

概要

11/06/27 10:34:23

文書間に違いがあります。

新文書：

[2nd_mae_03](#)

66 ページ (2.16 MB)

11/06/27 10:33:12

旧文書：

[1st_mae_03](#)

58 ページ (2.26 MB)


11/06/27 10:33:07


結果の表示に使用します。


[最初の変更箇所が 1 ページ目にあります。](#)


削除されたページはありません

このレポートの読み方

 は、変更箇所を示します。

 は、削除された内容を示します。

 は、ページが変更されたことを示します。

 は、ページが移動されたことを示します。

**PRAISE FOR THE FIRST EDITION OF
*PRACTICAL PACKET ANALYSIS***

“An essential book if you are responsible for network administration on any level.”

—LINUX PRO MAGAZINE

“A wonderful, simple to use and well laid out guide.”

—ARSGEEK.COM

“If you need to get the basics of packet analysis down pat, this is a very good place to start.”

—STATEOFSECURITY.COM

“Very informative and held up to the key word in its title, ‘Practical.’ It does a great job of giving readers what they need to know to do packet analysis and then jumps right in with vivid real life examples of what to do with Wireshark.”

—LINUXSECURITY.COM

“Are there unknown hosts chatting away with each other? Is my machine talking to strangers? You need a packet sniffer to really find the answers to these questions. Wireshark is one of the best tools to do this job and this book is one of the best ways to learn about that tool.”

—FREE SOFTWARE MAGAZINE

“Perfect for the beginner to intermediate.”

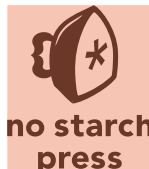
—DAEMON NEWS

PRACTICAL PACKET ANALYSIS

2ND EDITION

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

by Chris Sanders



San Francisco

PRACTICAL PACKET ANALYSIS, 2ND EDITION. Copyright © 2011 by Chris Sanders.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Printed in Canada

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

ISBN-10: 1-59327-266-9

ISBN-13: 978-1-59327-266-1

Publisher: William Pollock

Production Editor: Alison Law

Cover and Interior Design: Octopod Studios

Developmental Editor: William Pollock

Technical Reviewer: Tyler Reguly

Copyeditor: Marilyn Smith

Compositor: Susan Glinert Stevens

Proofreader: Ward Webber

Indexer: Nancy Guenther

For information on book distributors or translations, please contact No Starch Press, Inc. directly:

No Starch Press, Inc.

38 Ringold Street, San Francisco, CA 94103

phone: 415.863.9900; fax: 415.863.9950; info@nostarch.com; www.nostarch.com

The Library of Congress has cataloged the first edition as follows:

Sanders, Chris, 1986-

Practical packet analysis : using Wireshark to solve real-world network problems / Chris Sanders.

p. cm.

ISBN-13: 978-1-59327-149-7

ISBN-10: 1-59327-149-2

1. Computer network protocols. 2. Packet switching (Data transmission) I. Title.

TK5105.S55.S265 2007

004.6'6--dc22

2007013453

No Starch Press and the No Starch Press logo are registered trademarks of No Starch Press, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners. Rather than use a trademark symbol with every occurrence of a trademarked name, we are using the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor No Starch Press, Inc. shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in it.

This book, my life, and everything I will ever do is a direct result of faith given and faith received. This book is dedicated to God, my parents, and everyone who has ever shown faith in me.

I tell you the truth, if you have faith as small as a mustard seed, you can say to this mountain, "Move from here to there" and it will move. Nothing will be impossible for you.

Matthew 17:20

BRIEF CONTENTS

Acknowledgments	xv
Introduction	xvii
Chapter 1: Packet Analysis and Network Basics	1
Chapter 2: Tapping into the Wire	17
Chapter 3: Introduction to Wireshark	35
Chapter 4: Working with Captured Packets	47
Chapter 5: Advanced Wireshark Features	67
Chapter 6: Common Lower-Layer Protocols	85
Chapter 7: Common Upper-Layer Protocols	113
Chapter 8: Basic Real-World Scenarios	133
Chapter 9: Fighting a Slow Network	165
Chapter 10: Packet Analysis for Security	189
Chapter 11: Wireless Packet Analysis	215
Appendix: Further Reading	235
Index	241

CONTENTS IN DETAIL

ACKNOWLEDGMENTS

xv

INTRODUCTION

xvii

Why This Book?	xvii
Concepts and Approach	xviii
How to Use This Book	xix
About the Sample Capture Files	xx
The Rural Technology Fund	xx
Contacting Me	xx

1

PACKET ANALYSIS AND NETWORK BASICS

1

Packet Analysis and Packet Sniffers	2
Evaluating a Packet Sniffer	2
How Packet Sniffers Work	3
How Computers Communicate	4
Protocols	4
The Seven-Layer OSI Model	5
Data Encapsulation	8
Network Hardware	10
Traffic Classifications	14
Broadcast Traffic	14
Multicast Traffic	15
Unicast Traffic	15
Final Thoughts	16

2

TAPPING INTO THE WIRE

17

Living Promiscuously	18
Sniffing Around Hubs	19
Sniffing in a Switched Environment	20
Port Mirroring	21
Hubbing Out	22
Using a Tap	24
ARP Cache Poisoning	26
Sniffing in a Routed Environment	30
Sniffer Placement in Practice	31

3

INTRODUCTION TO WIRESHARK

35

A Brief History of Wireshark	35
The Benefits of Wireshark	36

Installing Wireshark.....	37
Installing on Microsoft Windows Systems.....	37
Installing on Linux Systems.....	39
Installing on Mac OS X Systems.....	40
Wireshark Fundamentals.....	41
Your First Packet Capture.....	41
Wireshark's Main Window.....	42
Wireshark Preferences.....	43
Packet Color Coding.....	45

4

WORKING WITH CAPTURED PACKETS

47

Working with Capture Files.....	47
Saving and Exporting Capture Files.....	48
Merging Capture Files.....	49
Working with Packets.....	49
Finding Packets.....	50
Marking Packets.....	51
Printing Packets.....	51
Setting Time Display Formats and References.....	52
Time Display Formats.....	52
Packet Time Referencing.....	52
Setting Capture Options.....	53
Capture Settings.....	53
Capture File(s) Settings.....	54
Stop Capture Settings.....	55
Display Options.....	56
Name Resolution Settings.....	56
Using Filters.....	56
Capture Filters.....	56
Display Filters.....	62
Saving Filters.....	65

5

ADVANCED WIRESHARK FEATURES

67

Network Endpoints and Conversations.....	67
Viewing Endpoints.....	68
Viewing Network Conversations.....	69
Troubleshooting with the Endpoints and Conversations Windows.....	70
Protocol Hierarchy Statistics.....	71
Name Resolution.....	72
Enabling Name Resolution.....	73
Potential Drawbacks to Name Resolution.....	73
Protocol Dissection.....	74
Changing the Dissector.....	74
Viewing Dissector Source Code.....	76
Following TCP Streams.....	76
Packet Lengths.....	78

Graphing.....	79
Viewing IO Graphs.....	79
Round-Trip Time Graphing.....	81
Flow Graphing.....	82
Expert Information.....	82

6 COMMON LOWER-LAYER PROTOCOLS 85

Address Resolution Protocol.....	86
The ARP Header.....	87
Packet 1: ARP Request.....	88
Packet 2: ARP Response.....	89
Gratuitous ARP.....	89
Internet Protocol.....	91
IP Addresses.....	91
The IPv4 Header.....	92
Time to Live.....	93
IP Fragmentation.....	95
Transmission Control Protocol.....	98
The TCP Header.....	98
TCP Ports.....	99
The TCP Three-Way Handshake.....	101
TCP Teardown.....	103
TCP Resets.....	105
User Datagram Protocol.....	105
The UDP Header.....	106
Internet Control Message Protocol.....	107
The ICMP Header.....	107
ICMP Types and Messages.....	107
Echo Requests and Responses.....	108
Traceroute.....	110

7 COMMON UPPER-LAYER PROTOCOLS 113

Dynamic Host Configuration Protocol.....	113
The DHCP Packet Structure.....	114
The DHCP Renewal Process.....	115
DHCP In-Lease Renewal.....	119
DHCP Options and Message Types.....	120
Domain Name System.....	120
The DNS Packet Structure.....	121
A Simple DNS Query.....	122
DNS Question Types.....	124
DNS Recursion.....	124
DNS Zone Transfers.....	127
Hypertext Transfer Protocol.....	129
Browsing with HTTP.....	129
Posting Data with HTTP.....	131
Final Thoughts.....	132

8**BASIC REAL-WORLD SCENARIOS****133**

Social Networking at the Packet Level	134
Capturing Twitter Traffic	134
Capturing Facebook Traffic	137
Comparing Twitter vs. Facebook Methods	140
Capturing ESPN.com Traffic	140
Using the Conversations Window	140
Using the Protocol Hierarchy Statistics Window	141
Viewing DNS Traffic	142
Viewing HTTP Requests	143
Real-World Problems	144
No Internet Access: Configuration Problems	144
No Internet Access: Unwanted Redirection	147
No Internet Access: Upstream Problems	150
Inconsistent Printer	153
Stranded in a Branch Office	155
Ticked-Off Developer	159
Final Thoughts	163

9**FIGHTING A SLOW NETWORK****165**

TCP Error-Recovery Features	166
TCP Retransmissions	166
TCP Duplicate Acknowledgments and Fast Retransmissions	169
TCP Flow Control	173
Adjusting the Window Size	174
Halting Data Flow with a Zero Window Notification	175
The TCP Sliding Window in Practice	175
Learning from TCP Error-Control and Flow-Control Packets	178
Locating the Source of High Latency	179
Normal Communications	180
Slow Communications—Wire Latency	180
Slow Communications—Client Latency	181
Slow Communications—Server Latency	182
Latency Locating Framework	182
Network Baseline	183
Site Baseline	184
Host Baseline	185
Application Baseline	186
Additional Notes on Baselines	186
Final Thoughts	187

10**PACKET ANALYSIS FOR SECURITY****189**

Reconnaissance	190
SYN Scan	190
Operating System Fingerprinting	194

Exploitation	197
Operation Aurora	197
ARP Cache Poisoning	202
Remote-Access Trojan	206
Final Thoughts.....	213

11

WIRELESS PACKET ANALYSIS

215

Physical Considerations	216
Sniffing One Channel at a Time	216
Wireless Signal Interference	217
Detecting and Analyzing Signal Interference	217
Wireless Card Modes.....	218
Sniffing Wirelessly in Windows	219
Configuring AirPcap.....	219
Capturing Traffic with AirPcap.....	221
Sniffing Wirelessly in Linux.....	222
802.11 Packet Structure	223
Adding Wireless-Specific Columns to the Packet List Pane	225
Wireless-Specific Filters.....	226
Filtering Traffic for a Specific BSS ID.....	226
Filtering Specific Wireless Packet Types	227
Filtering a Specific Frequency.....	227
Wireless Security	228
Successful WEP Authentication.....	229
Failed WEP Authentication	230
Successful WPA Authentication	231
Failed WPA Authentication.....	232
Final Thoughts.....	233

APPENDIX

FURTHER READING

235

Packet Analysis Tools.....	235
tcpdump and Windump	235
Cain & Abel	236
Scapy.....	236
Netdude	236
Colasoft Packet Builder	237
CloudShark	237
pcapr	237
NetworkMiner	238
Tcpreplay.....	238
ngrep	238
libpcap.....	239
hping.....	239
Domain Dossier	239
Perl and Python.....	239

Packet Analysis Resources	239
Wireshark Home Page.....	239
SANS Security Intrusion Detection In-Depth Course	239
Chris Sanders Blog	240
Packetstan Blog	240
Wireshark University	240
IANA	240
TCP/IP Illustrated (Addison-Wesley).....	240
The TCP/IP Guide (No Starch Press)	240

INDEX

241

ACKNOWLEDGMENTS

This book was made possible through the direct and indirect contributions of a great number of people.

First and foremost, all the glory goes to God. Writing a book brings forth a great deal of positive and negative emotion. When I am stressed, He brings me comfort. When I am frustrated, He brings me peace. When I am confused, He brings me resolve. When I am tired, He brings me rest. When I am prideful, He keeps me level-headed. This book, my career, and my existence are possible only because of God and his son Jesus Christ.

Dad, I draw motivation from a lot of sources, but nothing makes me happier than to hear you say that you are proud of me. I can't thank you enough for letting me know that you are.

Mom, the second edition of this book will be released right before the ten-year anniversary of your passing. I know you are watching over me and that you are proud, and I hope I can continue to make you even prouder.

Aunt Debi and Uncle Randy, you guys have been my biggest supporters since day one. I don't have a large family, but I treasure what I do have, especially you guys. Although we don't get together nearly as much as I'd like, I can't thank you enough for being like a second set of parents to me.

Tina Nance, we don't get to talk nearly as much as we used to, but I will always consider you my second mom. I wouldn't be doing what I'm doing today without your support and belief in me.

Jason Smith, you've listened to more of my frequent rants than anyone else, and just that has helped me keep sane. Thanks for being a great friend and coworker, providing input on various projects, and letting me use your garage for like six months that one time.

Regarding my coworkers (past and present), I've always believed that if a person surrounds himself with good people, he will become a better person. I have the good fortune of working with some great people who are some of the best and brightest in the business. You guys are my family.

Mike Poor, you are my packet-analysis idol without equivocation. Your work and approach to what you do are inspiring and help me do what I do.

Tyler Reguly, thanks so much for tech-editing this book. I'm sure it wasn't a fun process, but it was absolutely necessary and absolutely appreciated.

Thanks also to Gerald Combs and the Wireshark development team. It's the dedication of Gerald and the hundreds of other developers that makes Wireshark such a great analysis platform. If it weren't for their efforts, this book wouldn't exist ... or if it did, it would be based on tcpdump, and that wouldn't be fun for anyone.

Bill and the No Starch Press staff took a chance on a kid from Kentucky not just once but twice. Thanks for doing it, having patience with me, and helping me make my dreams come true.

INTRODUCTION



Practical Packet Analysis, 2nd Edition was written over the course of a year and a half, from late 2009 to mid 2011, approximately four years after the first edition's release. This book contains almost all new content, with completely new capture files and scenarios. If you liked the first edition, then you will like this one. It is written in the same tone and breaks down explanations in a simple, understandable manner. If you didn't like the first edition, you will like this one, because of the new scenarios and expanded content.

Why This Book?

You may find yourself wondering why you should buy this book as opposed to any other book about packet analysis. The answer lies in the title: *Practical Packet Analysis*. Let's face it—nothing beats real-world experience, and the closest you can come to that experience in a book is through practical examples of packet analysis with real-world scenarios.

The first half of this book gives you the prerequisite knowledge you will need to understand packet analysis and Wireshark. The second half of the book is devoted entirely to practical cases that you could easily encounter in day-to-day network management.

Whether you are a network technician, a network administrator, a chief information officer, a desktop technician, or even a network security analyst, you have a lot to gain from understanding and using the packet-analysis techniques described in this book.

