-14)。

File Transfer Protocol (FTP)
220 Chris Sanders FTP Server\r\n
Response code: Service ready for new user (220)
Response arg: Chris Sanders FTP Server

図6-14 FTP通信の開始 (4番目のパケット)

そしてクライアントがユーザー名 (csanders) とパスワード (echo) をサーバに送り、サーバはそれを受け取った旨をクライアントに伝えます (図6-15)。

No. -	Time	Source	Destination	Protocol	Info
5	0.005259	192.168.0.114	192.168.0.193	FTP	Request: USER csanders
6	0.006560	192.168.0.193	192.168.0.114	FTP	Response: 331 Password required for csanders.
7	0.007647	192.168.0.114	192.168.0.193	FTP	Request: PASS echo
8	0.009936	192.168.0.193	192.168.0.114	FTP	Response: 230 User csanders logged in.

図6-15 ユーザー名とパスワードがサーバに送られる

FTPの通信では、パケット一覧のペインの [Info] から十分な情報が得られます。より詳細な情報を知りたいければ、パケット詳細のペインのFTPの部分を広げてみてください。

この通信は暗号化されていないため、サンプルファイルの7番目のパケットにはパスワードがはっきりと記録されています (図6-16)。

Frame 7 (65 bytes on wire, 65 bytes captured)
Ethernet II, Src: HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24), Dst: AsustekC_40:76:ef (00:15:f2:40:76:ef)
Internet Protocol, Src: 192.168.0.114 (192.168.0.114), Dst: 192.168.0.193 (192.168.0.193)
Transmission Control Protocol, Src Port: 1137 (1137), Dst Port: ftp (21), Seq: 16, Ack: 68, Len: 11
File Transfer Protocol (FTP)
PASS echo\r\n
Request command: PASS
Request arg: echo

図6-16 ユーザー「csanders」のパスワードが見える

クライアントは、FTPサーバとの通信にコマンドを使用します。コマンドには、ディレクトリ一覧の表示、ディレクトリの移動、ファイルの削除などがあります。コマンド一覧はRFC 959を参照してください。サンプルファイルの15番目のパケットから始まるFTPコマンドを見てみましょう(図6-17)。

```
File Transfer Protocol (FTP)
  CWD /\r\n
    Request command: CWD
    Request arg: /
```

図6-17 15番目のパケットはCWDコマンドを実行している

6.5.1 CWD コマンド

CWD コマンドはカレントディレクトリの変更のためのコマンドで、クライアントがディレクトリを移動するごとに実行されます。

サンプルファイルには、サーバのルートディレクトリである「/」へ移動するためのCWD コマンドが含まれています。FTPサーバにログインした直後に、ルートディレクトリに移動するCWD コマンドが実行され、カレントディレクトリがルートディレクトリであることをクライアントに伝えます。

6.5.2 SIZE コマンド

次はSIZE コマンドです。このコマンドを実行すると、ファイルのサイズが表示されます。25番目のパケットで、クライアントはSIZE コマンドを使って「Music.mp3」のファイルサイズを要求しています(図6-18)。

```
File Transfer Protocol (FTP)
  SIZE Music.mp3\r\n
    Request command: SIZE
    Request arg: Music.mp3
```

図6-18 SIZE コマンドがサーバに送信されている

26番目のパケットで、ファイルサイズは4,980,924バイトであることをサーバが返答しています(図6-19)。

```
File Transfer Protocol (FTP)
  213 4980924\r\n
    Response code: File status (213)
    Response arg: 4980924
```

図6-19 SIZE コマンドの実行結果

6.5.3 RETR コマンド

RETR (retrieve) コマンドは、サーバからファイルをダウンロードするためのコマンドです(図6-20)。32番目のパケットで、クライアントはMusic.mp3をダウンロードするためにRETR コマンドをサーバに送っています。サーバはこのコマンドを受け取

ると、クライアントへのデータ送信を開始します。

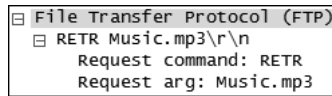


図6-20 RETR コマンドはサーバからファイルをダウンロードするコマンド



FTP-DATA というラベルのパケットは、ファイルをダウンロードまたはアップロードしているパケットです。

6.6 TELNET

telnet.pcap

TELNET (RFC 854) はコンピュータをリモートから操作するためのプロトコルです。平文で通信するため、セキュリティ上は問題があります。サーバ、スイッチ、ルータなどのネットワーク機器を管理するために使われます。

このサンプルファイルには、クライアント (192.168.0.2) がサーバ (192.168.0.1) に TELNET で接続している様子が記録されています。平文でデータのやり取りをするため、通信が丸見えです。そのため、重要なデータは TELNET で送信すべきではありません。



TELNET より安全な SSH を使うべきです。

クライアントサーバ間でどんな通信が行われているのでしょうか？ サンプルファイルの最初のほうのパケットで、TELNET 特有の通信が行われているのでそれが TELNET のトラフィックであることが分かります (図6-21)。

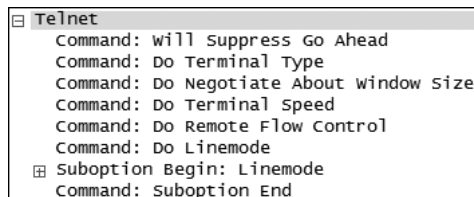


図6-21 TELNET のパケット (これは9番目のパケット)

TELNET のセッションでは、転送レートやデータ転送モードを指定するオプションが使われています。通信を始める前にクライアントとサーバでそれらを同期する必要があります。これらのオプションは、サンプルファイルの最初の30個くらいの

パケットで登場します。

まず興味深いのは27番目のパケットです。サーバがOpenBSDであることを告げています。29番目のパケットではクライアントへログインプロンプトを表示しており、クライアントは31番目のパケットでユーザー名「fake」をサーバに送信しています。36番目のパケットでサーバはクライアントにパスワードを要求し、クライアントが38番目のパケットでパスワード「user」を返しています(図6-22)。これで、TELNETがいかに危険かわかるでしょう。このユーザー名とパスワードは重要なサーバを管理するために使われているかもしれません。そして平文で行われる通信は、スニッファとちょっとした知識があれば簡単に読み取ることができるのです。

Frame 36 (75 bytes on wire, 75 bytes captured)
Ethernet II, Src: western0_9f:a0:97 (00:00:c0:9f:a0:97), Dst: Lite-OnC_3b:bf:fa (00:a0:cc:3b:bf:fa)
Internet Protocol, Src: 192.168.0.1 (192.168.0.1), Dst: 192.168.0.2 (192.168.0.2)
Transmission Control Protocol, Src Port: telnet (23), Dst Port: 1550 (1550), Seq: 143, Ack: 207, Len: 9
Telnet
Data: Password:
Frame 38 (72 bytes on wire, 72 bytes captured)
Ethernet II, Src: Lite-OnC_3b:bf:fa (00:a0:cc:3b:bf:fa), Dst: western0_9f:a0:97 (00:00:c0:9f:a0:97)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 1550 (1550), Dst Port: telnet (23), Seq: 207, Ack: 152, Len: 6
Telnet
Data: user\r\n

図6-22 TELNETでパスワードを送信すると、あっさりと他人に見られてしまう

サンプルファイルではこのあと、クライアントがサーバのシェルを利用していくつかのWebサイトにpingを送信しています。パケット詳細のペインで、TELNETによる通信の全データを見ることができます。

6.7 MSN メッセンジャーサービス

msnms.pcap

きっとそのうちインスタントメッセージの会話を分析しなければならないことがあ
るでしょう。5章では、企業の社員がインスタントメッセージを使って新人いび
りでハッキング(クラッキング)する相談をしていました。インスタントメッセン
ジャーにはいくつか種類があり、それぞれ似てはいますが独自のプロトコルを使っ
ています。ここでは、MSN メッセンジャーサービス(MSNMS)のサンプルファイルを
使って、社員の会話を覗いてみましょう[†]。

[†] 監訳注：一般的には、社員は会社のコンピュータとネットワークを使って仕事をしているので、その上でのやりとり(通信)を盗聴されたとしても文句は言えない立場です。だからといって、同意なしにいきなり盗聴することは、プライバシーを侵害したり、パワーハラスメントになりかねません。雇用契約などを結ぶ場面などで、ネットワーク障害やセキュリティの調査や重要情報保護の観点からの調査などで、責任者の承認を得て盗聴することはありえる、ということを説明し、同意を得ておくべきでしょう。



企業によっては、インスタントメッセンジャーの使用を禁止するポリシーがあるかもしれません。MSNMS プロトコルを見ておけば、警鐘を鳴らすことができますでしょう。

サンプルファイルの冒頭部分は、TCP の 3 ウェイハンドシェイクです (図 6-23)。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.114	207.46.26.167	TCP	3331 > 1863 [SYN] Seq=0 Len=0 MSS=1460
2	0.098754	207.46.26.167	192.168.0.114	TCP	1863 > 3331 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1452
3	0.098792	192.168.0.114	207.46.26.167	TCP	3331 > 1863 [ACK] Seq=1 Ack=1 Win=17424 Len=0

図 6-23 3 ウェイハンドシェイクによる通信の開始

3 ウェイハンドシェイクの後に、192.168.0.114 からローカルネットワークの外にあるサーバに MSNMS パケットが送信されます (図 6-24)。

No. -	Time	Source	Destination	Protocol	Info
4	0.098991	192.168.0.114	207.46.26.167	MSNMS	USR 93 tesla_brian@hotmail.com 1835953129.20013021.2623242

図 6-24 ローカルネットワークからインターネットのサーバと通信しようとしている

最初の MSNMS パケットはマイクロソフトのサーバと通信を開始するためのものです。パケット詳細のペインにあるように、このパケットには USR という文字が含まれます。チャットを始めようとしているのが tesla_brian@hotmail.com というメールアドレスを持つユーザーであることが分かります (図 6-25)。

MSN Messenger Service
USR 93 OK tesla_brian@hotmail.com Brian\r\n

図 6-25 5 番目のパケットのパケット詳細のペインを見ると、tesla_brian@hotmail.com というメールアドレスを持つユーザーがチャットしようとしていることが分かる

次の 2 つのパケットは CAL パケットと呼ばれ、チャット相手を指定するためにサーバに送られます (図 6-26)。

No. -	Time	Source	Destination	Protocol	Info
6	0.199942	192.168.0.114	207.46.26.167	MSNMS	CAL 94 tesla.thomas@hotmail.com
7	0.300257	207.46.26.167	192.168.0.114	MSNMS	CAL 94 RINGING 1835953129

図 6-26 CAL パケットはほかのユーザーとチャットを開始するために使われる

6 番目のパケットで、チャット相手のメールアドレスを送信しています (図 6-27)。

Frame 6 (87 bytes on wire, 87 bytes captured)
Ethernet II, Src: HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24), Dst: D-Link_21:99:4c (00:05:5d:21:99:4c)
Internet Protocol, Src: 192.168.0.114 (192.168.0.114), Dst: 207.46.26.167 (207.46.26.167)
Transmission Control Protocol, Src Port: 3331 (3331), Dst Port: 1863 (1863), Seq: 61, Ack: 42, Len: 33
MSN Messenger Service
CAL 94 tesla_thomas@hotmail.com\r\n

図6-27 CALパケットからチャット相手のメールアドレスが分かる

8番目のパケットで、サーバがCALパケットを受け取った旨をクライアントに伝えています (図6-28)。

No. -	Time	Source	Destination	Protocol	Info
7	0.300257	207.46.26.167	192.168.0.114	MSNMS	CAL 94 RINGING 1835953129
8	0.442314	192.168.0.114	207.46.26.167	TCP	3331 > 1863 [ACK] Seq=94 Ack=69 win=17356 Len=0

図6-28 8番目のパケットは7番目のパケットの応答確認

9番目のパケットで、チャットのための準備が整います。このパケットはJOIパケットと呼ばれ、チャット相手（この場合は tesla_thomas@hotmail.com）がチャットに参加できることをクライアントに伝えています (図6-29)。

No. -	Time	Source	Destination	Protocol	Info
9	0.510484	207.46.26.167	192.168.0.114	MSNMS	JOI tesla_thomas@hotmail.com Thomas 1616756780

図6-29 JOIパケットはユーザー同士がチャットできるようになったことを伝えている

サンプルファイルの残りの部分はMSGパケットです。ブライアンとトーマスのチャットメッセージが送信されています。

以上の文を読んで最初に思いつく疑問は、「本当にチャットの内容が読めるの?!」ということでしょう。恐ろしいことに、答えはyesです。どれでもよいのでMSGパケットを右クリックして、[Follow TCP Stream] (5章で学んだ機能)を選んでください。ブライアンとトーマスのチャットの内容が見えるはずですよ (図6-30)。これからはインスタントメッセンジャーでの会話には十分注意しましょう。

6.8 ICMP

icmp.pcap

ICMP (Internet Control Message Protocol : RFC 792) は、IPプロトコルの1つです。ICMPはユーティリティプロトコルといっていいほど、トラブルシューティングに役立つプロトコルです。pingコマンドはICMPを使っています。

ICMPのトラフィックがどのようなものか見てみましょう。サンプルファイルには8つのやりとりが記録されています。これは2つのホストに送ったpingパケットです。最初のパケットを見てみましょう (図6-31)。

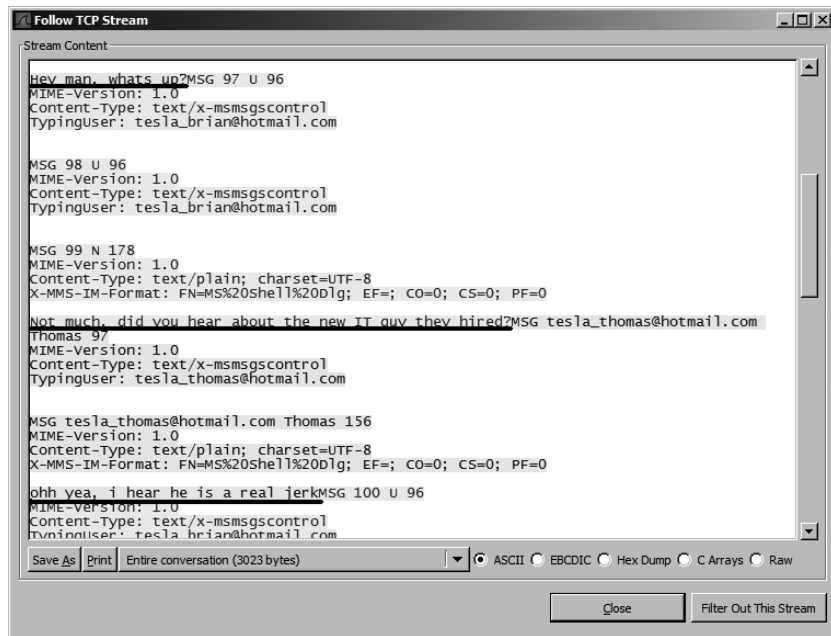


図6-30 これで誰がチャットしているか分かる。お前はクビだ！

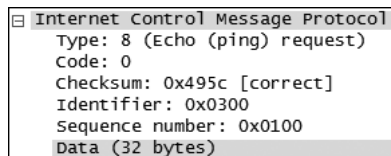


図6-31 最初のpingパケット

パケット詳細のペインのICMPの部分を広げると、ICMPがいかに小さいパケットか分かります。最初のパケットはタイプ8 (Echo リクエスト) です。ICMP パケットには必ず数字で表されるタイプが含まれています。その数字によって、送信先のコンピュータでの処理方法が変わるのです。ICMPのタイプ一覧についてはRFC 792を参照してください。

Echo リクエストを送信したらEcho リプライが返ってくる、というのが常識的な考えですが、実際にサンプルファイルでも、2番目のパケットはタイプ0のICMPパケット、すなわちEcho リプライです。

Windowsのpingコマンドでは、Echo リクエストが4回送信されます。図6-32のサンプルファイルもそのようになっています。最初のpingの宛先である192.168.0.1は、Echo リクエストを4回受信し4回リプライを返しています。同じように、72.14.207.99 (www.google.com) でも4回pingのやり取りが行われています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.114	192.168.0.1	ICMP	Echo (ping) request
2	0.001085	192.168.0.1	192.168.0.114	ICMP	Echo (ping) reply
3	0.996773	192.168.0.114	192.168.0.1	ICMP	Echo (ping) request
4	0.998983	192.168.0.1	192.168.0.114	ICMP	Echo (ping) reply
5	1.996801	192.168.0.114	192.168.0.1	ICMP	Echo (ping) request
6	1.999087	192.168.0.1	192.168.0.114	ICMP	Echo (ping) reply
7	2.996840	192.168.0.114	192.168.0.1	ICMP	Echo (ping) request
8	2.999177	192.168.0.1	192.168.0.114	ICMP	Echo (ping) reply

図6-32 リクエスト、リプライ、リクエスト、リプライ...

6.9 まとめ

この章の目的は、Wiresharkで一般的なプロトコルがどのように見えるかを学ぶことです。ここではプロトコルの簡単な紹介しかしていませんが、各プロトコルのRFCを読むことを強くお勧めします。次章からは、この章で学んだ概要を元に、さまざまなシナリオを読み解く練習をしていきます。

7章

ケーススタディ(基礎編)

さあ、いよいよ本題に入ります。ここからは、実際にネットワーク上で起きる問題についてのパケット解析をしていきます。

いくつかの単純な事例をパケット解析してみて、その裏で何が起きているのかを理解しましょう。さらに現実には日々起きているネットワーク障害についても調査します。

7.1 TCPの通信障害

tcp-con-lost.pcap

ネットワーク障害でもっとも多いのが、通信ができなくなってしまうことです。障害が起こる理由とはとりあえず置いておいて、障害がパケットレベルではどんなふうに見えるかを見てみましょう。実際のトラブルシューティングで、障害の切り分けができるようになります。

tcp-con-lost.pcapというサンプルファイルを開いてください(図7-1)。10.3.71.7と10.3.30.1の間で、普通のACKパケットが4つ、送受信されています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.3.71.7	10.3.30.1	TCP	1043 > 1048 [ACK] Seq=0 Ack=0 win=8760 Len=0
2	0.000000	10.3.30.1	10.3.71.7	TCP	1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648
3	0.000000	10.3.71.7	10.3.30.1	TCP	1043 > 1048 [ACK] Seq=0 Ack=2920 win=8760 Len=0
4	0.000000	10.3.71.7	10.3.30.1	TCP	1043 > 1048 [ACK] Seq=0 Ack=5840 win=8760 Len=0

図7-1 最初は普通にACKパケットのやり取りがされている

問題は5番目の、データを再送しているパケットから始まります(図7-2)。

No. -	Time	Source	Destination	Protocol	Info
5	0.206000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648
6	0.806000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648
7	2.006000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648
8	4.406000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648
9	9.211000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760 Len=648

図7-2 データの再送は通信が不安定になっていることを示す

TCPは、送信先にデータを送った後一定期間待っても返答がない場合、データを再送するよう設計されています。再送後、最初の待ち時間の2倍の時間待っても返答

がない場合、もう一度データを送りなおします。TCPの再送の仕組みは図7-3のとおりです。

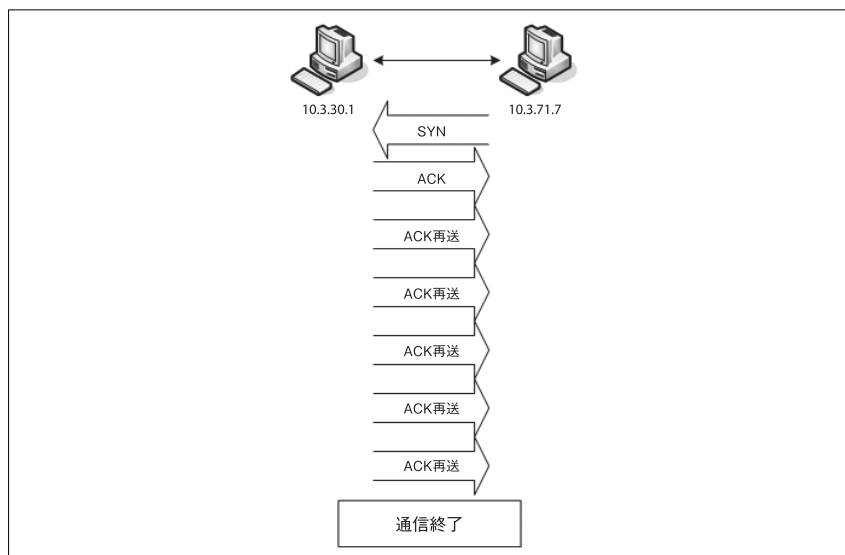


図7-3 何度もデータを再送している場合、通信に障害が起きている可能性が高い

Windowsでは、約9.6秒の間に5回再送を試みます（図7-3）。再送が5回とも失敗すると、通信が終了しデータは喪失します。

Wiresharkの時間の表示フォーマットを、キャプチャ開始時間からの相対的な時間に設定していれば（メインメニューの[View]から[Time Display Format]を選択し、[Seconds Since Beginning of Capture]をクリックする）、データが再送されるまでの時間が増えていっていることが分かるでしょう（図7-4）。

No.	Time	Source	Destination	Protocol	Info
4	0.000000	10.3.71.7	10.3.30.1	TCP	1043 > 1048 [ACK] Seq=0 Ack=5840 win=8760 Len=0
5	0.206000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760
6	0.806000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760
7	2.006000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760
8	4.406000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760
9	9.211000	10.3.30.1	10.3.71.7	TCP	[TCP Retransmission] 1048 > 1043 [PSH, ACK] Seq=5840 Ack=0 win=8760

図7-4 Windowsでは5回再送を試みる

図7-4をもう少し詳しく見てみましょう。4番目のパケットの応答確認番号（Ack=5840）がデータを再送している5つのパケットのシーケンス番号（Seq=5840）になっています。

6章で学んだとおり、TCPは複数の通信を識別するためにシーケンス番号と応答確認番号を保持しています。再送されたパケットのシーケンス番号が4番目のパケットの応答確認番号と一致しているということは、4番目のパケットが喪失してしまったということです。再送が始まった場所を見つけることができれば、通信障害がなぜ発生したかを知る手がかりになるかもしれません。

7.2 届かないパケットとICMPコード

ネットワークの導通を確認するときにもっともよく使われるのはpingコマンドでしょう。運が良ければpingパケットを送った相手から、pingパケットを受け取ったと返事が来るでしょう。運が悪ければ、送信先からのpingパケットの返事は来ず、宛先到達不可能通知 (Destination unreachable) が返ってきます。スニッファでICMPパケットをキャプチャすると、pingコマンドの返答だけを見るよりも多くの情報が得られます。ICMPエラーメッセージをもっと詳しく見てみましょう。

7.2.1 宛先到達不可能

destunreachable.pcap

destunreachable.pcapを開いてください。最初のパケットは10.2.10.2から10.4.88.8へのEchoリクエスト (ICMP タイプ8) です (図7-5)。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.2.10.2	10.4.88.88	ICMP	Echo (ping) request

図7-5 10.2.10.2から10.4.88.88へEchoリクエスト送信

パケット詳細のペインに表示されているICMPの部分を広げることでより詳しく見ることができます。Echoリクエストが宛先のコンピュータに届けば、Echoリプライ (ICMP タイプ0) が返ってくるはずですが。

図7-6の2番目のパケットを見ると、タイプ0でなくタイプ3のパケットが返ってきているのが分かります。これは、Echoリクエスト送信先に到達できなかったことを示します。

Internet Control Message Protocol
Type: 3 (Destination unreachable)
Code: 1 (Host unreachable)
Checksum: 0xa7a2 [correct]
Internet Protocol, Src: 10.2.10.2 (10.2.10.2), Dst: 10.4.88.88 (10.4.88.88)
Internet Control Message Protocol

図7-6 タイプ0でなくタイプ3のパケットが返ってきた



ICMPタイプだけではあまり有益な情報ではありませんが、ICMPはコード番号も返してくれます。たとえばコード番号1 (ホスト到達不能) はコンピュータに到達できなかったことを表します。たいていのタイプはコード番号も持っています。2番目のパケットを送信したのは、10.2.10.2が通信したかったコンピュータではありません。それはすなわちEchoリクエストが送信先に伝わらなかったということです。

コード1は、Echoリクエストがルータやスイッチを通りはしたものの、送信先のコンピュータには届かなかった、という意味です。コード1が返ってきたときにはルータやスイッチから送信されるARPブロードキャストを見てみましょう。ARPブロー

ドキャストに反応がなければ、送信先のコンピュータを見つけることができなかったということなので、タイプ3のコード1を送信元のコンピュータに返します。

7.2.2 ポート到達不能

トラブルシューティングでよくあるパターンとしてもう1つあげられるのが、特定のポートであるサービスが通信を受け入れられるようになっているかどうかを確認することです。

たとえば、FTPサーバの21番ポートが接続可能かどうかを実際に接続して調べたとしましょう。もしなんらかの理由で21番ポートに接続できない場合、タイプ0、コード2の「ポート到達不能」が返ってきます。

ネットワーク管理ではICMPを頻繁に使うので、基本的なタイプやコードについて押さえておきましょう。筆者は小さなクイックリファレンスを職場の机に置いています。

7.3 IPフラグメンテーション

ipfragments.pcap

IPはネットワークを介したデータ転送に使われています。一度にケーブルを通ることができるデータの量は限られていますので、IPにはフラグメンテーション（分割）という機能があります。IPフラグメンテーションを使えば、大きなデータを小さく分割して送信し、送信先でそれを元のデータに組み立てることができるようになります。

この節では、IPフラグメンテーションによって分割されたデータの流れを見ていきます。

サンプルファイルには、pingのやりとりが24個記録されています。前述のとおり、pingのやりとりは通常8個のパケットで済むはずですが、なぜ24個ものパケットが記録されているのでしょうか？ このサンプルファイルでは、1つのEchoリクエストに対してEchoリプライパケットが3つ返ってきています。ということはつまり、通常の3倍のパケットが送信されていることになります（図7-7）。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.114	192.168.0.193	IP	Fragmented IP protocol (proto=ICMP 0x01, off=0)
2	0.000085	192.168.0.114	192.168.0.193	IP	Fragmented IP protocol (proto=ICMP 0x01, off=1480)
3	0.000094	192.168.0.114	192.168.0.193	ICMP	Echo (ping) request
4	0.004244	192.168.0.193	192.168.0.114	IP	Fragmented IP protocol (proto=ICMP 0x01, off=0)
5	0.004545	192.168.0.193	192.168.0.114	IP	Fragmented IP protocol (proto=ICMP 0x01, off=1480)
6	0.004623	192.168.0.193	192.168.0.114	ICMP	Echo (ping) reply

図7-7 ここでは、Echoリクエスト1つに対するEchoリプライが3つ

pingのデータサイズがデフォルトよりも大きいと、このように分割されて送信されます。Windowsのデフォルトではpingのデータサイズは32バイトですが、サンプルファイルのpingのデータサイズは3,072バイトです。イーサネットでは1回に送信できるパケットのサイズの上限は1,500バイトですので、IPはパケットを分割しなければなりません。

7.3.1 IPフラグメンテーションを実行するかどうか

パケットを受信したコンピュータは、それが分割されているかどうかをどうやって知るのでしょうか？ パケット詳細のペインを見ればすぐに分かります。ipfragments.pcapで以下を実行してみてください。

1. 1番目のパケットで、パケット詳細のペインのIPの部分を広げます。
2. [Flags]の部分を広げてください。図7-8のように、3つの欄に分かれているはずです。More fragmentsの欄に注目してください。数値が1になっています。これは、このパケットは分割されており、この後に残りのパケットが続くことを意味しています。

```

Internet Protocol, Src: 192.168.0.114 (192.168.0.114), Dst: 192.168.0.193 (192.168.0.193)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 1500
  Identification: 0x61d1 (25041)
  Flags: 0x02 (More Fragments)
    0... = Reserved bit: Not set
    .0.. = Don't fragment: Not set
    ..1. = More fragments: Set ←
  Fragment offset: 1480
  Time to live: 128
  Protocol: ICMP (0x01)
  Header checksum: 0x3013 [correct]
  Source: 192.168.0.114 (192.168.0.114)
  Destination: 192.168.0.193 (192.168.0.193)
  Reassembled IP in frame: 3
  Data (1480 bytes)

```

図7-8 More fragmentsのフラグが1の場合、分割されたパケットが後に続く

3. 2番目のパケットの同じ部分を見てください。同じくMore fragmentsのフラグが1になっています。
4. 3番目のパケットのMore fragmentsの部分を見てください(図7-9)。1番目と2番目のパケットと違い、このパケットはMore fragmentsの値が0になっています。これでこのパケットのデータはすべて送信されたということです。More fragmentsの値は1か0にしかありません。

```

Internet Protocol, Src: 192.168.0.114 (192.168.0.114), Dst: 192.168.0.193 (192.168.0.193)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 140
  Identification: 0x61d1 (25041)
  Flags: 0x00
    0... = Reserved bit: Not set
    .0.. = Don't fragment: Not set
    ..0. = More fragments: Not set ←
  Fragment offset: 2960
  Time to live: 128
  Protocol: ICMP (0x01)
  Header checksum: 0x54aa [correct]
  Source: 192.168.0.114 (192.168.0.114)
  Destination: 192.168.0.193 (192.168.0.193)
  [IP Fragments (3080 bytes): #1(1480), #2(1480), #3(120)]
  Internet Control Message Protocol

```

図7-9 More fragmentsフラグが0ということは、このパケットのデータの転送が終了したということ

7.3.2 順番に組み立てる

次に浮かんでくる疑問は、分割されたパケットをどうやって正しい順番で組み立てていくのか、でしょう。IPは分割されたパケットを送信するとき、その順番を示すオフセット値を用意します。

パケット詳細のペインからオフセット値を見ることができます。たとえば1番目のパケットのIPの部分を見ると、オフセット値 (Fragment offset) が0になっているのが分かります。これは、このパケットが分割されたパケットの最初のパケットであることを示しています。

2番目のパケットでは、オフセット値が突然1480という大きな数値になっています (図7-10)。この値は、直前のパケットのデータのサイズによって決まります。直前のパケットのデータサイズとオフセット値を足したものが、次のパケットのオフセット値になります。2番目のパケットのオフセット値は、直前のパケットのデータサイズが1,480バイト、オフセット値が0なので、1480になるのです。

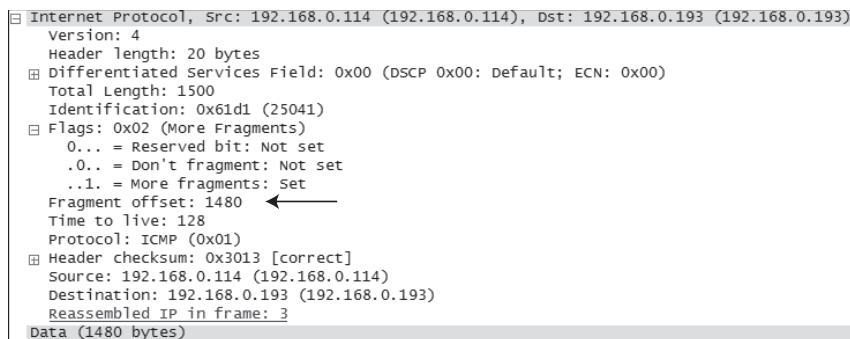


図7-10 2番目のパケットのオフセット値は直前のパケットのデータサイズに依存する

3番目のパケットのオフセット値は、2番目のパケットのデータサイズが1,480バイト、オフセット値が1480なので2960になります (図7-11)。

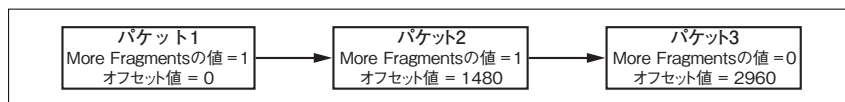


図7-11 IPフラグメンテーションによるパケットの分割

サンプルファイルのほかのIPフラグメンテーションでも、各パケットのオフセット値を見てみてください。大量のパケットが入り乱れるキャプチャファイルでは、これが思いのほか難しいのです。

