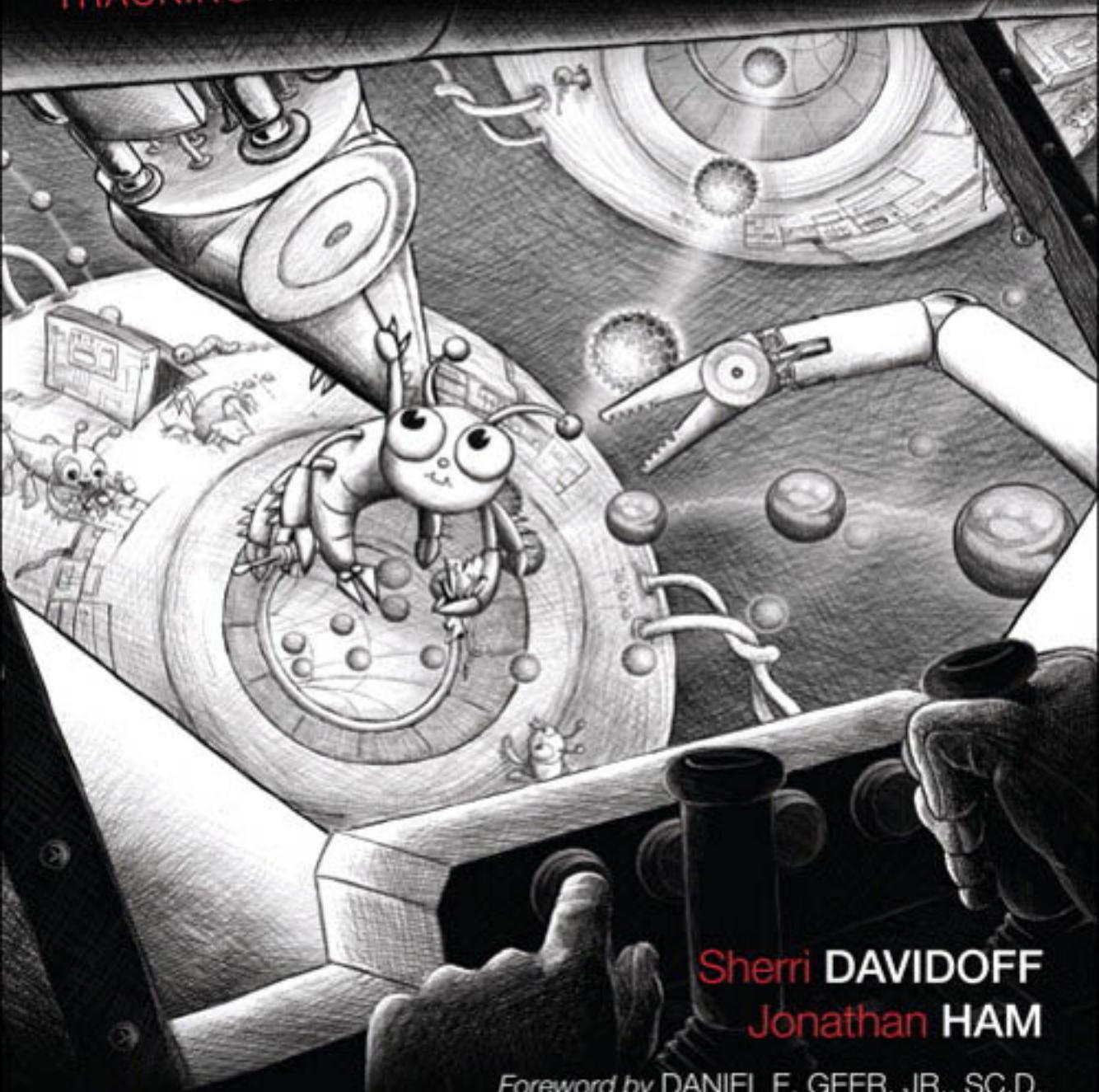




Network Forensics

TRACKING HACKERS THROUGH CYBERSPACE



**Sherri DAVIDOFF
Jonathan HAM**

Foreword by DANIEL E. GEER, JR., SC.D.

Network Forensics

This page intentionally left blank

Network Forensics

Tracking Hackers through Cyberspace

Sherri Davidoff

Jonathan Ham



Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: informati.com/ph

Library of Congress Cataloging-in-Publication Data

Davidoff, Sherri.

Network forensics : tracking hackers through cyberspace / Sherri Davidoff, Jonathan Ham.

p. cm.

Includes bibliographical references and index.

ISBN 0-13-256471-8 (hardcover : alk. paper)

1. Computer crimes—Investigation. 2. Computer hackers. 3. Forensic sciences. 4. Computer crimes—Investigation—Case studies. I. Ham, Jonathan. II. Title.