Concepts and Approach

I am generally a really laid-back guy, so when I teach a concept, I try to do so in a really laid-back way. This holds true for the language used in this book. It is very easy to get lost in technical jargon when dealing with technical concepts, but I have tried my best to keep things as casual as possible. I've made all the definitions clear, straightforward, and to the point, without any added fluff. After all, I'm from the great state of Kentucky, so I try to keep the big words to a minimum. (You'll have to forgive me for some of the backwoods country verbiage you'll find throughout the text.)

If you really want to learn packet analysis, you should make it a point to master the concepts in the first several chapters, because they are integral to understanding the rest of the book. The second half of the book is purely practical. You may not see these exact scenarios in your workplace, but you should be able to apply the concepts you learn from them in the situations you do encounter.

Here is a quick breakdown of the contents of the chapters in this book:

Chapter 1: Packet Analysis and Network Basics

What is packet analysis? How does it work? How do you do it? This chapter covers the basics of network communication and packet analysis.

Chapter 2: Tapping into the Wire

This chapter covers the different techniques you can use to place a packet sniffer on your network.

Chapter 3: Introduction to Wireshark

Here, we'll look at the basics of Wireshark—where to get it, how to use it, what it does, why it's great, and all of that good stuff.

Chapter 4: Working with Captured Packets

After you have Wireshark up and running, you will want to know how to interact with captured packets. This is where you'll learn the basics.

Chapter 5: Advanced Wireshark Features

Once you have learned to crawl, it's time to take off running. This chapter delves into the advanced Wireshark features, taking you under the hood to show you some of the less apparent operations.

Chapter 6: Common Lower-Layer Protocols

This chapter shows you what some of the most common lower-layer network communication protocols—such as TCP, UDP, and IP—look like at the packet level. In order to understand how these protocols can malfunction, you first need to understand how they work.

Chapter 7: Common Upper-Layer Protocols

Continuing with protocol coverage, this chapter shows you what the three of the most common upper-layer network communication protocols—HTTP, DNS, and DHCP—look like at the packet level.

Chapter 8: Basic Real-World Scenarios

This chapter contains breakdowns of some common traffic and the first set of real-world scenarios. Each scenario is presented in an easy-to-follow format, where the problem, analysis, and solution are given. These basic scenarios deal with only a few computers and involve a limited amount of analysis—just enough to get your feet wet.

Chapter 9: Fighting a Slow Network

The most common problems network technicians hear about generally involve slow network performance. This chapter is devoted to solving these types of problems.

Chapter 10: Packet Analysis for Security

Network security is the biggest hot-button topic in the information technology area. Chapter 10 shows you some scenarios related to solving security-related issues with packet-analysis techniques.

Chapter 11: Wireless Packet Analysis

This chapter is a primer on wireless packet analysis. It discusses the differences between wireless analysis and wired analysis, and includes some examples of wireless network traffic.

Appendix: Further Reading

The appendix of this book suggests some other reference tools and websites that you might find useful as you continue to use the packet-analysis techniques you have learned.

How to Use This Book

I have intended this book to be used in two ways:

- As an educational text that you will read through, chapter by chapter, in order to gain an understanding of packet analysis. This means paying particular attention to the real-world scenarios in the last several chapters.
- As a reference resource. There are some features of Wireshark that you will not use very often, so you may forget how they work. *Practical Packet Analysis* is a great book to have on your bookshelf when you need a quick refresher about how to use a specific feature. I've also provided some unique charts, diagrams, and methodologies that may prove to be useful references when doing packet analysis for your job.

About the Sample Capture Files

All of the capture files used in this book are available from the No Starch Press page for this book, <http://www.nostarch.com/packet2.htm>. In order to maximize the potential of this book, I highly recommend that you download these files and use them as you follow along with the examples.

The Rural Technology Fund

I couldn't write an introduction without mentioning the best thing to come from *Practical Packet Analysis*. Shortly after the release of the first edition of this book, I founded a 501(c)(3) nonprofit organization that is the culmination of one of my biggest dreams.

Rural students, even those with excellent grades, often have fewer opportunities for exposure to technology than their city or suburban counterparts. Established in 2008, the Rural Technology Fund (RTF) seeks to reduce the digital divide between rural communities and their more urban and suburban counterparts. This is done through targeted scholarship programs, community involvement, and general promotion and advocacy of technology in rural areas.

Our scholarships are targeted to students living in rural communities who have a passion for computer technology and intend to pursue further education in that field. I'm pleased to announce that 100 percent of the author proceeds from this book go directly to the Rural Technology Fund in order to provide these scholarships. If you want to learn more about the Rural Technology Fund or how you can contribute, visit our website at <http://www.ruraltechfund.org/>

Contacting Me

I'm always thrilled to get feedback from people who read my writing. If you would like to contact me for any reason, you can send all questions, comments, threats, and marriage proposals directly to me at chris@chrissanders.org. I also blog regularly at <http://www.chrissanders.org/> and can be followed on Twitter at @chrissanders88.

1

PACKET ANALYSIS AND NETWORK BASICS



A million different things can go wrong with a computer network on any given day—from a simple spyware infection to a complex router configuration error—and it's impossible to solve every problem immediately. The best we can hope for is to be fully prepared with the knowledge and tools we need to respond to these types of issues.

All network problems stem from the packet level, where even the prettiest looking applications can reveal their horrible implementations, and seemingly trustworthy protocols can prove malicious. To better understand network problems, we go to the packet level. Here, nothing is hidden from us—nothing is obscured by misleading menu structures, eye-catching graphics, or untrustworthy employees. At this level, there are no true secrets (only encrypted ones). The more we can do at the packet level, the more we can control our network and solve problems. This is the world of packet analysis.

This book dives into the world of packet analysis headfirst. You'll learn how to tackle slow network communication, identify application bottlenecks, and even track hackers through some real-world scenarios. By the time you have finished reading this book, you should be able to implement advanced packet-analysis techniques that will help you solve even the most difficult problems in your own network.

In this chapter, we'll begin with the basics, focusing on network communication, so you can gain some of the basic background you'll need to examine different scenarios.

Packet Analysis and Packet Sniffers

Packet analysis, often referred to as *packet sniffing* or *protocol analysis*, describes the process of capturing and interpreting live data as it flows across a network in order to better understand what is happening on that network. Packet analysis is typically performed by a *packet sniffer*, a tool used to capture raw network data going across the wire.

Packet analysis can help with the following:

- Understanding network characteristics
- Learning who is on a network
- Determining who or what is utilizing available bandwidth
- Identifying peak network usage times
- Identifying possible attacks or malicious activity
- Finding unsecured and bloated applications

There are various types of packet-sniffing programs, including both free and commercial ones. Each program is designed with different goals in mind. A few popular packet-analysis programs are `tcpdump`, `OmniPeek`, and `Wireshark` (which we will be using exclusively in this book). `tcpdump` is a command-line program. `OmniPeek` and `Wireshark` have graphical user interfaces (GUIs).

Evaluating a Packet Sniffer

You need to consider a number of factors when selecting a packet sniffer, including the following:

Supported protocols All packet sniffers can interpret various protocols. Most can interpret common network protocols (such as IPv4 and ICMP), transport layer protocols (such as TCP and UDP), and even application layer protocols (such as DNS and HTTP). However, they may not support nontraditional or newer protocols (such as IPv6, SMBv2, and SIP). When choosing a sniffer, make sure that it supports the protocols you're going to use.

User-friendliness Consider the packet sniffer's program layout, ease of installation, and general flow of standard operations. The program you choose should fit your level of expertise. If you have very little packet-analysis experience, you may want to avoid the more advanced command-line packet sniffers like tcpdump. On the other hand, if you have a wealth of experience, you may find an advanced program more appealing. As you gain experience with packet analysis, you may even find it useful to combine multiple packet-sniffing programs to fit particular scenarios.

Cost The great thing about packet sniffers is that there are many free ones that rival any commercial products. The most notable difference between commercial products and their free alternatives is their reporting engines. Commercial products typically include some form of fancy report-generation module, which is usually lacking or nonexistent in free applications.

Program support Even after you have mastered the basics of a sniffing program, you may occasionally need support to solve new problems as they arise. When evaluating available support, look for developer documentation, public forums, and mailing lists. Although there may be a lack of developer support for free packet-sniffing programs like Wireshark, the communities that use these applications will often fill the gap. These communities of users and contributors provide discussion boards, wikis, and blogs designed to help you to get more out of your packet sniffer.

Operating system support Unfortunately, not all packet sniffers support every operating system. Choose one that will work on all the operating systems that you need to support. If you are a consultant, you may be required to capture and analyze packets on a variety of operating systems, so you will need a tool that runs on most operating systems. Also keep in mind that you will sometimes capture packets on one machine and review them on another. Variations between operating systems may force you to use a different application for each device.

How Packet Sniffers Work

The packet-sniffing process involves a cooperative effort between software and hardware. This process can be broken down into three steps:

Collection In the first step, the packet sniffer collects raw binary data from the wire. Typically, this is done by switching the selected network interface into *promiscuous mode*. In this mode, the network card can listen to all traffic on a network segment, not only the traffic that is addressed to it.

Conversion In this step, the captured binary data is converted into a readable form. This is where most advanced command-line packet sniffers stop. At this point, the network data is in a form that can be interpreted only on a very basic level, leaving the majority of the analysis to the end user.

Analysis The third and final step involves the actual analysis of the captured and converted data. The packet sniffer takes the captured network data, verifies its protocol based on the information extracted, and begins its analysis of that protocol's specific features.

How Computers Communicate

In order to fully understand packet analysis, you must understand exactly how computers communicate with each other. In this section, we'll examine the basics of network protocols, the Open Systems Interconnections (OSI) model, network data frames, and the hardware that supports it all.

Protocols

Modern networks are made up of a variety of systems running on many different platforms. To aid this communication, we use a set of common languages called *protocols*. Common protocols include Transmission Control Protocol (TCP), Internet Protocol (IP), Address Resolution Protocol (ARP), and Dynamic Host Configuration Protocol (DHCP). A *protocol stack* is a logical grouping of protocols that work together.

One of the best ways to understand protocols is to think of them as similar to the rules that govern spoken or written human languages. Every language has rules, such as how verbs should be conjugated, how people should be greeted, and even how to properly thank someone. Protocols work in much the same fashion, allowing us to define how packets should be routed, how to initiate a connection, and how to acknowledge the receipt of data.

A protocol can be extremely simple or highly complex, depending on its function. Although the various protocols are often drastically different, many protocols commonly address the following issues:

Connection initiation Is it the client or server initiating the connection? What information must be exchanged prior to communication?

Negotiation of connection characteristics Is the communication of the protocol encrypted? How are encryption keys transmitted between communicating hosts?

Data formatting How is the data contained in the packet ordered? In what order is the data processed by the devices receiving it?

Error detection and correction What happens in the event that a packet takes too long to reach its destination? How does a client recover if it cannot establish communication with a server for a short duration?

Connection termination How does one host signify to the other that communication has ended? What final information must be transmitted in order to gracefully terminate communication?

The Seven-Layer OSI Model

Protocols are separated according to their functions based on the industry-standard OSI reference model. The OSI model divides the network communications process into seven distinct layers, as shown in Figure 1-1. This hierarchical model makes it much easier to understand network communication. The application layer at the top represents the actual programs used to access network resources. The bottom layer is the physical layer, through which the actual network data travels. The protocols at each layer work together to ensure data is properly handled by the protocols at layers above and below it.

NOTE The OSI model was originally published in 1983 by the International Organization for Standardization (ISO) as a document called ISO 7498. The OSI model is no more than an industry-recommended standard. Protocol developers are not required to follow it exactly. And the OSI model is not the only networking model that exists; for example, some people prefer the Department of Defense (DoD) model, also known as the TCP/IP model.

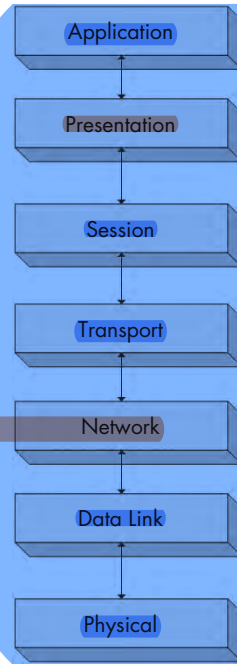


Figure 1-1: A hierarchical view of the seven layers of the OSI model

Each OSI model layer has a specific function, as follows:

Application layer (layer 7) The topmost layer of the OSI model provides a means for users to actually access network resources. This is the only layer typically seen by end users, as it provides the interface that is the base for all of their network activities.

Presentation layer (layer 6) This layer transforms the data it receives into a format that can be read by the application layer. The data encoding and decoding done here depends on the application layer protocol that is sending or receiving the data. The presentation layer also handles several forms of encryption and decryption used for securing data.

Session layer (layer 5) This layer manages the *dialogue*, or session between two computers. It establishes, manages, and terminates this connection among all communicating devices. The session layer is also responsible for establishing whether a connection is duplex or half-duplex, and for gracefully closing a connection between hosts, rather than dropping it abruptly.

Transport layer (layer 4) The primary purpose of the transport layer is to provide reliable data transport services to lower layers. Through flow control, segmentation/desegmentation, and error control, the transport layer makes sure data gets from point to point error-free. Because ensuring reliable data transportation can be extremely cumbersome, the OSI model devotes an entire layer to it. The transport layer utilizes both connection-oriented and connectionless protocols. Certain firewalls and proxy servers operate at this layer.

Network layer (layer 3) This layer is responsible for routing data between physical networks, and it is one of the most complex of the OSI layers. It is responsible for the logical addressing of network hosts (for example, through an IP address). It also handles packet fragmentation, and in some cases, error detection. Routers operate at this layer.

Data link layer (layer 2) This layer provides a means of transporting data across a physical network. Its primary purpose is to provide an addressing scheme that can be used to identify physical devices (for example, MAC addresses). Bridges and switches are physical devices that operate at the data link layer.

Physical layer (layer 1) The layer at the bottom of the OSI model is the physical medium through which network data is transferred. This layer defines the physical and electrical nature of all hardware used, including voltages, hubs, network adapters, repeaters, and cabling specifications. The physical layer establishes and terminates connections, provides a means of sharing communication resources, and converts signals from digital to analog and vice versa.

Table 1-1 lists some of the more common protocols used at each individual layer of the OSI model.

Table 1-1: Typical Protocols Used in Each Layer of the OSI Model

Layer	Protocol
Application	HTTP, SMTP, FTP, Telnet
Presentation	ASCII, MPEG, JPEG, MIDI
Session	NetBIOS, SAP, SDP, NWLink
Transport	TCP, UDP, SPX
Network	IP, IPX
Data link	Ethernet, Token Ring, FDDI, AppleTalk

Although the OSI model is no more than a recommended standard, you should know it by heart. As we progress through this book, you will find that the interaction of protocols on different layers will shape your approach to network problems. Router issues will soon become “layer 3 problems” and software issues will be recognized as “layer 7 problems.”