7.4 接続不能

barryscomputer.pcap
bethscomputer.pcap

それではこれから、Wiresharkを使って現実のネットワーク障害の解析をしていきましょう。このシナリオでは、2人のユーザー、バリーとベスが登場します。彼らはオフィスで隣同士に座っています。予算が増えたので、IT 部がちょうど2人に新しいコンピュータを購入したところです。あなたはこの2人のコンピュータを正しく機能するように設定しなければなりません。あなたは2台のコンピュータを箱から出してコンセントを差し込み、さまざまな設定を終えて、テストを始めました。そしてすぐに問題にぶつかりました。バリーのコンピュータは問題なくネットワークに接続できましたが、ベスのものはインターネットにアクセスすることができません。あなたはこれから、ベスのコンピュータがインターネットにアクセスできない理由を調査し、それを直さなければいけません。

7.4.1 分かっていること

トラブルシューティングで最初にやらなければいけないことは、その障害について自分が分かっていることのリストを作ることです。この場合、バリーとベスは真新しい同じコンピュータを使っているということが分かっています。また、IPアドレスは自分で設定し、ネットワークセグメント上のコンピュータにEcho リクエストを送信すると正しく返ってくるということも知っています。最後に、2台のコンピュータの設定は完璧に同じであることも分かっています。なにせあなたが自分で設定したのですから。

7.4.2 パケット解析開始

障害について自分が分かっていることを洗い出したら、今度は知らないことをどうやって調べるかについて考えましょう。どんなタイプのトラフィックをキャプチャするか、どこにスニッファマシンを設置するかを考えればよいのです。

インターネットにアクセスできない、というのが問題なので、理論的にはベスのコンピュータがインターネットに接続しようとしているときのパケットをキャプチャすればよいように思います。加えて、バリーとベスのコンピュータが接続されているネットワークをあまり知らないので、バリーのコンピュータのパケットもキャプチャしましょう。この2つのキャプチャファイル、接続できるものとできないものをキャプチャしたファイルを解析します。2つのファイルを比較することで問題がはっきりするでしょう。これをベースライニングと呼びます[†]。Wireshark を2台のコンピュータにインストールしてください。

[†] 監訳注：ベースライニングとは、たとえば正常な状態を詳細に記録しておいて、それを基軸＝ベースラインとして、現状は何がどう異なっているというところに着目する調査方法のことです。

7.4.3 解析

まずはインターネットにアクセスできているバリーのコンピュータのキャプチャファイル (barryscomputer.pcap) から見てみましょう。サンプルファイルを開くと、HTTPの通信が見えます。

図7-12のように、まずデフォルトゲートウェイ (192.168.0.10) のレイヤ2アドレスを要求するARPのプロードキャストがあります。バリーのコンピュータはARPリクエストの返答を受け取ると、Webサーバと3ウェイハンドシェイクを始めます。その後、サーバからデータが送信されています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	Microsof_2a:45:d2	Broadcast	ARP	who has 192.168.0.10? Tell 192.168.0.183
2	0.002196	D-Link_21:99:4c	Microsof_2a:45:d2	ARP	192.168.0.10 is at 00:05:5d:21:99:4c
3	0.002259	192.168.0.183	64.233.161.104	TCP	1125 > http [SYN] Seq=0 Len=0 MSS=1460
4	0.054708	64.233.161.104	192.168.0.183	TCP	http > 1125 [SYN, ACK] Seq=0 Ack=1 win=8190 Len=0 MSS=1452
5	0.054871	192.168.0.183	64.233.161.104	TCP	1125 > http [ACK] Seq=1 Ack=1 win=65535 Len=0
6	0.055737	192.168.0.183	64.233.161.104	HTTP	GET / HTTP/1.1
7	0.103969	64.233.161.104	192.168.0.183	TCP	http > 1125 [ACK] Seq=1 Ack=284 win=6432 Len=0
8	0.158478	64.233.161.104	192.168.0.183	TCP	[TCP segment of a reassembled PDU]
9	0.161865	64.233.161.104	192.168.0.183	HTTP	HTTP/1.1 200 OK (text/html)

図7-12 バリーのコンピュータは3ウェイハンドシェイク後、HTTPによるデータの送信が行われている

これでHTTPの通常の通信が、このネットワーク上でどのように見えるかが分かったので、ベスのコンピュータのキャプチャファイル (bethscomputer.pcap) を見てみましょう。おかしいことが起きているとすぐに分かるはずです。図7-13にあるように、最初はbarryscomputer.pcapと同じようなARPリクエストのパケットです。しかしながらARPリクエストは192.168.0.10でなく192.168.0.11のレイヤ2アドレスをリクエストしています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	Microsof_2a:45:d2	Broadcast	ARP	who has 192.168.0.11? Tell 192.168.0.122

図7-13 ベスのコンピュータは別のIPアドレスのレイヤ2アドレスを要求している

ARPリクエストの後には、NetBIOSのトラフィックが流れています (図7-14)。このNetBIOSのトラフィックが、何かおかしいということを証明しています。

No. -	Time	Source	Destination	Protocol	Info
2	1.271630	192.168.0.122	192.168.0.255	BROWSE	Request Announcement TESLA-MARKETING
3	2.424840	192.168.0.122	192.168.0.255	BROWSE	Host Announcement TESLA-MARKETING, workstation, Server, NT Workstation, Potential
4	2.425448	192.168.0.122	192.168.0.255	BROWSE	Host Announcement TESLA-MARKETING, workstation, Server, NT Workstation, Potential

図7-14 このNetBIOSのトラフィックはおかしい

NetBIOSは古いプロトコルで、現在はTCP/IPが機能しなかったときに使われることがほとんどです。NetBIOSのトラフィックが発生しているということは、ベスのコンピュータはTCP/IPを使ってインターネットにアクセスすることができず、かわりにNetBIOSを使った通信を試みた (そしてそれも失敗した) ということです。NetBIOSのトラフィックがキャプチャされたときは、ネットワークに何か問題があると考えてよいでしょう。

バリーのキャプチャファイルと違うところはなんでしょうか？ ARPパケットがレ

イヤ2アドレスを要求している相手のIPアドレスが異なるということです。バリーのコンピュータはデフォルトゲートウェイである192.168.0.10のレイヤ2アドレスを要求しており、ベスのコンピュータは192.168.0.11のレイヤ2アドレスを要求して失敗しています(図7-15)。デフォルトゲートウェイのアドレスが一致していません。

バリーのコンピュータ	
□	Address Resolution Protocol (request)
	Hardware type: Ethernet (0x0001)
	Protocol type: IP (0x0800)
	Hardware size: 6
	Protocol size: 4
	Opcode: request (0x0001)
	Sender MAC address: 00:03:ff:2a:45:d2 (00:03:ff:2a:45:d2)
	Sender IP address: 192.168.0.183 (192.168.0.183)
	Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
	Target IP address: 192.168.0.10 (192.168.0.10)
ベスのコンピュータ	
□	Address Resolution Protocol (request)
	Hardware type: Ethernet (0x0001)
	Protocol type: IP (0x0800)
	Hardware size: 6
	Protocol size: 4
	Opcode: request (0x0001)
	Sender MAC address: 00:03:ff:2a:45:d2 (00:03:ff:2a:45:d2)
	Sender IP address: 192.168.0.122 (192.168.0.122)
	Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
	Target IP address: 192.168.0.11 (192.168.0.11)

図7-15 ARPが要求しているIPアドレスが違うことが問題

TCP/IPの設定を確認してみたら、typoが見つかりました。バリーのコンピュータでは、デフォルトゲートウェイは192.168.0.10で、ベスのコンピュータでは192.168.0.11になっていました。192.168.0.11は間違ったアドレスです。

7.4.4 まとめ

この先あなたが出くわす障害の原因は、設定ミスであることが多いでしょう。その可能性があるときには、正常に動作しているコンピュータのキャプチャファイルと比べてみてください。今回のシナリオでは、正常なトラフィックと比較することで、どのパケットがおかしいのかを正確に見抜くことができました。障害の原因を特定できれば、その修正ははるかに簡単になるはずです。

7.5 Internet Explorerの悪魔

hauntedbrowser.pcap

このシナリオは、あなたが管理しているネットワークのユーザーであるチャドからの、ヘルプデスクへの電話から始まります。チャドによると、彼のコンピュータには最近悪魔が憑いているそうです。彼のブラウザのホームページは何をしても気象サイトを表示するようになってしまいました。手動で設定を変更しても、コンピュータを

再起動すると元に戻ってしまうそうです。これからあなたはことの真相を探り出し、チャドのコンピュータの悪魔払いをしなければなりません。

7.5.1 分かっていること

チャドは長い間この会社に務めており、技術的な知識があまりないことをあなたは知っています。実際に、彼にとってコンピュータは利益でなく害をもたらすものです。チャドのコンピュータは2年前に購入したもので、OSはWindows XP、ブラウザはInternet Explorer 6であることも分かっています。

7.5.2 パケット解析開始

これはチャドのコンピュータでのみ起きている問題なので、チャドのコンピュータに送受信されるパケットをキャプチャするだけで事足ります。コンピュータを再起動したときにホームページの設定が戻ってしまうということだったので、あなたのコンピュータで彼のコンピュータが再起動するときのパケットをキャプチャしてみましょう。

このシナリオでは、チャドのコンピュータにWiresharkをインストールことはできませんので、ハブを使ってパケットをキャプチャします。もしその方法を忘れてしまったのなら2章の「2.3.2 ハブの使用」を参照してください。チャドのコンピュータの電源を入れてから完全に起動するまでの間、パケットをキャプチャします。ユーザーが何かを操作する必要はありません。

7.5.3 解析

ユーザーがコンピュータにまったく触れていないにもかかわらず、相当な数のTCPとHTTPのパケットがサンプルファイルに記録されていることに驚くかもしれません（図7-16）。通常の起動時にはこのようなパケットがキャプチャされることはほとんどありません。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.184	24.46.230.187	TCP	1038 > 1706 [SYN] Seq=0 Len=0 MSS=1460
2	0.000019	192.168.0.184	69.206.254.66	TCP	1039 > 3531 [SYN] Seq=0 Len=0 MSS=1460
3	0.001354	192.168.0.184	24.46.230.187	TCP	1038 > 1706 [SYN] Seq=0 Len=0 MSS=1460
4	0.002375	192.168.0.184	69.206.254.66	TCP	1039 > 3531 [SYN] Seq=0 Len=0 MSS=1460
5	0.338822	192.168.0.184	64.124.109.200	HTTP	GET /command/commandv6.07.asp?key=&t=26962 HTTP/1.1
6	0.340546	192.168.0.184	64.124.109.200	HTTP	[TCP out-of-order] GET /command/commandv6.07.asp?key=&t=26962 HTTP/1.1
7	0.638241	192.168.0.184	64.124.109.200	TCP	1040 > http [ACK] Seq=286 Ack=243 win=65041 Len=0
8	0.638386	192.168.0.184	64.124.109.200	TCP	[TCP Dup ACK #1] 1040 > http [ACK] Seq=286 Ack=243 win=65041 Len=0
9	0.800253	192.168.0.184	64.124.109.200	TCP	1040 > http [ACK] Seq=286 Ack=613 win=64671 Len=0
10	0.800403	192.168.0.184	64.124.109.200	TCP	[TCP Dup ACK #1] 1040 > http [ACK] Seq=286 Ack=613 win=64671 Len=0

図7-16 ユーザーがなんの操作もしていないにもかかわらず大量のパケットが流れており、何か問題があることが分かる

これらのパケットをもう少し詳しく見てみると、すぐにある結論に達するでしょう。まず、ほとんどのGETリクエストがチャドのコンピュータのIPアドレスから送信されていることが分かります。さらに、5番目のパケットを見ると（図7-17）、データをダウンロードしようとGETリクエストをあるWebサーバに送信していることが分かります。

```
Hypertext Transfer Protocol
GET /command/Commandv6.07.asp?key=&t=26962 HTTP/1.1\r\n
Request Method: GET
Request URI: /command/Commandv6.07.asp?key=&t=26962
Request Version: HTTP/1.1
User-Agent: Mozilla/3.0 (compatible; MSIE 4.0; win32)\r\n
Host: command.weatherbug.com\r\n
Connection: Keep-Alive\r\n
Cookie: wxbug_cookie=has_cookies=1; RMID=4aecf9dc45a025d0; RMFD=011H3KJT0104ym|01058k; RMFS=011H3KhLU1052U; LMB1per12h=1\r\n\r\n
```

図 7-17 5 番目のパケットをよく見ると、インターネットからデータをダウンロードしようとしているのが分かる

このことから、コンピュータの起動時に、意図しない何かが実行されていることが分かります。パケット一覧のペインの下の方を見ると、問題の原因となっているものの一端が見えます。11 番目と 12 番目のパケットで、weatherbug.com というドメイン名の名前解決をしています (図 7-18)。

No. -	Time	Source	Destination	Protocol	Info
11	3.725242	192.168.0.184	205.152.37.23	DNS	Standard query A deskwx.weatherbug.com
12	3.734060	192.168.0.184	205.152.37.23	DNS	Standard query A deskwx.weatherbug.com

図 7-18 weatherbug.com の名前解決のパケットが犯人特定の手がかりだ

犯人はコンピュータを起動したときにチャドのホームページを気象のサイトに変更している何かです。さらに調査を続けると、WeatherBug のデスクトッププログラムがバックグラウンドで動作しており、コンピュータを再起動すると新しい天気予報をダウンロードし表示するように設定されていることが分かりました。このプログラムをアンインストールすると、問題は解決しました。

7.5.4 まとめ

問題の原因が特定のコンピュータやネットワークではなく、ソフトウェアだった、ということもよくあります。このシナリオでは、天気を追跡するプログラムがチャドのコンピュータにインストールされており、そのプログラムが Web ブラウザのホームページを変更していたことで、チャドはコンピュータが乗っ取られたと思ってしまったのです。Wireshark でパケットをキャプチャすることによって、このプログラムがバックグラウンドで知らずに動いていたということが分かりました。

問題をパケットレベルで見ることによって、トラブルシューティングはとても簡単になります。

7.6 FTPサーバとの通信

ftpclientdenied.pcap
ftpserverdenied.pcap

次のシナリオは、あなたが企業の FTP サーバを構築し終わった直後の話です。クライアントはネットワークの内側または外側から FTP サーバにアクセスし、データをダウンロードしたりアップロードしたりします。FTP サーバのソフトウェアをインストールし、従業員が使えるようにユーザー名とパスワードを作成しました。しか

しながら、なんらかの理由でFTPのクライアントソフトウェアからFTPサーバにアクセスすることができません。

7.6.1 分かっていること

このサーバは、最新のアップデートとサービスパックを適用した Windows Server 2003 上に構築されています。FTPサーバが正しく動作していることは確認済みです。また、クライアントはFTPサーバに接続するときに正しいIPアドレスと認証情報を使っていることも分かっています。

7.6.2 パケット解析開始

FTPはクライアントサーバ型のサービスなので、クライアントとサーバの両方のコンピュータのパケットをキャプチャします。クライアントソフトウェアがFTPサーバに接続しようとするときに、クライアントからのパケットをキャプチャします。サーバがクライアントソフトウェアに接続しようとするときに、サーバからのパケットをキャプチャします。こうすれば、問題の原因がクライアントにあるのかサーバにあるのかを確認し、より詳しく調査することができます。クライアントとサーバにWiresharkをインストールし、パケットをキャプチャしましょう。

7.6.3 解析

まずはクライアントが正しく通信を始めることができるかどうか確認してみましょう。サンプルファイルのftpclientdenied.pcapを開いてください(図7-19)。FTPサーバである192.168.0.182と3ウェイハンドシェイクをしようとしています、サーバが応答しません。クライアントはさらに2つのSYNパケットを送信し、セッションを確立しようとしています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.193	192.168.0.182	TCP	1596 > 182 [SYN] Seq=0 Len=0 MSS=1460
2	2.944417	192.168.0.193	192.168.0.182	TCP	1596 > 182 [SYN] Seq=0 Len=0 MSS=1460
3	8.979791	192.168.0.193	192.168.0.182	TCP	1596 > 182 [SYN] Seq=0 Len=0 MSS=1460

図7-19 クライアントがSYNパケットを送信するもサーバから返事が来ないので、さらに2つのSYNパケットを送信している

クライアントはこのあと9秒間サーバに接続を試み、失敗します。クライアントは3ウェイハンドシェイクを正しく始めようとしていますから、問題の原因はクライアントの側ではなさそうです。

それではftpserverdenied.pcapを見てみましょう。この2つのサンプルファイルはほとんど同じに見えますが、キャプチャしたタイミングが異なるため、パケットのソースポート番号(送信元ポート番号)が違っています(図7-20)。サーバにおいてパケットがキャプチャできている事実に注目してください。これはクライアントから送られてくるパケットがサーバまで届いているということを示しています。

つまり、ネットワーク障害などで通信がサーバまで到達していないのではなく、通

信は届いているがサーバが応答していない、ということが分かるわけです。この事実が分かったとって原因が推測できるわけではありませんが、少なくとも切り分けはできるでしょう。闇雲にいろんな可能性を探るより、はるかに効率的に調査ができるわけです。

クライアント

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.193	192.168.0.182	TCP	1596 > ftp [SYN] Seq=0 Len=0 MSS=1460
2	2.944417	192.168.0.193	192.168.0.182	TCP	1596 > ftp [SYN] Seq=0 Len=0 MSS=1460
3	8.979791	192.168.0.193	192.168.0.182	TCP	1596 > ftp [SYN] Seq=0 Len=0 MSS=1460

サーバ

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.193	192.168.0.182	TCP	1637 > ftp [SYN] Seq=0 Len=0 MSS=1460
2	2.992575	192.168.0.193	192.168.0.182	TCP	1637 > ftp [SYN] Seq=0 Len=0 MSS=1460
3	9.027733	192.168.0.193	192.168.0.182	TCP	1637 > ftp [SYN] Seq=0 Len=0 MSS=1460

図7-20 クライアントとサーバのキャプチャファイルはほとんど同じ

サーバがパケットを拒否する理由には、主に以下の3つがあります。

- サービスが稼働していない場合。FTPサーバが動作していることは確認済みなので、これは今回の理由ではありません。
- サーバに大量のアクセスがある場合。サーバの許容量を超えたトラフィックがあると、サーバにアクセスできなくなる場合があります。サーバは構築されたばかりでまだ使われていないので、これも今回の理由ではありません。
- パケットが意図的にブロックされている場合。そんなことがあるのでしょうか？あるのです！ 調査してみると、WindowsのファイアウォールがFTPポートのトラフィックをブロックしていることが分かりました。

7.6.4 まとめ

パケット解析では障害そのものの原因は分からないかもしれません。実際に、このシナリオではファイアウォールが問題であることを示すようなパケットはキャプチャされていませんでした。しかしながら、パケット解析によって問題がサーバ側にあることは分かります。

数十台、数百台に影響が及ぶ問題を解決しなければならないときに、パケット解析によって問題の原因がどのコンピュータにあるのかが分かれば、ものすごい時間の節約になるでしょう。

7.7 私のせいじゃない！

http-fault-post.pcap

ユーザーの中には本当にどうしようもない人がいます。あらゆる問題をすべてIT部のせいにするユーザーに出会ったことはありませんか？ エリンはまさにそういうタイプのユーザーです。ネットワークが少しでもおかしくなると、すぐに知らせてきます。

このシナリオでは、エリンはある製品をオンラインで注文しようとしています。問題は彼女が製品の注文フォームを送信しようとする、HTTP 403 (Forbidden) エラーが返ってくるということです。問題の原因はWebサイト側にあることがほとんどですが、エリンがあなたの上司にあまりに文句を言うので、上司はあなたにこれがこちら側のミスではないことを彼女に証明するように頼みにきました。あなたは彼女にこれがWebサイト側のエラーであることを証明しなければいけません。

7.7.1 分かっていること

エリンが件のWebサイトで首尾よくWebフォームからデータを送信できたことはありませんが、ほかのサイトではちゃんと送信できています。問題のWebサイトのフォームのソースには特におかしいところはありません。

7.7.2 パケット解析開始

エリンのコンピュータにWiresharkをインストールしパケットをキャプチャするのが一番簡単です。Wiresharkをインストールしてパケットキャプチャを開始し、エリンにフォームを入力して送信してもらいましょう。

7.7.3 解析

http-fault-post.pcapはエリンのコンピュータ24.4.97.251とWebサーバ216.23.168.114の間で3ウェイハンドシェイクを行うところから始まります(図7-21)。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	24.4.97.251	216.23.168.114	TCP	2580 > http [SYN] Seq=0 Len=0 MSS=1460
2	0.027956	216.23.168.114	24.4.97.251	TCP	http > 2579 [SYN, ACK] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
3	0.028045	24.4.97.251	216.23.168.114	TCP	2579 > http [ACK] Seq=0 Ack=1 Win=65535 Len=0

図7-21 ここまでは順調。通常の3ウェイハンドシェイクがエリンのコンピュータとWebサーバとで始まっている

その後、クライアントサーバ間でHTTPによる通信が始まります。すぐにパケット一覧の[Info]欄に、苦情の原因となったHTTP 403のメッセージが現れます(図7-22)。9番目のパケットで403のエラーが発生しています。パケットを右クリックして

No. -	Time	Source	Destination	Protocol	Info
9	0.065496	216.23.168.114	24.4.97.251	HTTP	HTTP/1.1 403 Forbidden (text/html)

図7-22 HTTP 403のメッセージがすぐに見えてくる

[Follow TCP Stream] を選択し、HTTP の通信を見てみてください (図 7-23)。

TCP ストリームを見ると、クライアントからサーバにデータが送信されているこ

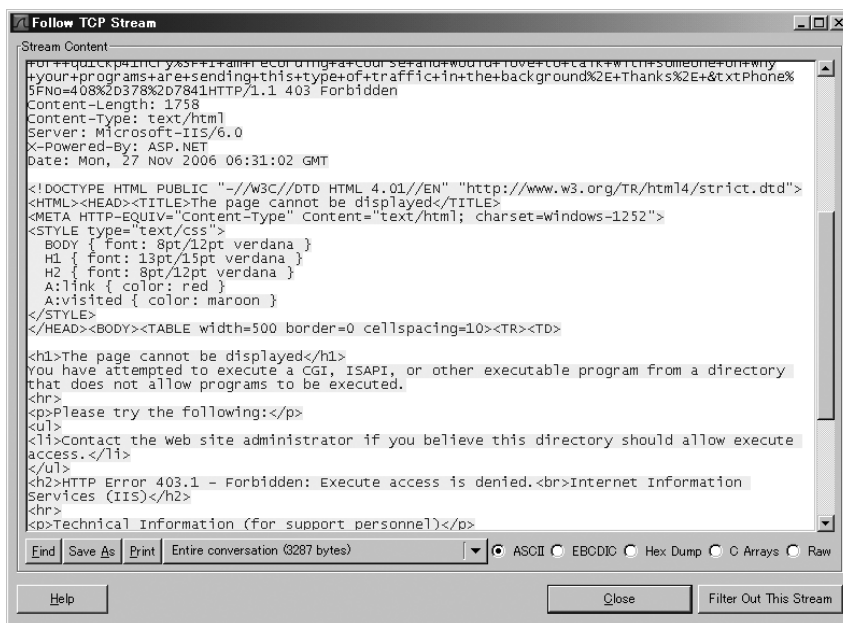


図 7-23 TCP ストリームから、今回の問題である 403 のエラーメッセージが見える

とが分かります。ここで、サーバからフォームを受信した旨をクライアントに伝えるべきなのにもかかわらず 403 のエラーが出ています。これで、問題があなたの管理するネットワークでなく Web サーバ側にあるということを説明できるでしょう。

7.7.4 まとめ

障害の原因について立てた仮説を確認するためだけではなく、濡れ衣を着せられたときに無実を証明するためにもパケット解析する必要があります。

このシナリオでは、TCP ストリームを上司に見せ、エリンに説明してもらえば、IT 部への非難はやむでしょう。

7.8 悪魔のプログラム

evilprogram.pcap

このシナリオは前述のチャドの話と似ています。しかしながら今回はもう少し複雑です。マンディというユーザーがあなたのネットワークを使っています。彼女のブラウザに何かおかしいことが起きているとあなたに言ってきました。1日のうち何度かブラウザのホームページが偽のセキュリティのWebサイトに変更されます。加えて、多くのポップアップがコンピュータ上に表示されます。

コンピュータの修理をしたことがある人なら、これがスパイウェアのせいであることが分かるでしょう。正解です。しかし、ここでスパイウェア除去ソフトを使うのではなく、マンディのコンピュータがスパイウェアの影響をどのくらい受けているのかを追跡してみましょう。

7.8.1 分かっていること

この問題を解決するのに多くの調査は必要ありません。あなたはマンディのコンピュータの動作が遅く、ブラウザが絶えずハイジャックされていることを知っています。彼女のコンピュータ上ではウイルス対策ソフトが動作しているので、ウイルスについてはあまり心配する必要はありません。

7.8.2 解析開始

スパイウェア関連のトラブルシューティングをするときは、コンピュータが起動するときのパケットをキャプチャするのが有効です。ほとんどのスパイウェアは、コンピュータが起動するとアップデートがないかどうかを確認するために自身のWebサイトにアクセスします。

コンピュータを起動させると同時にパケットをキャプチャし始め、起動が完了するまでのおよそ1分ほどキャプチャし続けます。この場合、ハブを使うかARPキャプシュポイズニングがパケットキャプチャするためのもっともよい方法でしょう。ネットワーク上には多くのトラフィックが存在するため、キャプチャフィルタを使ってマンディのコンピュータのトラフィックのみをキャプチャするようにします。

7.8.3 解析

かなり大きなキャプチャファイルになるので、最初から見ていきましょう。最初の2つのパケットはコンピュータの起動時にはよく見るもので、TCP/IPの初期化をしています（図7-24）。

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x9a71fd19
2	0.210908	Intel_b7:f2:f5	Broadcast	ARP	Gratuitous ARP for 24.6.125.19 (Request)

図 7-24 最初の2つのパケットでマンディのコンピュータはIPアドレスを取得し、それがネットワーク上で重複していないか確認している

最初のパケットで、DHCPサーバにIPアドレスを割り振ってくれるよう求めています。本来はDHCPサーバから応答のパケットが表示されているはずですが、これはブロードキャスト宛のパケットなのでキャプチャフィルタによってキャプチャされないようになっています。

2番目のパケットはGratuitous ARPと呼ばれるARPパケットです。Gratuitous ARPはブロードキャスト宛のARPパケットで、ネットワーク上に自分と同じIPアドレスを使っているコンピュータがないかどうかを確認します。もしGratuitous ARPリクエストに返事が返ってくれば、そのIPアドレスはすでに誰かに使われているということになります。このキャプチャファイルではそういった返事はありませんので問題ありません。

3番目のパケットがあなたが見るべきパケットです。このパケットの後でGratuitous ARPが送信されているので、この時点ではTCP/IPはまだ初期化している状態です(図7-25)。しかし3番目のパケットで、外部のネットワークからマンディのコンピュータの5554番ポートにアクセスしようとしているコンピュータがいます。

No. .	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x9a71fd19
2	0.210908	Intel_b7:f2:f5	Broadcast	ARP	Gratuitous ARP for 24.6.125.19 (Request)
3	0.938012	67.70.67.186	24.6.125.19	TCP	2431 > 5554 [SYN] Seq=0 Len=0 MSS=1440
4	0.958370	Intel_b7:f2:f5	Broadcast	ARP	Gratuitous ARP for 24.6.125.19 (Request)
5	1.755318	67.70.67.186	24.6.125.19	TCP	2860 > 9898 [SYN] Seq=0 Len=0 MSS=1440
6	1.959831	Intel_b7:f2:f5	Broadcast	ARP	Gratuitous ARP for 24.6.125.19 (Request)

図7-25 マンディのコンピュータは3番目と5番目のパケットを受け取る準備がまだできていない

初期化中のコンピュータはまだ通信を受け入れる準備ができていませんから、ほかのコンピュータはいかなる通信も試みるべきではありません。したがってマンディのコンピュータは、起動中はこれらのパケットを破棄します。5番目のパケットの後に同じようなパケットがありますが、このパケットは送信先のポートを9898番に変えています(図7-26)。とても怪しいです。

<div>Transmission Control Protocol, Src Port: 2860 (2860), Dst Port: 9898 (9898), Seq: 0, Len: 0</div> <div>Source port: 2860 (2860)</div> <div>Destination port: 9898 (9898)</div> <div>Sequence number: 0 (relative sequence number)</div> <div>Header length: 28 bytes</div> <div>Flags: 0x02 (SYN)</div> <div>Window size: 16384</div> <div>Checksum: 0x1ba8 [correct]</div> <div>Options: (8 bytes)</div>
--

図7-26 もう一度マンディのコンピュータにアクセスしようとしている

繰り返しますがマンディのコンピュータはまだ通信を受け入れる準備ができていないので、このパケットも破棄します。

マンディのコンピュータが通信できるようになると、このパケットを受け取ってしまいます(10番目のパケット)。マンディは3ウェイハンドシェイクを受け入れられるサービスを起動していないので、RSTパケットを返して通信を終了しています(図7-27)。

このやりとりは何回か繰り返され、マンディのコンピュータは通信を拒否し続けて

No. .	Time	Source	Destination	Protocol	Info
10	556.468939	203.101.42.68	24.6.125.19	TCP	1560 > 4899 [SYN] Seq=0 Len=0 MSS=1460
11	556.468943	Intel_b7:f2:f5	Broadcast	ARP	who has 24.6.112.1? Tell 24.6.125.19
12	556.478580	24.6.125.19	203.101.42.68	TCP	4899 > 1560 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
13	556.480712	Cadant_22:89:c2	Intel_b7:f2:f5	ARP	24.6.112.1 is at 00:01:5c:22:89:c2
14	557.229229	203.101.42.68	24.6.125.19	TCP	1560 > 4899 [SYN] Seq=0 Len=0 MSS=1460
15	557.229235	24.6.125.19	203.101.42.68	TCP	4899 > 1560 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
16	557.932751	203.101.42.68	24.6.125.19	TCP	1560 > 4899 [SYN] Seq=0 Len=0 MSS=1460
17	557.932857	24.6.125.19	203.101.42.68	TCP	4899 > 1560 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
18	596.714158	211.91.150.78	24.6.125.19	TCP	2597 > 9898 [SYN] Seq=0 Len=0 MSS=1440

図7-27 マンディのコンピュータが通信できるようになっても、RSTパケットを返して通信が終了する

います。

7.8.3.1 正常なパケットをフィルタリングする

68番目のパケットが、最初の通常の通信です（図7-28）。

マンディのコンピュータはウイルス対策ソフトのアップデートのための通信を始め

No. .	Time	Source	Destination	Protocol	Info
68	1132.623271	24.6.125.19	216.148.227.68	DNS	standard query A updatekeepalive.mcafee.com
69	1132.658710	216.148.227.68	24.6.125.19	DNS	standard query response A 216.49.88.118

図7-28 ウイルス対策ソフトのアップデートのためのパケット

ます。これらのパケットは正規のものなので、疑わしいパケットのみを表示するため、McAfeeのIPアドレスをフィルタリングしましょう（図7-29）。

Filter:	!ip.addr==216.49.88.118	▼	Expression...	Clear	Apply
---------	-------------------------	---	---------------	-------	-------

図7-29 正規のトラフィックをフィルタリングして、怪しいパケットに焦点を当てよう



フィルタの作り方は覚えていますか？ McAfeeのサーバとの通信をフィルタリングする場合には、!ip.addr==216.49.88.118と書けばよいのです。

7.8.3.2 インターネットからのアクセスの試み

フィルタを作成したら、147番目のパケットを見てみましょう（図7-30）。

No. .	Time	Source	Destination	Protocol	Info
147	1202.816709	221.143.42.254	24.6.125.19	Messen	NetrSendMessage request

図7-30 147番目のパケットはメッセージャーパケット。このパケットは詳しく調査する必要がある

このメッセージャーパケットはインターネットから送られています。バイナリのペインを見ると、どういうメッセージが送られてきているのかが分かります（図7-31）。

ありがたいことに、あなたのネットワークではメッセージャーサービスは無効になっており、マンディはこのメッセージを見たことはありません。その証拠に、マン

0060	00 00 d2 00 00 00 00 00 00 00 d2 00 00 00 41 4cAL
00c0	45 52 54 3a 20 44 41 4e 47 45 52 4f 55 53 20 53	ERT: DAN GERIOUS S
00d0	50 59 57 41 52 45 20 56 49 52 55 53 20 46 4f 55	PYWARE V IRUS FOU
00e0	4e 44 0a 44 45 4c 45 54 45 20 54 48 49 53 20 53	ND.DELET E THIS S
00f0	50 59 57 41 52 45 20 56 49 52 55 53 20 49 4d 4d	PYWARE V IRUS IMM
0100	45 44 49 41 54 45 4c 59 0a 56 49 53 49 54 20 54	EDIATELY .VISIT T
0110	48 45 20 57 45 42 53 49 54 45 20 57 57 57 2e 50	HE WEBSI TE WWW.P
0120	34 55 32 2e 43 4f 4d 20 54 4f 20 43 4f 4e 54 49	4U2.COM TO CONTI
0130	4e 55 45 0a 0a 4e 4f 54 45 3a 20 44 49 53 41 42	NUE..NOT E: DISAB
0140	4c 49 4e 47 20 59 4f 55 52 20 22 4d 45 53 53 45	LING YOU R "MESSE
0150	4e 47 45 52 22 20 41 4e 44 20 22 41 4c 45 52 54	NGER" AN D "ALERT
0160	45 52 22 20 57 49 4e 44 4f 57 53 20 53 45 52 56	ER" WIND OWS SERV
0170	49 43 45 20 57 49 4c 4c 20 42 4c 4f 43 4b 20 54	ICE WILL BLOCK T
0180	48 45 53 45 20 4d 45 53 53 41 47 45 53 0a 00	HESE MES SAGES...