HV8079.C65D348 2012

363.25'968—dc23

2012014889

Copyright © 2012 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-256471-7

ISBN-10: 0-13-256471-8

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.

First printing, May 2012

To my mother, father, and sister.
Thank you for teaching me to follow my dreams
and for your endless encouragement and support.

SD

For Charlie and Violet,
and the rest of my family,
whose patience has made this possible.

JH

“May you find what you seek.”

-An “ancient curse”; origin unknown.

Contents

Foreword	xvii
Preface	xix
0.1 The Changing Landscape	xix
0.2 Organization	xxi
0.2.1 Part I, “Foundation”	xxi
0.2.2 Part II, “Traffic Analysis”	xxii
0.2.3 Part III, “Network Devices and Servers”	xxii
0.2.4 Part IV, “Advanced Topics”	xxiii
0.3 Tools	xxiii
0.4 Case Studies	xxiii
0.5 Errata	xxiv
0.6 Final Notes	xxiv
Acknowledgments	xxv
About the Authors	xxvii
Part I Foundation	1
Chapter 1 Practical Investigative Strategies	3
1.1 Real-World Cases	3
1.1.1 Hospital Laptop Goes Missing	4
1.1.2 Catching a Corporate Pirate	6
1.1.3 Hacked Government Server	7
1.2 Footprints	8
1.3 Concepts in Digital Evidence	9
1.3.1 Real Evidence	10
1.3.2 Best Evidence	11
1.3.3 Direct Evidence	12
1.3.4 Circumstantial Evidence	12
1.3.5 Hearsay	13
1.3.6 Business Records	14
1.3.7 Digital Evidence	15
1.3.8 Network-Based Digital Evidence	15
	vii

1.4 Challenges Relating to Network Evidence	16
1.5 Network Forensics Investigative Methodology (OSCAR)	17
1.5.1 Obtain Information	17
1.5.2 Strategize	18
1.5.3 Collect Evidence	19
1.5.4 Analyze	20
1.5.5 Report	21
1.6 Conclusion	22
Chapter 2 Technical Fundamentals	23
2.1 Sources of Network-Based Evidence	23
2.1.1 On the Wire	24
2.1.2 In the Air	24
2.1.3 Switches	25
2.1.4 Routers	25
2.1.5 DHCP Servers	26
2.1.6 Name Servers	26
2.1.7 Authentication Servers	27
2.1.8 Network Intrusion Detection/Prevention Systems	27
2.1.9 Firewalls	27
2.1.10 Web Proxies	28
2.1.11 Application Servers	29
2.1.12 Central Log Servers	29
2.2 Principles of Internetworking	30
2.2.1 Protocols	30
2.2.2 Open Systems Interconnection Model	31
2.2.3 Example: Around the World ... and Back	33
2.3 Internet Protocol Suite	35
2.3.1 Early History and Development of the Internet Protocol Suite	36
2.3.2 Internet Protocol	37
2.3.3 Transmission Control Protocol	41
2.3.4 User Datagram Protocol	43
2.4 Conclusion	44
Chapter 3 Evidence Acquisition	45
3.1 Physical Interception	46
3.1.1 Cables	46
3.1.2 Radio Frequency	50
3.1.3 Hubs	51
3.1.4 Switches	52
3.2 Traffic Acquisition Software	54
3.2.1 libpcap and WinPcap	55
3.2.2 The Berkeley Packet Filter (BPF) Language	55
3.2.3 tcpdump	59
3.2.4 Wireshark	64
3.2.5 tshark	64

3.2.6	dumpcap	64
3.3	Active Acquisition	65
3.3.1	Common Interfaces	66
3.3.2	Inspection Without Access	70
3.3.3	Strategy	71
3.4	Conclusion	72

Part II Traffic Analysis 73

Chapter 4	Packet Analysis	75
4.1	Protocol Analysis	76
4.1.1	Where to Get Information on Protocols	76
4.1.2	Protocol Analysis Tools	79
4.1.3	Protocol Analysis Techniques	82
4.2	Packet Analysis	95
4.2.1	Packet Analysis Tools	96
4.2.2	Packet Analysis Techniques	99
4.3	Flow Analysis	103
4.3.1	Flow Analysis Tools	105
4.3.2	Flow Analysis Techniques	109
4.4	Higher-Layer Traffic Analysis	120
4.4.1	A Few Common Higher-Layer Protocols	120
4.4.2	Higher-Layer Analysis Tools	129
4.4.3	Higher-Layer Analysis Techniques	131
4.5	Conclusion	133
4.6	Case Study: Ann’s Rendezvous	135
4.6.1	Analysis: Protocol Summary	135
4.6.2	DHCP Traffic	136
4.6.3	Keyword Search	138
4.6.4	SMTP Analysis—Wireshark	141
4.6.5	SMTP Analysis—TCPFlow	143
4.6.6	SMTP Analysis—Attachment File Carving	146
4.6.7	Viewing the Attachment	147
4.6.8	Finding Ann the Easy Way	150
4.6.9	Timeline	154
4.6.10	Theory of the Case	155
4.6.11	Response to Challenge Questions	155
4.6.12	Next Steps	157