NOTE In discussing our work, a colleague told me about a user complaining that he could not access a network resource. The issue was the result of the user entering an incorrect password. My colleague referred to this as a “layer 8 issue.” Layer 8 is the unofficial user layer. This term is commonly used among those who live at the packet level.

How does data flow through the OSI model? The initial data transfer on a network begins at the application layer of the transmitting system. Data works its way down the seven layers of the OSI model until it reaches the physical layer, at which point the physical layer of the transmitting system sends the data to the receiving system. The receiving system picks up the data at its physical layer, and the data proceeds up the remaining layers of the receiving system to the application layer at the top.

Services provided by various protocols at any given level of the OSI model are not redundant. For example, if a protocol at one layer provides a particular service, then no other protocol at any other layer will provide this same service. Protocols at different levels may have features with similar goals, but they will function a bit differently.

Protocols at corresponding layers on the sending and receiving computers are complementary. For example, if a protocol on layer 7 of the sending computer is responsible for encrypting the data being transmitted, the corresponding protocol on layer 7 of the receiving machine is expected to be responsible for decrypting that data.

Figure 1-2 shows a graphical representation of the OSI model as it relates to two communicating clients. You can see communication going from top to bottom on one client, and then reversing when it reaches the second client.

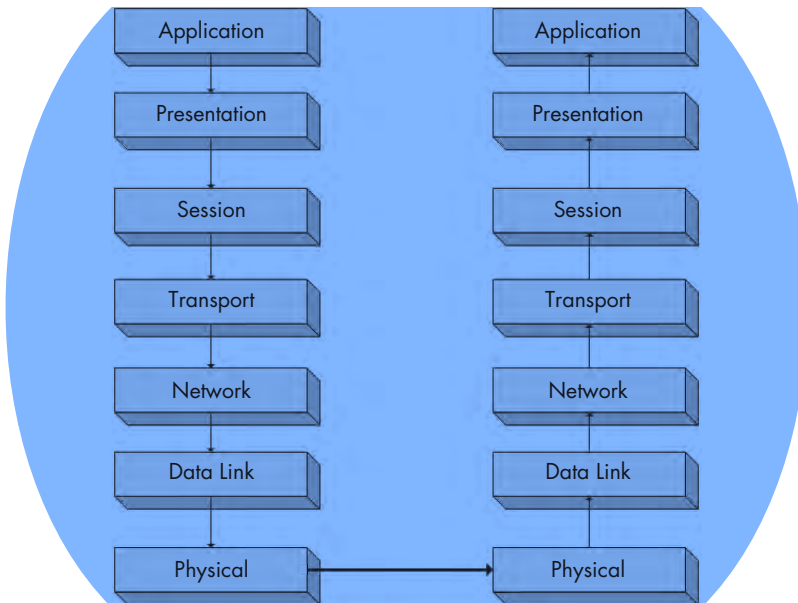


Figure 1-2: Protocols working at the same layer on both the sending and receiving systems

Each layer in the OSI model is capable of communicating with only the layers directly above and below it. For example, layer 2 can send and receive data only from layers 1 and 3.

Data Encapsulation

The protocols on different layers of the OSI model communicate with the aid of *data encapsulation*. Each layer in the stack is responsible for adding a header or footer—extra bits of information that allow the layers to communicate—to the data being communicated. For example, when the transport layer receives data from the session layer, it adds its own header information to that data before passing it to the next layer.

The encapsulation process creates a *protocol data unit (PDU)*, which includes the data being sent and all header or footer information added to it. As data moves down the OSI model, the PDU changes and grows as header and footer information from various protocols is added to it. The PDU is in its final form once it reaches the physical layer, at which point it is sent to the destination computer. The receiving computer strips the protocol headers and footers from the PDU as the data climbs up the OSI layers. Once the PDU reaches the top layer of the OSI model, only the original data remains.

NOTE The term *packet* refers to a complete PDU that includes header and footer information from all layers of the OSI model.

Understanding how encapsulation of data works can be a bit confusing, so we'll look at a practical example of a packet being built, transmitted, and received in relation to the OSI model. Keep in mind that as analysts, we don't often talk about the session or presentation layers, so those will be absent in this example (and the rest of this book).

In this scenario, we are on a computer that is attempting to browse to <http://www.google.com/>. For this process to take place, we must generate a request packet that is transmitted from our source client computer to the destination server computer. This scenario assumes that a TCP/IP communication session has already been initiated. Figure 1-3 illustrates the data-encapsulation process in this example.

We begin on our client computer at the application layer. We are browsing to a website, so the application layer protocol being used is HTTP, which will issue a command to download the file *index.html* from *google.com*.

Once our application layer protocol has dictated what we want to accomplish, our concern is with getting the packet to its destination. The data in our packet is passed down the stack to the transport layer. HTTP is an application layer protocol that utilizes, or sits on, TCP. Therefore, TCP serves as the transport layer protocol used to ensure reliable delivery of the packet. As a result, a TCP header is generated. This TCP header includes sequence numbers and other data that is appended to the packet, and ensures that the packet is properly delivered.

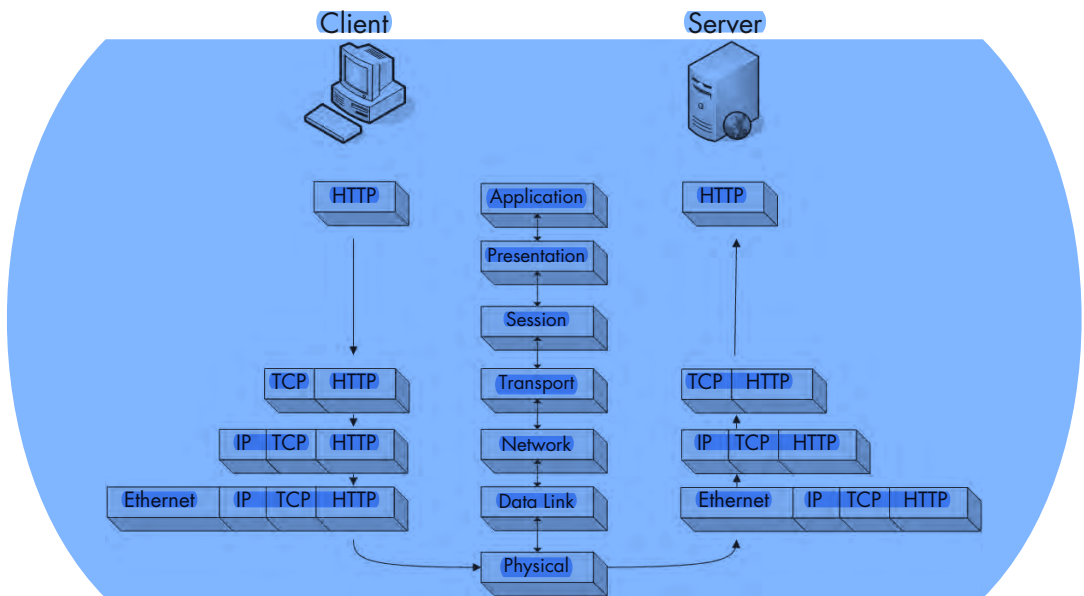


Figure 1-3: A graphical representation of encapsulation of data between client and server

NOTE We often say that one protocol “sits on” another protocol because of the top-down design of the OSI model. An application protocol such as HTTP provides a particular service and relies on TCP to ensure delivery of its service. As you will learn, DNS sits on UDP, and TCP sits on IP.

Having done its job, TCP hands the packet off to IP, which is the layer 3 protocol responsible for the logical addressing of the packet. IP creates a header containing logical addressing information and passes the packet along to Ethernet on the data link layer. Physical Ethernet addresses are stored in the Ethernet header. The packet is now fully assembled and passed to the physical layer, where it is transmitted as zeros and ones across the network.

The completed packet traverses the network cabling system, eventually reaching the Google web server. The web server begins by reading the packet from the bottom up, meaning that it first reads the data link layer, which contains the physical Ethernet addressing information that the network card uses to determine that the packet is intended for a particular server. Once this information is processed, the layer 2 information is stripped away, and the layer 3 information is processed.

The IP addressing information is read in the same manner as the layer 2 information to ensure proper addressing and that the packet is not fragmented. This data is also stripped away so that the next layer can be processed.

Layer 4 TCP information is now read to ensure that the packet has arrived in sequence. Then the layer 4 header information is stripped away, leaving only the application layer data, which can be passed to the web server application hosting the website. In response to this packet from the client, the server should transmit a TCP acknowledgment packet so the client knows its request was received followed by the *index.html* file.

All packets are built and processed as described in this example, regardless of which protocols are used. But at the same time, keep in mind that not every packet on a network is generated from an application layer protocol, so you will see packets that contain only information from layer 2, 3, or 4 protocols.

Network Hardware

Now it's time to look at network hardware, where the dirty work is done. We'll focus on just a few of the more common pieces of network hardware: hubs, switches, and routers.

Hubs

A *hub* is generally a box with multiple RJ-45 ports, like the NETGEAR hub shown in Figure 1-4. Hubs range from very small 4-port devices to larger 48-port ones designed for rack mounting in a corporate environment.



Figure 1-4: A typical 4-port Ethernet hub

Because hubs can generate a lot of unnecessary network traffic and are capable of operating only in *half-duplex mode* (they cannot send and receive data at the same time), you won't typically see them used in most modern or high-density networks (switches are used instead). However, you should know how hubs work, since they will be very important to packet analysis when using the “hubbing out” technique discussed in Chapter 2.

A hub is no more than a *repeating device* that operates on the physical layer of the OSI model. It takes packets sent from one port and transmits (repeats) them to every other port on the device. For example, if a computer on port 1 of a 4-port hub needs to send data to a computer on port 2, the hub sends those packets to ports 1, 2, 3, and 4. The clients connected to ports 3 and 4 examine the destination Media Access Control (MAC) address field in the Ethernet header of the packet, and they see that the packet is not for them, so they *drop* (discard) the packet. Figure 1-5 illustrates an example in which computer A is transmitting data to computer B. When computer A sends this data, all computers connected to the hub receive it. Only computer B actually accepts the data; the other computers discard it.

As an analogy, suppose that you sent an email with the subject line “Attention all marketing staff” to every employee in your company, rather than to only those people who work in the marketing department. The marketing department employees will know it is for them, and they will probably open

it. The other employees will see that it is not for them, and they will probably discard it. You can see how this would result in a lot of unnecessary communication and wasted time, yet this is exactly how a hub functions.

The best alternatives to hubs in production and high-density networks are switches, which are full-duplex devices that can send and receive data synchronously.

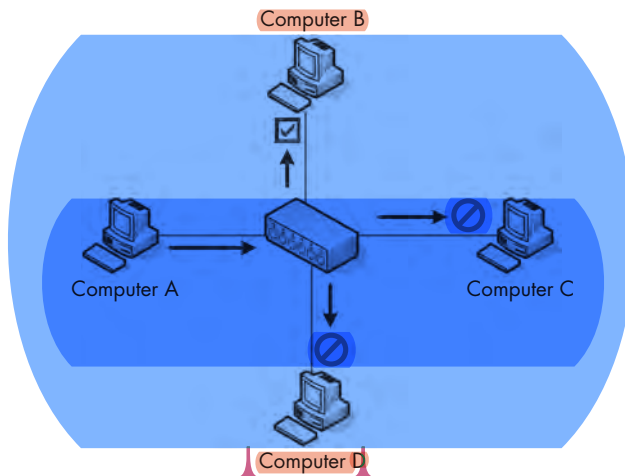


Figure 1-5: The flow of traffic when computer A transmits data to computer B through a hub

Switches

Like a hub, a switch is designed to repeat packets. However, unlike a hub, rather than broadcasting data to every port, a switch sends data to only the computer for which the data is intended. Switches look just like hubs, as shown in Figure 1-6.



Figure 1-6: A rack-mountable 24-port Ethernet switch

Several larger switches on the market, such as Cisco-branded ones, are managed via specialized, vendor-specific software or web interfaces. These switches are commonly referred to as managed switches. Managed switches provide several features that can be useful in network management, including the ability to enable or disable specific ports, view port specifics, make configuration changes, and remotely reboot.

Switches also offer advanced functionality when it comes to handling transmitted packets. In order to be able to communicate directly with specific devices, switches must be able to uniquely identify devices based on their MAC addresses, which means that they must operate on the data link layer of the OSI model.

Switches store the layer 2 address of every connected device in a CAM table, which acts as a kind of traffic cop. When a packet is transmitted, the switch reads the layer 2 header information in the packet and, using the CAM table as reference, determines to which port(s) to send the packet. Switches send packets only to specific ports, thus greatly reducing network traffic.

Figure 1-7 illustrates traffic flow through a switch. In this figure, computer A is sending data to only the intended recipient, computer B. Multiple conversations can happen on the network at the same time, but information is communicated directly between the switch and intended recipient, not between the switch and all connected computers.

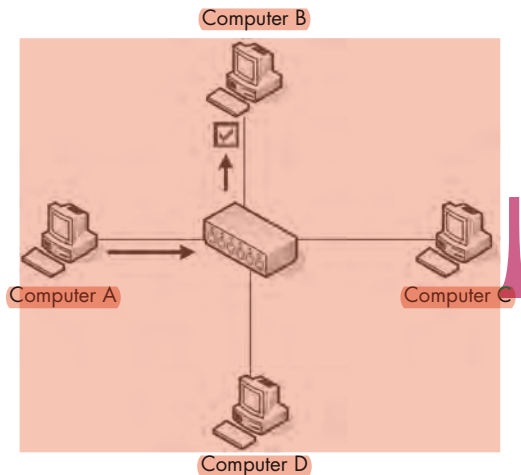


Figure 1-7: The flow of traffic when computer A transmits data to computer B through a switch

Routers

A router is an advanced network device with a much higher level of functionality than a switch or a hub. A router can take many shapes and forms, but most have several LED indicator lights on the front and a few network ports on the back, depending on the size of the network. Figure 1-8 shows an example of a router.

Routers operate at layer 3 of the OSI model, where they are responsible for forwarding packets between two or more networks. The process routers use to direct the flow of traffic among networks is called routing. Several types of routing protocols dictate how different types of packets are routed to other networks. Routers commonly use layer 3 addresses (such as IP addresses) to uniquely identify devices on a network.



Figure 1-8: A low-level Cisco router suitable for use in a small to mid-sized network

One way to illustrate the concept of routing is by using the analogy of a neighborhood with several streets. Think of the houses, with their addresses, as computers, and each street as a network segment, as shown in Figure 1-9. From your house on your street you can easily communicate with your neighbors in the other houses on the street. This is similar to the operation of a switch that allows communication among all computers on a network segment. However, communicating with a neighbor on another street is like communicating with a computer that is not on the same segment.