図7-31 147番目のパケットのデータ

ディのコンピュータはこのパケットの後にICMPのポート到達不能を返しています(図7-32)。

No. -	Time	Source	Destination	Protocol	Info
147	1202.816709	221.143.42.254	24.6.125.19	Messen	NetrSendMessage request
148	1202.816878	24.6.125.19	221.143.42.254	ICMP	Destination unreachable (Port unreachable).

図7-32 メッセージャーサービスが無効になっているため、コンピュータがメッセージを受け取ることはない

210番目のパケットから問題が起っています(図7-33)。

No. -	Time	Source	Destination	Protocol	Info
210	1617.530663	24.136.28.59	24.6.125.19	TCP	3092 > 1025 [SYN] Seq=0 Len=0 MSS=1460
211	1617.530814	24.6.125.19	24.136.28.59	TCP	1025 > 3092 [SYN, ACK] Seq=0 Ack=1 win=17520 Len=0 MSS=1460
212	1617.534487	24.136.28.59	24.6.125.19	TCP	3093 > 1025 [SYN] Seq=0 Len=0 MSS=1460
213	1617.534608	24.6.125.19	24.136.28.59	TCP	1025 > 3093 [SYN, ACK] Seq=0 Ack=1 win=17520 Len=0 MSS=1460
214	1617.598282	24.136.28.59	24.6.125.19	TCP	3099 > 3127 [SYN] Seq=0 Len=0 MSS=1460

図7-33 今度は通信を受け入れている

今までと同じように、リモートのコンピュータがマンディのコンピュータの1025番ポートに向かって3ウェイハンドシェイクをしようとします。今度は彼女のコンピュータはそれを受け入れてしまいます。彼女のコンピュータ上で、1025番ポートで提供されるサービスが動いているのです。これはおかしい！

7.8.3.3 問題の詳細

ここからしばらくは同じやりとりの繰り返しです。マンディのコンピュータのさまざまなポートに対して通信を試み、あるものは成功しあるものは失敗します。とりあえず357番目のパケットまでは特に見るべきものはありません(図7-34)。

No. -	Time	Source	Destination	Protocol	Info
357	9901.421104	24.191.223.102	24.6.125.19	DCERPC	bind: call_id: 127 IsystemActivator v0.0
358	9901.421594	24.6.125.19	24.191.223.102	DCERPC	bind_ack: call_id: 127 provider rejection, reason: Abstract syntax not supported
359	9901.522097	24.191.223.102	24.6.125.19	TCP	[TCP segment of a reassembled PDU]
360	9901.535034	24.191.223.102	24.6.125.19	Isystema	RemoteCreateInstance request[Long frame (2780 bytes)]
361	9901.535166	24.6.125.19	24.191.223.102	TCP	1025 > 4ipop [ACK] Seq=0 Ack=2977 win=17520 Len=0
362	9901.535260	24.6.125.19	24.191.223.102	DCERPC	Fault: call_id: 229 ctx_id: 0 status: nca_unk_if

図7-34 357番目のパケットは外部ネットワークからのDCERPCパケット

357番目のパケットはDCERPCとかRPC (Remote Procedure Call) パケットとか呼ばれるものです。RPCはリモートからコンピュータ上でプログラムを実行するため

のプロトコルです。このパケットは、外部ネットワークからプログラムを実行しようとしています。これが怪しいことくらいは誰もが分かります。

それではここから、マンディのコンピュータとリモートのコンピュータとの通信を詳しく見ていきましょう。381 番目のパケットを見てください。updates.virtumonde.com というドメインの名前解決をするための DNS パケットです (図 7-35)。

No. *	Time	Source	Destination	Protocol	Info
381	11477.76286	24.6.125.19	216.148.227.68	DNS	Standard query A updates.virtumonde.com
382	11477.83280	216.148.227.68	24.6.125.19	DNS	Standard query response A 208.48.15.13 A 208.48.15.11

図 7-35 ここで、マンディのコンピュータは DNS を使ってリモートのコンピュータの名前解決を試みる

Web サイトについて何か見慣れないものが表示されたら、インターネットで検索してみましょう。virtumonde について検索すると、スパイウェアやサーバのホスティングについての情報がいろいろ出てくるでしょう。

マンディのコンピュータと virtumonde のサーバとの通信をより詳しく見てみます。メインメニューの [Statistics] から [Conversations] を選択して [Conversations] ダイアログを開き、TCP タブを表示させます。マンディのコンピュータ 246.125.19 と virtumonde のサーバ 208.48.15.13 のみを表示するようフィルタリング[†]してください (図 7-36)。これで表示されるパケットがかなり少なくなり、見やすくなるはずです。

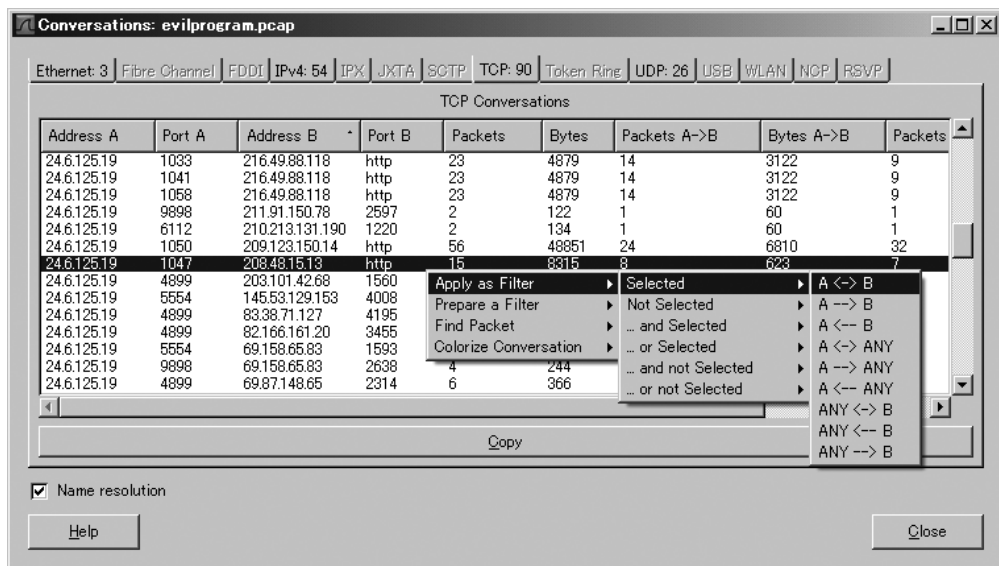


図 7-36 [Conversations] ダイアログで 2 つのエンドポイントの通信のみを表示する

[†] 訳注：[Conversations] ダイアログの TCP タブページで [Address B] をクリックしてソートすれば、[Address B] が「208.48.15.13」で [Address A] が「24.6.125.19」のパケットを探せます。この通信を右クリックし、[Apply as Filter] から [Selected] を選択して [A<->B] をクリックすればフィルタリングできます。[Conversations] ダイアログでのフィルタリングについては、「8.7.3 解析」の解説が参考になります。

順にパケットを見ていくと、386番目のパケットで彼女のコンピュータがvirtumondeからbkinst.exeというファイルをダウンロードしていることが分かります(図7-37)。

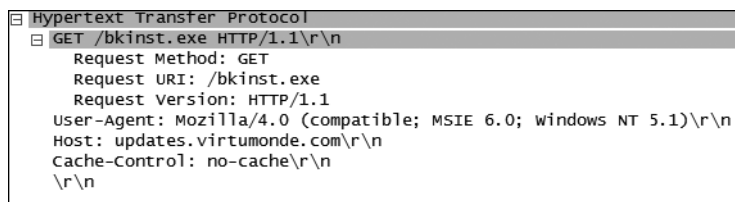


図7-37 マンディのコンピュータがvirtumondeのサーバからファイルをダウンロードしようとしている

このファイルについてインターネットで調べてみると、スパイウェア、ブラウザハイジャックなど、悪行の限りを尽くすファイルであることが分かるでしょう。マンディのコンピュータに悪さをしているものがこれで判明しました。

7.8.4 まとめ

このシナリオでは、マンディのコンピュータの問題はバックグラウンドで動作しているRPCを利用してスパイウェアをダウンロードしていることが原因でした。しかしほかにも学んだことがあります。

この解析は、ネットワーク上で何が起きているかをより理解するために行いました。マンディのコンピュータがこのスパイウェアに感染したということは、ほかの誰かでも起こりうるということです。どのポートとサービスが使われているかがもう分かったので、ファイアウォールでこのスパイウェアをブロックすることができます。スパイウェアは簡単に削除できるとはいえ、ちょっとした設定をしておけばスパイウェアを削除する手間が大幅に省けます。

7.9 考察

この章で取り上げたシナリオはとても単純ですが、Wiresharkによるパケット解析やネットワークの一般的なトラブルシューティングに慣れるための大事な練習になります。本書の残りはこの章のような形式で書かれていますが、この章とはまた違った分野での、実際のパケット解析について述べています。

8章

ケーススタディ (ネットワークの遅延と戦う)

ネットワーク管理者は、日々ネットワークの遅延と戦わなくてはなりません。ネットワークを利用している人たちからの苦情でもっとも多いのが、ネットワークが遅いということです。しかしながら、ネットワークの遅延は恥ずべき問題というわけではありません。

ネットワークの遅延に取り組む前に、本当にネットワークが遅いのかということを確認する必要があります。この章では、ユーザーが「ネットワークが遅い」と文句を言ってくるさまざまなシナリオをご紹介します。

8.1 ダウンロードの遅延の原因

slowdownload.pcap

ダウンロードが遅延する原因について、パケットレベルで見してみましょう。

サンプルファイルをスクロールしていくと、たくさんのHTTPやTCPのトラフィックと、ファイルがダウンロードされている様子が表示されているのが分かります(図8-1)。6章で学んだとおり、HTTPの通信では、HTTPを使ってWebサーバにデータを要求し、TCPを使って要求したデータをダウンロードします。

No. -	Time	Source	Destination	Protocol	Info
12	5.516729	216.251.114.10	10.0.52.164	TCP	[TCP segment of a reassembled PDU]
13	5.517004	10.0.52.164	216.251.114.10	TCP	2468 > http [ACK] Seq=1605 Ack=1893 win=258060 Len=0
14	5.517708	216.251.114.10	10.0.52.164	TCP	[TCP segment of a reassembled PDU]
15	5.517965	10.0.52.164	216.251.114.10	TCP	2468 > http [ACK] Seq=1605 Ack=3273 win=258060 Len=0
16	5.518722	216.251.114.10	10.0.52.164	TCP	[TCP segment of a reassembled PDU]
17	5.518771	10.0.52.164	216.251.114.10	TCP	2468 > http [ACK] Seq=1605 Ack=4653 win=258060 Len=0
18	5.709920	10.0.52.164	216.251.114.10	TCP	2470 > http [SYN] Seq=0 Len=0 MSS=1460 WS=2
19	5.723110	216.251.114.10	10.0.52.164	TCP	[TCP segment of a reassembled PDU]
20	5.723346	10.0.52.164	216.251.114.10	TCP	2468 > http [ACK] Seq=1605 Ack=6033 win=258060 Len=0
21	5.724099	216.251.114.10	10.0.52.164	TCP	[TCP segment of a reassembled PDU]

図8-1 HTTPとTCPのトラフィックをフィルタする

ダウンロードを遅くしている異常なトラフィックを見分けるために、[Expert Infos] ウィンドウを使います。メインメニューの [Analyze] から [Expert Infos] を選択してください(図8-2)。

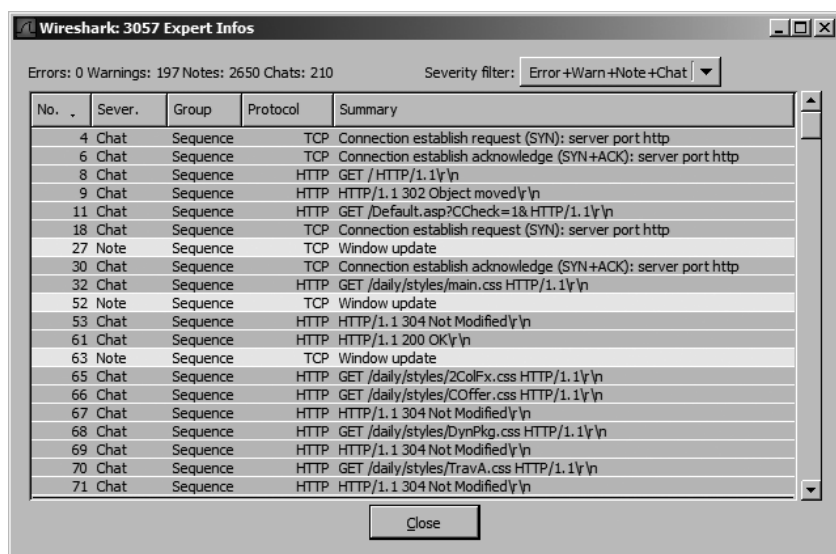


図8-2 [Expert Infos] ウィンドウには、Error、Warning、Note、Chatの4種類の通信が表示される

デフォルトでは、[Expert Infos] ウィンドウには、Error、Warning、Note、Chatの4種類の通信が表示されます。今回はChatは関係ないので、[Severity filter]の横にあるドロップダウンメニューから [Error+Warn+Note] を選択し、デフォルトの設定を変更しましょう。この変更によって図8-3のようになります。

TCPのウィンドウサイズを変更するパケットが大量にあることに注目してください。データの通信速度は、TCPのウィンドウサイズに依存しています。クライアントがデータを転送しているとき、データ受信速度が速くなったり遅くなったりするのに応じて、[Window update] のパケットが送信されています[†]。これらのパケットは転送するデータのサイズを増やす、または減らす必要があることをクライアントに知らせています。誰かがあなたのために冷水器のボタンを押してくれているところを想像してください。ボタンが強く押されると、水はあなたの口からこぼれてしまいますので、ボタンを弱く押すように言う必要があります。逆にその人がボタンを弱く押していれば、水を十分に飲むことができません。

次に、問題が起こる最初のパケットを見ることができます。ダウンロードが始まると同時に、[TCP Previous segment lost] というパケットが表示されはじめます(図8-4)。

[†] 監訳注：ここでのWindow updateは、Windows OS (2000、XP、2003 Server、Vistaなど)のパッチ当てを行う仕掛けのことではなく、ウィンドウサイズの更新を知らせるパケットのことです。TCPのウィンドウサイズというのはデータを受け渡す際の器のサイズで、これを適宜変更しながらデータのやりとりを行います。

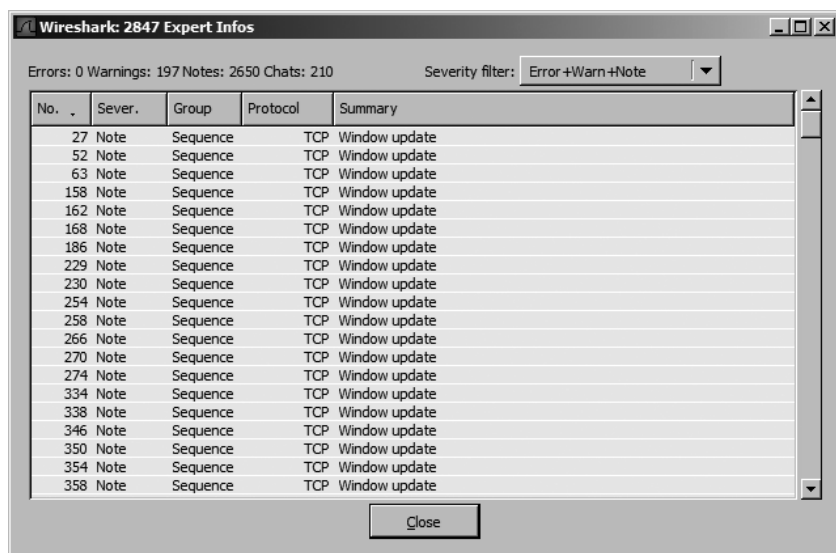


図8-3 [Expert Infos] ウィンドウ (Chatを除く) には、ダウンロードの遅延に関するサマリが表示されている

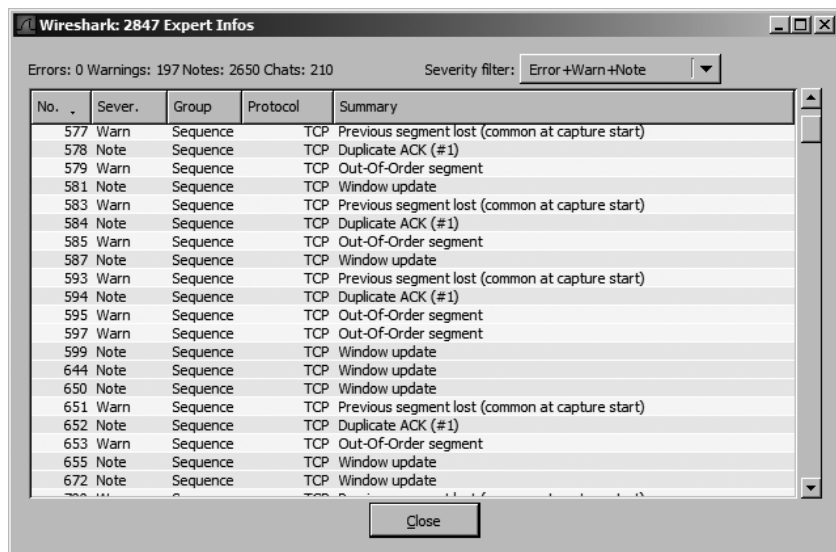


図8-4 [TCP Previous segment lost] が問題のパケット

これらのパケットは、データを転送している間にパケットが突然破棄されたということを意味します。クライアントはこのパケットに対しDuplicate ACKパケットをサーバに送り、受け取れなかったパケットを再度送信するように求めます。クライアントは要求したデータを受信するまでDuplicate ACKパケットを送信し続けます。破

棄されたパケットの再送信は、[Expert Infos] ウィンドウの [TCP Retransmission] として表示されています (図8-5)。

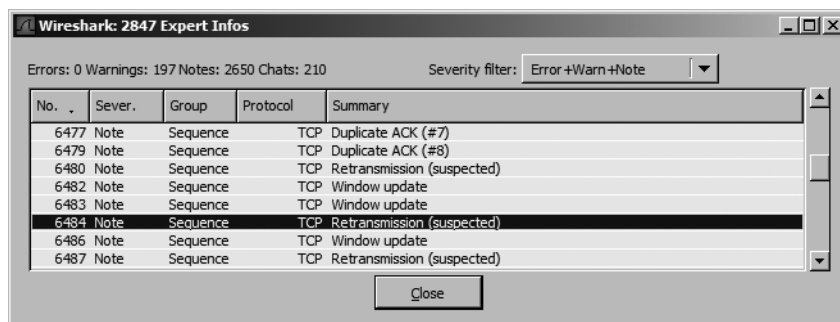


図8-5 パケットが破棄されると速やかに再送される

ダウンロードが始まるときには、Duplicate ACKパケットは1つか2つしかありませんが、ダウンロードが進むにつれ増えていきます。サンプルファイルの残りを見ていくと、[TCP Previous Segment Lost] と Duplicate ACK でいっぱいになっていることが分かります。これがダウンロードの遅延のサインです。

Wiresharkには、TCPストリームをグラフ化する便利な機能があります (図8-6)。解析したいストリームのパケットをクリックし (図では1023番目のパケットをクリックしています)、メインメニューの [Statistics] から [TCP Stream Graph] を選択し、さらに [Round Trip Time Graph] を選択します。TCPストリームをグラフ化する機能を使えば、データのスループットを簡単に確認することができます。

このグラフは少しみにくいかもしれませんが、通信のRTT (Round Trip Time : 往復遅延時間)を確認する便利な機能です。たとえばこの通信の最初のほうではRTTが1秒以上の値を示しています。ファイルをダウンロードするときには、0.1秒以下のRTTでなければいけません。理想的なのは0.04秒 (40ミリ秒) です。このグラフを見れば何か問題が発生していることがすぐに分かるでしょう。

8.2 ルーティングの不具合

icmp-tracert-slow.pcap

ネットワークの遅延を解決するために最初にすべきことは、問題の原因を探ることです。次のシナリオでは、ヘルプデスクがオーウェンからの電話を受け、インターネットの接続が非常に遅いと文句を言われることから始まります。

8.2.1 分かっていること

単純な苦情のほかに分かっていることはほとんどありません。どのWebサイトを見ようとしても、インターネットの接続が遅いということが分かっています。さらなる調査によって、オーウェンと同じネットワークに接続しているすべてのコンピュー

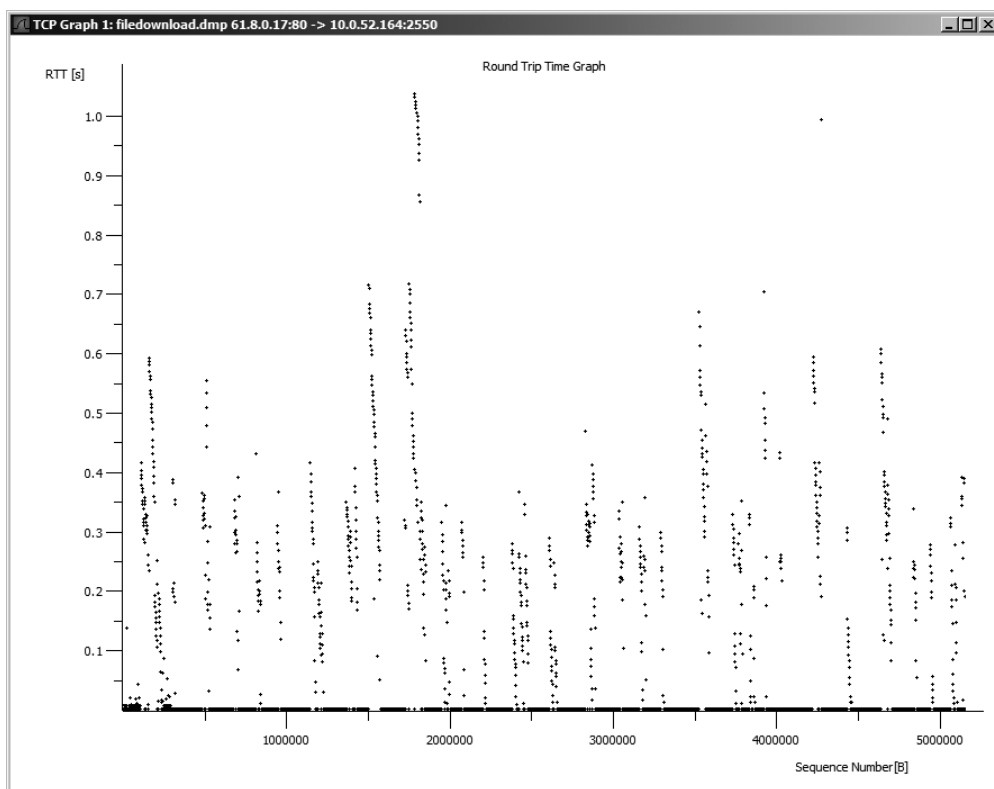


図 8-6 この通信の RTT (Round Trip Time : 往復遅延時間)

タが同じ問題を抱えているということが分かりました。

8.2.2 パケット解析開始

最初に文句を言ってきたのはオーウェンなので、彼のコンピュータで解析を試みます。彼のコンピュータに Wireshark をインストールし、パケットをキャプチャします。

複数のコンピュータで同じ問題が出ているので、オーウェンのコンピュータには問題がないことは分かっています。そのため、オーウェンのコンピュータがインターネットにアクセスするところをキャプチャする意味はありません。かわりに、ICMP の traceroute を使って問題を解析していきましょう。

traceroute は ICMP ベース (Unix では UDP ベース) の診断用ユーティリティで、ある宛先に達するまでのすべてのルータにパケットを送信します。traceroute を使えば、遅延に関するなんらかの情報が得られるでしょうが (図 8-7)、ボトルネックになっているところをより詳しく知るために、Wireshark でパケットをキャプチャしつつ、traceroute を使ってみます。

図 8-7 は traceroute (Windows OS では tracert) を実行した結果です。各行には、

```

C:\>tracert www.webafrica.co.za

Tracing route to www.webafrica.co.za [196.31.65.20]
over a maximum of 30 hops:

  0  <1 ms    <1 ms    <1 ms    fw1.d.barn.za.net [172.16.0.254]
  1  <1 ms    <1 ms    <1 ms    i.tsb.barn.za.net [196.7.14.112]
  2  3 ms     2 ms     2 ms     router.barn.za.net [196.7.14.110]
  3  6 ms     8 ms     8 ms     router.cabinet.barn.za.net [196.31.167.115]
  4  6 ms     8 ms     8 ms     i.cabinet.barn.za.net [196.30.11.73]
  5  7 ms     8 ms     8 ms     vlan512.hr3.cpt1.alter.net [196.31.93.49]
  6  8 ms     7 ms     8 ms     vlan10.hr1.cpt1.alter.net [196.30.176.36]
  7  8 ms     7 ms     8 ms     cpt.h-gw.net [196.7.5.134]
  8  10 ms    6 ms     7 ms     ns10.pcnets.co.za [196.31.65.20]

Trace complete.

C:\>_

```

図8-7 tracerouteの一般的な結果

宛先までの経路に存在するルータに到達するまでにかかる時間が表示されています。

8.2.3 解析

サンプルファイル (icmp-tracert-slow.pcap) を見ると、最初にオーウェンのコンピュータがEcho リクエストを送信していることが分かります (図8-8)。

これらのパケットには、通常の ping との重要な違いがあります。パケット詳細の

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request
2	3.364382	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request
3	6.368126	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request
4	9.371704	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request

図8-8 Echo リクエストがオーウェンのコンピュータから送信されている

ペインでIPの部分を見てください。TTL (Time To Live : パケットの生存期間) が設定されています (図8-9)。

TTL とは、宛先にたどり着くまでにルータを通ることができる回数を決める値で

```

Internet Protocol, Src: 24.6.126.218 (24.6.126.218), Dst: 198.173.244.32 (198.173.244.32)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 92
  Identification: 0xb5f6 (46582)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 1
  Protocol: ICMP (0x01)
  Header checksum: 0xb1fc [correct]
  Source: 24.6.126.218 (24.6.126.218)
  Destination: 198.173.244.32 (198.173.244.32)

```

図8-9 このパケットにはTTLが設定されている

す。もしこの値が1だと、パケットが宛先に着くまでの間にルータを1台通ると TTL が切れてしまい、ICMP の生存時間超過 (ICMP Time Exceeded) パケットが返され

ます。tracertは生存時間超過パケットを受け取ると、今度はTTLの値を2にして送信します。すると宛先までの間の2番目のルータでTTLが切れます。パケットが宛先にたどり着くまでこの処理を繰り返します (図8-10)。

TTLの知識を前提にサンプルファイルを見ると、tracertの最初のパケットが

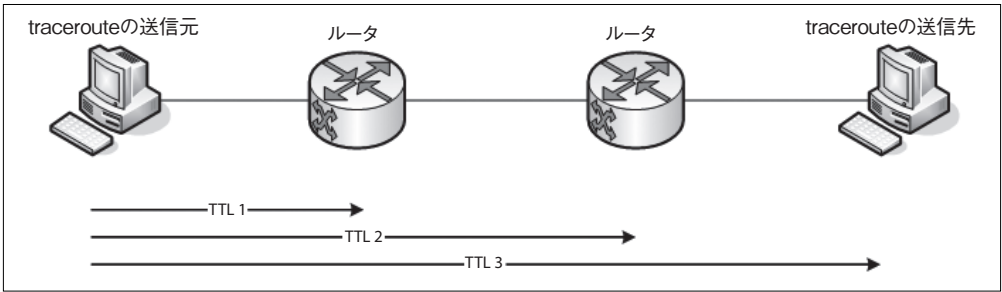


図8-10 宛先までの経路にあるルータが増えると、TTLの値も増える

おかしいということがすぐに分かります。このパケットのTTLの値は1なので、すぐに生存時間超過パケットを受け取るはずなのに、受け取っていません。

オーウェンのコンピュータはこの返答を受け取っておらず、3秒ほどたってからリクエストを再送しています (図8-11のTimeの部分)。

オーウェンのコンピュータは2度目のリクエストの返答も受け取っていません。3

No. ↓	Time	Source	Destination	Protocol	Info
2	3.364382	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request
3	6.368126	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request

図8-11 オーウェンのコンピュータは最初のリクエストの返答は受け取っておらず、3秒後にもう一度リクエストを送っている

秒ほど待ってから3度目のリクエストを送っていますがこれも失敗しています (図8-12)。

ここで、tracertは最初のルータから返答をもらうことをあきらめ、TTLの値を2にしてパケットを送信しています (4番目のパケット)。このパケットは2番目の

No. ↓	Time	Source	Destination	Protocol	Info
3	6.368126	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request
4	9.371704	24.6.126.218	198.173.244.32	ICMP	Echo (ping) request

図8-12 返答が帰ってこないで、もう一度リクエストを送っている

ルータに無事到達し、オーウェンのコンピュータはICMPのタイプ11、コード0の生存時間超過パケットを受け取っています (図8-13)。

No. -	Time	Source	Destination	Protocol	Info
59.393904	12.244.25.161	24.6.126.218	ICMP	Time-to-live exceeded (Time to live exceeded in transit)	

図8-13 生存時間超過パケットを受け取っている

あとは、パケットが宛先に届くまでTTLの値を増やしながらパケットを送信し、生存時間超過パケットを受け取るという処理を繰り返しています。

tracertの解析で分かることはなんでしょうか？ 生存時間超過パケットを受け取ることができなかった内部のルータに問題があるということです。ルータは非常に複雑なネットワーク機器なので、ここではルータの何が問題だったかということは調査しません。重要なのは、この解析によってどこに問題があったのかを知ることができたということです[†]。

8.2.4 まとめ

Wiresharkを使うことで問題の原因をすばやく探ることができ、トラブルシューティングの時間を大幅に減らすことができます。Wiresharkはルータの何が悪いのかまでは教えてくれませんが、とにかくルータの設定を変えなければならないということは分かります。

また、今回のシナリオでICMPの知識がさらに増え、tracertの機能を知ることでもできました。tracertにはほかにもさまざまなオプションが使えます。インターネットで検索してみてください。

8.3 二重に見える

double-vision.pcap

あなたはジェフのために新しいコンピュータを設定しました。通常、コンピュータを新しく設定したときには、ほかのコンピュータより早く通信ができるはずです。しかしながら、しばらくするとジェフは、ネットワーク使用量のピーク時にネットワークが非常に遅くなり、あるネットワークサービスが使えなくなってしまうと言っていました。

8.3.1 分かっていること

ジェフのコンピュータは新しいものであり、最低限の機能しか動いていないということは分かっています。また、ネットワークの使用量のピーク時もオフピーク時も、ほかの人からはネットワークが遅いという報告は受けていません。ジェフの仕事のほとんどはネットワークを使わないとできないため、かなりたくさんネットワークを使っています。また、ネットワークを使うアプリケーションを複数起動することも多いです。これらのアプリケーションはブラウザやメールクライアントとともに多くの

[†] 監訳注：パケットを取得し、解析することですべての事実を明らかにできるわけではありません。パケット解析はあくまで調査の一部分で、それ以上のものではありませんし、どこまでできるのか、どこからができないのかを見極めて効率よく調査、解析を行うとよいでしょう。