Chapter 5 Statistical Flow Analysis 159

5.1	Process Overview	160
5.2	Sensors	161
5.2.1	Sensor Types	162
5.2.2	Sensor Software	163
5.2.3	Sensor Placement	164

5.2.4	Modifying the Environment	165
5.3	Flow Record Export Protocols	166
5.3.1	NetFlow	166
5.3.2	IPFIX	167
5.3.3	sFlow	167
5.4	Collection and Aggregation	168
5.4.1	Collector Placement and Architecture	169
5.4.2	Collection Systems	170
5.5	Analysis	172
5.5.1	Flow Record Analysis Techniques	172
5.5.2	Flow Record Analysis Tools	177
5.6	Conclusion	183
5.7	Case Study: The Curious Mr. X	184
5.7.1	Analysis: First Steps	185
5.7.2	External Attacker and Port 22 Traffic	186
5.7.3	The DMZ Victim—10.30.30.20 (aka 172.30.1.231)	189
5.7.4	The Internal Victim—192.30.1.101	193
5.7.5	Timeline	194
5.7.6	Theory of the Case	195
5.7.7	Response to Challenge Questions	196
5.7.8	Next Steps	196
Chapter 6 Wireless: Network Forensics Unplugged		199
6.1	The IEEE Layer 2 Protocol Series	201
6.1.1	Why So Many Layer 2 Protocols?	201
6.1.2	The 802.11 Protocol Suite	202
6.1.3	802.1X	212
6.2	Wireless Access Points (WAPs)	214
6.2.1	Why Investigate Wireless Access Points?	214
6.2.2	Types of Wireless Access Points	215
6.2.3	WAP Evidence	218
6.3	Wireless Traffic Capture and Analysis	219
6.3.1	Spectrum Analysis	220
6.3.2	Wireless Passive Evidence Acquisition	221
6.3.3	Analyzing 802.11 Efficiently	222
6.4	Common Attacks	224
6.4.1	Sniffing	224
6.4.2	Rogue Wireless Access Points	225
6.4.3	Evil Twin	227
6.4.4	WEP Cracking	228
6.5	Locating Wireless Devices	229
6.5.1	Gather Station Descriptors	229
6.5.2	Identify Nearby Wireless Access Points	229
6.5.3	Signal Strength	231
6.5.4	Commercial Enterprise Tools	233

6.5.5	Skyhook	233
6.6	Conclusion	235
6.7	Case Study: HackMe, Inc.	236
6.7.1	Inspecting the WAP	236
6.7.2	Quick-and-Dirty Statistics	242
6.7.3	A Closer Look at the Management Frames	248
6.7.4	A Possible Bad Actor	250
6.7.5	Timeline	251
6.7.6	Theory of the Case	252
6.7.7	Response to Challenge Questions	253
6.7.8	Next Steps	255
Chapter 7 Network Intrusion Detection and Analysis		257
7.1	Why Investigate NIDS/NIPS?	258
7.2	Typical NIDS/NIPS Functionality	258
7.2.1	Sniffing	259
7.2.2	Higher-Layer Protocol Awareness	259
7.2.3	Alerting on Suspicious Bits	260
7.3	Modes of Detection	261
7.3.1	Signature-Based Analysis	261
7.3.2	Protocol Awareness	261
7.3.3	Behavioral Analysis	261
7.4	Types of NIDS/NIPSs	262
7.4.1	Commercial	262
7.4.2	Roll-Your-Own	263
7.5	NIDS/NIPS Evidence Acquisition	264
7.5.1	Types of Evidence	264
7.5.2	NIDS/NIPS Interfaces	266
7.6	Comprehensive Packet Logging	267
7.7	Snort	268
7.7.1	Basic Architecture	268
7.7.2	Configuration	269
7.7.3	Snort Rule Language	269
7.7.4	Examples	273
7.8	Conclusion	275
7.9	Case Study: InterOptic Saves the Planet (Part 1 of 2)	276
7.9.1	Analysis: Snort Alert	277
7.9.2	Initial Packet Analysis	278
7.9.3	Snort Rule Analysis	279
7.9.4	Carving a Suspicious File from Snort Capture	281
7.9.5	“INFO Web Bug” Alert	283
7.9.6	“Tcp Window Scale Option” Alert	284
7.9.7	Timeline	285
7.9.8	Theory of the Case	286
7.9.9	Next Steps	287