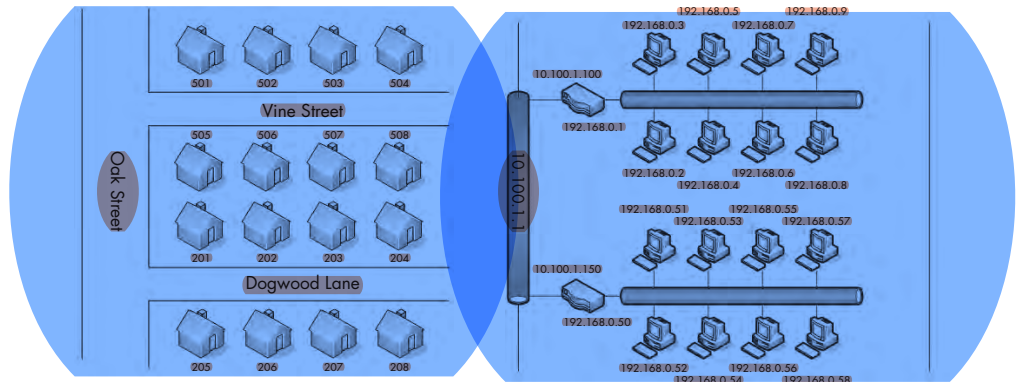


Figure 1-9: Comparison of a routed network to neighborhood streets

Referring to Figure 1-9, let's say that you're sitting at 503 Vine Street and need to get to 202 Dogwood Lane. In order to do this, you must cross onto Oak Street, and then onto Dogwood Lane. Think of this as crossing network segments. If the device at 192.168.0.3 needs to communicate with the device at 192.168.0.54, it must cross a router to get to the 10.100.1.1 network, and then cross the destination network segment's router before it can get to the destination network segment.

The size and number of routers on a network will typically depend on the network's size and function. Personal and home-office networks may have only a small router located at the center of the network. A large corporate network might have several routers spread throughout various departments, all connecting to one large central router or layer 3 switch (an advanced type of switch that also has built-in functionality to act as a router).

As you begin looking at more and more network diagrams, you will come to understand how data flows through these various points. Figure 1-10 shows the layout of a very common form of routed network. In this example, two separate networks are connected via a single router. If a computer on network A wishes to communicate with a computer on network B, the transmitted data must go through the router.

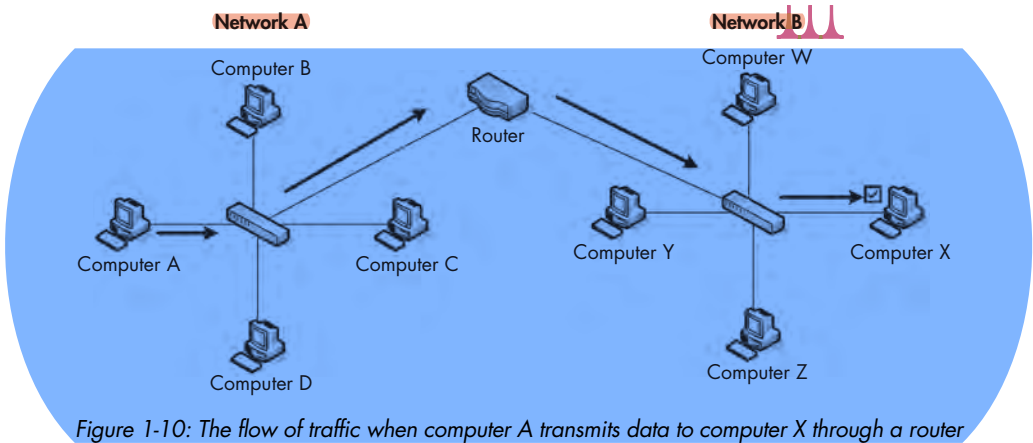


Figure 1-10: The flow of traffic when computer A transmits data to computer X through a router

Traffic Classifications

Network traffic can be divided among three main classes: broadcast, multicast, and unicast. Each classification has a distinct characteristic that determines how packets in that class are handled by networking hardware.

Broadcast Traffic

A *broadcast packet* is one that is sent to all ports on a network segment, regardless of whether that port is a hub or switch.

All broadcast traffic is not created equally, however. There are layer 2 and layer 3 forms of broadcast traffic. For instance, on layer 2, the MAC address FF:FF:FF:FF:FF:FF is the reserved broadcast address, and any traffic sent to this address is broadcast to the entire network segment. Layer 3 also has a specific broadcast address.

The highest possible IP address in an IP network range is reserved for use as the broadcast address. For example, in a network configured with a 192.168.0.xxx IP range and a 255.255.255.0 subnet mask, the address 192.168.0.255 is the broadcast address.

In larger networks with multiple hubs or switches connected via different media, broadcast packets transmitted from one switch reach all the way to the ports on the other switches on the network, as they are repeated from switch to switch. The extent to which broadcast packets travel is called the *broadcast domain*, which is the network segment where any computer can directly transmit to another computer without going through a router. Figure 1-11

shows an example of two broadcast domains on a small network. Because each broadcast domain extends until it reaches the router, broadcast packets circulate only within this specified broadcast domain.

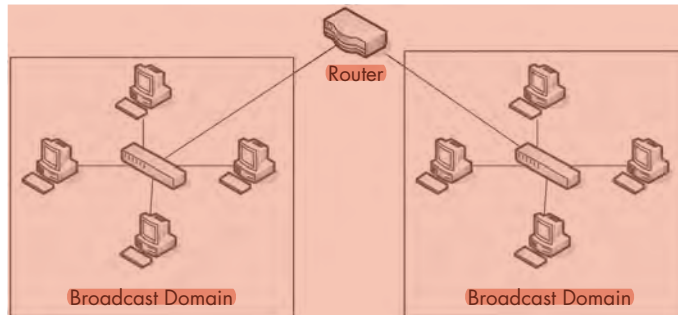


Figure 1-11: A broadcast domain extends to everything behind the current routed segment.

Our earlier example describing how routing relates to a neighborhood also provides good insight into how broadcast domains work. You can think of a broadcast domain as being like a neighborhood street. If you stand on your front porch and yell, only the people on your street will be able to hear you. If you want to talk to someone on a different street, you need to find a way to speak to that person directly, rather than broadcasting (yelling) from your front porch.

Multicast Traffic

Multicast is a means of transmitting a packet from a single source to multiple destinations simultaneously. The goal of multicasting is to simplify this process by using as little bandwidth as possible. The optimization of this traffic lies in the number of times a stream of data is replicated in order to get to its destination. The exact handling of multicast traffic is highly dependent on its implementation in individual protocols.

The primary method of implementing multicast is via an addressing scheme that joins the packet recipients to a multicast group, which is how IP multicast works. This addressing scheme ensures that the packets cannot be transmitted to computers to which they are not destined. In fact, IP devotes an entire range of addresses to multicast. If you see an IP address in the 224.0.0.0 to 239.255.255.255 range, it is most likely multicast traffic.

Unicast Traffic

A unicast packet is transmitted from one computer directly to another. The details of how unicast functions depend on the protocol using it.

For example, consider a device that wishes to communicate with a web server. This is a one-to-one connection, so this communication process would begin with the client device transmitting a packet to only the web server. This form of communication is an example of unicast traffic.

Final Thoughts

This chapter covered the absolute basics that you need as a foundation for packet analysis. You *must* understand what is going on at this level of network communication before you can begin troubleshooting network issues. In the next chapter, we will build on these concepts and discuss more advanced network communication principles.

2

TAPPING INTO THE WIRE



A key decision for effective packet analysis is where to position a packet sniffer to appropriately capture the data. This is most often referred to by packet analysts as *sniffing the wire*, *tapping the network*, or *tapping into the wire*. Simply put, this is the process of placing a packet sniffer on a network in the correct physical location.

Unfortunately, sniffing packets is not as simple as plugging a laptop into a network port and capturing traffic. In fact, it is sometimes more difficult to place a packet sniffer on a network's cabling system than it is to actually analyze the packets.

The challenge with sniffer placement is that a large variety of networking hardware is used to connect devices. Figure 2-1 illustrates a typical situation. Because the three main devices on a modern network (hubs, switches, and routers) each handles traffic differently, you must be very aware of the physical setup of the network you are analyzing.

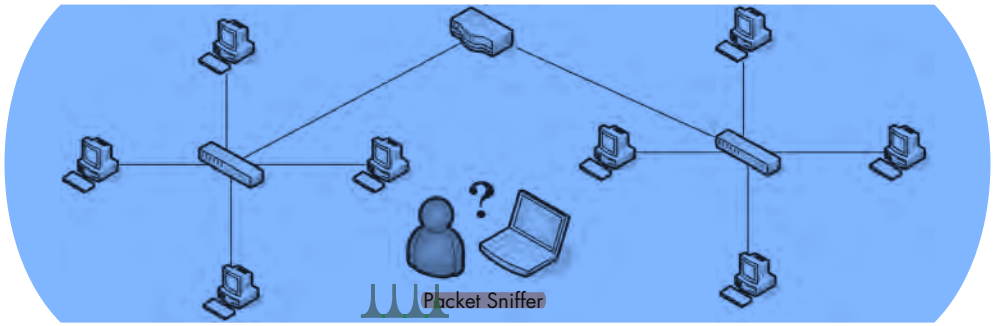


Figure 2-1: Placing your sniffer on the network is sometimes the biggest challenge you will face.

The goal of this chapter is to help you develop an understanding of packet-sniffer placement in a variety of different network topologies. But first, let's look at how we're actually able to see all the packets that cross the wire we're tapping into.

Living Promiscuously

Before you can sniff packets on a network, you need a network interface card (NIC) that supports a promiscuous mode driver. *Promiscuous mode* is what allows a NIC to view all packets crossing the wire.

As you learned in Chapter 1, with network broadcast traffic, it's common for clients to receive packets that are not actually destined for them. ARP, which is used to determine which MAC address corresponds to a particular IP address, is a crucial fixture on any network, and it's a great example of traffic sent to hosts other than the intended recipient. To find the matching MAC address, ARP sends a broadcast packet to every device on its broadcast domain in hopes that the correct client will respond.

A broadcast domain (the network segment where any computer can directly transmit to another computer without going through a router) can consist of several computers, but only one client on that domain should be interested in the ARP broadcast packet that is transmitted. It would be terribly inefficient for every computer on the network to actually process the ARP broadcast packet. Instead, the NICs of the devices on the network for whom the packet is not destined recognize that the packet is of no use to them, and the packet is discarded rather than being passed to the CPU for processing.

The discarding of packets not destined for the receiving host improves processing efficiency, but it's not so great for packet analysts. As analysts, we typically want to see every packet sent across the wire so that we don't risk missing some crucial piece of information.

We can ensure we capture all of the traffic by using the NIC's promiscuous mode. When operating in promiscuous mode, the NIC passes every packet it sees to the host's processor, regardless of addressing. Once the packet makes it to the CPU, it can then be grabbed by a packet-sniffing application for analysis.

Most modern NICs support promiscuous mode, and Wireshark includes the libpcap/WinPcap driver, which allows it to switch your NIC directly into promiscuous mode from the Wireshark GUI. (We'll talk more about libpcap/WinPcap in Chapter 3.)

For the purposes of this book, you must have a NIC and an operating system that support the use of promiscuous mode. The only time you do not need to sniff in promiscuous mode is when you want to see only the traffic sent directly to the MAC address of the interface from which you are sniffing.

NOTE *Most operating systems (including Windows) will not let you use a NIC in promiscuous mode unless you have elevated user privileges. If you cannot legally obtain these privileges on a system, chances are that you should not be performing any type of packet sniffing on that particular network.*

Sniffing Around Hubs

Sniffing on a network that has hubs installed is a dream for any packet analyst. As you learned in Chapter 1, traffic sent through a hub goes through every port connected to that hub. Therefore, to analyze the traffic running through a computer connected to a hub, all you need to do is connect a packet sniffer to an empty port on the hub. You will be able to see all communication to and from that computer, as well as all communication between any other devices plugged into that hub. As illustrated in Figure 2-2, your visibility window is limitless when your sniffer is connected to a hub-based network.

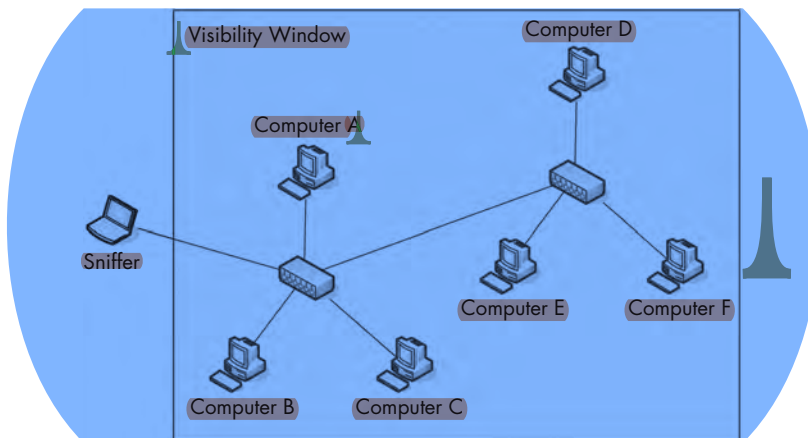


Figure 2-2: Sniffing on a hub network provides a limitless visibility window.

NOTE The visibility window, as shown in various diagrams throughout this book, represents the devices on the network whose traffic you can see with a packet sniffer.

Unfortunately for us, hub-based networks are pretty rare because of the headaches they cause network administrators. Because only one device can communicate at any one time, a device connected through a hub must compete for bandwidth with the other devices trying to communicate through the hub. When two or more devices communicate at the same time, packets collide, as shown in Figure 2-3. The result may be packet loss, and the communicating devices will compensate for that loss by retransmitting packets, which increases network congestion and collisions. As the level of traffic and number of collisions increase, devices may need to transmit a packet three or four times, decreasing network performance dramatically. It's easy to understand why most modern networks of any size use switches.

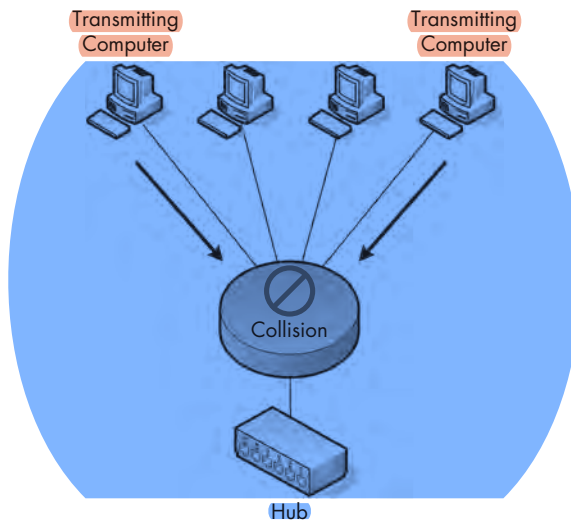


Figure 2-3: Collisions occur on a hub network when two devices transmit at the same time.

Sniffing in a Switched Environment

As discussed in Chapter 1, switches are the most common type of connection device used in modern network environments. They provide an efficient way to transport data via broadcast, unicast, and multicast traffic. As a bonus, switches allow full-duplex communication, meaning that machines can send and receive data simultaneously.

Unfortunately for packet analysts, switches add a whole new level of complexity. When you connect a sniffer to a port on a switch, you can see only broadcast traffic and the traffic transmitted and received by your machine, as shown in Figure 2-4.