負荷がネットワークにかかりますが、あなたのネットワークは帯域が広く、処理に時間はかかりません。

8.3.2 パケット解析開始

この問題はジェフのコンピュータでのみ起こっているため、Wiresharkをジェフのコンピュータにインストールしましょう。ネットワークの使用がピークに達するときにパケットをキャプチャするのが一番よさそうです。ジェフには日々のルーチンワークをこなしてもらい、その間にキャプチャしたパケットを見ることにします。

8.3.3 解析

サンプルファイルを開けば、このシナリオのタイトルの由来が分かるでしょう。すべてのパケットが2回送信されています(図8-14)。これは普通では考えられません。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	12.234.13.89	12.234.14.63	TCP	47063 > http [ACK] Seq=0 Ack=0 win=4096 Len=0
2	0.025583	12.234.13.89	12.234.14.63	TCP	[TCP Dup ACK 1#1] 47063 > http [ACK] Seq=0 Ack=0 win=4096 Len=0
3	0.025776	12.234.14.63	12.234.13.89	TCP	http > 47063 [RST] Seq=0 Len=0
4	0.066826	12.234.14.63	12.234.13.89	TCP	http > 47063 [RST] Seq=0 Len=0
5	15.001667	12.234.13.89	12.234.14.63	TCP	1092 > 424 [SYN] Seq=0 Len=0 MSS=1460
6	15.086461	12.234.13.89	12.234.14.63	TCP	1092 > 424 [SYN] Seq=0 Len=0 MSS=1460

図8-14 2重にばやけているわけではなく、パケットが繰り返し送信されている



便宜上、このサンプルファイルには8つのパケットしか載せていません。それだけで問題の原因が十分に分かるからです。本当は、ジェフのコンピュータの通信はすべて2重になっています。

一般的に、パケットが2重に送信される理由は2つあります。ルーティングの間違いとポートミラーリングの設定ミスです。核心に迫る前に、2つのパケットが本当に同じかどうかを確認してみましょう。

IPヘッダのIDが同じであれば、2つのパケットは同じものであるといえます。パケット詳細のペインで、IPの部分を広げてIdentificationを見てください。1番目と2番目のIDが0xc509になっています(図8-15)。

□ Internet Protocol, Src: 12.234.13.89 (12.234.13.89), Dst: 12.234.14.63 (12.234.14.63) Version: 4 Header length: 20 bytes □ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00) Total Length: 40 Identification: 0xc509 (50441) ← □ Flags: 0x00 Fragment offset: 0 Time to live: 47 Protocol: TCP (0x06) □ Header checksum: 0x915b [correct] Source: 12.234.13.89 (12.234.13.89) Destination: 12.234.14.63 (12.234.14.63)
--

図8-15 最初の2つのパケットは、同じIDを持っている

同じことが3番目と4番目のパケットについてもいえます。この2つのIDは0xaca7です（図8-16）。

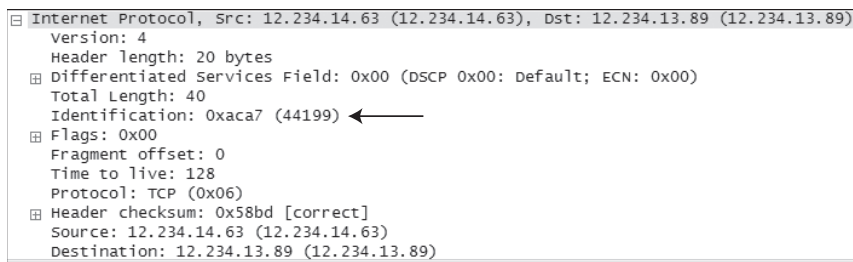


図8-16 3番目と4番目のパケットも同じIDを持っている

パケットが本当に2重になっているということが分かったので、今度はその原因がルーティングの間違いなのかポートミラーリングの設定ミスなのかを調べてみましょう。パケットのTTLの値を見てください。値が異なれば、ルーティングが原因です。同じならば、ポートミラーリングが原因です。

図8-17のとおり、1番目のパケットのTTLは47、2番目のパケットのTTLは46です。ということは、これはルーティングの問題です。2番目のパケットはどこかのルータを通過してから戻ってきているのでTTLが1減っているのです。

<p>パケット1:</p> <p>Fragment offset: 0</p> <p>Time to live: 47</p> <p>Protocol: TCP (0x06)</p>
<p>パケット2:</p> <p>Fragment offset: 0</p> <p>Time to live: 46</p> <p>Protocol: TCP (0x06)</p>

図8-17 TTLの値が異なるのでルーティングが原因

この問題はジェフのコンピュータでのみ起こるので、ネットワーク上のルータでなく彼のコンピュータを調査しなければいけません。さらに調査を進めると、彼のコンピュータはサブネットマスクの設定にミスがあったことが分かりました。

8.3.4 まとめ

サブネットマスクが間違っていて設定されていると、さまざまな問題が発生します。コンピュータがまったく通信できなくなってしまうこともあります。今回の場合では、ジェフのコンピュータから送信されたパケットが返ってきていたので、コンピュータが処理しなければならないトラフィックの量が倍になり、ネットワーク使用量のピーク時に通信が非常に遅くなっていたのです。

8.4 サーバが私を拒否してる？

[http-client-refuse.pcap](#)

別のユーザーがインターネット接続が遅いと文句を言ってきました。エリックは Novell の Web サイトにアクセスすることができず、必要なソフトウェアをダウンロードすることができないと言っています。そのサイトを見ようとする、ブラウザはページをロードしようとしませんが何も起きません。これはネットワークの問題に違いないですよね？

8.4.1 分かっていること

ネットワークを徹底的に調査すると、エリック以外のコンピュータでは問題がないことが分かりました。したがって、この問題はエリックのコンピュータ特有のもので、彼のコンピュータでは Windows が動いており、最新のサービスパックやパッチが適用されています。さらに調査をすると、問題は Novell の Web サイトのある場所を閲覧するときのみに起こっていることが分かりました。

8.4.2 パケット解析開始

問題はエリックのコンピュータでのみ起こっている、Wireshark を彼のコンピュータにインストールしてパケットをキャプチャします。問題が起こる Novell の Web サイト上のある場所を訪問し、何が起きているのか見てみましょう。

8.4.3 解析

`http-client-refuse.pcap` を開くと、HTTP の通信が表示されていることが分かります (図 8-18)。3 ウェイハンドシェイクが正しく行われています。実際に、HTTP リクエストは 28 番目と 29 番目のパケット以外は正常です。何が問題なのか順に見ていきましょう。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	67.161.32.69	130.57.5.25	TCP	1782 > http [SYN] Seq=0 Len=0 MSS=1460 WS=2
2	0.027029	130.57.5.25	67.161.32.69	TCP	http > 1782 [SYN, ACK] Seq=0 Ack=1 Win=6144 Len=0 MSS=1460
3	0.027068	67.161.32.69	130.57.5.25	TCP	1782 > http [ACK] Seq=1 Ack=1 Win=258060 Len=0
4	0.028241	67.161.32.69	130.57.5.25	HTTP	GET /img/flash/load_stream.html?temp=1&id=webex_conve
5	0.061432	130.57.5.25	67.161.32.69	TCP	http > 1782 [ACK] Seq=1 Ack=926 Win=5219 Len=0
6	0.072229	130.57.5.25	67.161.32.69	TCP	[TCP segment of a reassembled PDU]
7	0.073391	130.57.5.25	67.161.32.69	TCP	[TCP segment of a reassembled PDU]
8	0.073430	67.161.32.69	130.57.5.25	TCP	1782 > http [ACK] Seq=926 Ack=2761 Win=258060 Len=0
9	0.074556	130.57.5.25	67.161.32.69	TCP	[TCP segment of a reassembled PDU]
10	0.074752	130.57.5.25	67.161.32.69	TCP	[TCP segment of a reassembled PDU]
11	0.074770	67.161.32.69	130.57.5.25	TCP	1782 > http [ACK] Seq=926 Ack=4301 Win=258060 Len=0

図 8-18 HTTP の通常の通信がキャプチャされている

時間 (Time) の部分に注目してください。28 番目以降のパケットを受信するのにかかる時間がかかっています。27 番目と 28 番目のパケットの間には 9 秒も間があります。

ネットワークの世界では、次のパケットを受け取るのに 9 秒も間が開いてしまうということは、ユーザーが入力したフォームを送信しているようなとき以外には考えら

8.5 BitTorrentの大雨

torrential-slowness.pcap

あなたのネットワークのユーザーがヘルプデスクを呼び出し、ネットワークが非常に遅くなっていると苦情を言ってきました。彼のコンピュータはインターネットにアクセスしにくくなり、ネットワークを使うアプリケーションも動作が遅くなってしまったため、仕事に支障が出てきています。何が原因なのでしょう。

8.5.1 分かっていること

ほかのユーザーについて調査した結果、その問題が広範囲にわたっていることが分かりました。すべてのユーザーが、インターネットへのアクセスが遅すぎてほとんど使えないと報告してきました。また、ネットワークの境界ルータは高い負荷がかかっており、大量のトラフィックを処理していることが分かりました。



境界ルータとは、内部ネットワークと外部ネットワーク（インターネット）をつなぐルータのことです。

8.5.2 パケット解析開始

境界ルータは内部ネットワークとインターネットとでやりとりされるトラフィックを処理しており、非常に高い負荷がかかっていることから、このルータが解析のポイントとなります。

8.5.3 解析

ルータにWiresharkをインストールすることはできないので、ポートミラーリングを使います。

torrential-slowness.pcapのパケットは、ネットワーク上で大量のトラフィックが発生していることを示しているにすぎません（図8-21）。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	203.211.67.195	192.168.0.193	TCP	11766 > 1534 [PSH, ACK] Seq=0 Ack=0 win=65535 Len=1418
2	0.000033	192.168.0.193	203.211.67.195	TCP	1534 > 11766 [ACK] Seq=7090 Ack=4294965878 Win=65535 Len=0
3	0.000053	124.197.17.30	192.168.0.193	TCP	26507 > 4512 [ACK] Seq=0 Ack=0 win=64227 Len=0
4	0.000070	192.168.0.193	124.197.17.30	TCP	4512 > 26507 [PSH, ACK] Seq=8117 Ack=0 win=65108 Len=1380
5	0.000121	213.17.90.77	192.168.0.193	TCP	14173 > 3331 [ACK] Seq=0 Ack=0 win=65535 Len=0
6	0.000136	192.168.0.193	213.17.90.77	TCP	3331 > 14173 [PSH, ACK] Seq=216 Ack=0 win=65492 Len=1452
7	0.000147	192.168.0.193	213.17.90.77	TCP	3331 > 14173 [PSH, ACK] Seq=1668 Ack=0 win=65492 Len=1452
8	0.009806	189.142.91.6	192.168.0.193	TCP	3326 > 6881 [ACK] Seq=0 Ack=0 win=64240 Len=0
9	0.009844	192.168.0.193	189.142.91.6	TCP	6881 > 3326 [PSH, ACK] Seq=8280 Ack=0 win=65006 Len=1380
10	0.025255	142.68.42.31	192.168.0.193	TCP	6881 > 4853 [ACK] Seq=0 Ack=0 win=17280 Len=0

図8-21 大量のトラフィックが記録されている

† 監訳注：Internet Explorer 6以上で、かつWindows XPのSP2以上という環境であれば、Internet Explorerの機能によってすべてのマシンでポップアップがブロックされる可能性があります。しかし、古いOSがまだに使われている環境などでは、Internet Explorerの機能ではなく各種セキュリティソフトウェアによってポップアップがブロックされている場合もあります。そして、そのセキュリティソフトウェアがエリックのマシンにのみ入っていたという前提であれば、このような状況もありえるといえるでしょう。

ネットワーク上のあるコンピュータ（192.168.0.193）がサンプルファイルに繰り返し現れて外部ネットワークからの通信を多く受け入れています。より不吉なことに、PSH フラグが立っています。このフラグが立っていると、受信バッファをスキップして上位アプリケーションにデータが届けられることになります。つまり、処理が優先的に行われるということです[†]。これは悪い兆しです。

さらに悪いことに、これらの通信のほとんどは3ウェイハンドシェイクをすでに終えて、どんどんデータをクライアントに送受信しています。[Conversations] ダイアログを見れば、どれほどの通信が行われているかが分かります（図 8-22）。

Conversations: torrential-slowness.pcap

Ethernet: 3 | Fibre Channel | FDDI | **IPv4: 26** | IPX | JXTA | SCTP | TCP: 21 | Token Ring | UDP: 3 | WLAN | NCP | RSVP

IPv4 Conversations

Address A	Address B	Packets	Bytes	Packets A->B	Bytes A->B	Packets A<-B	Bytes A<-B
24.113.183.206	192.168.0.193	1	71	0	0	1	71
68.208.248.1	192.168.0.193	1	70	1	70	0	0
87.64.232.92	192.168.0.193	1	42	0	0	1	42
192.168.0.193	205.152.132.23	1	148	0	0	1	148
192.168.0.193	201.37.178.124	1	1506	1	1506	0	0
192.168.0.193	216.144.228.100	1	92	1	92	0	0
192.168.0.193	201.42.186.196	1	1434	0	0	1	1434
192.168.0.193	203.211.67.195	2	1538	1	66	1	1472
124.197.17.30	192.168.0.193	2	1494	1	60	1	1434
192.168.0.193	201.250.26.129	2	1494	1	1434	1	60
151.46.136.178	192.168.0.193	2	1494	1	60	1	1434
192.168.0.193	224.0.0.251	2	482	2	482	0	0
192.168.0.193	217.201.172.34	2	1494	1	1434	1	60
82.2.118.24	192.168.0.193	2	114	1	60	1	54

Copy

☒ Name resolution

Close

図 8-22 [Conversations] ダイアログから、どれほど多くの通信が行われているかが分かる

1 秒間に 26 個もの通信が行われています！

この問題を解決するもっとも簡単な方法は、192.168.0.193 のコンピュータを調査することですが、ここではパケット解析でも詳しく調べることができることを示す意味で、あえてパケット解析によってこの問題の原因を探ってみましょう。

まず最初にやることは、これらのパケットの送信先がどこなのか、WHOIS などを使って調べることでしょう。しかしながら、今回の場合、送信先の IP アドレスは 1 つではなく世界中に散らばっています。

パケットをさらに詳しく調査するために、TCP ストリームに何か有益な情報がないかどうかを確認しておきましょう。今回の場合、TCP ストリームには意味の分からない文字列が羅列されているだけで、そこから何か情報を得ることはできません（図 8-23）。

TCP ストリームから何も読み取れないとなると、今度は何をすべきでしょう？ 問

[†] 監訳注：RFC 上はこのとおりだが、実際に上位アプリケーションにすぐにデータが渡されるかどうかは実装依存で、このフラグ自体意味を持たないことも多い。

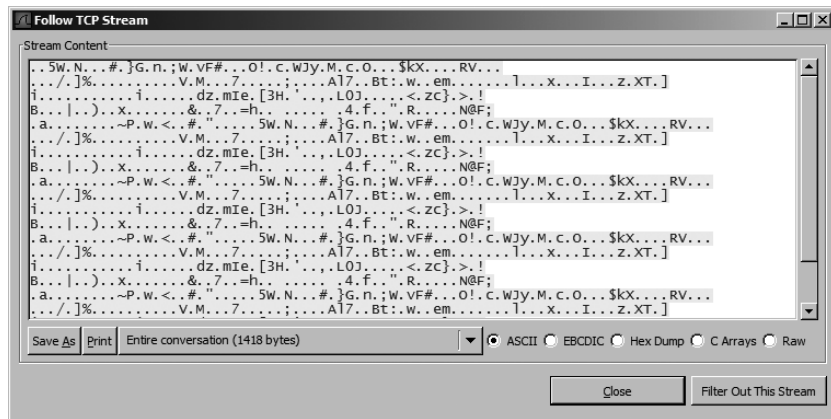


図8-23 TCPストリームを見て何も分からない

題を起こしているコンピュータを調査する以外のもっとも簡単な方法は？ パケットを1つ1つ見ていくことです。何か重要なものを探しましょう。

サンプルファイルのパケットを見ていくと、44番目のパケットがおかしいということに気づくでしょう(図8-24)。このパケットは[Info]欄にあるとおり、リモートのBitTorrentサーバにお伺いを立てています。このBitTorrent(P2Pのファイル転送サービス)が問題の原因です。

No.	Time	Source	Destination	Protocol	Info.
44	0.313562	192.168.0.193	224.0.0.251	MDNS	Standard query PTR BitTorrent-12b98806c576922f092c37f22e2
99	0.774636	192.168.0.193	224.0.0.251	MDNS	Standard query PTR BitTorrent-e7393164158de2fac0f58a13a0f1

図8-24 パケット1つ1つを見ていくと、BitTorrentが犯人であることが分かる

8.5.4 まとめ

今回のシナリオでは、BitTorrentをインストールしたコンピュータが音楽をダウンロードしており、かつ無制限に通信を許可していたことが問題でした。ネットワークの帯域を独占していたのです。

パケット解析をするときには高度な機能を使う必要があるものの、ときには1つ1つパケットを見ていくほうが早く問題が解決する場合があるということが、今回のシナリオから分かります。

8.6 メールサーバに流れ込むPOP

email-troubles.pcap

従業員の目から見ると、インターネットを使うもっとも重要な目的はメールです。今回のシナリオでは、メールの送受信に問題があるときの対応の仕方を学びます。

あなたのネットワークのユーザーが、メールが到達するまでに非常に長い時間がかかると文句を言っています。ほかのドメインにメールを送るときだけでなく、同じ組織の中でメールを送るときにも大変な時間がかかっています。この真相を探りましょう。

8.6.1 分かっていること

企業のすべてのメールを1台のメールサーバが管理しています。調査してみると、この問題はあなたのネットワーク上にあるすべてのメールクライアントで起こっていることが分かりました。通常は職場内のメールであれば一瞬で届くところが、10分から15分ほどかかっています。外部からメールを受信するときにも同じくらいの時間がかかります。

8.6.2 パケット解析開始

メールを処理しているサーバは1台しかないので、そこにWiresharkをインストールします。この問題は1日中起きているので、キャプチャする時間はいつでもよいです。

8.6.3 解析

email-troubles.pcapを見ると、メールサーバ上でキャプチャできるパケット（つまり、メールのパケット）が表示されているのが分かります。ものすごい量のPOP (Post Office Protocol) パケットがメールサーバに流れ込んでいます（図8-25）。おそらくメールサーバには負荷がかかりすぎているでしょう。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	12.234.13.202	161.58.73.170	POP	Request: RETR 20
2	0.079320	161.58.73.170	12.234.13.202	TCP	pop3 > 1567 [ACK] Seq=0 Ack=9 win=49152 Len=0
3	0.090650	161.58.73.170	12.234.13.202	POP	Response: +OK 100220 octets
4	0.091089	161.58.73.170	12.234.13.202	POP	Continuation
5	0.091117	12.234.13.202	161.58.73.170	TCP	1567 > pop3 [ACK] Seq=9 Ack=2920 win=64512 Len=0
6	0.092467	161.58.73.170	12.234.13.202	POP	Continuation

図8-25 サンプルファイルには大量のPOPパケットが記録されている

どのくらいの量のPOPパケットが受信されているかを調べるために、時間の表示フォーマットを「Seconds Since Beginning of Capture」に変更しましょう。最後のパケットを見てください。サンプルファイルに記録されているパケットは2分の間にキャプチャされたものであることが分かります（図8-26）。

No. -	Time	Source	Destination	Protocol	Info
360	121.664143	12.234.13.202	161.58.73.170	TCP	1567 > pop3 [ACK] Seq=27 Ack=301035 win=63337 Len=0

図8-26 時間の表示フォーマットを変えれば合計時間内にどれだけのパケットを受信したかが分かる

それでは各通信で何かおかしいことが起こっていないかを見てみましょう。

POPパケットのすごいところは、TCPストリームを見れば、メールの中身を見ることができるといことです。たとえば、1番目のパケットのTCPストリームを見れば、メールにはdocument_9446.pifというファイルが添付されていることが分かります（図8-27）。

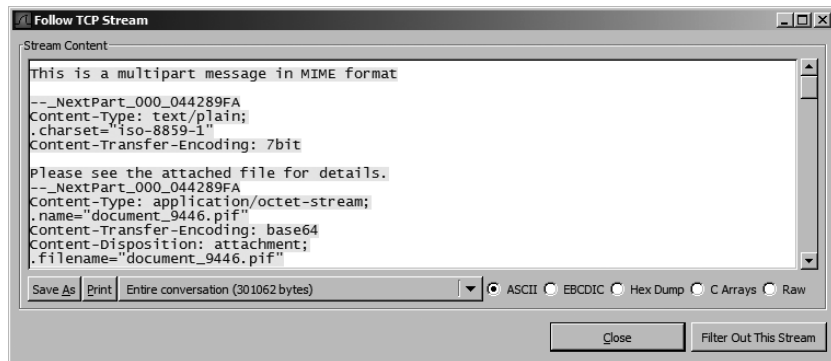


図8-27 1 番目のパケットの詳細

ほかのTCPストリームを見てみると、やはりPIFファイルが添付されています。これは怪しい。

PIFとはProgram Information Fileで[†]、通常はメールでやりとりされるものではありません。それだけでなく、ファイルサイズが非常に大きい傾向にあります。サンプルファイル全体にわたってこのファイルを添付したメールが方々に行きかっています。

これは（おそらくウイルスやワームを含む）スパムメールであり、そのせいでメールサーバに高い負荷がかかってメールの送受信が遅くなっているのです。

8.6.4 まとめ

メールサーバには巨大な添付ファイルを含んだスパムメールのために、高い負荷がかかっていました。メールサーバのパフォーマンスを監視していると、こういうことはよく起こります。組織が大きくなるにつれ、受信するスパムメールの数も増えていきます。ネットワークのユーザーに遅延を我慢してもらうか、スパムフィルタを導入することが今回のシナリオの解決策になるでしょう。

8.7 Gnutella も大雨

gnutella.pcap

今回のシナリオはBitTorrentのシナリオと似たところがあります。ティナはあなたのネットワークのユーザーであり、ネットワークに接続しているときでもしていないときでも、コンピュータが驚くほど遅くなっていると言っています。

8.7.1 分かっていること

BitTorrentのシナリオと同じようなことが分かっています。すなわち、この問題が

[†] 訳注：Windowsにおいて、MS-DOSプログラムを実行するための各種の設定が記述されたファイルです。

ほかのユーザーにも起こっています。ユーザーがインターネットに接続したりネットワークを使うアプリケーションを使うときだけ速度が遅くなります。境界ルータには高い負荷がかかっており、大量のデータが行き来しています。

8.7.2 パケット解析開始

今回の場合、BitTorrentのシナリオと同じようにすべてのコンピュータに問題があります。ただし、彼女のコンピュータだけは、ネットワークを使うアプリケーションが遅いだけでなく、コンピュータそのものが遅くなっています。

彼女のコンピュータはほかと違う症状があるので、問題の原因は彼女のコンピュータにあるという前提で解析をしていきます。しかしながらティナのコンピュータは非常に処理が遅くなっているため、Wiresharkを彼女のコンピュータにインストールするのは得策ではありません。パケットがうまくキャプチャできない可能性があります。そのため、ポートミラーリングを使います。

8.7.3 解析

このサンプルファイル（gnutella.pcap）は長いですが、BitTorrentのときと似ています。図8-28のとおり、ティナのコンピュータのIPアドレスは10.1.4.176で、外部ネットワークのさまざまなコンピュータとの通信を試みています。ほとんどはSYNパケットを送っても返事がないか、RSTパケットを受け取っています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.1.4.176	66.68.99.53	TCP	3663 > 6346 [SYN] Seq=0 Len=0 MSS=1460
2	0.028257	10.1.4.176	198.82.59.65	TCP	3684 > 6346 [SYN] Seq=0 Len=0 MSS=1460
3	0.060831	198.82.59.65	10.1.4.176	TCP	6346 > 3684 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
4	0.499894	10.1.4.176	198.82.59.65	TCP	3684 > 6346 [SYN] Seq=0 Len=0 MSS=1460
5	0.531212	198.82.59.65	10.1.4.176	TCP	6346 > 3684 [RST, ACK] Seq=0 Ack=1 win=0 Len=0
6	0.999962	10.1.4.176	198.82.59.65	TCP	3684 > 6346 [SYN] Seq=0 Len=0 MSS=1460
7	1.030592	198.82.59.65	10.1.4.176	TCP	6346 > 3684 [RST, ACK] Seq=0 Ack=1 win=0 Len=0

図8-28 ほとんどの通信が失敗に終わっている

問題のいくつかはこの通信の失敗が引き起こしていると思われますが、さらに調査を進める前に、問題の範囲を特定するために、調査すべきトラフィックがどのくらいあるかを確認しておきましょう。[Conversations] ダイアログを開いて、TCPとIPの通信がどのくらいあるのか見てください（図8-29）。

[Conversations] ダイアログから、81のIPの通信と243のTCPの通信が行われていることが分かります（図8-29の上部）。これがサーバの通信であればさほど珍しくありませんが、ティナのコンピュータは個人で使用しているものです。短い期間にこれほどの通信を行うのは普通ではありません。

TCPの通信をいくつか見てみれば、それらがすべてリモートホストとの通信であることが分かります。パケットの数が非常に少ないことから、これらの通信のほとんどは失敗していることが予想できます。

これらの通信について本当に必要な情報を得るには、成功している通信を見る必要

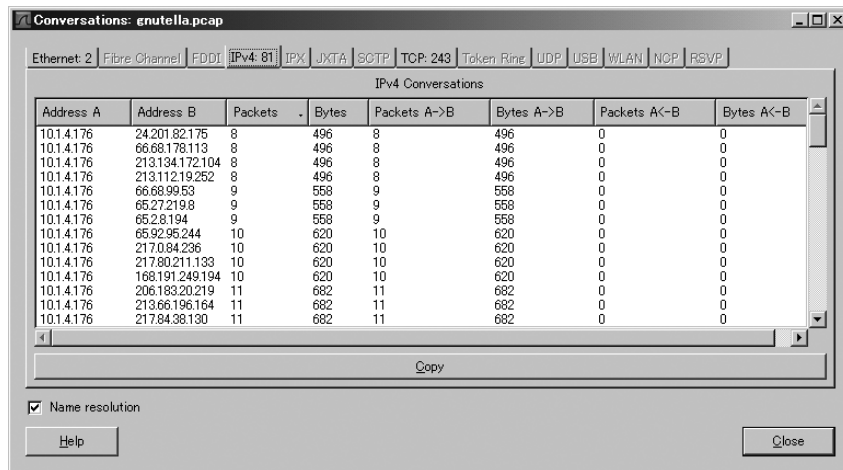


図 8-29 [Conversations] ダイアログを見れば、どのくらい通信が行われているのか分かる

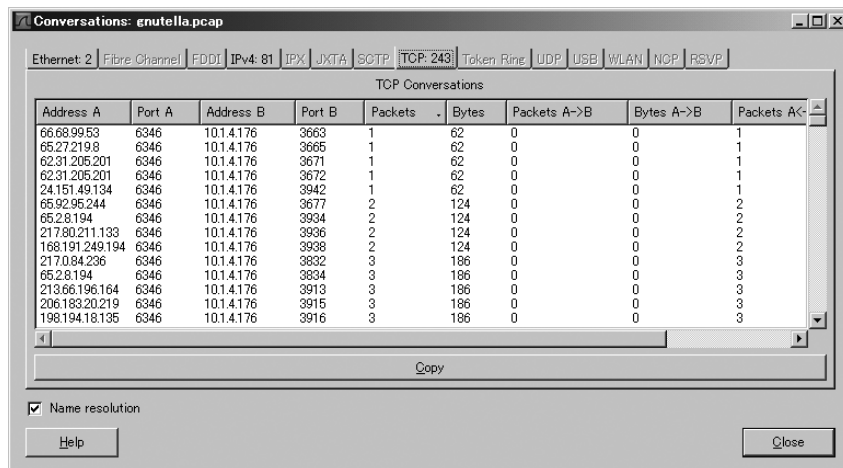


図 8-30 [Conversations] ダイアログの [TCP] タブページ

があります。[Conversations] ダイアログの [TCP] タブを選択してください (図 8-30)。そして、[Packets] をクリックしてください。各通信がパケット数順にソートされます (図 8-31)。

TCP のタブを開き通信量が多い順にソートしてみると、65.34.1.56 をはじめとする多数のコンピュータと通信を行っているのが見えてきます。まずはもっとも通信量が多い 65.34.1.56 との通信を詳細に見てみましょう。

この通信を右クリックし、[Apply as Filter] から [Selected] を選択して [A<->B] をクリックしてください。これで、この通信のパケットだけを表示させることができます (図 8-32)。

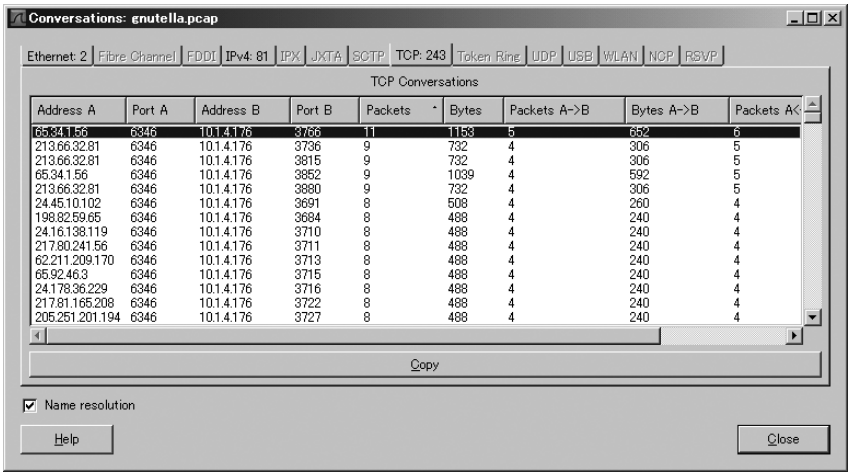


図 8-31 パケット数順にソートしてみる

No. -	Time	Source	Destination	Protocol	Info
426	52.436286	10.1.4.176	65.34.1.56	TCP	3766 > 6346 [SYN] Seq=0 Len=0 MSS=1460
429	52.514080	65.34.1.56	10.1.4.176	TCP	6346 > 3766 [SYN, ACK] Seq=0 Ack=1 win=17520 Len=0 MSS=1460
430	52.514799	10.1.4.176	65.34.1.56	TCP	3766 > 6346 [ACK] Seq=1 Ack=1 win=8760 Len=0
431	52.515422	10.1.4.176	65.34.1.56	Gnutella	
432	52.722234	65.34.1.56	10.1.4.176	TCP	6346 > 3766 [ACK] Seq=1 Ack=31 win=17490 Len=0
433	52.724068	10.1.4.176	65.34.1.56	Gnutella	
434	52.844222	65.34.1.56	10.1.4.176	Gnutella	

図 8-32 関連するパケットのみが表示される

図 8-32 のパケットから、問題の原因を突き止めるための情報が得られます。431 番目、433 番目、434 番目のパケットは Gnutella のパケットです。Gnutella はファイル共有のためのサービスです。クリックして詳細を見てみましょう（図 8-33）。

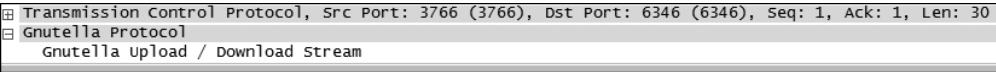


図 8-33 Gnutella パケットに興味深い情報が入っている

431 番目のパケットのパケット詳細のペインには、有益な情報は含まれていません。単に Gnutella ネットワークにファイルをダウンロードまたはアップロードしているという情報のみです。しかしバイナリのペインには、怪しい情報が含まれています（図 8-34）。

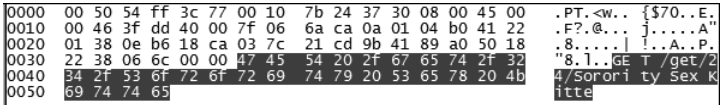


図 8-34 バイナリのペインに何がダウンロードされているか表示されている

このGnutellaパケットには、「sorority sex kitte」という名前を含むファイルをダウンロードするGETコマンドが含まれています。疑わしいトラフィックです。