There are four primary ways to capture traffic from a target device on a switched network: port mirroring, hubbing out, using a tap, and ARP cache poisoning.

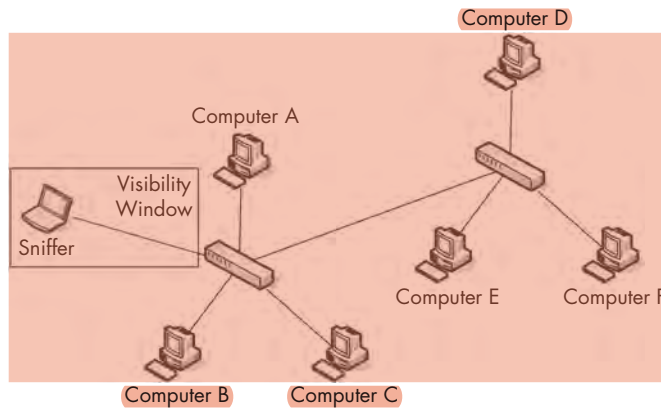


Figure 2-4: The visibility window on a switched network is limited to the port you are plugged into.

Port Mirroring

Port mirroring, or *port spanning*, is perhaps the easiest way to capture the traffic from a target device on a switched network. In this type of setup, you must have access to the command-line or web-management interface of the switch on which the target computer is located. Also, the switch must support port mirroring and have an empty port into which you can plug your sniffer.

To enable port mirroring, you issue a command that forces the switch to copy all traffic on one port to another port. For instance, to capture the traffic from a device on port 3 of a switch, you could simply plug your analyzer into port 4 and mirror port 3 to port 4, allowing you to see all traffic transmitted and received by your target device. Figure 2-5 illustrates port mirroring.

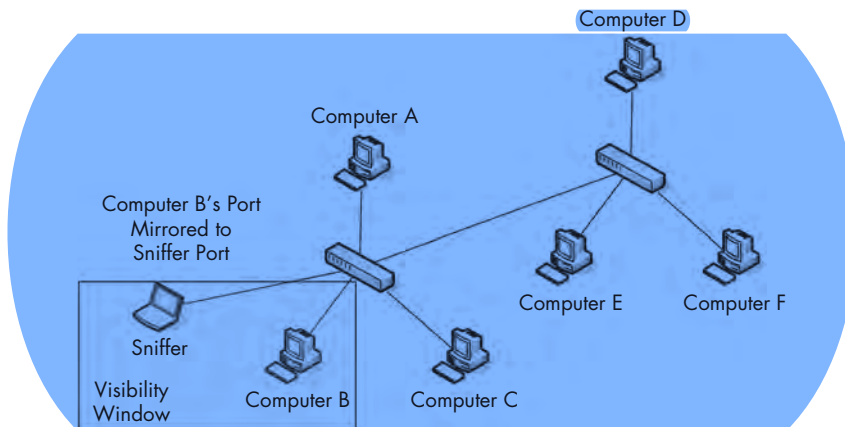


Figure 2-5: Port mirroring allows you to expand your visibility window on a switched network.

The way that you set up port mirroring depends on the manufacturer of your switch. For most switches, you'll need to log in to a command-line interface and enter the port mirroring command. You'll find a list of common port-mirroring commands in Table 2-1.

NOTE *Some switches provide web-based GUIs that offer port mirroring as an option, but these aren't as common and aren't standardized. However, if your switch provides an effective way to configure port mirroring through a GUI, by all means use it.*

Table 2-1: Commands Used to Enable Port Mirroring

Manufacturer	Command
Cisco	<code>set span <source port> <destination port></code>
Enterasys	<code>set port mirroring create <source port> <destination port></code>
Nortel	<code>port-mirroring mode mirror-port <source port> monitor-port <destination port></code>

When port mirroring, be aware of the throughput of the ports you are mirroring. Some switch manufacturers allow you to mirror multiple ports to one individual port, which may be very useful when analyzing the communication between two or more devices on a single switch. However, let's consider what will happen using some basic math. If you have a 24-port switch and you mirror 23 full-duplex 100Mbps ports to one port, you could potentially have 4,600Mbps flowing to that port. This is well beyond the physical threshold of a single port, so it could cause packet loss or network slowdowns if the traffic reached a certain level. In these situations, switches have been known to completely drop excess packets or even "pause" their internal circuitry, preventing communication altogether. Be sure that this type of situation doesn't occur when you are trying to perform your capture.

Hubbing Out

Another way to capture the traffic through a target device on a switched network is by *hubbing out*. This is a technique by which you segment the target device and your analyzer system on the same network segment by plugging them directly into a hub. Many people think of hubbing out as cheating, but it's really a perfect solution in situations where you can't perform port mirroring but still have physical access to the switch the target device is plugged into.

To hub out, all you need is a hub and a few network cables. Once you have your hardware, connect it as follows:

1. Go to the switch the target device resides on and unplug the target from the network.
2. Plug the target's network cable into your hub.
3. Plug in another cable that connects your analyzer to the hub.
4. Plug in a network cable from your hub to the network switch to connect the hub to the network.

Now you have basically put the target device and your analyzer in the same broadcast domain, and all traffic from your target device will be broadcast so that the analyzer can capture those packets, as illustrated in Figure 2-6.

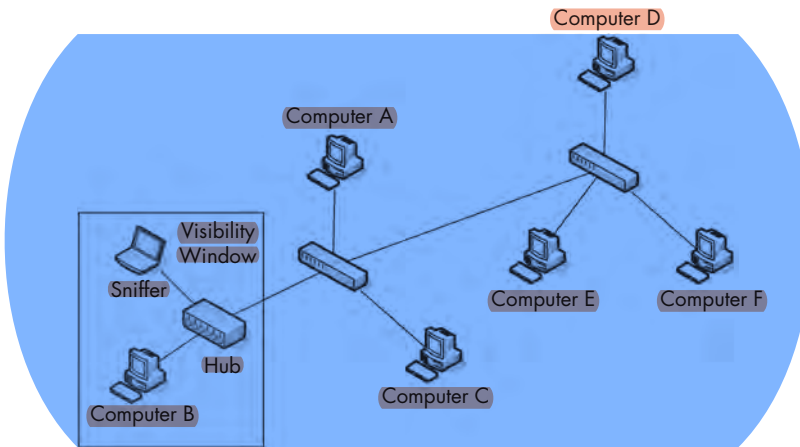


Figure 2-6: Hubbing out isolates your target device and analyzer.

In most situations, hubbing out will reduce the duplex of the target device from full to half. While this method isn't the cleanest way to tap into the wire, it's sometimes your only option when a switch does not support port mirroring. But keep in mind that your hub will also require a power connection, which can be difficult to find in some instances.

NOTE As a reminder, it is usually a nice gesture to alert the user of the device that you will be unplugging it, especially if that user happens to be the company CEO!

FINDING "TRUE" HUBS

When hubbing out, be sure that you're using a true hub and not a falsely labeled switch. Several networking hardware vendors have a bad habit of marketing and selling a device as a hub when it actually functions as a low-level switch. If you aren't working with a proven, tested hub, you will see only your own traffic, not that of the target device.

When you find a hub, test it to make sure it really is a hub. If it is, it's a keeper! The best way to determine whether or not a device is a true hub is to hook up a pair of computers to it and see if one computer can sniff traffic between the other computer and various other devices on the network, such as another computer or a printer. If so, that's a true hub.

Since hubs are so antiquated, they are not really mass-produced anymore. It's almost impossible to buy a true hub off the shelf, so you'll need to be creative in order to find one. A great source is often a surplus auction at your local school district. Public schools are required to attempt to auction surplus items before disposing of them, and they often have older hardware sitting around. I've seen people walk away from surplus auctions with several hubs for less than the cost of a plate of white beans and cornbread. Alternatively, eBay can be a good source of hubs, but be wary, as you may run into the same issue with switches mislabeled as hubs.

Using a Tap

Everybody knows the phrase, “Why have chicken when you can have steak?” (Or if you are from the South, “Why have ham when you can have fried bologna?”) This choice also applies to hubbing out versus using a tap.

A network *tap* is a hardware device that you can place between two points on your cabling system in order to capture the packets between those two points. As with hubbing out, you place a piece of hardware on the network that allows you to capture the packets you need. The difference is that rather than using a hub, you use a specialized piece of hardware designed for network analysis.

There are two primary types of network taps: *aggregated* and *nonaggregated*. Both types of taps sit in between two devices in order to sniff the communications. The primary difference between an aggregated tap and a nonaggregated tap is that the nonaggregated tap has four ports, as shown in Figure 2-7, and the aggregated tap only has three ports.

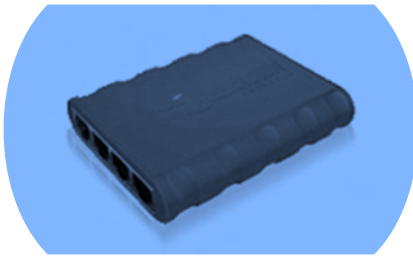


Figure 2-7: A Barracuda nonaggregated tap

Taps also typically require a power connection, although some include batteries for brief stints of packet sniffing without the need to plug into a power receptacle.

Aggregated Taps

The aggregated tap is the simplest to use. It has only one physical monitor port for sniffing bidirectional traffic.

To capture all traffic to and from a single computer plugged into a switch using an aggregated tap, follow these steps:

1. Unplug the computer from the switch.
2. Plug one end of a network cable into the computer, and plug the other end into the tap’s “in” port.
3. Plug one end of another network cable into the tap’s “out” port, and plug the other end into the network switch.
4. Plug one end of a final cable into the tap’s “monitor” port, and plug the other end into the computer that is acting as your sniffer.

The aggregated tap should be connected as shown in Figure 2-8. At this point, your sniffer should be capturing all traffic in and out of the computer you've plugged into the tap.

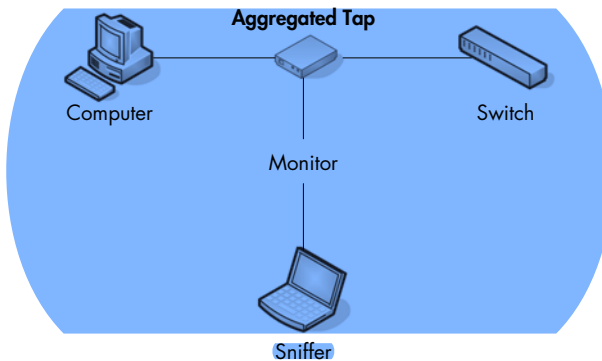


Figure 2-8: Using an aggregated tap to intercept network traffic

Nonaggregated Taps

The nonaggregated tap is slightly more complex than the aggregated type, but it allows a bit more flexibility when capturing traffic. Instead of a single monitor port that can be used to listen to bidirectional communication, the nonaggregated type has two monitor ports. One monitor port is used for sniffing traffic in one direction (from the computer connected to the tap), and the other monitor port is used for sniffing traffic in the other direction (to the computer connected to the tap).

To capture all traffic to and from a single computer plugged into a switch, follow these steps:

1. Unplug the computer from the switch.
2. Plug one end of a network cable into the computer, and plug the other end into the tap's "in" port.
3. Plug one end of another network cable into the tap's "out" port, and plug the other end into the network switch.
4. Plug one end of a third network cable into the tap's "monitor A" port, and plug the other end into one NIC on the computer that is acting as your sniffer.
5. Plug one end of a final cable into the tap's "monitor B" port, and plug the other end into a second NIC on the computer that is acting as your sniffer.

The nonaggregated tap should be connected as shown in Figure 2-9.

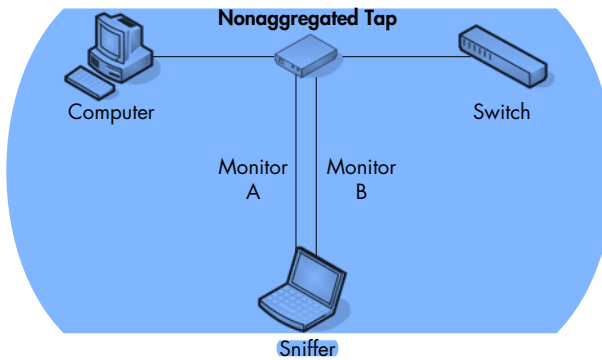


Figure 2-9: Using a nonaggregated tap to intercept network traffic

Choosing a Network Tap

Given the difference between these two types of taps, which one is better? In most situations, aggregated taps are preferred, because they require less cabling and don't need two NICs on your sniffer computer. However, in situations where you are capturing a high volume of traffic or care about traffic going in only one direction, nonaggregated taps are beneficial.

You can purchase taps of all sizes, ranging from about US\$150 for simple Ethernet taps to enterprise-grade fiber-optic taps in the five-figure range. I've used taps from Net Optics and Barracuda Networks, and have been very happy with them. I'm sure that there are many other great taps available.

ARP Cache Poisoning

One of my favorite techniques for tapping into the wire is ARP cache poisoning. We will cover the ARP protocol in detail in Chapter 6, but a brief explanation is necessary in order to understand how this technique works.

The ARP Process

Recall from Chapter 1 that the two main types of packet addressing are at layers 2 and 3 of the OSI model. These layer 2 addresses, or MAC addresses, are used in conjunction with whichever layer 3 addressing system you are using. In this book, in accordance with industry-standard terminology, I refer to the layer 3 addressing system as the *IP addressing system*.

All devices on a network communicate with each other on layer 3 using IP addresses. Because switches operate on layer 2 of the OSI model, they are cognizant of only layer 2 MAC addresses, so devices must be able to include this information in packets they construct. When a MAC address is not known, it must be obtained using the known layer 3 IP addresses to be able to forward traffic to the appropriate device. This translation process is done through the layer 2 protocol ARP.

The ARP process, for computers connected to Ethernet networks, begins when one computer wishes to communicate with another. The transmitting computer first checks its ARP cache to see if it already has the

MAC address associated with the IP address of the destination computer. If it does not, it sends an ARP request to the data link layer broadcast address FF:FF:FF:FF:FF:FF, as discussed in Chapter 1. As a broadcast packet, this packet is received by every computer on that particular Ethernet segment. The packet basically asks, “Which IP address owns the XX:XX:XX:XX:XX:XX MAC address?”

Devices without the destination computer’s IP address simply discard this ARP request. The destination machine replies to the packet with its MAC address via an ARP reply. At this point, the original transmitting computer now has the data link layer addressing information it needs to communicate with the remote computer, and it stores that information in its ARP cache for fast retrieval.

How ARP Cache Poisoning Works

ARP cache poisoning, sometimes called *ARP spoofing*, is the process of sending ARP messages to an Ethernet switch or router with fake MAC (layer 2) addresses in order to intercept the traffic of another computer. Figure 2-10 illustrates this setup.

ARP cache poisoning is an advanced form of tapping into the wire on a switched network. It is commonly used by attackers to send falsely addressed packets to client systems in order to intercept certain traffic or cause denial-of-service (DoS) attacks on a target. However, it can also be a legitimate way to capture the packets of a target machine on a switched network.