フィルタリングをしなくても、これがGnutellaトラフィックであることを示すもう1つの証拠があります。通信を見ていくと、すべてが6346番ポートで行われていることに気づくでしょう(図8-35)。

No. -	Time	Source	Destination	Protocol	Info
583	75.544302	10.1.4.176	65.27.229.23	TCP	3803 > 6346 [SYN] Seq=0 Len=0 MSS=1460
584	75.544499	10.1.4.176	24.28.233.147	TCP	3802 > 6346 [SYN] Seq=0 Len=0 MSS=1460
585	75.544750	10.1.4.176	65.2.243.188	TCP	3801 > 6346 [SYN] Seq=0 Len=0 MSS=1460
586	75.545190	10.1.4.176	24.178.197.5	TCP	3804 > 6346 [SYN] Seq=0 Len=0 MSS=1460
587	75.545446	10.1.4.176	24.249.29.119	TCP	3799 > 6346 [SYN] Seq=0 Len=0 MSS=1460

図8-35 ポート番号を見るのも、なんのトラフィックかを探るよい手

<http://www.iana.org>のポート番号表を見れば、どのポートがなんのサービスを提供しているのかが分かります。

8.7.4 まとめ

Gnutellaはさまざまなファイルをダウンロードしたり配布したりするのに使われています。便利なサービスですが、残念ながらポルノ画像や海賊版のソフトウェア、映画、音楽などを共有する場としても利用されています。

このシナリオでは、ティナかティナのコンピュータを利用している誰かがGnutellaクライアントをインストールして、ポルノ画像をダウンロードしていました。

8.8 考察

これらのシナリオを読んで、実際にはネットワークの障害ではない問題がほとんどだということに気づいたでしょうか。ネットワークの遅延に関してはこういうことがよくあります。ネットワークが遅いのではなく、各コンピュータやアプリケーションに関する問題なのです。

9章

ケーススタディ (セキュリティ解析)

この章では、ネットワークセキュリティに関するシナリオを用意し、Wiresharkで解析します。個人情報や企業機密の漏えい、ハッカーなどの脅威をパケットレベルで解析できるようがんばりましょう。

9.1 OSのフィンガープリント

`osfingerprinting.pcap`

OS (Operating System) のフィンガープリントとは、リモートのコンピュータ上で稼動しているOS特有の情報で、ハッカーはフィンガープリントを収集してどんなOSが稼動しているかを知ること、より効果的に攻撃しようとしています。OSのフィンガープリントは、さまざまなコマンドをターゲットコンピュータに送ることで採取できます。リモートのコンピュータが返してくるメッセージを見ると、稼動しているOSを推測できます。こうして、そのOS特有の脆弱性をついた攻撃がしやすくなるのです。

osfingerprinting.pcapを開くと、いくつかのICMPトラフィックが表示されていることが分かります(図9-1)。トラフィックのいくつかはEchoリクエストとEchoリプライで、特に重要ではありません。しかしながら、[Timestamp request]、[Timestamp reply]、[Address mask request]、[Information request]は通常使われることはありません。

No. ↓	Time	Source	Destination	Protocol	Info
11	1.863030	10.0.0.29	10.0.0.2	ICMP	Timestamp request
12	1.863238	10.0.0.2	10.0.0.29	ICMP	Timestamp reply
13	1.869470	10.0.0.29	10.0.0.2	ICMP	Timestamp request
14	1.869609	10.0.0.2	10.0.0.29	ICMP	Timestamp reply
15	2.739445	10.0.0.29	10.0.0.2	ICMP	Address mask request
16	2.742531	10.0.0.29	10.0.0.2	ICMP	Address mask request
17	7.062589	10.0.0.29	10.0.0.2	ICMP	Information request
18	7.064628	10.0.0.29	10.0.0.2	ICMP	Information request
19	11.354823	10.0.0.29	10.0.0.2	ICMP	Echo (ping) request
20	11.355045	10.0.0.2	10.0.0.29	ICMP	Echo (ping) reply
21	11.359669	10.0.0.29	10.0.0.2	ICMP	Echo (ping) request
22	11.359816	10.0.0.2	10.0.0.29	ICMP	Echo (ping) reply

図9-1 通常はこんなICMPトラフィックは流れない

これらのICMPトラフィックは、ICMPを使ったOSのフィンガープリントの収集と考えられます。攻撃者はこれらのリクエストのレスポンスを見て、どのOSが稼動しているかを予測するのです。



ICMPのタイプ13、15、17のパケットは通常では考えられないパケットなので、フィルタを作成してこれらのパケットのみを表示するようにしましょう。icmp.type==13 || icmp.type==15 || icmp.type==17というフィルタを作成してください。

9.2 ポートスキャン

portscan.pcap

攻撃者はポートスキャンによってネットワークに関する重要な情報を得ようとしています。ポートスキャンのソフトウェアを使ってターゲットのポートに順に接続していき、接続可能なポートを見つけ出すことができます。接続可能なポートは、堅牢な域にある秘密のトンネルのようなものです。攻撃者はこれらのトンネルの存在を知ると、あらゆる手段を使って侵入してくるでしょう。図9-2はportscan.pcapの一部で、ポートスキャンされている様子をキャプチャしています。

図9-2を見てください。ローカルコンピュータ（10.100.25.14）とリモートコンピュータ（10.100.18.12）を行き来するパケットが表示されています。これらのパケットをよく見ると、なぜ怪しいのかが分かるでしょう。

No. -	Time	Source	Destination	Protocol	Info
7	0.607512	10.100.25.14	10.100.18.12	TCP	16748 > telnet [SYN] Seq=0 Len=0
8	0.707986	10.100.25.14	10.100.18.12	TCP	12502 > ftp [SYN] Seq=0 Len=0
9	0.808340	10.100.25.14	10.100.18.12	TCP	30382 > 6000 [SYN] Seq=0 Len=0
10	0.904949	10.100.25.14	10.100.18.12	TCP	27986 > 1025 [SYN] Seq=0 Len=0
11	1.004235	10.100.25.14	10.100.18.12	TCP	25488 > smtp [SYN] Seq=0 Len=0
12	1.110883	10.100.25.14	10.100.18.12	TCP	6729 > sunrpc [SYN] Seq=0 Len=0
13	1.212836	10.100.25.14	10.100.18.12	TCP	29169 > 1028 [SYN] Seq=0 Len=0
14	1.307771	10.100.25.14	10.100.18.12	TCP	24305 > 9100 [SYN] Seq=0 Len=0
15	1.407052	10.100.25.14	10.100.18.12	TCP	17851 > 1029 [SYN] Seq=0 Len=0
16	1.512738	10.100.25.14	10.100.18.12	TCP	10985 > finger [SYN] Seq=0 Len=0

図9-2 ポートスキャンによってさまざまなポートへの接続が試みられている

サンプルファイルには、リモートコンピュータからローカルコンピュータのさまざまなポート（21番ポート、1028番ポートなど）に対してパケットが送信されている様子が記録されています。

もっと重要なのは、ポートスキャンはTELNET、microsoft-ds、FTP、SMTPなど、攻撃がしやすいポートに対して行われることが多いということです。リモートのコンピュータが複数のパケットをこれらのポートに送っているときは、ポートスキャンされている可能性が高いでしょう。

9.3 プリンタの氾濫

printerproblem.pcap

どんなに小さい企業でも、ネットワークに接続されたプリンタを何台か持っていることでしょう。用紙、インク、メンテナンスのコストなど、たとえ小さなプリンタでも、トータルコストはすぐに跳ね上がります。今回は、あなたのネットワーク上にあるプリンタが、誰もプリントアウトしてないのにおかしなものを印刷し始めるというシナリオです。その発信元を探り、解決するのが目的です。

9.3.1 分かっていること

問題のプリンタはサーバを介して共有されている大規模なものです。プリンタに接続するための特別な権限は必要なく、ログをとる機能也没有。問題は今も続いています。プリンタのキューを消しても、すぐにいっぱいになって印刷が始まります。

9.3.2 パケット解析開始

問題のプリンタはサーバと接続しているため、サーバ周辺のトラフィックを調べるのがよいでしょう。おそらく大量のパケットを見ることになりますが、Wiresharkをサーバにインストールするのが一番よさそうです。問題はずっと続いているため、パケットをキャプチャする時間に制限はありません。

9.3.3 解析

サンプルファイルのprinterproblem.pcapは、プリンタのトラフィックがどのようなのかを知るためのよい例です。図9-3のように、サーバ(10.100.16.15)は10.100.17.47のクライアントから大量にSPOOLSSパケットを受信しています。

これでプリンタにデータを送信している犯人は分かりましたが、まだ問題は解決し

No. .	Time	Source	Destination	Protocol	Info
49	0.058610	10.100.17.47	10.100.16.15	SPOOLSS	DeletePrinterIC request
50	0.058619	10.100.16.15	10.100.17.47	SPOOLSS	DeletePrinterIC response
51	0.059582	10.100.17.47	10.100.16.15	SPOOLSS	OpenPrinterEx request, \\e205000n5\Tech-office
52	0.059799	10.100.16.15	10.100.17.47	SPOOLSS	OpenPrinterEx response
53	0.060312	10.100.17.47	10.100.16.15	SPOOLSS	ClosePrinter request
54	0.060396	10.100.16.15	10.100.17.47	SPOOLSS	ClosePrinter response
55	0.061042	10.100.17.47	10.100.16.15	SPOOLSS	StartDocPrinter request, OpenPrinterEx(\\e205000n5\Tech-office)
56	0.062040	10.100.16.15	10.100.17.47	SPOOLSS	StartDocPrinter response

図9-3 SPOOLSSパケットがプリンタに流れ込んでいる

ていません。何が起きているかもう少し詳しく知るために、TCPストリームを見てみましょう。送信されているデータはMicrosoft Wordの文書で、ユーザー名がcsandersであることが分かります(図9-4)。

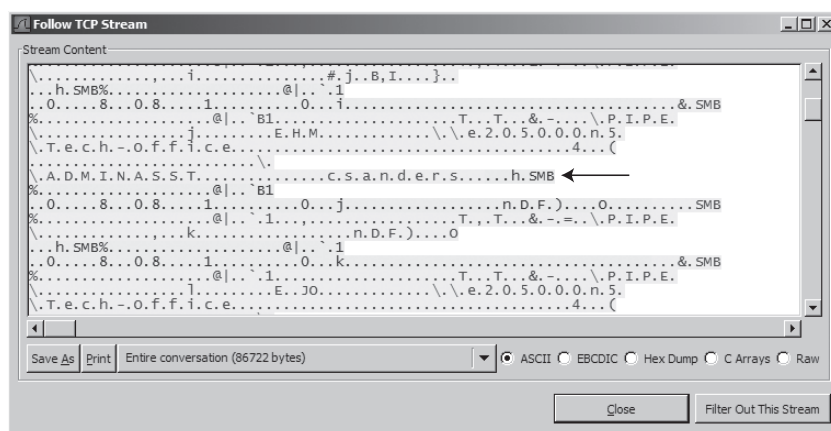


図9-4 TCPストリームを見れば、問題を解決するヒントが得られる

9.3.4 まとめ

SPOOLSパケットを止める前に、Wiresharkによって怪しいプリンタの挙動の原因を探ることができました。おそらく、10.100.17.47のコンピュータはなんらかの方法で侵入されたのでしょう[†]。

9.4 FTPサーバへの侵入

ftp-crack.pcap

FTPはデータを大量に転送するためのもっとも一般的な手段です。今回のシナリオに出てくる企業は、内部ネットワークにFTPサーバがあり、リリース前のソフトウェアすべてをそのサーバで管理しています。最近、このサーバを管理/運用している技術者が、サーバに対して何時間もの間大量のトラフィックが発生していることに気づきました。不幸なことにこのFTPサーバにはログをとる機能が付いていないため、パケットキャプチャのみで問題を解決しなければいけません。サーバに対するトラフィックが増えた原因を突き止め、その原因を排除しましょう。

9.4.1 分かっていること

このFTPサーバは非常に古く、ログをとる機能がありません。ほとんどの開発者が、このサーバのアカウントを持っており、すべてのファイルにアクセスすることができます。このサーバは外部からのアクセスも許可しており、開発者は家で仕事をすることもできます^{††}。

[†] 監訳注：プリンタのOSにはWindowsが使われていることも多く、きちんとメンテナンスされていないと当然バッチも当たりません。したがって、ウイルスに感染することも現実的にありえることなのです。ここでは別のコンピュータが原因であると判明しましたが、プリンタ自身が侵入される可能性も考慮する必要がありますでしょう。

9.4.2 パケット解析開始

このサーバはあなたのネットワークで稼動しているので、Wiresharkをサーバにインストールするのがよい方法のように思えます。しかしながら、このサーバには大量のトラフィックが流れ込んでいるため、パケットをキャプチャしても正しくキャプチャできない可能性があります。かわりにポートミラーリングを使いましょう。

9.4.3 解析

ftp-crack.pcap ファイルを開けばすぐに、何が起きているのか即座に理解できるでしょう。6章で学んだ、FTPの認証について思い出してください。

3ウェイハンドシェイクの後、ログインの処理が行われ、ユーザーはサーバとやりとりできるようになります。このサンプルファイルでは、ユーザー名とパスワードの認証が行われ、4番目のパケットのようにそれに失敗しています(図9-5)。

```

File Transfer Protocol (FTP)
  530 Login incorrect.\r\n
    Response code: Not logged in (530)
    Response arg: Login incorrect.
  
```

図9-5 4番目のパケットが認証に失敗している最初のパケット

きっとユーザーがパスワードを打ち間違えたのだろう、と最初は思うでしょうが、続くいくつかのパケットを見ればその仮定は打ち砕かれます。図9-6のとおり、何度も何度も認証に失敗しています。

No.	Time	Source	Destination	Protocol	Info
27	0.104111	10.121.70.151	10.234.125.254	TCP	ftp > 2220 [FIN, ACK] Seq=22 Ack=1 win=49152 [TCP CHECKSUM INCORRECT] Len=0
28	0.104155	10.234.125.254	10.121.70.151	TCP	2220 > ftp [ACK] Seq=1 Ack=23 win=17447 [TCP CHECKSUM INCORRECT] Len=0
29	0.108560	10.121.70.151	10.234.125.254	FTP	Response: 530 Login incorrect. ←
30	0.108773	10.121.70.151	10.234.125.254	TCP	ftp > 2221 [ACK] Seq=34 Ack=14 win=49152 [TCP CHECKSUM INCORRECT] Len=0
31	0.112332	10.234.125.254	10.121.70.151	TCP	2222 > ftp [FIN, ACK] Seq=13 Ack=56 win=17447 [TCP CHECKSUM INCORRECT] Len=0
32	0.120024	10.121.70.151	10.234.125.254	FTP	Response: 530 Login incorrect. ←
33	0.121851	10.234.125.254	10.121.70.151	TCP	2221 > ftp [FIN, ACK] Seq=14 Ack=56 win=17447 [TCP CHECKSUM INCORRECT] Len=0
34	0.122830	10.121.70.151	10.234.125.254	TCP	ftp > 2223 [ACK] Seq=34 Ack=11 win=49152 [TCP CHECKSUM INCORRECT] Len=0
35	0.141432	10.121.70.151	10.234.125.254	TCP	ftp > 2222 [ACK] Seq=56 Ack=14 win=49152 [TCP CHECKSUM INCORRECT] Len=0
36	0.141886	10.121.70.151	10.234.125.254	TCP	ftp > 2222 [FIN, ACK] Seq=56 Ack=14 win=49152 [TCP CHECKSUM INCORRECT] Len=0
37	0.141939	10.234.125.254	10.121.70.151	TCP	2222 > ftp [ACK] Seq=14 Ack=57 win=17447 [TCP CHECKSUM INCORRECT] Len=0
38	0.145312	10.121.70.151	10.234.125.254	TCP	ftp > 2221 [ACK] Seq=56 Ack=13 win=49152 [TCP CHECKSUM INCORRECT] Len=0
39	0.145896	10.121.70.151	10.234.125.254	FTP	Response: 530 Login incorrect. ←

図9-6 何度も認証に失敗している

認証に失敗すると、すぐにまた認証を試みています。認証を試みているクライアントはあなたのネットワーク上のコンピュータ(10.234.125.254)です。これらの認証は管理者のアカウント(admin)で行われています。図9-7は10番目のパケットの詳細ペインです。

⚠ 監訳注：そもそも古いFTPサーバをそのまま使うこと自体危険なので止めたほうがよいのですが、古いだけでなく、サーバのパフォーマンス等、諸事情があってやむを得ずログを取得できない場合には、パケットをキャプチャするしかないかもしれません。Wiresharkでデータをすべてキャプチャするとデータ量が多くなってしまうことが懸念される場合には、tcpdumpなどでパケットヘッダなどの部分的な記録を残す手もあります。tcpdumpの詳しい使い方については、マニュアル(man tcpdump)やWebサイト(<http://www.tcpdump.org>)などを参考にしてください。

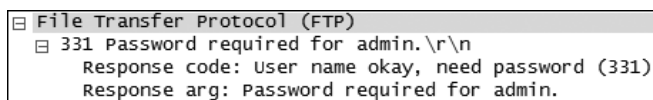


図9-7 10番目のパケットは管理者アカウントでログインを試みている

FTPのログイン処理のみを表示するために、ディスプレイフィルタを使ってみましょう。

```
ftp.request.command == "USER" || ftp.request.command == "PASS"
```

図9-8はフィルタを作成した結果表示されるパケットです。

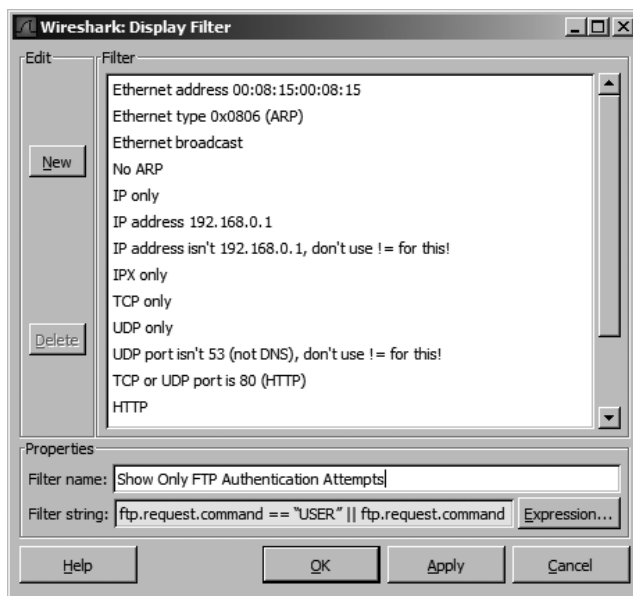


図9-8 [Display Filter] ダイアログにフィルタを入力して、認証のパケットのみを表示する

[Info] 欄を見ると、攻撃者がアルファベット順にパスワードを試していることが分かります[†]。これは、パスワードを辞書攻撃で探っている証拠です。辞書攻撃とは、ユーザー、またはマシンが作成した単語の辞書を使ってパスワードを推測する手法です。1つ1つパケットを見ていると、人間には不可能な速さでユーザー名とパスワードが入力されていることが分かるでしょう。おそらく、クラッキングツールを使って

[†] 監訳注：ネットワークを経由したパスワード攻撃には、大別すると「辞書攻撃」と「ブルートフォース攻撃」の二種類があります。両方とも機械的に何度もログインを試行するという、同じ特徴を持っていますので、厳密に言えば両方の可能性があるといえるでしょう。しかし多くの場合、「機械的な攻撃によってパスワードが危機に晒されている」というふうに事象を捕捉できれば、解析の目的は達成できるはずなので、攻撃の厳密な定義はあまり問題ではありません。

いるのです。これで、サーバに負荷がかかっている理由が分かりました。

9.4.4 まとめ

さらに調査をすると、あなたのネットワーク上のコンピュータが攻撃され、辞書攻撃によってFTPサーバの認証を突破するクラッキングツールがインストールされていることが分かりました。しかし仕事はまだ終わっていません。今度はFTPサーバを攻撃しているコンピュータの持ち主が意図的に攻撃を行っているのか、それとも外部からの侵入によって知らないうちに攻撃に加担しているのかを調べなければいけません。

9.5 Blasterワーム

blaster.pcap

インターネット上を飛び交うウイルスやワームは、システム管理者やエンドユーザーにとって大きな脅威です。このシナリオでは、エディのコンピュータがウイルスに感染します。彼がコンピュータを起動すると、60秒後にシャットダウンするというメッセージが表示されます。そして60秒後に本当にシャットダウンします。この症状はコンピュータを起動するたびに起こり、彼は60秒以上コンピュータを操作することができません。

9.5.1 分かっていること

あなたの企業はウイルス対策ソフトを使っていますが、集中管理しているのではなく、ユーザー個人が管理しています。

9.5.2 パケット解析開始

ウイルスやワームがコンピュータに悪さをしている可能性があるときは、そのコンピュータにスニッファをインストールするのは賢明な考えではありません。悪意あるプログラムがスニッファになんらかの影響を与え、パケットを正しくキャプチャできない可能性があります。そのため、ここではポートミラーリングを使うことにします。コンピュータを起動する瞬間から60秒後にシャットダウンするまでの間、パケットをキャプチャします。

9.5.3 解析

サンプルファイル `blaster.pcap` には、TCPパケットがいくつか記録されており、感染したと思われるコンピュータが1793番ポートと4444番ポートを使って、ほかのコンピュータにパケットを送信しています (図9-9)。これらのパケットは、「60秒タイマー」が付いているエディのコンピュータ以外ではキャプチャされません。怪しいですね。

No.	Time	Source	Destination	Protocol	Info
1	0.000000	10.234.0.239	10.234.2.116	TCP	1793 > 4444 [ACK] Seq=0 Ack=0 win=17330 [TCP CH
2	0.000191	10.234.2.116	10.234.0.239	TCP	4444 > 1793 [PSH, ACK] Seq=0 Ack=0 win=64475 [TC
3	0.218319	10.234.0.239	10.234.2.116	TCP	1793 > 4444 [ACK] Seq=0 Ack=20 win=17310 [TCP Cl
4	1.673435	10.234.0.239	10.234.2.116	TCP	1793 > 4444 [PSH, ACK] Seq=0 Ack=20 win=17310 [TC
5	1.673773	10.234.2.116	10.234.0.239	TCP	4444 > 1793 [PSH, ACK] Seq=20 Ack=18 win=64457 [TC
6	1.859752	10.234.0.239	10.234.2.116	TCP	1793 > 4444 [ACK] Seq=18 Ack=38 win=17292 [TCP C
7	3.713980	10.234.0.239	10.234.2.116	TCP	1793 > 4444 [PSH, ACK] Seq=18 Ack=38 win=17292 [TC
8	3.900264	10.234.2.116	10.234.0.239	TCP	4444 > 1793 [ACK] Seq=38 Ack=30 win=64445 [TCP C

図9-9 エディのコンピュータのみがこのパケットを送信しているのはおかしい

ウイルスやワームを識別するもっともよい方法は、パケットのバイナリを見ることです。バイナリのペインを見てください。最初のパケットには異常はありません(図9-10)。

0000	00 d0 59 aa af 80 00 01 96 3c 3f a8 08 00 45 00	..Y....<?...E.
0010	00 28 08 ed 40 00 7f 06 d9 ac 0a ea 00 ef 0a ea	(..@... .K...t..
0020	02 74 07 01 11 5c 76 be 16 50 cd 5a 82 b2 50 10	.t...V..P.Z..P.
0030	43 b2 59 73 00 00 00 00 00 00 00 00	C.Ys....

図9-10 最初のパケットには特に異常は見当たらない

しかし2番目のパケットを見ると、C:\WINNT\System32というディレクトリが見えます(図9-11)。これはWindows 2000ではもっとも重要なディレクトリの1つで、Windowsのコアとなるシステムファイルが含まれています。このディレクトリにアクセスしようとするというのは通常考えられません。

0000	00 80 ad d1 84 d7 00 d0 59 aa af 80 08 00 45 00Y....E.
0010	00 3c 00 3a 40 00 80 06 e1 4b 0a ea 02 74 0a ea	<.:@... .K...t..
0020	00 ef 11 5c 07 01 cd 5a 82 b2 76 be 16 50 50 18	...\....Z..V..PP.
0030	fb db 73 31 00 00 0d 0a 43 3a 5c 57 49 4e 4e 54	...s1... C:\WINNT
0040	5c 73 79 73 74 65 6d 33 32 3e	\system32>

図9-11 何かがC:\WINNT\System32のシステムファイルにアクセスしようとしている

3番目のパケットにもたいした情報はありますが、4番目のパケットには怪しい部分があります(図9-12)。

0000	00 d0 59 aa af 80 00 01 96 3c 3f a8 08 00 45 00	..Y....<?...E.
0010	00 3a 08 ef 40 00 7f 06 d9 98 0a ea 00 ef 0a ea	...@... ..K...t..
0020	02 74 07 01 11 5c 76 be 16 50 cd 5a 82 c6 50 18	.t...V..P.Z..P.
0030	43 9e a0 4d 00 00 f3 74 61 72 74 20 6d 73 62 6c	C..M...st art msbl
0040	61 73 74 2e 65 78 65 0a	ast.exe.

図9-12 4番目のパケットには「msblast.exe」という文字が含まれている

4番目のパケットのバイナリのペインを見ると、「msblast.exe」というファイル名が含まれています。もし2003年にITに携わっていたのなら、これがなんなのかすぐに分かるでしょう。msblast.exeを知らないという人はGoogleで調べてみてください。このファイルはBlaster ワームが使うものです。これが問題の原因だったのです†。

9.5.4 まとめ

このシナリオでは、コンピュータのウイルス対策ソフトがうまく機能していなかったため、Blaster ワームに感染してしまいました。

ウイルスやワームに感染した疑いがあるときは、その症状をインターネットで検索すれば、それがなんなのか特定できるでしょう。ウイルスやワームを特定したら今度はそれをどうやって駆除すればよいのかを再度検索してください。

9.6 隠された情報

`covertinfo.pcap`

このシナリオでは、あなたは巨大企業のネットワークのセキュリティ管理者です。あなたは上司から、2人の従業員が会社の資産を持ち出そうと相談しているところがある従業員が聞いたという話をされました。あなたは彼らのコンピュータを監視し、彼らの計画を探らなければいけません。

9.6.1 分かっていること

このシナリオの疑惑はあくまで疑惑です。問題の2人の従業員はコンピュータに非常に詳しいため、立ち聞きしたことが本当かどうか確認できるまでは、非常に慎重に調査する必要があります。

9.6.2 パケット解析開始

2人の従業員にあなたが調査していることを気づかれてしまうので、Wiresharkを彼らのコンピュータにインストールするわけにはいきません。そのため、ポートミラーリングを使います。各コンピュータに対してポートミラーリングする必要があります。

9.6.3 解析

この2人の従業員は、1日を通して大量のパケットを送受信しています。ほとんどの場合それは通常のパケットなので、まずは疑わしいトラフィックを見つけることから始めなければなりません。MSRPC (DCERPC)、NetBIOS、ICMPなどを表示するディスプレイフィルタを作成しましょう。これらは通常の業務では発生しないものです。ディスプレイフィルタを使いフィルタリングしたものがcovertinfo.pcapに記録されています(図9-13)。

† 監訳注：blaster.pcapにはBlaster本体は含まれていませんが(そんな危険なことではできませんので(笑))、キャプチャしたデータにblaster.exeのような怪しいデータが含まれているかどうか簡単に見分けるには、バイナリのペインや[Follow TCP Stream]などを用いて通信データを目視して、中に「MZP」とか「This program must be run under Win32」という文字列が存在するかどうか見てみるとよいでしょう。両方ともWindows OSのNT、2000、XP、2003 Serverなどで動くプログラム(.exe)ファイルの特徴です。該当する通信に.exeファイルが含まれていることがそもそもおかしいのであれば、その通信は「怪しい」可能性が高いといえるでしょう。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.100.17.48	10.100.18.5	ICMP	Echo (ping) request
2	0.000015	10.100.18.5	10.100.17.48	ICMP	Echo (ping) reply

図9-13 ICMPが？なぜこの2人はpingのやり取りをしているんでしょう？

これらのパケットは普通のICMPパケットに見えますが、送信元と送信先があの2人の従業員のコンピュータのアドレスになっています。なぜ日中にpingのやり取りをしているのでしょうか。

次に、バイナリのペインを見てみましょう。何か面白いものが見つかるかもしれません（図9-14）。

0000	00 15 c5 37 e1 c1 00 0b	db 71 d7 39 08 00 45 00	...7.... .q.9..E.
0010	00 a8 52 87 00 00 80 01	af d1 0a 64 11 30 0a 64	..R..... ..d.0.d
0020	12 05 08 00 1c 2f e4 0e	a3 0d bc 44 8d 15 00 00/. ...D...
0030	00 00 00 00 00 00 42 6c	75 65 43 68 61 74 31 30Bl ueChat10
0040	2e 31 30 30 2e 31 37 2e	34 38 20 20 20 54 72 61	.100.17. 48 Tra
0050	6e 73 66 65 72 20 61 6c	6c 20 6f 66 20 74 68 65	nsfer al l of the
0060	20 66 75 6e 64 73 20 74	6f 20 61 63 63 6f 75 6e	funds t o account
0070	74 20 6e 75 6d 62 65 72	20 31 31 39 32 38 32 38	t number 1192828
0080	32 33 31 2d 30 20 20 20	20 20 20 20 20 20 20 20	231-0
0090	20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20	
00a0	20 20 20 20 20 20 20 20	20 20 20 20 20 20 20 20	
00b0	20 20 20 20 20 20 20 20		

図9-14 これは普通のpingじゃない

このpingは普通のものとは程遠いです。実は、データの部分で機密情報をやり取りしていたのです！

9.6.4 まとめ

このシナリオで使用されている技術を、Lokiといいます。情報をこっそりと送信することです。LokiはICMPパケットにデータを埋め込むことを意味します。今回は、2人の従業員がICMPを機密情報を運ぶ手段として利用していました。

隠れた通信網を使うというのは新しい技術ではありませんが、それは絶えず発展しています。TCPヘッダやARPパケットにも不穏なデータが隠されていることも珍しくありません。常にバイナリのペインに注意しててください。それがパケットに含まれる秘密のデータを見つけるための唯一の方法になるかもしれません。

9.7 ハッカーの視点

hackersview.pcap

本書は、ネットワーク管理者の視点から書かれています。しかし、パケット解析の知識を持っているハッカーがネットワークに侵入しようとしたら？ このシナリオでは、あなたには自分の会社のネットワーク上で機密情報にアクセスしようとするハッカーになってもらいます。

9.7.1 分かっていること

あなたはその会社の従業員ですが、ネットワークリソースへのアクセスは制限されています。ネットワークはありふれたイーサネットで、いくつかのスイッチとルータがあります。ネットワークに接続されているコンピュータでは、さまざまなバージョンのWindowsが稼動しており、ユーザーごとに権限が設定されています。

9.7.2 パケット解析開始

ハッカーは、ネットワークに管理者権限でアクセスするために、ネットワーク管理者のパスワードを知りたいがります。または、単にネットワーク障害を引き起こそうとします。今回は、ルータに侵入して何か深刻なダメージを与えることにしましょう。ネットワーク管理者はいつもネットワークをいじくり回しているので、ネットワーク管理者とルータのトラフィックを監視するだけでパスワードを盗むことができそうです。

運がよいことに、ネットワーク管理者のコンピュータとルータはあなたと同じサブネットワークに接続しています。Cain & AbelでARPキャッシュポイズニングを実行し、2章で学んだようにネットワーク管理者のコンピュータ10.100.18.5とルータ10.100.16.1の間に割り込みます。

9.7.3 解析

しばらくして、あなたはネットワーク管理者がTELNETを使ってルータと通信しているところを捕らえました。図9-15は、このシナリオに関連するTELNETのトラフィックのみを記録しています。

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	10.100.18.5	10.100.16.1	TCP	3756 > telnet [SYN] Seq=0 Len=0 MSS=1460
2	0.001244	10.100.16.1	10.100.18.5	TCP	telnet > 3756 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460
3	0.001263	10.100.18.5	10.100.16.1	TCP	3756 > telnet [ACK] Seq=1 Ack=1 win=65535 Len=0
4	0.003143	10.100.16.1	10.100.18.5	TELNET	Telnet Data ...
5	0.003201	10.100.18.5	10.100.16.1	TELNET	Telnet Data ...
6	0.004161	10.100.16.1	10.100.18.5	TELNET	Telnet Data ...
7	0.150539	10.100.18.5	10.100.16.1	TCP	3756 > telnet [ACK] Seq=4 Ack=6 win=65530 Len=0
8	0.151553	10.100.16.1	10.100.18.5	TELNET	Telnet Data ...
9	0.351722	10.100.18.5	10.100.16.1	TCP	3756 > telnet [ACK] Seq=4 Ack=16 win=65520 Len=0
10	1.285806	10.100.18.5	10.100.16.1	TELNET	Telnet Data ...
11	1.287056	10.100.16.1	10.100.18.5	TCP	telnet > 3756 [ACK] Seq=16 Ack=5 win=8192 Len=0
12	1.287275	10.100.16.1	10.100.18.5	TELNET	Telnet Data ...

図9-15 探しているものを見つけた

6章で、TELNETはデータを平文でやり取りしているということを学びました。TELNETはここでは、スイッチやサーバ、ルータをリモートから操作するために使われています。これらのネットワーク機器はSSHを使って安全に通信ができるようになっていますが、ここのシステム管理者は横着者のようです。平文で通信しているので、すぐにログイン情報を発見することができました。

TELNETはすべてのことが一連の通信の中で順番通りに起こるシーケンシャルなプロトコルです。したがって、ログインプロセスを見つける一番よい方法は、TELNETのパケットを1つ1つ見ていくことです。8番目のパケットを見てください。認証が

そこから始まっています（図9-16）。

Frame 8 (64 bytes on wire, 64 bytes captured)
Ethernet II, Src: Enterasy_31:4a:f0 (00:11:88:31:4a:f0), Dst: Dell_37:e1:c1 (00:15:c5:37:e1:c1)
Internet Protocol, Src: 10.100.16.1 (10.100.16.1), Dst: 10.100.18.5 (10.100.18.5)
Transmission Control Protocol, Src Port: telnet (23), Dst Port: 3756 (3756), Seq: 6, Ack: 4, Len: 10
Telnet
Data: Username:

図9-16 認証を開始するためにユーザー名を要求している

パケット詳細のペインのTelnetの部分を見れば、サーバがユーザー名を要求していることが分かります。次のパケットではクライアントがサーバにユーザー名を返すだろうと思うでしょうが、ことはもう少し複雑です。

図9-17のとおり、10番目のパケットには「a」という文字しか含まれていません。これはユーザー名ではありません。

0000	00 11 88 31 4a f0 00 15 c5 37 e1 c1 08 00 45 00	...1j... .7....E.
0010	00 29 50 9c 40 00 80 06 73 65 0a 64 12 05 0a 64	.)P.@... se.d...d
0020	10 01 0e ac 00 17 24 0f 27 9a e0 a9 10 7c 50 18\$. '.... P.
0030	ff f0 cc 7a 00 00 61	...Z..

図9-17 このパケットにはパズルの最初のピース「a」が含まれている

13番目のパケットでは、クライアントがサーバに「d」という文字を渡しています（図9-18）。管理者の返答は1回につき1文字ずつ送られ、「admin」という文字列を送り終わるまで繰り返されます。ありふれたユーザー名ですね？ きっとデフォルトのままなのでしょう。

0000	00 11 88 31 4a f0 00 15 c5 37 e1 c1 08 00 45 00	...1j... .7....E.
0010	00 29 50 9f 40 00 80 06 73 62 0a 64 12 05 0a 64	.)P.@... \$b.d...d
0020	10 01 0e ac 00 17 24 0f 27 9b e0 a9 10 7d 50 18\$. '....}P.
0030	ff ef c9 79 00 00 64	...y..

図9-18 パケットごとに1文字ずつ判明していく。ここでは「d」

24番目のパケットではパスワードが要求されています（図9-19）。

Frame 24 (64 bytes on wire, 64 bytes captured)
Ethernet II, Src: Enterasy_31:4a:f0 (00:11:88:31:4a:f0), Dst: Dell_37:e1:c1 (00:15:c5:37:e1:c1)
Internet Protocol, Src: 10.100.16.1 (10.100.16.1), Dst: 10.100.18.5 (10.100.18.5)
Transmission Control Protocol, Src Port: telnet (23), Dst Port: 3756 (3756), Seq: 23, Ack: 11, Len: 10
Telnet
Data: Password:

図9-19 サーバがパスワードを要求している

ここでも、1文字ずつパスワードが送信されています (図9-20)。

0000	00 11 88 31 4a f0 00 15 c5 37 e1 c1 08 00 45 00	...1j... 7....E.
0010	00 29 50 b2 40 00 80 06 73 4f 0a 64 12 05 0a 64	.)P.@... 5O.d...d
0020	10 01 0e ac 00 17 24 0f 27 a1 e0 a9 10 8d 50 18\$. '.....P.
0030	ff df cb 73 00 00 52	...s..0

図9-20 パスワードの最初の文字は「b」

パスワード (barrymanilow) がすべて判明するまでスニффイングします。あなたはルータのパスワードを手にしただけでなく、ネットワーク管理者のすばらしい音楽の趣味まで分かってしまったのです！

9.7.4 まとめ

これであなたはこのネットワークを破滅させることができます。ルータで設定されているサブネットを削除し、ヘルパーアドレスを変え、いろいろないたずらをしてネットワーク管理者に頭痛を起こさせることができます。

このシナリオのポイントは、ネットワーク管理者をどうやって怒らせるかではなく、パケット解析の知識がある人はどのように攻撃できるかということです。WiresharkやCain & Abelを使うことで、このネットワークの機能すべてを止めてしまうことができるということが分かりました。

10章

無線LANのスニффイング

無線LANの世界は、伝統的な有線LANとはまったく違うものです。無線LANの世界では、周波数、規格、独特のセキュリティを考慮する必要があります。そういった付加的な事柄があるため、スニッフイングの方法もまったく違ったものになります。

この章では、WindowsシステムおよびUnixシステムの無線LANでのスニッフイングについて述べられています。実際のサンプルを見つつ、無線LANでのスニッフイングがどうして特殊なのかを学びます。

10.1 1つのチャンネルをスニッフイング

無線LANでのスニッフイングでまず理解しなければいけないのは、同時に1つのチャンネルしかスニッフイングできないということです。アメリカ合衆国の無線LANは、11のチャンネルのうち1つのチャンネルを使っています。したがって、パケットをキャプチャするにはまずどのチャンネルが使われているかを調べる必要があります(図10-1)。

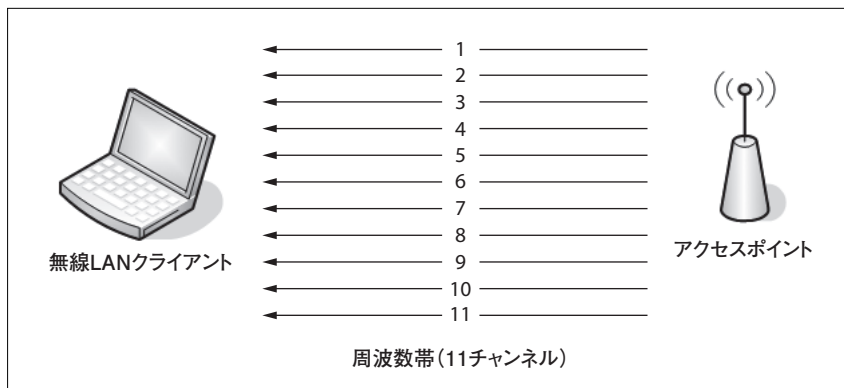


図10-1 同時に1つのチャンネルしかスニッフイングできないので、退屈になるかもしれない

チャンネルを1つずつ切り替えて、どのチャンネルが使用されているかを確認しましょう。データをキャプチャできるまで、チャンネルを切り替えたらパケットキャプチャを開始するという作業を繰り返しましょう。チャンネルの切り替えは技術的な解決策ではありませんが、効果的です。

10.2 無線LANのインターフェース

残念ながら、無線LANではいつでも信用できる通信が行われているわけではありません。データは空気を通して送られるため、何かがその信号を妨げることがままあります。無線LANには干渉を制御する機能が備わっていますが、それがうまく動作しないときもあります。したがって無線LANでパケットをキャプチャするときには、電波を反射するもの、硬くて大きなもの、電子レンジ、2.4GHzの携帯電話、厚い壁、高密度のものなどが近くにないことを確認しましょう。

また、できるだけ解析するコンピュータの近くにできるようにしてください。パケットをキャプチャしたいコンピュータがある場所の上の階からでは、うまくキャプチャすることはできないでしょう。

10.3 無線LANカードのモード

無線LANのパケットをキャプチャする前に、無線LANカードのモードについて学んでおきましょう。

無線LANカードのほとんどはアドホックモードになっていますが、マスターモードとモニターモードというものもあります。各モードの動作を図10-2に示します。

インフラストラクチャモード (マネージドモード)

インフラストラクチャモードでは、クライアントはアクセスポイント (WAP : Wireless Access Point) に直接接続します。このモードでは、クライアントはアクセスポイントに通信の制御を任せます[†]。

アドホックモード

アドホックモードは、クライアント同士が直接無線を介して通信をするときに使います。このモードでは通信を行う2つのクライアントが、アクセスポイントの代わりに通信を制御します。

マスターモード

ハイレベルな無線LANカードは、マスターモードもサポートしています。このモードでは、クライアントがアクセスポイントのような役割を担うことができます。

[†] 訳注：Linux上で無線LANを設定するコマンド `iwconfig` では、インフラストラクチャモードは「Managed mode」と表記されます。`iwconfig` に関しては後述。

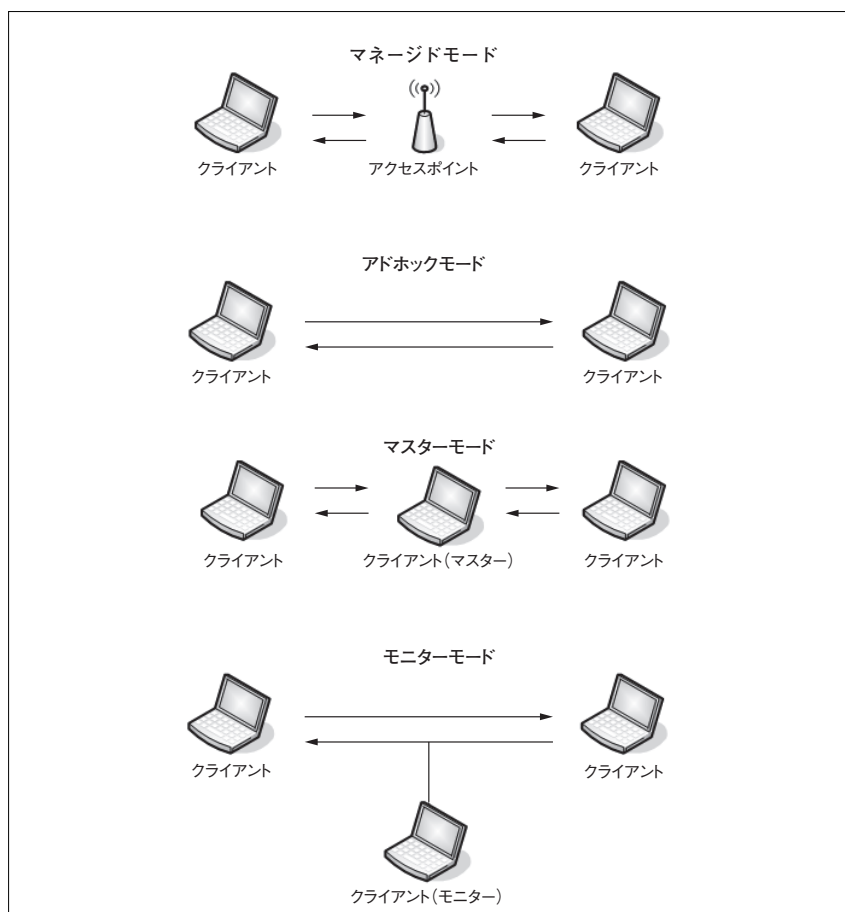


図 10-2 無線LANカードのモード

モニターモード

これがもっとも重要なモードです。モニターモードの無線LANカードは、データの送受信は行わず、飛び交うパケットを監視します。Wiresharkで無線LANのパケットをキャプチャする場合は、キャプチャするコンピュータの無線LANカードがモニターモードをサポートする必要があります。パケットキャプチャするために無線LANカードを買うときは、モニターモード (RFMONモードとも呼ばれます) をサポートしていることを確認してください[†]。

[†] 監訳注：現状、Wiresharkでは設定オプションの [Capture packets in promiscuous mode] をオフにすれば、無線LANに接続したあとの通信はキャプチャできます（無線LAN特有のパケット（WEPキーのやり取りなど）はキャプチャできません）。

10.4 Windows上での無線LANのスニッフイング

モニターモードをサポートしている無線LANカードを使っている、Windowsではそのモードを使うことができません。ほかのドライバが必要です。

10.4.1 AirPcapの設定

AirPcap (CACE Technologies : <http://www.cacotech.com>) は、Windows上で無線LANのパケット解析を行うために設計されたものです。AirPcapは無線LANでのパケットキャプチャのために設計されたUSBフラッシュドライブです(図10-3)。AirPcapは3章で説明したWinPcapドライバを使っており、専用の設定画面があります。



図10-3 AirPcapはノートPCと一緒に簡単に持ち運べるくらいコンパクト

AirPcapの設定はオプションが少ないので非常に簡単です。[AirPcap Control Panel]から以下のオプションが設定可能です(図10-4)。

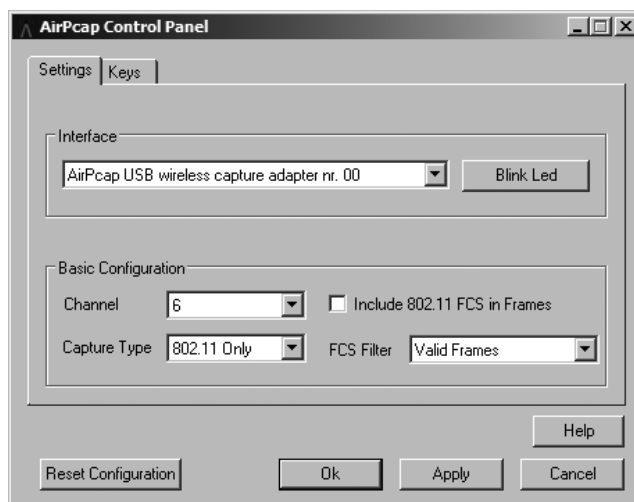


図10-4 AirPcapの設定用プログラム

[Interface]

キャプチャに使うデバイスを選択できます。解析するときに、複数の AirPcap を使って複数のチャンネルを同時にスニッフイングする場合があります。

[Blink Led]

AirPcap の LED を点滅させます。複数の AirPcap を使っているときに、どれを使っているかを示すためのものです。

[Channel]

ここでは、AirPcap を使ってキャプチャするチャンネルを選択します。

[Include 802.11 FCS in Frames]

OS によっては、デフォルトで無線 LAN のパケットのチェックサム最後の 4 ビットを取り除いてしまうことがあります。このチェックサムは FCS (Frame Check Sequence) と呼ばれており、転送している間にデータが破損していないことを保証するために使われています。特に理由がなければ、チェックボックスをオンにして FCS チェックサムを削除しないようにしましょう。

[Capture Type]

[802.11 Only] と [802.11+Radio] という 2 つのオプションがあります。[802.11 Only] というオプションは、標準的な 802.11 のパケットのヘッダもキャプチャするということです。[802.11+Radio] は、レート、周波数、信号レベルやノイズレベルを含むラジオタップヘッダもキャプチャします。入手可能なすべての情報を見られるようにするため、[802.11+Radio] を選択しましょう。

[FCS Filter]

[Include 802.11 FCS in Frames] のチェックボックスをオンにしていなくても、このオプションを有効にしておけば FCS のチェックによりデータが破損していると判断されたパケットはフィルタされます。[Valid Frames] オプションをオンにすれば、FCS のチェックによりデータが正しく受信されたと判断されたものだけが表示されます。

[WEP Configuration]

ここでは、スニッフイングしたいネットワークの WEP キーを入力し、WEP によって暗号化されたデータを解釈できるようにします。

10.4.2 AirPcap を使ったパケットキャプチャ

AirPcap をインストールして設定したら、以下の手順でパケットをキャプチャしてみましょう。

1. Wireshark のメインメニューの [Capture] から [Options] を選択します。
2. [Interface] から AirPcap を選択します (図 10-5)。

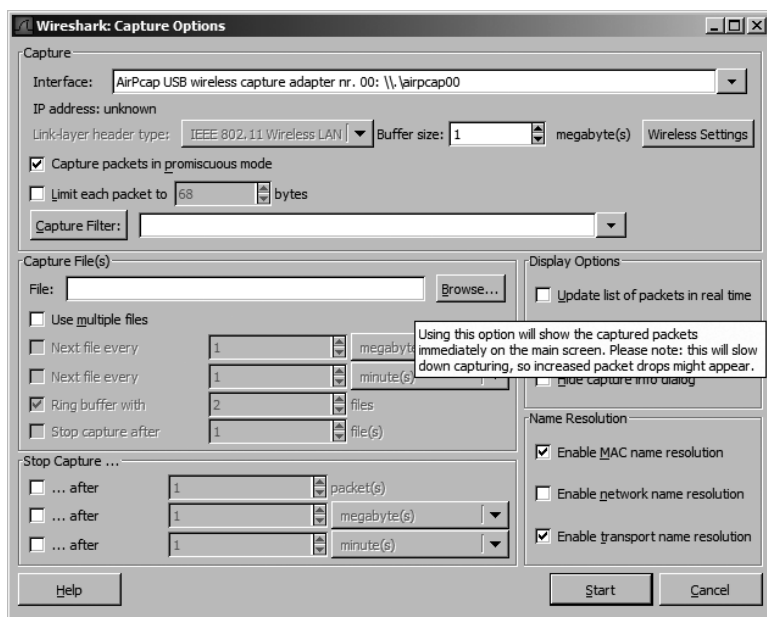


図 10-5 キャプチャするインターフェースとして、AirPcap のデバイスを選択する

[Wireless Settings] というボタン以外は慣れた画面だと思います。[Wireless Settings] ボタンをクリックすると AirPcap と同じオプションが表示されます (図 10-6)。Wireshark は AirPcap と完全に一体化しているため、AirPcap で設定できることは Wireshark でも設定できます。

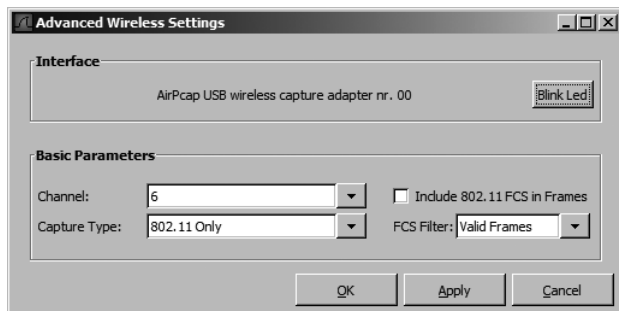


図 10-6 [Advanced Wireless Settings] ダイアログでは Wireshark から AirPcap の設定ができる

3. 設定がすべて終わったら、[Start] ボタンを押してパケットキャプチャを開始してください。

10.5 Linux上での無線LANのスニッフィング

Linuxでのスニッフィングに必要なのは、無線LANカードをモニターモードにすることだけです。残念ながらモニターモードに変更する手順は無線LANカードごとに異なるため、ここでそのやりかたを説明することはできません。ご自分の無線LANカードについて、Webで調べてみてください。

Linux上で無線LANカードをモニターモードに変更するもっとも一般的な方法は、Linuxにビルトインされている機能を使うことです。iwconfig コマンドを使えば、無線LANカードを設定できます。コンソール上でiwconfigを実行すると、以下のような結果になります。

```
$ iwconfig
Eth0    no wireless extensions
Lo0     no wireless extensions
Eth1    IEEE 802.11g ESSID:"Tesla Wireless Network"
        Mode:Managed Frequency:2.462 GHz Access Point: 00:02:2D:8B:70:2E
        Bit Rate: 54 Mb/s Tx-Power=20 dBm Sensitivity=8/0
        Retry Limit:7 RTS thr:off Fragment thr:off
        Power Management:off
        Link Quality=75/100 Signal level=-71 dBm Noise level=-86
        Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
        Tx excessive retries:0 Invalid misc:0 Missed beacon:2
```

iwconfig コマンドの結果から、802.11g という無線LANのプロトコルについての情報が表示されている Eth1 が無線LANのインターフェースであることが分かります。Eth0 と Lo0 では、無線LANは使えません。

Eth1 と表示されている行の下の方を見てください。iwconfig コマンドを実行して得られる無線LANカードのモードや周波数などの情報とともに、モードがManaged であると表示されています。ここを変更する必要があります。

Eth1 をモニターモードに変更するには root 権限が必要なので、su コマンドでユーザーを変更します。

```
$ su
Password: 〈root のパスワードを入力〉
```

root になれば、無線LANカードのオプションを設定するコマンドを実行することができます。Eth1 をモニターモードにするには、以下のコマンドを実行してください。

```
# iwconfig eth1 mode monitor
```

モニターモードに変更したら、iwconfig をもう一度実行して変更を有効にします。以下のコマンドを実行してください。

```
# iwconfig eth1 up
```

iwconfig コマンドでチャンネルを切り替えることもできます。Eth1 のチャンネルを3に切り替えるには、以下のコマンドを実行してください。

```
# iwconfig eth1 channel 3
```



パケットキャプチャしてる間にもチャンネルを切り替えることができるので、遠慮なく変更してください。スクリプトを作ってしまうと簡単に実行することができます。

設定が終わったら Wireshark を起動し、パケットキャプチャを開始してください。

10.6 802.11 のパケット

80211traffic.pcap

無線LANと有線LANのパケットの違いは、802.11ヘッダがあるかないかです。このヘッダにはデータの転送に使う媒体の情報が含まれています(図10-7)。

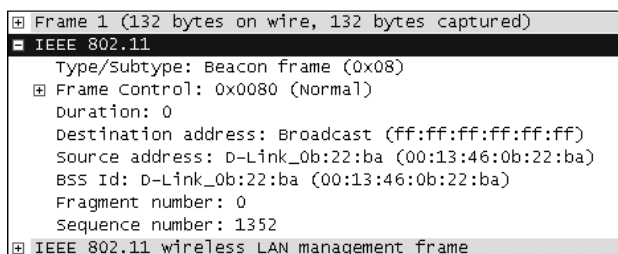


図10-7 802.11ヘッダには無線LANについての情報が含まれている

80211traffic.pcapを開き、パケットを詳しく見てみましょう。いくつか興味深い項目があります。

Type/Subtype

802.11パケットのタイプやサブタイプを指定しています。タイプにはManagement、Data、Controlがあります。

各タイプにはサブタイプがあります。たとえばManagementタイプにはBeacon frame、Authentication request、Disassociation noticeというサブタイプがあります。

Destination address、Source address、BSS Id

ここには送信元および送信先のアドレスとBSSIDが含まれます。

Fragment number、Sequence number

これらの番号は無線LANのパケットを正しく組み立てられるようにするために

振られるものです。

10.6.1 802.11のフラグ

802.11のパケットには、無線LAN特有の情報を含むフラグがあります(図10-8)。フラグには以下の項目があります。

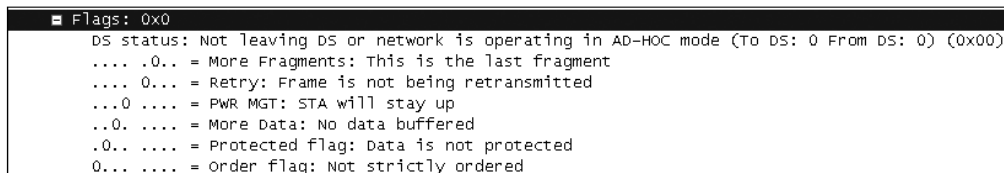


図10-8 フラグには無線LAN特有の情報が多く含まれている

DS status

パケットからどこからどこに転送されるかはDS (Distribution Status) によって決まります。もしFrom DSが1でTo DSが0なら、パケットはアクセスポイントからクライアントに向かって転送されます。値が反対なら、パケットはクライアントからアクセスポイントに転送されます。両方の値が0の場合は、通常はアクセスポイントからのブロードキャストです。

More Fragments

分割されたパケットがほかに存在するかどうかを示します。

Retry

Retryオプションは、受信したパケットがオリジナルのもの (0) なのか再送されたもの (1) なのかを示します。

PWR MGT

クライアントが省電力モードになっているかどうかを示します。

More Data

この後送信されるのを待っているパケットがあるかどうかをアクセスポイントがクライアントに教えるためのフラグです。

Protected flag

パケットのデータが暗号化されているかどうかを示します。

Order flag

パケットが特定の順番どおりになっている必要があるかどうかをクライアントに伝えます。クライアントはこのフラグによって、スループットを上げるために勝手に順番を変えてよいかどうかを判断します。

10.6.2 ビーコンフレーム

ビーコンフレームは無線LANでの通信においてもっとも有益なパケットの1つです。ビーコンフレームはアクセスポイントからブロードキャスト宛に送信されるパケットで、そのアクセスポイントに接続可能なクライアントに対して、接続に必要なパラメータを教えるために送信されます。つまり、このパケットにはいろいろと便利な情報が詰まっているわけです(図10-9)。

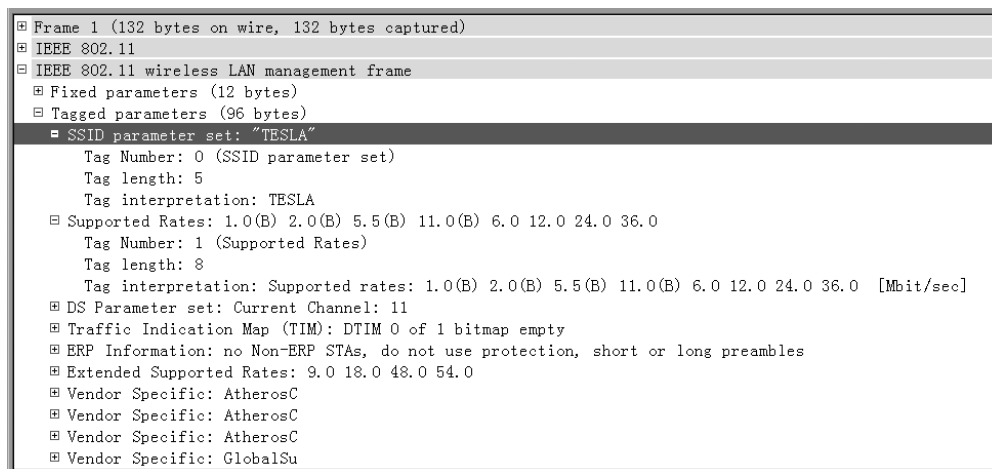


図10-9 ビーコンフレームにはアクセスポイントについて知りたいことがすべて詰まっている

ビーコンフレームに含まれる情報のいくつかをここで説明します。

SSID parameter set

アクセスポイントがブロードキャストしている無線LANセグメントのSSIDです。

Supported Rates

アクセスポイントがサポートしているデータのスループットのレートと、プロトコルが802.11bなのか802.11gなのかを示しています。

DS Parameter set

アクセスポイントがブロードキャストしているチャンネルの情報です。

Extended Supported Rates

アクセスポイントがサポートしているスループットのレートの一覧です。

Vendor-specific

アクセスポイントのベンダ特有の情報です。チップセットの製造元、タグの番号、タグの長さも含まれます(チップセットの製造元とアクセスポイントの製造元は異なる場合があります)。

10.7 無線LAN特有の情報

Wiresharkのパケット一覧のペインには通常6つのコラムがありますが、無線LANのパケットを表示すると、さらに2つの非常に便利なコラムが表示されます。RSSIとTX Rateです。RSSI (Received Signal Strength Indication : 受信信号強度) コラムはRF (Radio Frequency : 無線周波数) 信号の強さを、TX Rateコラムはキャプチャしたパケットのレートを表示します (図 10-10)。これらの情報は無線LANでのトラブルシューティングにおいて大きな助けになるでしょう。実際に、無線LANのクライアントが信号が強いということを示しているとき、これらのコラムがあればそれが本当なのかどうかを確認することができます。

No. •	Time	Source	Destination	Protocol	Info	TX Rate	RSSI
-------	------	--------	-------------	----------	------	---------	------

図 10-10 2つのコラムがあるかないかで解析のしやすさが大きく変わる

これらのコラムをパケット一覧のペインに表示させるには、以下の手順に従ってください。

1. メインメニューの [Edit] から [Preferences] をクリックします。
2. [Columns] を選択して [New] ボタンをクリックします。
3. [Title] のテキストボックスに RSSI と入力し、[Format] ドロップダウンボックスで [IEEE 802.11 RSSI] を選択します。
4. TX Rate についても同じ手順を繰り返し、[Format] ドロップダウンボックスで

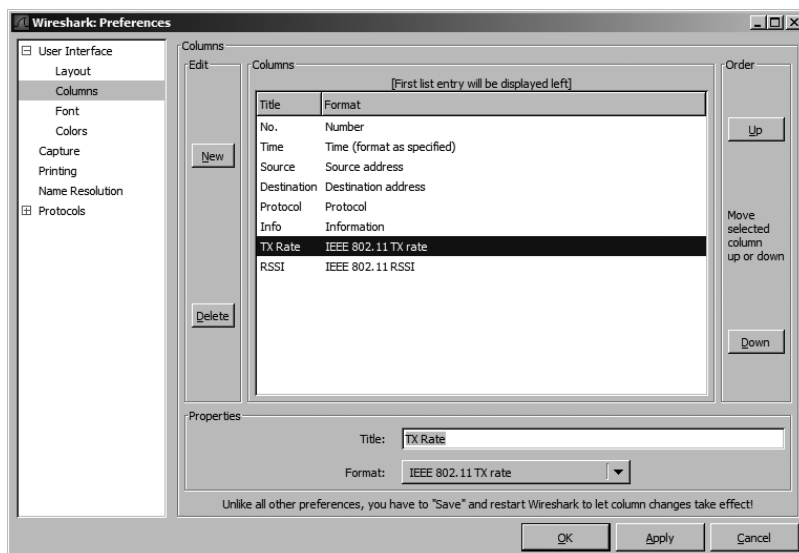


図 10-11 無線LAN特有の情報を表示するコラムをパケット一覧のペインに追加する

[IEEE 802.11 TX Rate]を選択します。図10-11は以上の手順を終えた後のウィンドウです。

5. [Apply]をクリックすると、新しいカラムが追加されます。
6. [OK]をクリックして変更を保存します。

10.8 無線LAN特有のフィルタ

フィルタの有用性については4章で議論しました。有線LANでは各デバイスごとにケーブルが存在するため、キャプチャしたいパケットのみをキャプチャするフィルタは簡単に作ることができました。しかしながら、無線LANでは各クライアントによって発生するすべてのトラフィックがチャンネル上に共存しており、1つのチャンネルをキャプチャするとさまざまなクライアントのトラフィックが混在したものが記録されます。ここでは、自分が求めるパケットのみをキャプチャできるようなフィルタの作り方を学びます。

10.8.1 特定のBSSIDでフィルタリング

各アクセスポイントには、BSSID (Basic Service Set Identifier) と呼ばれるユニークな名前が割り振られています。アクセスポイントが発信する、無線LANの管理用パケットとデータフレームの中には、この名前が含まれています (「10.6 802.11のパケット」参照)。

あなたが解析しようとしている無線LANのBSSIDが分かれば、あとは特定のアクセスポイントから送信されるパケットを見つけるだけです。Wiresharkでは、パケット一覧のペインの [Info] 欄でパケットを転送しているアクセスポイントを表示してくれますので、目的のパケットを見つけ出すのは簡単でしょう。

解析したい無線LANのアクセスポイントから送信されるパケットを見つけたら、802.11 ヘッダからBSSIDを見つけ出しましょう (図10-9)。

パケット詳細のペインに表示されるBSSIDのMACアドレスを見つけたら、`wlan.bssid eq 00:11:22:33:44:55`というフィルタを作成します。これで、特定のアクセスポイントを介して送受信されるパケットのみがキャプチャされるようになります。

10.8.2 無線LANのタイプでフィルタリング

この章の最初で、無線LANのパケットにはいくつかのタイプがあるということを述べました。時折、これらのタイプやサブタイプによってパケットをフィルタリングする必要があります。表10-1はタイプ別のフィルター一覧です。