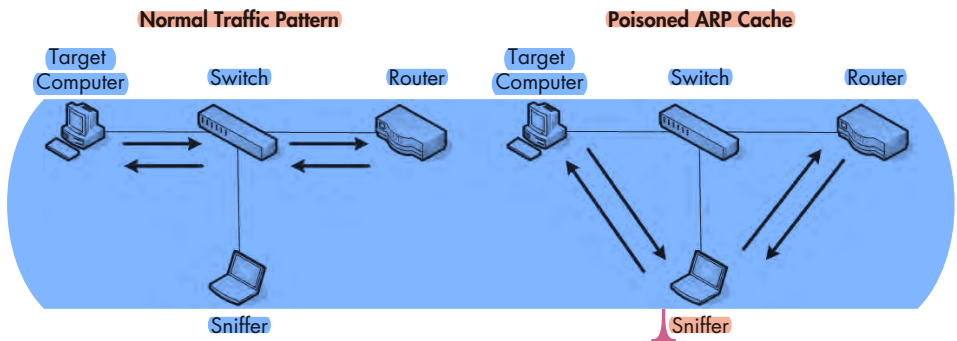


Figure 2-10: ARP cache poisoning allows you to intercept the traffic of your target computer.

Using Cain & Abel

When attempting to poison the ARP cache, the first step is to acquire the required tools and collect some information. For our demonstration, we’ll use the popular security tool Cain & Abel from oxid.it (<http://www.oxid.it/>), which supports Windows systems. Download and install it now, according to the directions on the website.

Before you can use Cain & Abel, you’ll need to collect certain information, including the IP address of your analyzer system, the remote system from which you wish to capture the traffic, and the router from which the remote system is downstream.

When you first open Cain & Abel, you will notice a series of tabs near the top of the window. (ARP cache poisoning is only one of Cain & Abel's features.) For our purposes, we'll be working in the Sniffer tab. When you click this tab, you should see an empty table, as shown in Figure 2-11.

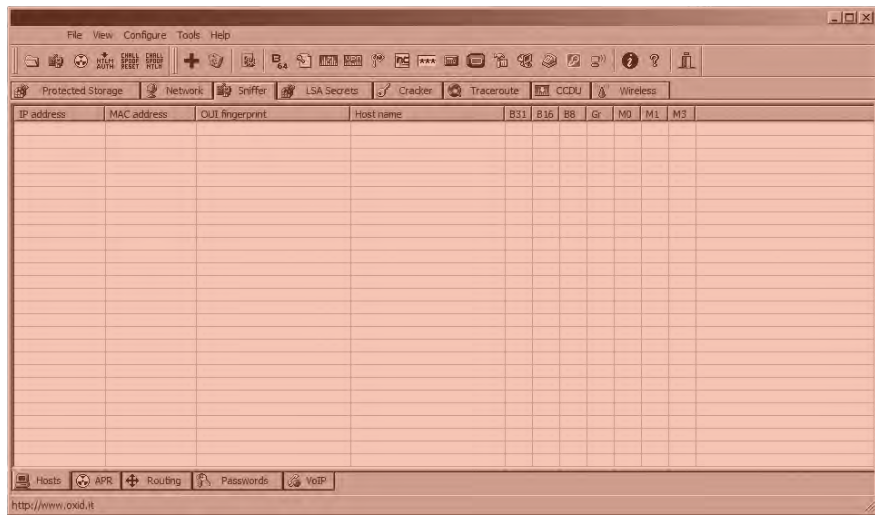


Figure 2-11: The Sniffer tab in the Cain & Abel main window

To complete this table, you will need to activate the program's built-in sniffer and scan your network for hosts. To do so, follow these steps:

1. Click the second icon from the left on the toolbar, which resembles a NIC.
2. You will be asked to select the interface you wish to sniff. This interface should be the one that is connected to the network on which you will be performing your ARP cache poisoning. Select this interface and click **OK**. (Ensure that this button is depressed in order to activate Cain & Abel's built-in sniffer.)
3. To build a list of available hosts on your network, click the plus symbol (+) icon. The MAC Address Scanner dialog appears, as shown in Figure 2-12. The **All hosts in my subnet** radio button should be selected (or you can specify an address range if necessary). Click **OK** to continue.

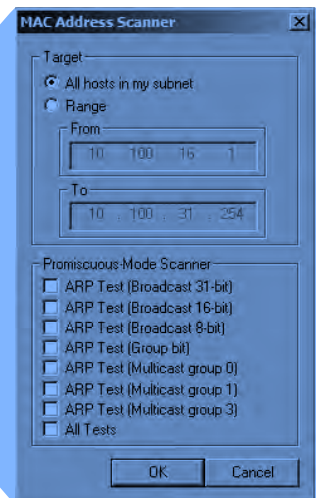


Figure 2-12: The Cain & Abel network discovery tool

The grid should now be filled with a list of all the hosts on your attached network, along with their MAC addresses, IP addresses, and vendor information. This is the list you will work from when setting up ARP cache poisoning.

At the bottom of the program window, you should see a set of tabs that will take you to other windows under the Sniffer heading. Now that you have built your host list, you will be working from the APR (for ARP Poison Routing) tab. Switch to the APR window now by clicking the tab.

Once in the APR window, you are presented with two empty tables. After you've completed the setup steps, the upper table will show the devices involved in your ARP cache poisoning, and the lower one will show all communication between your poisoned machines.

To set up your poisoning, follow these steps:

1. Click in the blank area in the upper portion of the screen, and then click the plus sign (+) icon on the program's standard toolbar.
2. The window that appears has two selection panes. On the left side, you will see a list of all available hosts on your network. Click the IP address of the target computer whose traffic you wish to sniff, and the pane on the right will show a list of all hosts in the network, except for the target machine's IP address.
3. In the right pane, click the IP address of the router that is directly upstream from the target machine, as shown in Figure 2-13, and then click OK. The IP addresses of both devices should now be listed in the upper table in the main application window.
4. To complete the process, click the yellow-and-black radiation symbol on the standard toolbar. This will activate Cain & Abel's ARP cache poisoning features and allow your analyzing system to be the middleman for all communications between the target system and its upstream router.

You should now be able to fire up your packet sniffer and begin the analysis process. When you are finished capturing traffic, simply click the yellow-and-black radiation symbol again to stop ARP cache poisoning.

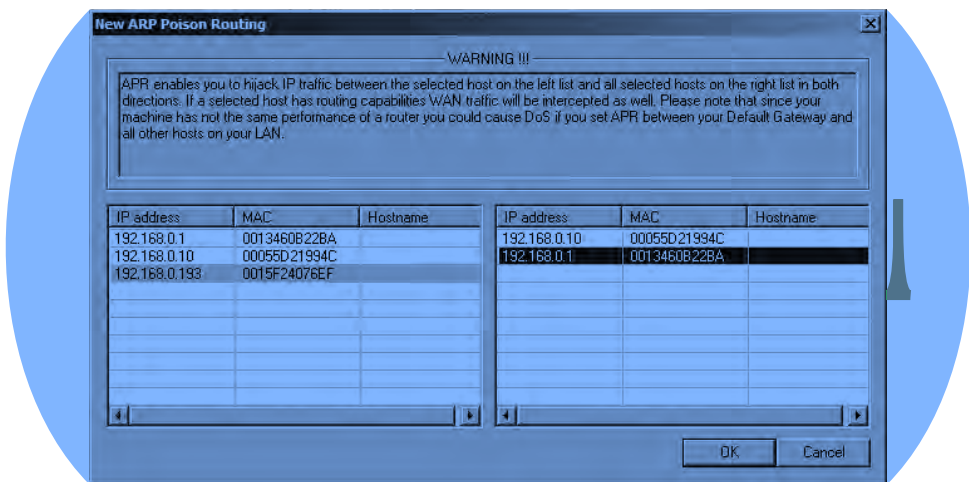


Figure 2-13: Selecting the devices for which you wish to enable ARP cache poisoning

A Word of Caution on ARP Cache Poisoning

As a final note on ARP cache poisoning, you should be very aware of the roles of the systems for which you implement this process. For instance, do not use this technique when the target device is something with very high network utilization, such as a file server with a 1Gbps link to the network (especially if your analyzer system provides only a 100Mbps link).

When you reroute traffic using the technique shown in this example, all traffic transmitted and received by the target system must first go through your analyzer system, therefore making your analyzer the bottleneck in the communication process. This rerouting can create a DoS-type effect on the machine you are analyzing, which will result in degraded network performance and faulty analysis data.

NOTE *You can avoid all the traffic going through your analyzer system by using a feature called asymmetric routing. For more information about this technique, see the oxid.it User Manual (http://www.oxid.it/ca_um/topics/apr.htm).*

Sniffing in a Routed Environment

All of the techniques for tapping into the wire on a switched network are available on routed networks as well. The only major consideration when dealing with routed environments is the importance of sniffer placement when you are troubleshooting a problem that spans multiple network segments.

As you've learned, a device's broadcast domain extends until it reaches a router, at which point the traffic is handed off to the next upstream router. In situations where data must traverse multiple routers, it is important to analyze the traffic on all sides of the router.

For example, consider the communications problem you might encounter in a network with several network segments connected via a variety of routers. In this network, each segment communicates with an upstream segment in order to store and retrieve data. Based on Figure 2-14, the problem we're trying to solve is that a downstream subnet, network D, cannot communicate with any devices on network A.

If you sniff the traffic of a device on network D that is having trouble communicating with devices on other networks, you may clearly see data being transmitted to another segment, but you may not see data coming back. If you rethink the positioning of your sniffer and begin sniffing the traffic in the next upstream network segment (network B), you will have a clearer picture of what is happening. At this point, you may find that traffic is dropped or routed incorrectly by the router of network B. Eventually, this leads you to a router configuration problem that, when corrected, solves your larger dilemma. Although this scenario is a bit broad, the moral of the story is that when dealing with multiple routers and network segments, you may need to move your sniffer around a bit to get the entire picture.

This is a prime example of why it is often necessary to sniff the traffic of multiple devices on multiple segments in order to pinpoint a problem.

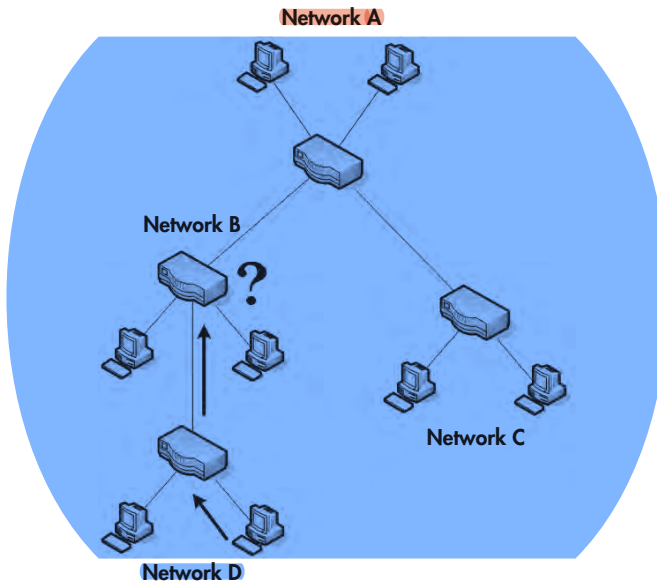


Figure 2-14: A computer on network D can't communicate with one on network A.

NETWORK MAPS

In our discussion of network placement, we have examined several different network maps. A *network map*, or *network diagram*, is a diagram that shows all technical resources on a network and how they are connected.

There is no better way to determine the placement of your packet sniffer than to be able to visualize a network. If you have a network map available, keep it handy, as it will become a valuable asset in the troubleshooting and analysis process. You may even want to make a detailed network map of your own network. Remember that sometimes half the battle in troubleshooting is ensuring you are collecting the right data.

Sniffer Placement in Practice

We have looked at four different ways to capture network traffic in a switched environment. We can add one more if we consider simply installing a packet-sniffing application on a single device from which we want to capture traffic (the *direct install method*). Given these five methods, it can be a bit confusing to determine which one is the most appropriate. Table 2-2 provides some general guidelines for each method.

Table 2-2: Guidelines for Packet Sniffing in a Switched Environment

Technique	Guidelines
Port mirroring	<ul style="list-style-type: none">• Usually preferred because it leaves no network footprint and no additional packets are generated as a result of it.• Can be configured without taking the client offline, which is convenient when mirroring router or server ports.
Hubbing out	<ul style="list-style-type: none">• Ideal when you are not concerned about taking the host temporarily offline.• Ineffective when you must capture traffic from multiple hosts, as collisions and dropped packets will be imminent.• Can result in lost packets on modern 100/1000Mbps hosts because most true hubs are only 10Mbps.
Using a tap	<ul style="list-style-type: none">• Ideal when you are not concerned about taking the host temporarily offline.• The only option when you need to sniff traffic from a fiber-optic connection.• Since taps are made for the task at hand and are up to par with modern network speeds, this method is superior to hubbing out.• May be cost prohibitive when budgets are tight.
ARP cache poisoning	<ul style="list-style-type: none">• Considered very sloppy, as it involves injecting packets onto the network in order to reroute traffic through your sniffer.• Can be effective when you need to grab a quick capture of traffic from a device without taking it offline and where port mirroring is not an option.
Direct install	<ul style="list-style-type: none">• Usually not recommended because if there is an issue with a host, that issue could cause packets to be dropped or manipulated in such a way that they are not represented accurately.• The NIC of the host does not need to be in promiscuous mode.• Best for test environments, examining/baselining performance, and examining capture files created elsewhere.

As analysts, we need to be as stealthy as possible. In a perfect world, we collect the data we need without leaving a footprint. Just as forensic investigators don't want to contaminate a crime scene, we don't want to contaminate our captured network traffic.

As we step through practical scenarios in later chapters, we'll discuss the best ways to capture the data we require on a case-by-case basis. For the time being, the flowchart in Figure 2-15 should help you to decide on the best method to use for capturing traffic. Remember that this flowchart is simply a general reference, and it does not cover every possible iteration of tapping into the wire.