10.8.3 特定のデータタイプでフィルタリング

無線LANのManagementパケットはあるタイプの解析では非常に重要ですが、単

表 10-1 無線LANのタイプ/サブタイプ別フィルタ一覧

タイプ/サブタイプ	フィルタ構文
マネージメントフレーム	wlan.fc.type eq 0
コントロールフレーム	wlan.fc.type eq 1
データフレーム	wlan.fc.type eq 2
アソシエーションリクエスト (Association request)	wlan.fc.type_subtype eq 0
アソシエーションレスポンス (Association response)	wlan.fc.type_subtype eq 1
リアソシエーションリクエスト (Reassociation request)	wlan.fc.type_subtype eq 2
リアソシエーションレスポンス (Reassociation response)	wlan.fc.type_subtype eq 3
プローブリクエスト (Probe request)	wlan.fc.type_subtype eq 4
プローブレスポンス (Probe response)	wlan.fc.type_subtype eq 5
ビーコン	wlan.fc.type_subtype eq 8
ディスアソシエート (Disassociate)	wlan.fc.type_subtype eq 10
オーセンティケーション (Authentication)	wlan.fc.type_subtype eq 11
デオーセンティケーション (Deauthentication)	wlan.fc.type_subtype eq 12
アクションフレーム (Action frames)	wlan.fc.type_subtype eq 13
ブロック ACK リクエスト (Block ACK requests)	wlan.fc.type_subtype eq 24
ブロック ACK (Block ACK)	wlan.fc.type_subtype eq 25
PS-Poll (Power save poll)	wlan.fc.type_subtype eq 26
RTS (Request to send)	wlan.fc.type_subtype eq 27
CTS (Clear to send)	wlan.fc.type_subtype eq 28
ACK	wlan.fc.type_subtype eq 29
CF-End (Contention free period end)	wlan.fc.type_subtype eq 30
NULLデータ (NULL data)	wlan.fc.type_subtype eq 36
QoSデータ (QoS data)	wlan.fc.type_subtype eq 40
Null QoSデータ (Null QoS data)	wlan.fc.type_subtype eq 44

に空中を飛び交うデータが必要なだけということもあります。たとえば不正に無線LANにアクセスしているクライアントを見つけ出したり、公開したくない情報が流れてしまっていないかを確かめたりしたい場合などです。こういったことを実現するには、データパケットだけをフィルタする必要があります。

キャプチャファイルのデータパケットのみを表示するには、wlan.fc.type eq 2というフィルタを用います (表10-1ではタイプ2はデータフレームに関係するすべてのデータのことを指します)。

このフィルタを使用することでデメリットがあるとすれば、NULLデータパケットも表示されるということです。NULLデータパケットは、あるアクセスポイントとクライアントがチャンネルを切り替えようとしていることをネットワークに警告するために使用されるものです。NULLデータパケットが必要なければ、先に作ったフィルタにもう一工夫してNULLパケットのサブタイプもフィルタリングしましょう。フィルタは以下のようになります。

```
(wlan.fc.type eq 2) and !(wlan.fc.type_subtype eq 36)
```

暗号化されたデータと暗号化されていないデータを区別するのは、不正なアクセスポイントの特定や機密情報が平文で送信されていないかの確認のためのよい方法です。

「10.6.1 802.11のフラグ」に掲載されている Protected flag は、パケットが暗号化されているかどうかを示すフラグです。このフラグを使ってフィルタを作りましょう。

Protected flag が0の場合はそのパケットは暗号化されておらず、1の場合はWEP、WPA、TKIPなどを使って暗号化されています。したがって、以下のようなフィルタを使えば、暗号化されていないパケットのみを表示することができます。

```
wlan.fc.protected eq 0
```

同じように、以下のようなフィルタを使えば、暗号化されたパケットのみを表示することができます。

```
wlan.fc.protected eq 1
```

無線LANのためのフィルタを作成する方法は何百とあります。キャプチャフィルタのサンプルは<http://wiki.wireshark.org>のWireshark Wikiを参照ください。

10.9 無線LANに接続できない

SuccessfulWepAuth.pcap
FailedWepAuth.pcap

それでは、無線LANでのパケット解析の実際のシナリオを見てみましょう。ジャスティンはオフィスで無線LANにアクセスするために、ノートPCを設定しています。しかし残念ながら正しく機能していません。

10.9.1 分かっていること

ジャスティンが接続しようとしている無線LANは、チャンネル1でのWEPによる認証が必要です。無線LANクライアントをそのように設定すればいいだけで、実際彼はそう設定しましたが、接続できません。

10.9.2 パケット解析開始

今回の場合、有線LANでのパケットキャプチャと同じように考える必要があります。ジャスティンは無線LANの接続に失敗しているのです。接続しようとしたときにパケットをキャプチャします。AirPcapをチャンネル1に設定しましょう。

10.9.3 解析

無線LANのパケットを見たことがないと、無線LANで認証が成功した場合や接続の流れなどがどのようになっているのかわかりません。サンプルファイルを見て、無線LANに接続するときに流れを学びましょう。SuccessfulWepAuth.pcapを開いてくだ

さい。認証に成功するまでの流れが記録されています。

ジャスティンが接続しようとしている無線LANでは、WEP キーによる暗号化がなされています。WEP (Wired Equivalent Privacy) キーとは、16 進数または英数字のコードで、アクセスポイントとクライアント間の通信を暗号化するパスワードのようなものです。アクセスポイントに接続するには、クライアントはアクセスポイントとまずチャレンジレスポンス方式で WEP キーが正しいかどうかを確認しなければいけません。このチャレンジレスポンスはサンプルファイルの 4 番目のパケットから始まっています (図 10-12)。

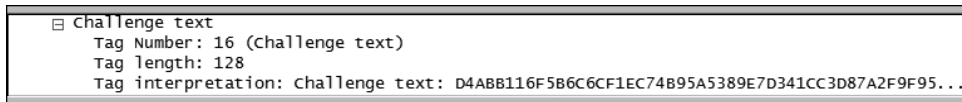


図 10-12 アクセスポイントからクライアントにチャレンジが送信されている

アクセスポイントはチャレンジを送り、接続の試みに応じます。このチャレンジは暗号化されたテキストで、WEP キーを使ってクライアントによって復号され、アクセスポイントに返されます。

6 番目のパケットでは、クライアントが復号したチャレンジを返しています (図 10-13)。8 番目のパケットでは、アクセスポイントは認証が成功したことをクライアントに

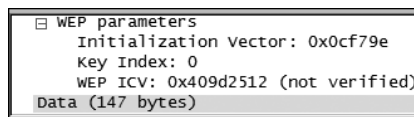


図 10-13 クライアントは復号したチャレンジをアクセスポイントに返す

伝えています (図 10-14)。

認証が成功すると、クライアントが接続要求を送信し、応答確認を受信して、つい

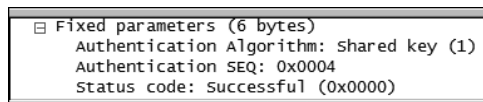


図 10-14 アクセスポイントがクライアントに認証が成功したことを伝えている

に無線LANに接続します (図 10-15)。

これでアクセスポイントとの接続がどういうふうに行われるか分かったので、サン

No. .	Time	Source	Destination	Protocol	Info
10	0.145465	GemtekTe_30:b0:af	Enterasy_6b:68:30	IEEE 8	Association Request, SN=44, FN=0, SSID: "DENVEROFFICE"
11	0.145839	GemtekTe_30:b0:af	GemtekTe_30:b0:af	IEEE 8	Acknowledgement
12	0.148466	Enterasy_6b:67:28	Broadcast	IEEE 8	Data, SN=1390, FN=0
13	0.149090	Enterasy_6b:68:30	GemtekTe_30:b0:af	IEEE 8	Association Response, SN=1391, FN=0
14	0.149464	Enterasy_6b:68:30	Enterasy_6b:68:30	IEEE 8	Acknowledgement

図 10-15 認証の処理はシンプルな接続要求と接続許可によって行われる

ブルファイル (FailedWepAuth.pcap) のジャスティンが接続を試みている部分を見てみましょう。3番目のパケットで、アクセスポイントがチャレンジをジャスティンのコンピュータに送信しています (図10-16)。ということで、とりあえずアクセスポイントとクライアントはパケットを送信しあうことはできるということが分かります。

5番目のパケットでクライアントがアクセスポイントに復号したチャレンジを返し

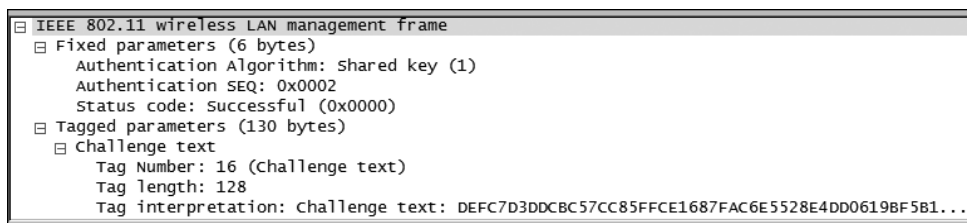


図10-16 アクセスポイントからジャスティンのコンピュータにチャレンジを送っている

ています (図10-17)。

本来はここで、アクセスポイントから認証が成功したという返事をもらうはずで

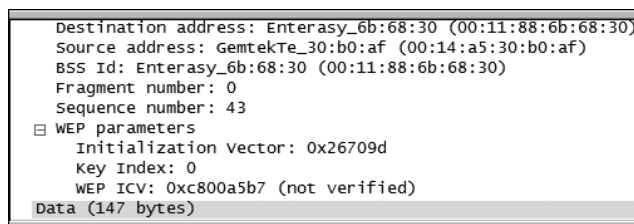


図10-17 ジャスティンのコンピュータがアクセスポイントにチャレンジを返している

が、そのかわりに認証が失敗したという返事が返ってきています (図10-18)。

アクセスポイントから送られてくるメッセージには何が起っているのかざり書

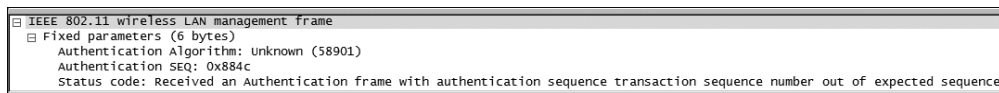


図10-18 どうやら認証に失敗したらしい

いてあります。シーケンス番号がおかしいのです。これは、ジャスティンのコンピュータが送ってきたチャレンジが間違っているということです。したがって復号に使われたWEPキーが間違っているということになります。

10.9.4 まとめ

無線LANのトラブルシューティングの悲しいところは、クライアントソフトウェアは問題に関するメッセージをあまり出さないということです。クライアントはただ接続するか失敗するだけです。無線LANでのパケット解析の技術があれば、トラブルシューティングをより効率的に行うことができます。

10.10 考察

無線LANは企業の重要なインフラになりつつあります。無線LANへの移行とともに、われわれも有線LANと同様のトラブルシューティングができるようにならなければいけません。この章で学んだ技術と概念は、無線LANでのパケット解析によるトラブルシューティングの細部を理解する助けになるでしょう。

11 章

推薦文献

パケット解析はWiresharkだけで事足りる場合が多いですが、他のツールやWebサイトも役に立つかもしれません。

Cain & Abel

Cain & Abel (<http://www.oxid.it>) は、2章で説明したARPキャッシュポイズニングを実行するためのツールです。ARPキャッシュポイズニングだけでなく、Cain & Abelにはパスワードスニッファやパスワードの復元、VoIPの盗聴、ネットワークのさまざまな情報の奪取など、ほかにもたくさんの機能があります。

PingPlotter

このプログラムはICMPのpingの拡張機能を提供します。ネットワークの接続状況をより容易に解析できるように、pingの結果をテキストやグラフで出力します。長期間の解析を行うときにこの機能が役に立つでしょう。PingPlotterは<http://www.pingplotter.com/download.html>からダウンロード可能です。

Superscan 4

Superscan 4は簡単なネットワークスキャナです。特筆すべきなのはスキャンの速さです。早急にネットワークの情報が欲しい場合は、Superscanを使うのが効率的です。ホストやネットワークの情報を集めたいときに、このツールがさまざまな形で役に立つでしょう。Superscanは<http://www.foundstone.com/resources/proddesc/superscan.htm>からダウンロード可能です。

RUMINT

RUMINT (ルーミント) はキャプチャしたパケットのデータをビジュアル化するツールで、フリーで配布されています。詳細なグラフとビジュアル化のオプションによって、キャプチャしたパケットをより理解しやすくします。詳しい情報は<http://www.rumint.org>を参照ください。

Engage Packet Builder

Engage Securityが提供するEngage Packet Builder (<http://www.engagesecurity.com/products/engagepacketbuilder>) を使えば、パケットをカスタマイズして送

信することができます(図11-1)。ファイアウォールや侵入検知システムのテスト、Flood攻撃に影響されやすいデバイスの検知、または単なる教育目的でカスタマイズされたパケットが必要になるときにはこのツールを使えばよいでしょう。Engage Packet Builderを使えば、さまざまなオプションを設定したパケットを生成することができます[†]。また、パケット生成を自動化するスクリプトを使用すること可能です。

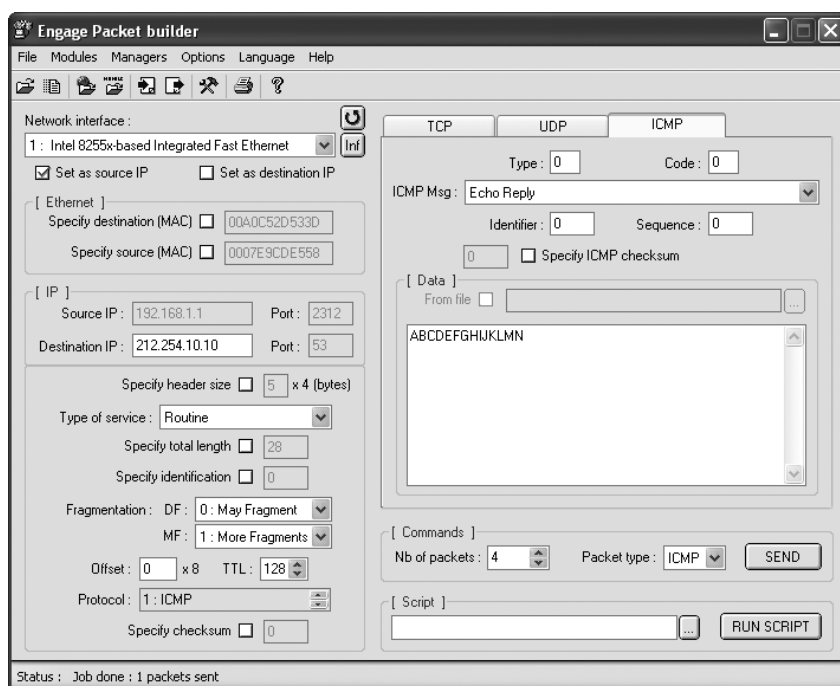


図11-1 Engage SecurityのEngage Packet Builder

IANA

IANA (Internet Assigned Number Authority) はIPアドレスとプロトコル番号を管理している組織です。IANAのWebサイト (<http://www.iana.org>) には、ポート番号の検索やトップレベルドメインに関する情報、RFCの検索や閲覧ができるサイトの一覧など、貴重な情報が掲載されています。

Wireshark Wikiとメーリングリスト

Wireshark (<http://www.wireshark.org>) はコミュニティベースのプロジェクトなので、主なサポートはWireshark Wikiとメーリングリストによって行われてい

[†] 監訳注: 同様なツールにhping3 (<http://wiki.hping.org>) もあります。

ます。

Wireshark University

Wireshark University (<http://www.wiresharktraining.com>) は Wireshark や パケット解析のコミュニティの主要な参加者によって 2007 年 3 月に設立されました。設立者には Gerald Combs (Wireshark の作者)、Laura Chappell (Packet Analysis Institute の優秀なプロトコル解析者)、John Bruno (CACE technologies の共同創立者)、Loris Degioanni (WinPcap の作者) がいます。

Wireshark University は初めての Wireshark のトレーニングサイトです。自分のペースでできるトレーニング用ビデオに加え、Wireshark の認証プログラムも提供しています。

あとがき

あなたが本書に書かれていることをすべて習得できますように。

パケット解析は薬に似ています。科学と芸術両方を兼ね備えているのです。ネットワークは患者であり、あなたは医者なのです。医者が人体解剖学や薬を生み出した科学に関する知識があるように、ネットワーク管理者もネットワークアーキテクチャやネットワークを生み出すプロトコルに関する知識があるのです。加えて、机上の理論をどれほど理解していようとも、実際に経験しなければ最良の結果を生み出すことはできません。複雑な病気にかかったときに経験豊かな医者に治療をお願いしたくなるのと同じです。

本書の主な目的は、ネットワーク管理のために習得する必要があるツールと概念を紹介することです。パケット解析をやればやるほど、実際の経験を積み重ねるほど、複雑なネットワークの問題でも効率的に解決することができるようになります。さまざまなネットワーク上で（もちろん許可を得て）、Wiresharkを使ってパケットレベルで通信を解析してみてください。それが、ネットワークを調べることとはどういうことかを学び、ネットワーク上で何が起きているかを正確に見抜くための唯一の手段です。これこそが「実践 パケット解析」です。

付録

Winnyやボットのパケット解析

本書の原著では扱われていない重要な分野（であり、ネットワーク管理者にとっていちばん気になるトピック）に、Winnyやボット関連のトラブルシューティングがあります。本書日本語版オリジナルのこの付録では、このWinnyやボットのパケット解析について解説します。本書の後半の章ではサンプルのキャプチャファイルを提示しながら解説してきましたが、この付録では、サンプルのパケットキャプチャファイルを用意していません。というのも、Winny通信の場合は通信相手が個人ですし、ボット通信の場合はボットやボットがダウンロードしてきたモジュール本体が含まれてしまうためです。なお、ボットの通信例に関しては、監訳者である園田がリーダーを務める、日本ネットワークセキュリティ協会（JNSA）のハニーボットワーキンググループの協力で採取しました。ご協力いただいたメンバーの皆さんに感謝いたします。

A.1 その1：Winnyパケットを追え

どんな企業や組織においても、Winnyに代表されるP2Pによるファイル共有ネットワークの「問題」は頭の痛いところです。P2P（Peer To Peer：ピアツーピア）という通信形態は特定のサーバとの通信ではなく、ネットワークにつながる各コンピュータがそれぞれ通信を行うため、ファイアウォールなどによる通信制限を行いつらいともいえます。通信制限を行いつらいということは、その種の通信を「検出」することも簡単ではないということでもあります。

ここでは実際にWinnyが行う通信を題材として、その特徴による通信の見分け方、解析方法などを見ていきましょう。

ネットワークを飛び交う通信は多岐にわたります。ルーティングのために必要な通信や、各種のブロードキャスト（存在を知らしめるための通信）など、実際にネットワークをモニターしてみると、想像以上に多くの通信が行われていることに驚くことでしょう。

パケットキャプチャを武器にその中から特定の通信をあぶり出すには、まずはその通信の特徴を知る必要があります。

A.1.1 P2P 通信 Winny

Winny の通信の特徴としてはまず、P2P という通信形態なので「通信する相手が多いこと」があげられます。LAN につながったと、あるコンピュータがやたらと多数の相手と通信を行っているというだけでも、Winny もしくは他の P2P 通信を行っている可能性は高いといえるでしょう。また、Winny は使用する通信ポート番号を自由に設定できるため、相手が不特定多数で、しかも通信ポート番号が一定ではないとすると、可能性はさらに高くなるわけです。

ネットワークの構成と通信要件次第では、その疑いを確信に変えることもできます。たとえば、LAN から出て行く通信はすべて DMZ (非武装中立地帯) 経由になるようなネットワークでは、そもそも直接インターネットに出て行く通信があることだけで怪しいといえます。ネットワークを管理する立場でいえば、直接外部に出る通信などの不確定な要素を少なくすればするほど、異常を検出しやすいわけです。

Winny には、ファイアウォールの内部にいる場合の通信を想定した Port0 (ゼロ) という機能があります。これは少々厄介です。というのも、この機能は通信先を特定の相手だけに絞り込む傾向が強くなるものだからです (図 A-1)。

Winny の各ノード (Winny が動くコンピュータ、もしくは Winny そのもののことをこう呼びます) は、クライアントでもありサーバでもあるわけですが、サーバの機

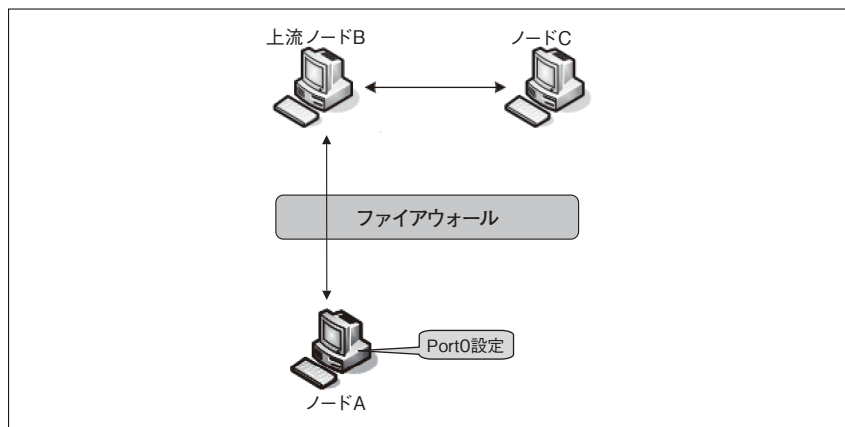


図 A-1 Winny の Port0 通信。ファイアウォールによって外部からの通信を自由に受信できないノード A は Port0 機能を使って上流のノード B にまず接続し、ノード B を代理サーバにしてサーバとしての役割を果たす

能を果たそうとすると、直接外部から LAN にアクセスさせないファイアウォールの存在が邪魔になるわけです。

そこで、まず特定の相手と LAN → 外部という形で通信を行って、その相手を経由してサーバの役割を果たす機能が実装されたのです。この Port0 機能は、サーバとしての役割を果たすだけでなく、結果として特定の相手とのみ通信を行うという形で、不特定多数と通信を行うという特徴を弱めることにもなってしまいます。

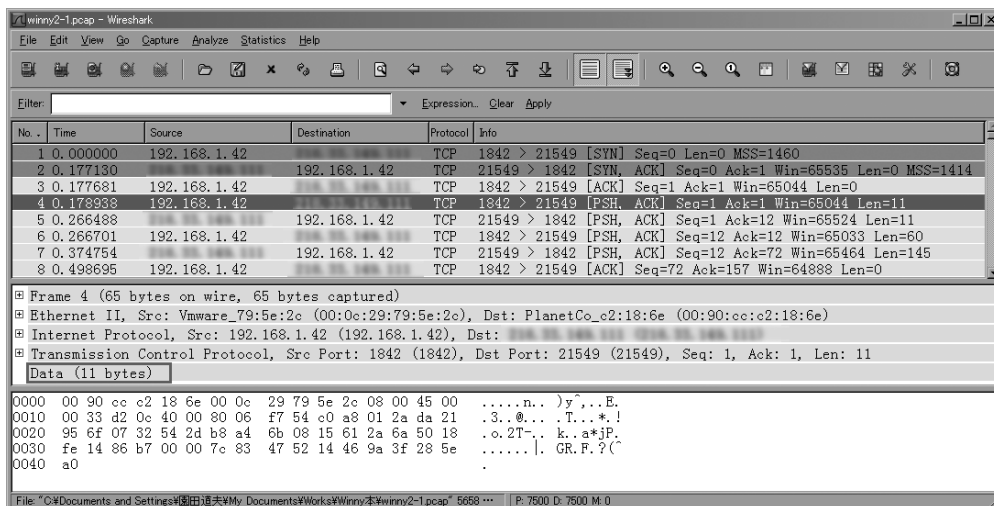
ただ、Port0機能はそのWinnyが「サーバ」として動作するときのものであり、もっぱらダウンロード専用で使用しているのならば、相変わらず不特定多数と通信するはずです。

「不特定多数」との通信がまったくありえないようなネットワークでは、それだけでも十分怪しむに足る材料ですが、他のP2Pソフトウェア（たとえばSkypeなど）の通信を許可していたりすると、そもそもそういう特徴だけでは特定できなくなってしまいます。そうするとやはり、パケットキャプチャなどでさらに「深掘り」する必要があります。許可されていないかつたとしても、パケットキャプチャでさらに証拠を固めておきたいところです。

A.1.2 Winny通信の解析

Winnyの通信は実際にはどう見えるでしょうか。図A-2はWinnyの通信です。

まずTCPの通信セッションを確立し（3ウェイハンドシェイク：SYN、SYN/ACK、ACK）、その後PSH ACKでデータのやり取りを開始しています。このPSH ACKと



図A-2 Winnyパケットのキャプチャデータ。データの長さが11バイトであることが分かる

いうパケットの中身はWinnyプロトコルでのやり取りです。Winnyの場合、この最初のパケットの長さの特徴があります。パケット一覧画面の右端にデータの長さが表示（Len=11）されていますが、Winnyが通信を開始するときに投げるパケットは、長さは必ず11バイトです。Winnyはまず、この11バイトのパケットをお互いにやり取りして通信を開始するわけですが、通信の中身は暗号化されていますので、そのままでは読めません。試しに「Follow TCP Stream」機能を使ってやり取りを表示させてみても、何をやり取りしているのかさっぱり分かりません（図A-3）。

ただ、Winnyはこの暗号通信の仕組みが脆弱なので、通信の中身を突き止めること

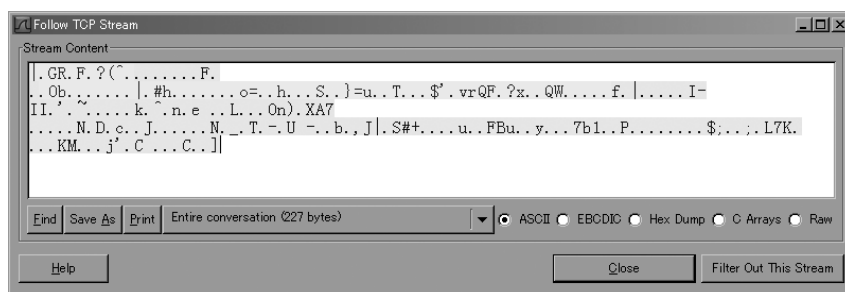


図 A-3 「Follow TCP Stream」ASCII モード。Winny 通信の場合は「Follow TCP Stream」機能を ASCII モードで用いても中身がよく分からない

は比較的簡単にできます。Winny 通信の暗号化は、RC4 というストリーム暗号形式で行われています。暗号通信の場合、まず当事者同士で暗号化に使う「鍵」のやり取りを行う必要があるわけですが、Winny はこの最初の packets で鍵自体を交換しています。つまり、最初の packets をキャプチャ（捕捉）できれば、その後の通信の暗号を解けてしまうのです。設計者によればこれは通信のパフォーマンスを重視したためだということですが、確かに Winny は通信セッションを多数確立しながら通信を行うので、その都度安全で複雑な鍵の交換を行っていたらパフォーマンスに影響が出てきてしまいそうです。逆にいえば、パフォーマンスのために通信の「安全性」を犠牲にしたのが Winny の通信なのです。

11 バイトの内訳ですが、最初の 2 バイトにはダミーの乱数、次に来る 4 バイトには暗号通信用の鍵、残る 5 バイトには情報と命令（コマンド）が格納されています（図 A-4）。

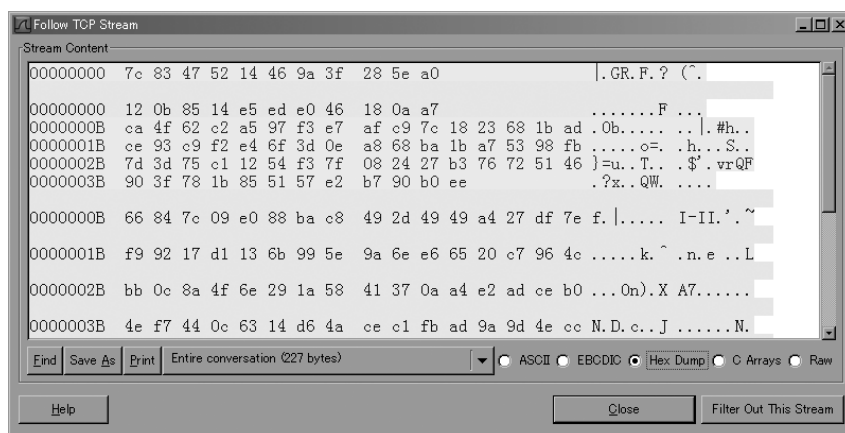
鍵が毎回異なるため、第 1 packets の「見た目」はいつも違って見えます。もう一度「Follow TCP Stream」機能を用いてデータを表示させてみましょう。今度は [HEX Dump] を選択して、16 進表示させてみます。

最初の 2 列に注目してください。両方ともデータ長は 11 バイトですが、中身はまったく別物です。データが暗号化されているため異なっているのですが、ももとの情報はまったく同じものです。具体的にはその中身は「01 00 00 00 61」となり、これは Winny のバージョンをお互いに確認するための情報です。したがって、packets の中身に含まれる鍵を用いてうしろ 5 バイトを復号し、その結果が「01 00 00 00 61」になればその通信は Winny のものであると断定できます[†]。

ちなみにこのデータのうち 01 はコマンドの長さが 1 バイトであることを示し、16 進数の 61 は 10 進数では 97 で、これがコマンドになります。

しかし、データを復号するまでもなく、たとえば LAN 内のどこかにあるコンピュータが長さ 11 バイトのデータを見慣れない通信ポート宛に発信し、その相手からも 11 バイトのデータが戻ってきたとしたら、その通信は Winny であると見て間違いはな

[†] もちろん、偶然まったく同じバイト長、まったく同じ内容のデータがやり取りされる可能性はゼロではありません。しかし、その確率はかなり低いといえます。



図A-4 Winny第1パケットの構造。「Follow TCP Stream」機能を「Hex Dump」で使えば実データが見えやすくなる。暗号解析などはASCIIやEBCDICなどで変換されないデータを見ることが重要になる

いでしょう。そこまで材料があれば、あとはその通信を行ったパソコンを突き止めて、そこでWinnyが動いているかどうかを確認すればよいということになるわけです。

A.1.3 まとめ

1つ1つの通信をキャプチャして解析すること、その考え方を学ぶことがこの本の目的ですが、実際の調査や解析においては、なんらかの徴候、ヒント、材料、情報を元に、ネットワークを流れる大量の通信の中から怪しげなものを抽出しなければなりません。となると、調査の初動において、絞り込みを行うことが重要になってきます。

漠然とWinnyユーザーを追いかけるよりも、ファイアウォールなどで通信要件を整理し、たとえばそもそもLANからインターネットに直接通信することがないようにしておけば、絞り込みはぐっと容易になるわけです(図A-5)。

絞り込みさえ適切に行えたら、あとは事象とそれに現れる特徴を適切に把握するだけです。そのためにパケットキャプチャデータは非常に役立つでしょう。

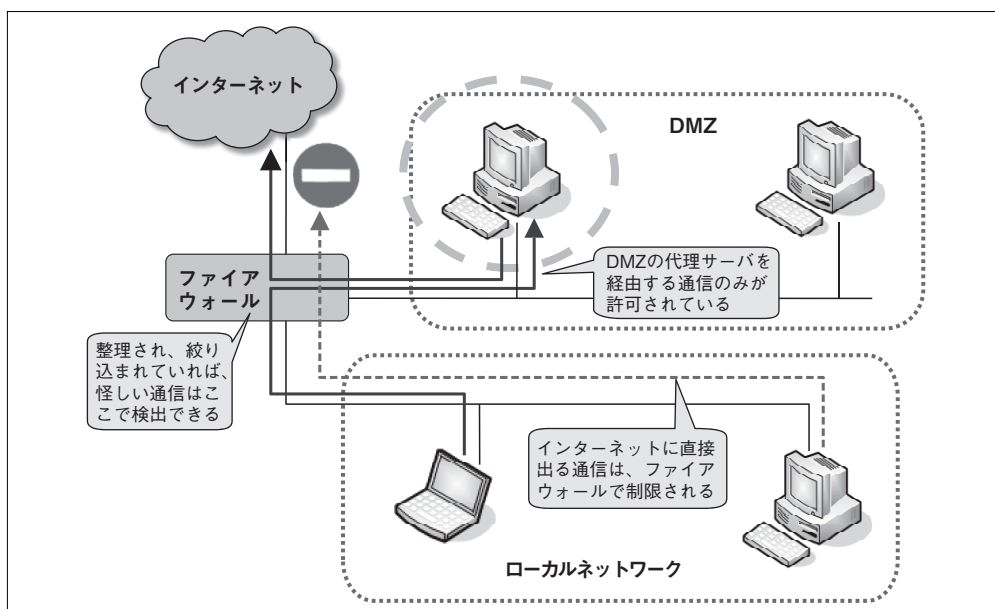


図 A-5 Winny はどのような設定であっても必ず他のノードと「直接通信」する。ファイアウォールの内側から外への直接通信が許可されていないか、あるいは記録されていれば、ファイアウォールのところで検出できる