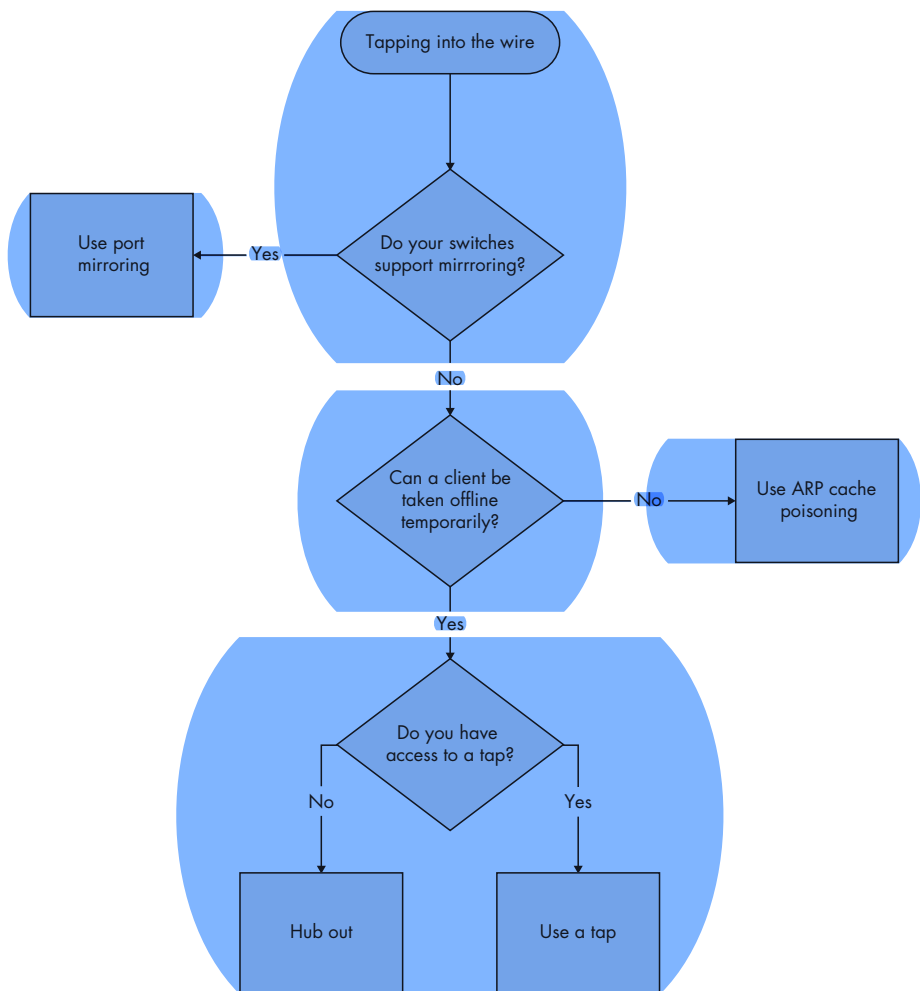


Figure 2-15: A diagram to help determine which method is best for tapping into the wire

3

INTRODUCTION TO WIRESHARK



As mentioned in Chapter 1, several packet-sniffing applications are available for performing network analysis, but we'll use Wireshark in this book. This chapter introduces Wireshark.

A Brief History of Wireshark

Wireshark has a very rich history. Gerald Combs, a computer science graduate of the University of Missouri at Kansas City, originally developed it out of necessity. The first version of Combs's application, called Ethereal, was released in 1998 under the GNU Public License (GPL).

Eight years after releasing Ethereal, Combs left his job to pursue other career opportunities. Unfortunately, his employer at that time had full rights to the Ethereal trademarks, and Combs was unable to reach an agreement that would allow him to control the Ethereal "brand." Instead, Combs and the rest of the development team rebranded the project as *Wireshark* in mid-2006.

Wireshark has grown dramatically in popularity, and its collaborative development team now boasts more than 500 contributors. The program as it exists under the Ethereal name is no longer being developed.

The Benefits of Wireshark

Wireshark offers several benefits that make it appealing for everyday use. It is aimed at both the journeyman and the expert packet analyst, and offers a variety of features to entice each. Let's examine Wireshark according to the criteria defined in Chapter 1 for selecting a packet-sniffing tool.

Supported protocols Wireshark excels in the number of protocols that it supports—more than 850 as of this writing. These range from common ones like IP and DHCP to more advanced proprietary protocols like AppleTalk and BitTorrent. And because Wireshark is developed under an open source model, new protocol support is added with each update.

NOTE *In the unlikely case that Wireshark doesn't support a protocol you need, you can code that support yourself and submit your code to the Wireshark developers for inclusion in the application (if your code is accepted, of course).*

User-friendliness The Wireshark interface is one of the easiest to understand of any packet-sniffing application. It is GUI-based, with very clearly written context menus and a straightforward layout. It also provides several features designed to enhance usability, such as protocol-based color coding and detailed graphical representations of raw data. Unlike some of the more complicated command-line-driven alternatives, like tcpdump, the Wireshark GUI is great for those who are just entering the world of packet analysis.

Cost Since it is open source, Wireshark's pricing can't be beat: Wireshark is released as free software under the GPL. You can download and use Wireshark for any purpose, whether personal or commercial.

NOTE *Although Wireshark may be free, some people have made the mistake of paying for it by accident. If you search for packet sniffers on eBay, you may be surprised by how many people would love to sell you a "professional enterprise license" for Wireshark for the low, low price of \$39.95. Of course, this is a farce, but if you decide you really want to buy it, give me a call, and we can talk about some oceanfront property in Kentucky I have for sale!*

Program support A software package's level of support can make or break it. When dealing with freely distributed software such as Wireshark, there may not be any formal support, which is why the open source community often relies on its user base to provide support. Luckily for us, the Wireshark community is one of the most active of any open source project.

The Wireshark web page links directly to several forms of support, including online documentation, a support and development wiki, FAQs, and a place to sign up for the Wireshark mailing list, which is monitored by most of the program's top developers. Paid support for Wireshark is also available from CACE Technologies through its SharkNet program.

Operating system support Wireshark supports all major modern operating systems, including Windows, Mac OS X, and Linux-based platforms. You can view a complete list of supported operating systems on the Wireshark home page.

Installing Wireshark

The Wireshark installation process is surprisingly simple. However, before you install Wireshark, make sure that your system meets the following requirements:

- 400 MHz processor or faster
- 128MB RAM
- At least 75MB of available storage space
- NIC that supports promiscuous mode
- WinPcap capture driver

The WinPcap capture driver is the Windows implementation of the pcap packet-capturing application programming interface (API). Simply put, this driver interacts with your operating system to capture raw packet data, apply filters, and switch the NIC in and out of promiscuous mode.

Although you can download WinPcap separately (from <http://www.winpcap.org/>), it is typically better to install WinPcap from the Wireshark installation package, because the included version of WinPcap has been tested to work with Wireshark.

Installing on Microsoft Windows Systems

The first step when installing Wireshark under Windows is to obtain the latest installation build from the official Wireshark web page, <http://www.wireshark.org/>. Navigate to the Downloads section on the website and choose a mirror. Once you've downloaded the package, follow these steps:

1. Double-click the .exe file to begin installation, and then click **Next** in the introductory window.
2. Read the licensing agreement, and then click **I Agree** if you agree.
3. Select the components of Wireshark you wish to install, as shown in Figure 3-1. For our purposes, you can accept the defaults by clicking **Next**.

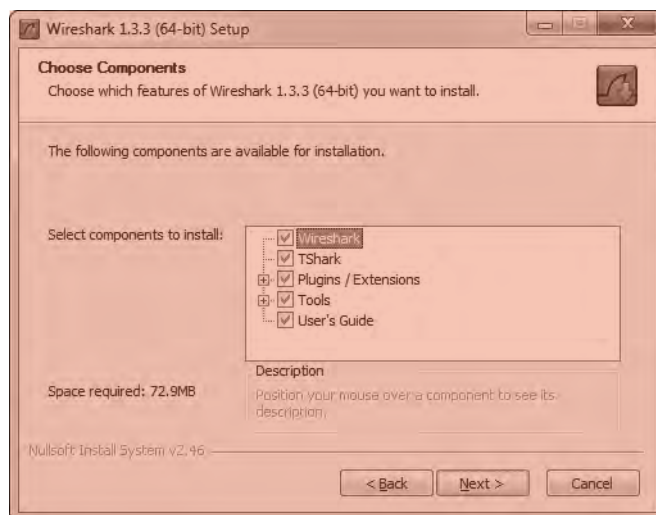


Figure 3-1: Choosing the Wireshark components you wish to install

4. Click **Next** in the Additional Tasks window.
5. Select the location where you wish to install Wireshark, and then click **Next**.
6. When the dialog asks whether you want to install WinPcap, make sure the **Install WinPcap** box is checked, as shown in Figure 3-2, and then click **Install**. The installation process should begin.

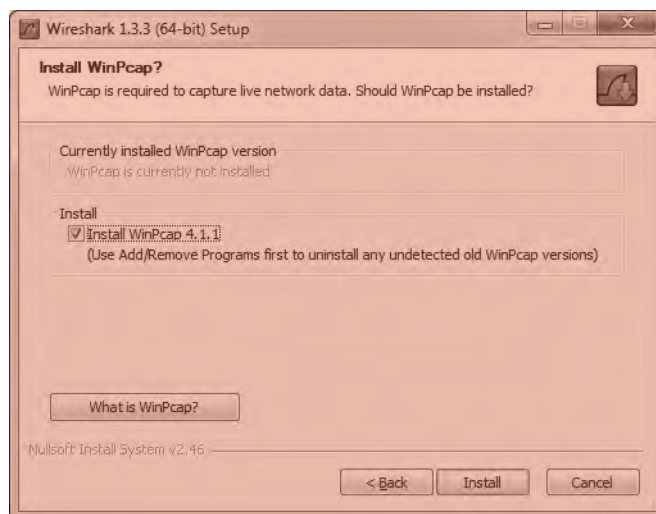


Figure 3-2: Selecting the option to install the WinPcap driver

7. About halfway through the Wireshark installation, the WinPcap installation should start. When it does, click **Next** in the introductory window, read the licensing agreement, and then click **I Agree**.

8. WinPcap should install on your computer. After this installation is complete, click **Finish**.
9. Wireshark should complete its installation. When it's finished, click **Next**.
10. In the installation confirmation window, click **Finish**.

Installing on Linux Systems

The first step when installing Wireshark on a Linux system is to download the appropriate installation package. Not every version of Linux is supported, so don't be surprised if your specific distribution doesn't have its own install package.

Typically, for system-wide software, root access is a requirement. However, local software installations compiled from source can usually be installed without root access.

RPM-based Systems

For RPM-based distributions, such as Red Hat Linux, download the appropriate installation package from the Wireshark web page. Then open a console window and enter the following (substituting the filename of your downloaded package as appropriate):

```
rpm -ivh wireshark-0.99.3.i386.rpm
```

If any dependencies are missing, install them and repeat the Wireshark installation.

DEB-based Systems

On a DEB-based distribution such as Debian or Ubuntu, you can install Wireshark from the system repositories. Open a console window and type the following:

```
apt-get install wireshark
```

Compiling from Source

If your Linux distribution doesn't use an automated package management software, the most effective way to install Wireshark is to compile it from source. To do this, complete the following steps:

1. Download the source package from the Wireshark web page.
2. Extract the archive by typing the following (substituting the filename of your downloaded package as appropriate):

```
tar -jxvf wireshark-1.2.2.tar.bz2
```

3. Change into the newly created directory where the files were extracted.

4. As a root-level user, configure the source so that it will build correctly for your distribution of Linux by using the command `./configure`. If you wish to deviate from the default installation options, you can specify those options at this point in the installation. If any dependencies are missing, you will most likely receive an error. If installation is successful, you should see a message noting success, as shown in Figure 3-3.

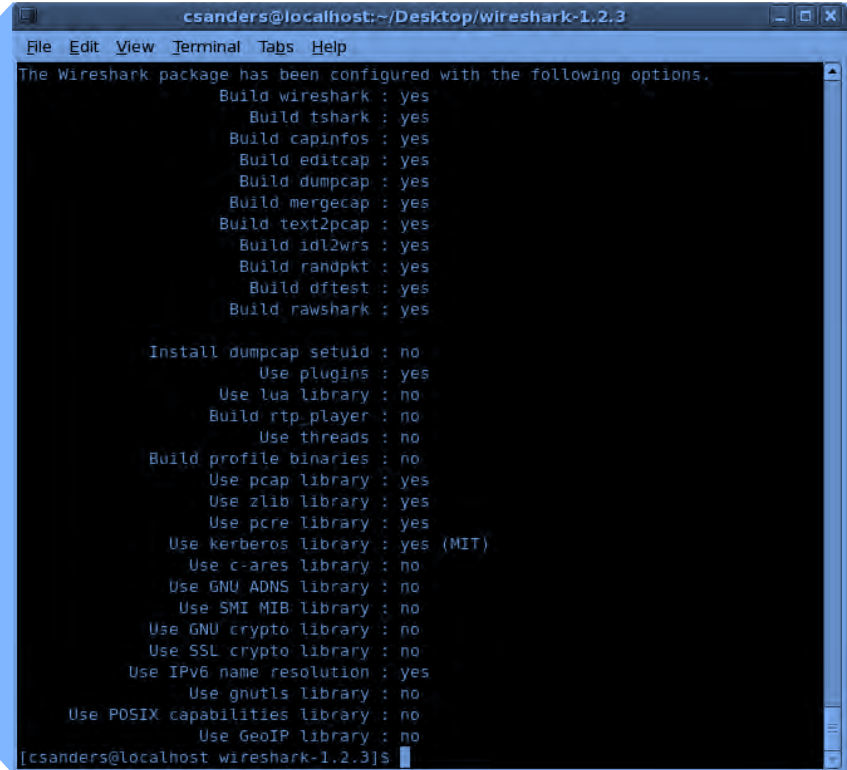
A terminal window titled 'csanders@localhost: ~/Desktop/wireshark-1.2.3' showing the output of the `./configure` command. The output lists various build options and their status, indicating a successful configuration. The options listed are: Build wireshark: yes, Build tshark: yes, Build capinfos: yes, Build editcap: yes, Build dumpcap: yes, Build mergecap: yes, Build text2pcap: yes, Build idl2wrs: yes, Build randpkt: yes, Build oftest: yes, Build rawshark: yes, Install dumpcap setuid: no, Use plugins: yes, Use lua library: no, Build rtp_player: no, Use threads: no, Build profile binaries: no, Use pcap library: yes, Use zlib library: yes, Use pcre library: yes, Use kerberos library: yes (MIT), Use c-ares library: no, Use GNU ADNS library: no, Use SMI MIB library: no, Use GNU crypto library: no, Use SSL crypto library: no, Use IPv6 name resolution: yes, Use gnutls library: no, Use POSIX capabilities library: no, Use GeoIP library: no. The prompt at the bottom is '[csanders@localhost wireshark-1.2.3]\$'.

Figure 3-3: Successful output from the `./configure` command

5. Enter the `make` command to build the source into a binary.
6. Initiate the final installation with `make install`.

Installing on Mac OS X Systems

There are a few caveats for installing Wireshark on Mac OS X Snow Leopard, but installation is not a difficult task and I've outlined the installation steps here. The steps are:

1. Download the DMG package from the Wireshark web page.
2. Copy *Wireshark.app* to the *Applications* folder.
3. Open the *Utilities* folder in *Wireshark.app*.

4. In Finder, click **Go**, and select **Go To Folder**. Enter `/usr/local/bin/` to open that directory.
5. Copy the contents of the *Command Line* folder into `/usr/local/bin/`. You must enter your password in order to do this.
6. In the *Utilities* folder, copy the *ChmodBPF* folder into the *StartupItems* folder. You will need to enter your password again to perform this action and complete the installation.

Wireshark Fundamentals

Once you've successfully installed Wireshark on your system, you can begin to familiarize yourself with it. Now you finally get to open your fully functioning packet sniffer and see... absolutely nothing!

Okay, so Wireshark isn't very interesting when you first open it. In order for things to really get exciting, you need to get some data.

Your First Packet Capture

To get packet data into Wireshark, you'll perform your first packet capture. You may be thinking, "How am I going to capture packets when nothing is wrong on the network?"

First, there is *always* something wrong on the network. If you don't believe me, then go ahead and send an email to all of your network users and let them know that everything is working perfectly.

Secondly, there doesn't need to be something wrong in order for you to perform packet analysis. In fact, most packet analysts spend more time analyzing problem-free traffic than traffic that they are troubleshooting. You need a baseline to compare to in order to be able to effectively troubleshoot network traffic. For example, if you ever hope to solve a problem with DHCP by analyzing its traffic, you must understand what the flow of working DHCP traffic looks like.

More broadly, in order to find anomalies in daily network activity, you must know what normal daily network activity looks like. When your network is running smoothly, you can set your baseline so that you'll know what its traffic looks like in a normal state.

So, let's capture some packets!

1. Open Wireshark.
2. From the main drop-down menu, select **Capture** and then **Interfaces**. You should see a dialog listing the various interfaces that can be used to capture packets, along with their IP addresses.
3. Choose the interface you wish to use, as shown in Figure 3-4, and click **Start**, or simply click the interface under the Interface List section of the welcome page. Data should begin filling the window.

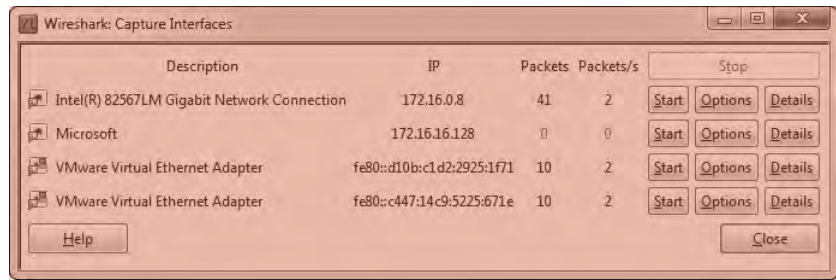


Figure 3-4: Selecting an interface on which to perform your packet capture

- Wait about a minute or so, and when you are ready to stop the capture and view your data, click the **Stop** button from the Capture drop-down menu.

Once you have completed these steps and finished the capture process, the Wireshark main window should be alive with data. As a matter of fact, you might be overwhelmed by the amount of data that appears, but it will all start to make sense very quickly as we break down the main window of Wireshark one piece at a time.

Wireshark's Main Window

You'll spend most of your time in the Wireshark main window. This is where all of the packets you capture are displayed and broken down into a more understandable format. Using the packet capture you just made, let's take a look at Wireshark's main window, as shown in Figure 3-5.

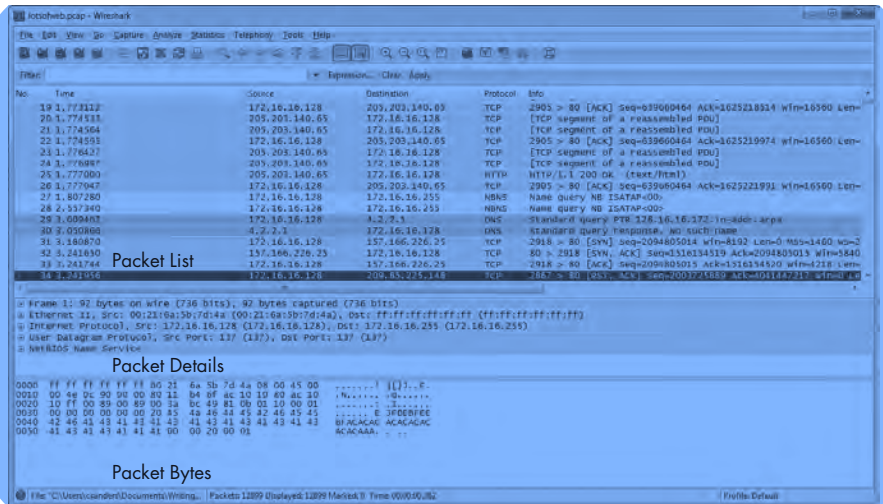


Figure 3-5: The Wireshark main window uses a three-pane design.

The three panes in the main window depend on one another. In order to view the details of an individual packet in the Packet Details pane, you must first select that packet by clicking it in the Packet List pane. Once you've selected your packet, you can see the bytes that correspond with a certain portion of the packet in the Packet Bytes pane when you click that portion of the packet in the Packet Details pane.

NOTE Notice that Figure 3-5 lists a few different protocols in the Packet List pane. There is no visual separation of protocols on different layers; all packets are shown as they are received on the wire.

Here's what each pane contains:

Packet List The top pane displays a table containing all packets in the current capture file. It has columns containing the packet number, the relative time the packet was captured, the source and destination of the packet, the packet's protocol, and some general information found in the packet.

NOTE When I refer to traffic, I am referring to all packets displayed in the Packet List pane. When I refer to DNS traffic specifically, I mean the DNS protocol packets in the Packet List pane.

Packet Details The middle pane contains a hierarchical display of information about a single packet. This display can be collapsed and expanded to show all of the information collected about an individual packet.

Packet Bytes The lower pane—perhaps the most confusing—displays a packet in its raw, unprocessed form; that is, it shows what the packet looks like as it travels across the wire. This is raw information with nothing warm or fuzzy to make it easier to follow.

Wireshark Preferences

Wireshark has several preferences that can be customized to meet your needs. To access Wireshark's preferences, select **Edit** from the main drop-down menu and click **Preferences**. You'll see the Preferences dialog, which contains several customizable options, as shown in Figure 3-6.

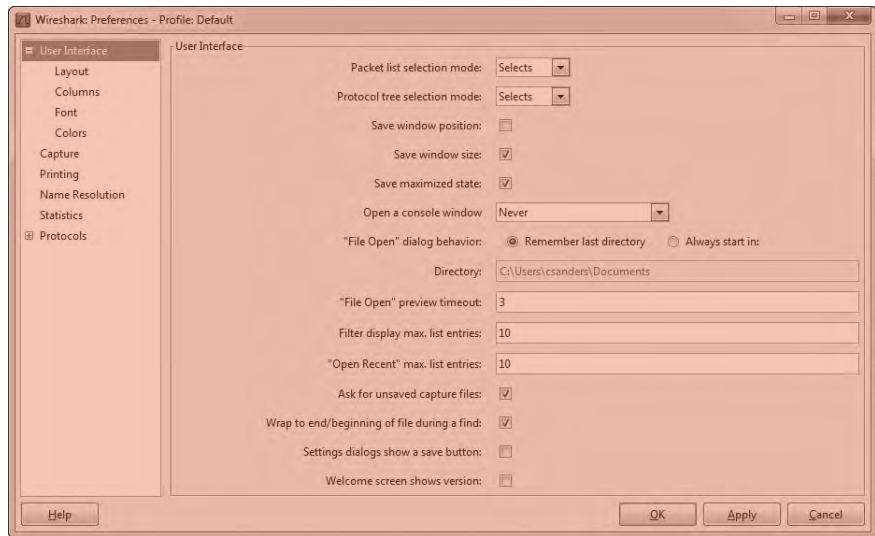


Figure 3-6: You can customize Wireshark using the Preferences dialog options.

Wireshark's preferences are divided into six major sections:

User Interface These preferences determine how Wireshark presents data. You can change most options here according to your personal preferences, including whether or not to save window positions, the layout of the three main panes, the placement of the scroll bar, the placement of the Packet List pane columns, the fonts used to display the captured data, and the background and foreground colors.

Capture These preferences allow you to specify options related to the way packets are captured, including your default capture interface, whether to use promiscuous mode by default, and whether to update the Packet List pane in real time.

Printing The preferences in this section allow you to specify various options related to the way Wireshark prints your data.

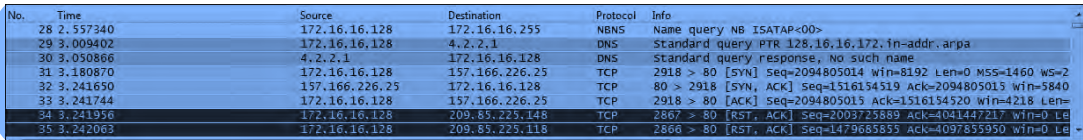
Name Resolution Through these preferences, you can activate features of Wireshark that allow it to resolve addresses into more recognizable names (including MAC, network, and transport name resolution) and specify the maximum number of concurrent name resolution requests.

Statistics This section provides a few configurable options for Wireshark's statistical features.

Protocols The preferences in this section allow you to manipulate options related to the capture and display of the various packets Wireshark is capable of decoding. Not every protocol has configurable preferences, but some have several options that can be changed. These options are best left at their defaults unless you have a specific reason to change them.

Packet Color Coding

If you are anything like me, you may enjoy shiny objects and pretty colors. If that is the case, you probably got excited when you saw all those different colors in the Packet List pane, as in the example in Figure 3-7 (well, the figure is in black and white, but you get the idea). It may seem as if these colors are randomly assigned to each individual packet, but this is not the case.

A screenshot of the Wireshark Packet List pane. The table has five columns: No., Time, Source, Destination, Protocol, and Info. The packets are color-coded: DNS (blue), TCP (green), and RST (red).

No.	Time	Source	Destination	Protocol	Info
28	2.557340	172.16.16.128	172.16.16.255	NDNS	Name query nb ISATAP<00>
29	3.009402	172.16.16.128	4.2.2.1	DNS	Standard query PTR 128.16.16.172.in=addr.arpa
30	3.050866	4.2.2.1	172.16.16.128	DNS	Standard query response, No such name
31	3.180870	172.16.16.128	157.166.226.25	TCP	2918 > 80 [SYN] Seq=2094805014 Win=8192 Len=0 MSS=1460 WS=2
32	3.241650	157.166.226.25	172.16.16.128	TCP	80 > 2918 [SYN, ACK] Seq=1516154519 Ack=2094805013 Win=5840
33	3.241744	172.16.16.128	157.166.226.25	TCP	2918 > 80 [ACK] Seq=2094805015 Ack=1516154520 Win=4218 Len=0
34	3.241956	172.16.16.128	209.85.225.148	TCP	2867 > 80 [RST, ACK] Seq=2003725889 Ack=4041447217 Win=0 Len=0
35	3.242063	172.16.16.128	209.85.225.118	TCP	2866 > 80 [RST, ACK] Seq=1479685859 Ack=4097859550 Win=0 Len=0

Figure 3-7: Wireshark's color coding allows for quick protocol identification.

Each packet is displayed as a certain color for a reason. These colors reflect the packet's protocol. For example, all DNS traffic is blue, and all HTTP traffic is green. The color coding allows you to quickly differentiate between various protocols so that you don't need to read the protocol field in the Packet List pane for each individual packet. You will find that this greatly speeds up the time it takes to browse through large capture files.

Wireshark makes it easy to see which colors are assigned to each protocol through the Coloring Rules window, shown in Figure 3-8. To open this window, select **View** from the main drop-down menu and click **Coloring Rules**.

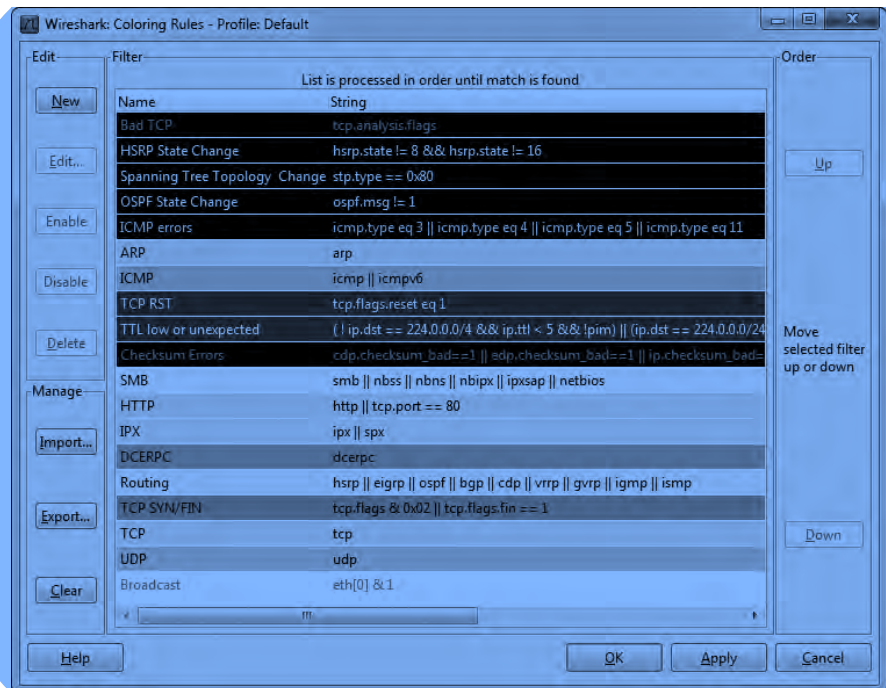


Figure 3-8: The Coloring Rules window allows you to view and modify the coloring of packets.

You can define your own coloring rules and modify existing ones. For example, to change the color used as the background for HTTP traffic from the default green to lavender, follow these steps:

1. Open Wireshark and access the Coloring Rules window (**View ▶ Coloring Rules**).
2. Find the HTTP coloring rule in the coloring rules list and select it by clicking it once.
3. Click the **Edit** button. You'll see the Edit Color Filter dialog, as shown in Figure 3-9.

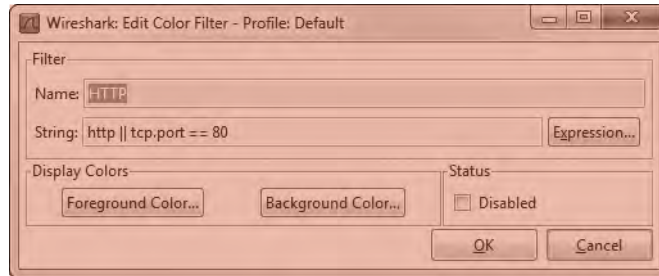


Figure 3-9: When editing a color filter, you can modify both the foreground and background colors.

4. Click the **Background Color** button.
5. Select the color you wish to use on the color wheel, and then click **OK**.
6. Click **OK** twice more to accept the changes and return to the main window. The main window should then reload itself to reflect the updated color scheme.

As you work with Wireshark on your network, you will begin to notice that you deal with certain protocols more than others. Here's where color-coded packets can make your life a lot easier. For example, if you think that there is a rogue DHCP server on your network handing out IP leases, you could simply modify the coloring rule for the DHCP protocol so that it shows up in bright yellow (or some other easily identifiable color). This would allow you to pick out all DHCP traffic much more quickly, and make your packet analysis more efficient.

These coloring rules can also be further extended by creating them based on your own custom filters.

Now that you have Wireshark up and running, you're ready to do some packet analysis. The next chapter describes how you can work with the packets you've captured.