A.2 その2：ウイルス、ワーム、ボットの追跡

A.2.1 ウイルス、ワームの変質

ウイルス対策は以前より万全とはいえなくなりつつあります。これまではウイルス対策ソフトウェアやファイアウォールなどがあれば、まずウイルスに感染することはなかったのですが、要であるウイルス対策ソフトウェアのチェックをくぐり抜けるようなものが出現し始めています。ウイルスを作る側の多くがビジネス志向となり、結果としてウイルスの開発環境やツール、サポートなどが充実してきていますので、特別な知識がなくても十分実効性の高いウイルスを作ることができるようになってきているからです。

さらに、金儲けを目的としたボットやウイルスは、以前のように自己顕示して世間を騒がせて喜ぶというよりも、むしろ長く静かに潜行して情報を盗み出したり、悪者の手下となって迷惑メールを送信したりするわけです。コンピュータの性能が向上し、見て分かるような変化（動きが重くなるとか、何もやらせていないつもりなのに CPU 利用率が上がり機体が熱くなるとか）が起これにくくなってきたこともあり、ウイルスを検出することは非常に難しいこととなりつつあります。

そしてここでも、怪しい通信を捕捉してそのやり取りを調査するということが重要になってくるのです。

A.2.2 ボット(踏み台)の特徴

ボットというのは、ネットワークに組み込まれて悪者の手下となるプログラムです。今のところはクライアントサーバ型の通信を行うところが多いようですが、WinnyのようにP2Pという形態で通信を行うものも出てきています。

主な目的はコンピュータを悪用することですが、そのためには命令を受信する必要がありますし、迷惑メールや個人情報を送信する必要があるわけです[†]。そうなるとWinnyと同様に、ネットワーク要件を整理し、直接外部に通信する行為を「異常」とあるとすることで、検出しやすくする方策や、通信の特徴を把握して捕捉する方法などが使えそうです。

そもそもボットとはIRC (Internet Relay Chat) 用語で「人の代理で言われたことをやる」みたいな意味だったりします。このことから分かるように、ボットは初期よりIRCプロトコルをベースにした通信を行ってきました。単純なボットであればIRC通信があるかどうか、IRCのポート番号6667を目指す通信が発生しているかどうかを見ればよいのですが、使用するポート番号を変えることはそれほど難しいことではありません。したがって、検出、調査する際にポート番号だけを頼りにするというのは少々心許ないといえるでしょう。

ボットの通信の特徴はもう1つあります。それは、踏み台となったボットの場合、迷惑メール送信などの既知のプロトコルによる通信を行うということです。直接外部に通信を行うという特徴に加えて、ごく普通のSMTP通信を発信しようとしているという特徴があれば、特に中身を解析しなくても「おかしい」というフラグを立ててマークすることができるでしょう。

メール以外のこの手のソフトウェアによる通信には、攻撃的なアクセスがあります。ワームにしろボットにしろ、自己複製を行って感染を拡大しようとする傾向があるわけです。ワームやボットはマイクロソフトネットワークのファイル共有や、いわゆるネットワーク攻撃によって拡大を狙います。

その際にもう1つ傾向として言えるのは、以前のように騒がしく攻撃を行うものばかりではないということです。たとえばSQLスラマーという有名なワームは、侵入した先のネットワークがダウンするほどの大騒ぎを引き起こしました。爆発的に攻撃パケットをまき散らしてSQLサーバが起動していそうなコンピュータを狙いまくったため、ファイアウォールがそのパケットをさばききれずにダウンしてしまったりしたのです。一時的には大きな被害になりますが、それも極論すれば1日程度仕事が停滞するだけにすぎませんでした。

現在の悪者の目的は完全にビジネスにシフトしてきています。つまり、長く静かに悪用して金儲けに利用するというわけです。そして、このビジネスへのシフトという動きが、ワームやボットの「完成度」を上げてきているのです。以前のように脆弱性検証のためのコンセプトコードをそのまま利用するなどの「乱暴な」作り方は影を潜

[†] 逆に言えば、通信を行わないウイルスというのは、それほど大きな危険はないといえるでしょう。

め、効率よく静かに悪用するようになってきつつあるということです。

これはつまり、拡散も静かに行う傾向があるということでもあります。

攻撃的な通信で攻略するにしても、数個の packets をばらばらと投げて相手の素性を探って攻撃につなげようとし、ファイル共有を伝って感染するときも、ごく普通のマイクロソフトネットワークのブロードキャストを装って感染先を探ったりします。

さらに言えば、ネットワークを使って感染するのではなく、犠牲者に自らインストールさせるといった攻略法も進化してきています。少量とはいえ怪しい packets を投げてネットワーク管理者などに見つかるリスクをとるよりも、確実に見つかることなく感染することができそうです。

つまり、ここで言いたいのは、感染しようとするときの通信を捕捉してワームやボットを捉えようとするのは今後ますます困難になるであろうということです。それほど厳しくセキュリティを設定していないコンピュータが発する無作為の packets が多数飛び交うようなネットワークでは（実際に packets キャプチャを行ってみれば、おそらくみなさんの想像よりもはるかに多くの packets を見ることができるでしょう）、少量の packets に反応することが難しいわけです。

もちろんあきらめる必要はありませんが、いわゆるトラディショナルな「ネットワーク攻撃」の通信を捕捉するだけでなく、サイト全体としてありえない通信というのを元に絞り込む必要もあるということになるわけです。

A.2.3 「怪しい通信」の解析

ネットワーク攻撃やありえない通信を捕捉できたら、今度はそれを解析します。

まずはその得体の知れない通信の素性をざっくり把握することを考えます。その通信が既知の「攻撃」に該当するかどうか、Snort (<http://www.snort.gr.jp>) というフリーの IDS (Intrusion Detection System : 侵入検知システム) を用いれば簡単に素性を知ることができます。

Wireshark でキャプチャしたデータを保存するとき、何も考えなければ pcap 形式で保存されます。pcap 形式は packets キャプチャデータ形式のデファクトスタンダードともいえる形式ですので、どの OS やソフトウェアでもまず読めないことはありません。この形式で保存したデータをそのまま Snort に読み込ませれば、そのデータがいわゆるネットワーク攻撃に該当するかどうか判明します[†]。Snort の -r オプションを用いれば、ファイルを読み込ませることができます。

ほかには、ウイルス対策ソフトウェアのスキャンを用いて素性を確認する方法があ

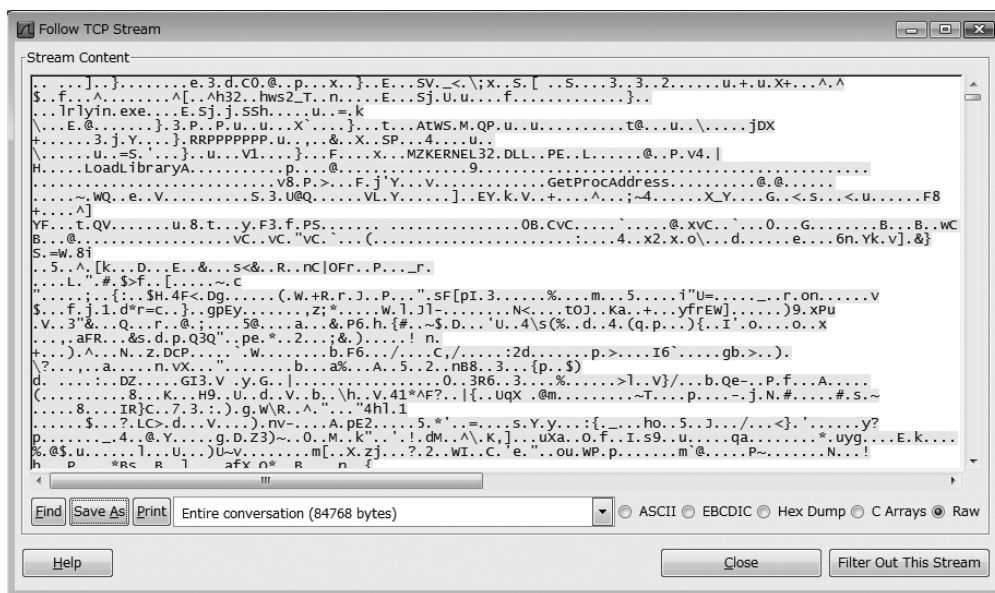
[†] Snort のインストール方法については、Snort の開発者の方針がよく変わるのでここでは触れません。その都度、インターネット上の文書などを参考にしてください。基本的には rpm 形式でもソースファイルからのインストールでも、どちらでも可能ですし、Windows OS にインストールすることも可能です。ルールファイルも開発サイトでユーザー登録すれば使えるルールや、ユーザーコミュニティが作り上げた Bleeding Edge ルールセット (<http://www.bleedingthreats.net>) を使う手もあります。

ります。キャプチャしたデータファイルをそのままウイルス対策ソフトウェアにチェックさせると、通信の中にウイルスやワーム、ボットなどが含まれていたら反応する場合もあるでしょう。オンラインチェックなどを併用すれば、別なベンダの対策ソフトウェアを使ってチェックすることもできます。

キャプチャデータの中に含まれる怪しいファイルを、元通りにファイルとして保存することもできます。

データのファイル化は以下の手順で行います。

1. まず、ファイル化したいデータを含む通信を特定します。「Follow TCP Stream」機能を用いれば簡単です(図A-6)。



図A-6 [Save As] を用いてRaw形式でファイルに保存すると、送受信されているデータの中身のみを保存することができる

2. 次にストリームデータをRaw形式で保存([Save As])します。ここでは仮にccc.exeというファイル名で保存しておきます。
3. このファイルをバイナリエディタで開きます。ここではStirling (<http://www.vector.co.jp/soft/win95/util/se079072.html>) というバイナリエディタを用います。Windows OSで動作するソフトウェアは、その中身が一般的に「MZ」、あるいは「MZP」という文字列で始まります。
4. 保存したストリームデータの中身にあるMZを探し、その直前までのデータを削除します(図A-7)。削除したら別名で保存、もしくは上書き保存します。
5. これでファイルが復元できたことになります。試しに中身をpedumpというソフ

トウェアで見てみると、実行できる形式のファイルであることが分かります (図 A-8)。

ファイル化すれば、ウイルス対策ソフトウェアももっと精度高くチェックしてくれるでしょう。

攻撃とは異なる通信を解析する場合、踏み台となって迷惑メールを出しまくって

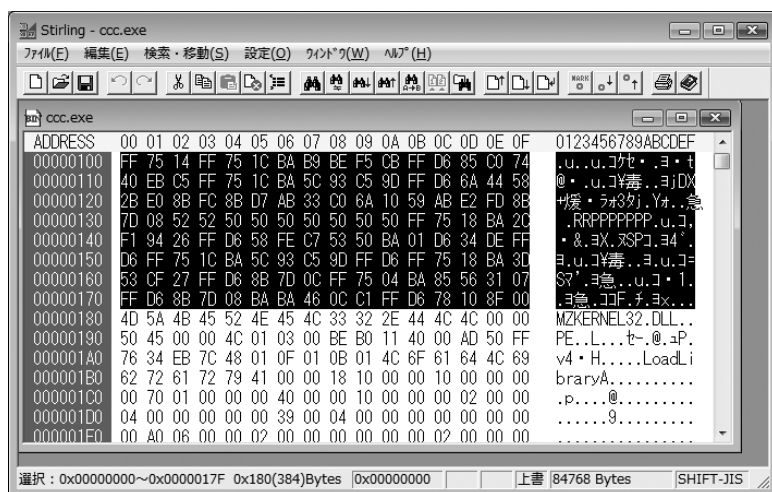


図 A-7 バイナリエディタで保存したファイルを直接開き、MZ もしくは MZP よりも前のデータを削除する。これでファイルを復元することができる

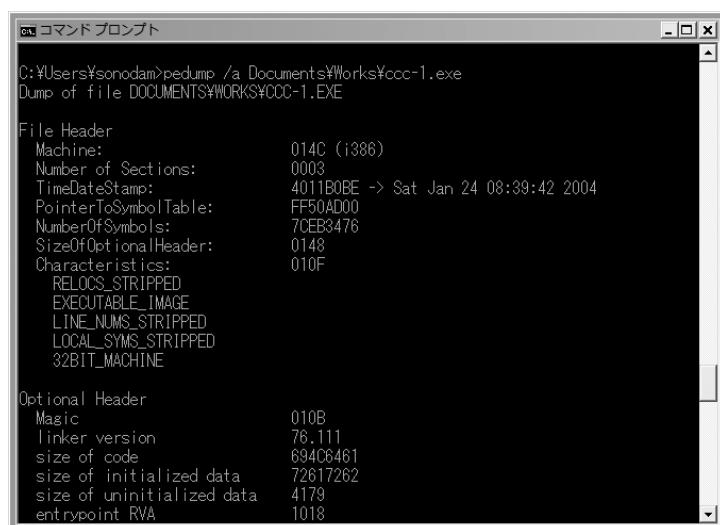


図 A-8 pedump というツールを用いれば、復元したファイルの中身を確認することができる。デバッガなどを用いることもできるが、誤って起動させないよう慎重に取り扱う必要がある

る場合などはむしろ簡単です。その場合はSMTPプロトコルそのものを用いているので、キャプチャデータを見れば何をやっているかはすぐ分かります (図 A-9)。

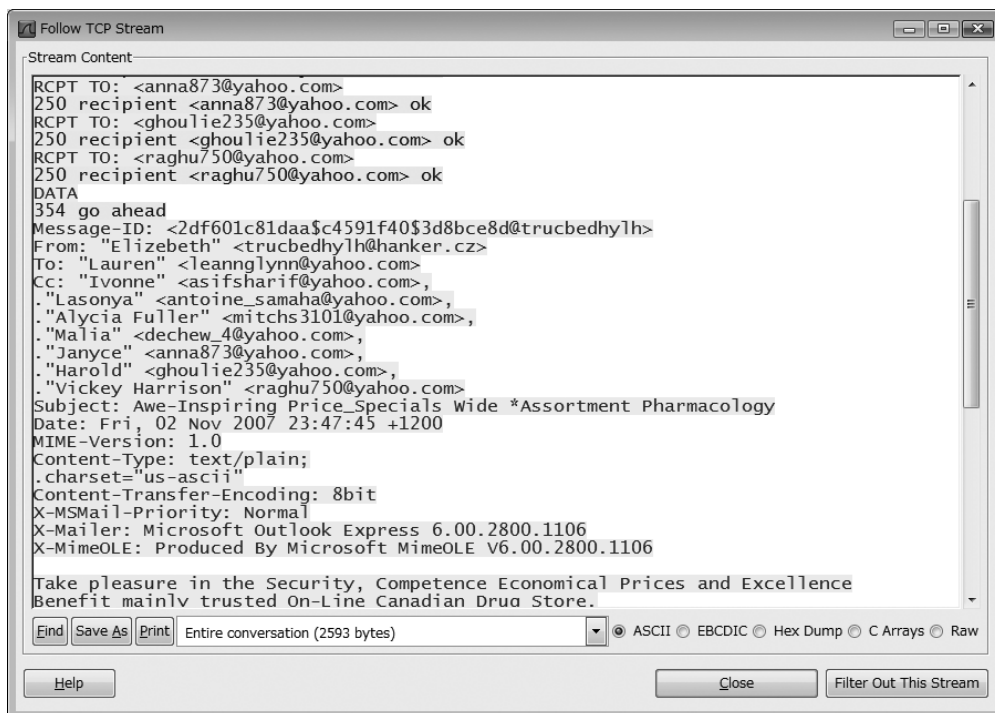


図 A-9 迷惑メールを多数発信している。ヘッダもそれらしく偽装していることが分かる

そのほかに可能性があるのは、踏み台となったソフトウェアが命令を受信する通信です。この通信は最近暗号化されていることも多く、中身を見ることは難しいかもしれません。キャプチャした通信内容のサンプルを以下に示します (図 A-10)。

```
USER qwtemi qwtemi qwtemi :cajomaqonncvizrk
NICK yJxGcfaF
:hub.61171.com 001 yJxGcfaF :education, yJxGcfaF!qwtemi@221.186.11.201
:hub.61171.com 005 yJxGcfaF MAP KNOCK SAFELIST HCN MAXCHANNELS=80 MAXBANS=60
NICKLEN=30 TOPICLEN=307 KICKLEN=307 MAXTARGETS=15 AWAYLEN=307 :are supported
by this server
:hub.61171.com 005 yJxGcfaF WALLCHOPS WATCH=128 SILENCE=15 MODES=12
CHANTYPES=# PREFIX=(qaoHV)?&%+ CHANMODES=be,kfL,l,psmntirRcoAQKVGcuzNSMT
NETWORK=education CASEMAPPING=ascii EXTBAN=?,cqr :are supported by this server
:yJxGcfaF MODE yJxGcfaF :+iRp
MODE yJxGcfaF +xi
JOIN #kok2
:yJxGcfaF!qwtemi@221.186.11.201 JOIN :#kok2
```



```
:hub.61171.com 482 yJxGcfaF #prox :You're not channel operator
MODE #63 +smntu
:hub.61171.com 482 yJxGcfaF #63 :You're not channel operator
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
PRIVMSG #kok2 :-.04.dcom2....04.c.- 1. Raw transfer to 221.186.132.45
complete.
:hub.61171.com 404 yJxGcfaF #kok2 :You need voice (+v) (#kok2)
:EH!Y@hoo.net PRIVMSG #kok2 :* ipscan s.s.s.s dcom2 -s
PRIVMSG #kok2 :-.04.dcom2....04.c.- 2. Raw transfer to 74.205.233.170
complete.
:hub.61171.com 404 yJxGcfaF #kok2 :You need voice (+v) (#kok2)
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
PRIVMSG #kok2 :-.04.dcom2....04.c.- 3. Raw transfer to 221.187.30.120
complete.
:hub.61171.com 404 yJxGcfaF #kok2 :You need voice (+v) (#kok2)
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
PING :hub.61171.com
PONG :hub.61171.com
:EH!Y@hoo.net PRIVMSG #kok2 :* ipscan s.s.s.s dcom2 -s
PING :hub.61171.com
PONG :hub.61171.com
PRIVMSG #kok2 :-.04.dcom2....04.c.- 4. Raw transfer to 221.187.250.118
complete.
:hub.61171.com 404 yJxGcfaF #kok2 :You need voice (+v) (#kok2)
```

しかし、仮に中身を読むことができたなら素性はほぼ判明するでしょうし、もし暗号化されているようで中身が分からなかったとしても、そういう通信を外部と直接行っている、もしくは試みている時点で「怪しさ満点」であるといえそうです。

索引

記号		
!= (等しくない)	50	
< (小なり)	50	
<= (以下)	50	
== (等しい)	50	
> (大なり)	50	
>= (以上)	50	
3ウェイハンドシェイク	66, 161	
802.11	141	
～のパケット	144	
～のフラグ	145	
A		
ACKパケット	67, 161	
AirPcap	140	
and (論理積)	50	
AppleTalk	7	
ARP (Address Resolution Protocol)	7, 23, 63	
ARPキャッシュポイズニング	23, 133	
ARPスプーフィング	23	
ARPパケット	95	
ASCII	7	
B		
BitTorrent	113	
Blaster ワーム	129	
Bleeding Edge ルールセット	166	
BSSID (Basic Service Set Identifier)	148	
～でフィルタリング	148	
C		
CACE Technologies	140	
Cain & Abel	133, 155	
～の使用	24	
CALパケット	75	
CAMテーブル	11	
[Capture] セクション	36	
[Coloring Rules] ダイアログ	38	
[Conversations] ダイアログ	59, 98, 114, 118	
CWD コマンド	72	
D		
Destination unreachable (宛先到達不可能通知)	81	
DHCP (Dynamic Host Configuration Protocol)	64	
DHCPACK パケット	65	
DHCPDISCOVER パケット	64	
DHCPOFFER パケット	65	
DHCPREQUEST パケット	65	
[Display Filter] ダイアログ	51, 128	
DNS (Domain Name System)	70	
DoS 攻撃 (Denial of Service attack :		
サービス不能攻撃)	23	
Duplicate ACK パケット	103	
E		
Engage Packet Builder	155	
Ethereal	29	
Ethernet	7	
[Expert Infos] ウィンドウ	102	
F		
FDDI	7	
[Filter Expression] ダイアログ	48	
FIN パケット	68	
FIN/ACK パケット	68	

[Follow TCP Stream]	57, 93
Forbidden	92
FTP (File Transfer Protocol)	7, 71
FTPサーバ	71
～との通信	89
～への侵入	126

G

Gnutella	117
Gratuitous ARP	95

H

HTTP (Hypertext Transfer Protocol)	7, 65
～の通信	68, 86, 93
HTTP 403	92

I

IANA (Internet Assigned Number Authority)	156
ICMP (Internet Control Message Protocol)	7, 76
ICMP Time Exceeded (生存時間超過) パケット	107
ICMPコード	81
IDS (Intrusion Detection System : 侵入検知システム)	166
IM (Instant Messenger : インスタントメッセンジャー)	56
Internet Explorer	87, 113
[IO Graphs] ウィンドウ	61
IP (Internet Protocol)	7, 66
IPアドレス	23
～の名前解決	54
IPフラグメンテーション	82
IPX	7
iwconfig コマンド	143

J

JPEG	7
------	---

M

MACアドレス	23
～の名前解決	53
MIDI	7
MPEG	7
MSGパケット	76
MSN Messenger	56
MSNMS (MSN メッセンジャーサービス)	57, 74

N

[Name Resolution] セクション	37
NetBIOS	7, 55
～のトラフィック	86
NIC (Network Interface Card : ネットワークインターフェースカード)	18
not (否定)	50
NWLink	7

O

or (論理和)	50
OSのフィンガープリント	123
OSI参照モデル	5
Oxid.it	24

P

P2P (Peer To Peer : ピアツーピア)	159
P2P通信Winny	160
PDU (Protocol Data Unit : プロトコルデータユニット)	8
PIF (Program Information File)	117
ping	132
pingコマンド	77
pingパケット	81
PingPlotter	155
POP (Post Office Protocol)	116
[Printing] セクション	37
[Protocol Hierarchy Statistics] ウィンドウ	58
[Protocols] セクション	37

R

RETR コマンド	72
RFC (Request For Comments)	63
RFC 791	66
RFC 792	76
RFC 793	66
RFC 854	73
RFC 959	71
RFC 1034	70
RFC 2131	65
RFC 2616	65
RIP	7
RJ-45ポート	9
RTT (Round Trip Time : 往復遅延時間)	104
RUMINT	155

S

SAP	7
SDP	7
SIZE コマンド	72
SMTP	7
Snort	166
SPOOLS パケット	125
SPX	7
Superscan 4	155
SYN パケット	66, 161
SYN/ACK パケット	67, 161

T

TCP (Transmission Control Protocol)	7, 66
～の通信障害	79
[TCP Stream Graph]	104
TCP ストリームの表示	56
TCP/IP	66
tcpdump	2, 127
TELNET	7, 73, 133
Token Ring	7
traceroute	105
Transaction ID	65
TTL (Time To Live : パケットの生存期間)	106

U

UDP	7
[User Interface] セクション	36

W

Winny	159
～のパケット解析	159
Winny 通信の解析	161
Winny パケット	159
WinPcap パケットドライバ	31
Wireshark	29
～でのパケットキャプチャ	41
～のインストール	30
～の基本	33
～の設定画面	36
～の名前解決ツール	53
～の歴史	29
サポートされているプロトコル	29
パケットの色分け	37
Wireshark University	157
Wireshark Wiki とメーリングリスト	156

X

xor (排他的論理和)	50
--------------	----

あ行

圧縮	4
宛先到達不可能通知 (Destination unreachable)	81
アドホックモード	138
アプリケーション層 (レイヤ7)	5, 6
「怪しい通信」の解析	166
誤り検出	4
暗号化	4
以下 (<=)	50
以上 (>=)	50
印刷	125
パケットの～	45
インスタントメッセージャー (Instant Messenger : IM)	56
インフラストラクチャモード	138
ウイルスの追跡	164
エラー訂正	4
エンドポイント	58
応答確認	4
往復遅延時間 (Round Trip Time : RTT)	104

か行

解析	3
Winny 通信の～	161
「怪しい通信」の～	166
障害の～	85
ネットワークの～	34
隠された情報	131
カプセル化	8
キャプチャデータのエクスポート	43
キャプチャファイル	36
～のエクスポート	43
～の保存	43
～のマージ	44
キャプチャフィルタ	47
クラッキングツール	128
グラフ	61, 104
検索	41

さ行

サービス不能攻撃 (Denial of Service attack : DoS 攻撃)	23
最初のパケットキャプチャ	34
サブネットマスク	110

シェアードハブ	9
時間の表示フォーマット	45
辞書攻撃	128
収集	3
小なり (<)	50
侵入検知システム (Intrusion Detection System : IDS)	166
スイッチ	10
〜で構成されたネットワークでの スニッフィング	20
スイッチングハブ	10
スニッファ	1
〜を配置する	17
スニッフィング	3
スイッチで構成されたネットワークでの〜	20
ハブで構成されたネットワークでの〜	18
無線LANの〜	137
ルータで構成されたネットワーク上での〜	27
スパイウェア	94
生存時間超過 (ICMP Time Exceeded) パケット	107
セッション層 (レイヤ5)	5, 6
セッション	64
〜の確立	66, 161
〜の終了	68
接続不能	85
全二重モード	10, 22
相対時間表示	46

た行

大なり (>)	50
代表的なプロトコル	7
ダウンロードの遅延	101
通信障害	79
ディスプレイフィルタ	47, 128
データ	
〜の圧縮	4
〜の暗号化	4
〜のカプセル化	8
データ送信の開始	67
データリンク層 (レイヤ2)	5, 7
届かないICMPコード	81
届かないパケット	81
トラフィックの分類	14
トランスポート層 (レイヤ4)	5, 6

な行

名前解決	53
ネットワークインターフェースカード (Network Interface Card : NIC)	18
ネットワーク図	28
ネットワークセグメント	12
ネットワーク層 (レイヤ3)	5, 7
ネットワークハードウェア	9
ネットワークプロトコル	4

は行

排他的論理和 (xor)	50
パケット	9
〜の色分け	37
〜の印刷	45
〜の応答確認	4
〜の検索	41
〜のマーキング	42
802.11の〜	144
届かない〜	81
パケットの生存期間 (Time To Live : TTL)	106
パケット解析	1, 33, 79
Winnyの〜	159
ポットの〜	159
パケットキャプチャ	34
〜のテクニック	41
パケットスニッファの評価	2
ハッカーの視点	132
ハブ	9
〜で構成されたネットワークでの スニッフィング	18
半二重モード	10, 22
ピアツーピア (Peer To Peer : P2P)	159
ピーコンフレーム	146
比較演算子	50
否定 (not)	50
等しい (==)	50
等しくない (!=)	50
フィルタ	47, 128
〜のサンプル	51
〜の保存	51
〜を作る	49
無線LAN特有の〜	148
フィルタリング	49, 98
BSSIDで〜	148
フィンガープリント	123
物理層 (レイヤ1)	5, 7

フラグ	66, 83, 114, 145	無線LANカードのモード	138
プリンタ	125	無線LAN特有	
ブルートフォース攻撃	128	～の情報	147
プレゼンテーション層 (レイヤ6)	5, 6	～のフィルタ	148
フロー制御	4	メインウィンドウ	35
ブロードキャスト	14	メールサーバ	115
ブロードキャストドメイン	14	モニターモード	139
プロトコル	7		
～の相互作用	7	や行	
～の分析	55	ユニキャスト	14
～をフィルタリングする	49		
プロトコルスタック	4	ら行	
プロトコルデータユニット (Protocol Data Unit : PDU)	8	リピータハブ	9
プロミスキヤスモード	3	ルータ	11
～の使用	18	～で構成されたネットワーク上での スニッフィング	27
分割	4	ルーティング	12
ベースライニング	85	～の不具合	104
変換	3	ルーティングプロトコル	12
ポートスキャン	124	レイヤ1 (物理層)	5, 7
ポート到達不能	82	レイヤ2 (データリンク層)	5, 7
ポートミラーリング	20, 110	～のアドレス (MAC アドレス)	65
ポット (踏み台)	165	レイヤ3 (ネットワーク層)	5, 7
～の追跡	164	～のアドレス (IPアドレス)	12, 23, 65
～のパケット解析	159	レイヤ3スイッチ	13
		レイヤ4 (トランスポート層)	5, 6
ま行		～のプロトコル	66
マーキング	42	レイヤ5 (セッション層)	5, 6
マージ	44	レイヤ6 (プレゼンテーション層)	5, 6
マスターモード	138	レイヤ7 (アプリケーション層)	5, 6
マネージドモード	138	～のプロトコル	8, 71
マネージメントスイッチ	10	論理演算子	50
マルチキャスト	14	論理積 (and)	50
無線LAN	137	論理和 (or)	50
～に接続できない	150		
～のインターフェース	138	わ行	
～のスニッフィング	137	ワームの追跡	164

●著者紹介

Chris Sanders (クリス・サンダース)

ケンタッキー州のGraves County Schoolのネットワーク管理者 (1,800台以上のワークステーションと20台のサーバで構成される学内ネットワークの利用者数は約5,000人)。彼のWebサイト (<http://chrissanders.org>) にはチュートリアルやガイダンス、「Packet School 101」などの技術解説が公開されている。彼はまた、WindowsNetworking.comとWindowsDevCenter.comのライターでもある。彼はほぼ毎日、Wiresharkを使ってパケットを解析している。

●監訳者紹介

園田 道夫 (そのだ みちお)

サイバー大学IT総合学部准教授、NPO日本ネットワークセキュリティ協会研究員、独立行政法人情報処理推進機構 (IPA) 非常勤研究員、株式会社セキュアスカイ・テクノロジー社外取締役。情報セキュリティ関連の教育を中心に活動中。著作に漫画『アクセス探偵IHARA』シリーズ (技術評論社)、『Winnyはなぜ破られたのか』 (九天社)、監訳に『ハニーネットプロジェクト』 (毎日コミュニケーションズ)、共訳に『実用SSH 第2版』 (オライリー・ジャパン)、『暗号技術大全』 (ソフトバンククリエイティブ) など多数。

●訳者紹介

一瀬 小夜 (いちのせ さよ)

一般社団法人JPCERTコーディネーションセンターにて情報セキュリティに従事。共著に『ウィルスの原理と対策—インターネットセキュリティ』 (ソフトバンク クリエイティブ)、共訳に『セキュアプログラミング』 (オライリー・ジャパン)、『Solarisセキュリティ入門』 (翔泳社) などがある。

実践 パケット解析 — Wiresharkを使ったトラブルシューティング

2008年1月22日 初版第1刷発行

2010年9月27日 初版第6刷発行

著 者	Chris Sanders (クリス・サンダース)
監 訳 者	園田 道夫 (そのだ みちお)
訳 者	一瀬 小夜 (いちのせ さよ)
発 行 人	ティム・オライリー
制 作	有限会社はるにれ
印刷・製本	株式会社平河工業社
発 行 所	株式会社オライリー・ジャパン 〒160-0002 東京都新宿区坂町26番地27 インテリジェントプラザビル 1F TEL (03) 3356-5227 FAX (03) 3356-5263 電子メール japan@oreilly.co.jp
発 売 元	株式会社オーム社 〒101-8460 東京都千代田区神田錦町3-1 TEL (03) 3233-0641 (代表) FAX (03) 3233-3440

Printed in Japan (ISBN978-4-87311-351-7)

落丁、乱丁の際はお取り替えいたします。

本書は著作権上の保護を受けています。本書の一部あるいは全部について、株式会社オライリー・ジャパンから文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。