Part III Network Devices and Servers	289
Chapter 8 Event Log Aggregation, Correlation, and Analysis	291
8.1 Sources of Logs	292
8.1.1 Operating System Logs	292
8.1.2 Application Logs	300
8.1.3 Physical Device Logs	302
8.1.4 Network Equipment Logs	305
8.2 Network Log Architecture	306
8.2.1 Three Types of Logging Architectures	306
8.2.2 Remote Logging: Common Pitfalls and Strategies	308
8.2.3 Log Aggregation and Analysis Tools	309
8.3 Collecting and Analyzing Evidence	311
8.3.1 Obtain Information	311
8.3.2 Strategize	313
8.3.3 Collect Evidence	314
8.3.4 Analyze	316
8.3.5 Report	317
8.4 Conclusion	317
8.5 Case Study: L0ne Sh4rk's Revenge	318
8.5.1 Analysis: First Steps	319
8.5.2 Visualizing Failed Login Attempts	319
8.5.3 Targeted Accounts	322
8.5.4 Successful Logins	323
8.5.5 Activity Following Compromise	324
8.5.6 Firewall Logs	325
8.5.7 The Internal Victim—192.30.1.101	328
8.5.8 Timeline	330
8.5.9 Theory of the Case	332
8.5.10 Response to Challenge Questions	332
8.5.11 Next Steps	333
Chapter 9 Switches, Routers, and Firewalls	335
9.1 Storage Media	336
9.2 Switches	336
9.2.1 Why Investigate Switches?	337
9.2.2 Content-Addressable Memory Table	337
9.2.3 Address Resolution Protocol	338
9.2.4 Types of Switches	338
9.2.5 Switch Evidence	340
9.3 Routers	340
9.3.1 Why Investigate Routers?	341
9.3.2 Types of Routers	341
9.3.3 Router Evidence	343
9.4 Firewalls	344

9.4.1	Why Investigate Firewalls?	344
9.4.2	Types of Firewalls	344
9.4.3	Firewall Evidence	347
9.5	Interfaces	348
9.5.1	Web Interface	348
9.5.2	Console Command-Line Interface (CLI)	349
9.5.3	Remote Command-Line Interface	350
9.5.4	Simple Network Management Protocol (SNMP)	351
9.5.5	Proprietary Interface	351
9.6	Logging	352
9.6.1	Local Logging	352
9.6.2	Simple Network Management Protocol	353
9.6.3	syslog	354
9.6.4	Authentication, Authorization, and Accounting Logging	355
9.7	Conclusion	355
9.8	Case Study: Ann's Coffee Ring	356
9.8.1	Firewall Diagnostic Commands	357
9.8.2	DHCP Server Logs	358
9.8.3	The Firewall ACLs	359
9.8.4	Firewall Log Analysis	360
9.8.5	Timeline	364
9.8.6	Theory of the Case	365
9.8.7	Responses to Challenge Questions	367
9.8.8	Next Steps	367
Chapter 10 Web Proxies		369
10.1	Why Investigate Web Proxies?	369
10.2	Web Proxy Functionality	371
10.2.1	Caching	371
10.2.2	URI Filtering	373
10.2.3	Content Filtering	373
10.2.4	Distributed Caching	374
10.3	Evidence	375
10.3.1	Types of Evidence	375
10.3.2	Obtaining Evidence	376
10.4	Squid	377
10.4.1	Squid Configuration	377
10.4.2	Squid Access Logfile	378
10.4.3	Squid Cache	379
10.5	Web Proxy Analysis	381
10.5.1	Web Proxy Log Analysis Tools	381
10.5.2	Example: Dissecting a Squid Disk Cache	384
10.6	Encrypted Web Traffic	392
10.6.1	Transport Layer Security (TLS)	394
10.6.2	Gaining Access to Encrypted Content	396

10.6.3 Commercial TLS/SSL Interception Tools	400
10.7 Conclusion	401
10.8 Case Study: InterOptic Saves the Planet (Part 2 of 2)	402
10.8.1 Analysis: pwny.jpg	403
10.8.2 Squid Cache Page Extraction	405
10.8.3 Squid Access.log File	408
10.8.4 Further Squid Cache Analysis	411
10.8.5 Timeline	415
10.8.6 Theory of the Case	417
10.8.7 Response to Challenge Questions	418
10.8.8 Next Steps	419
Part IV Advanced Topics	421
Chapter 11 Network Tunneling	423
11.1 Tunneling for Functionality	423
11.1.1 Background: VLAN Trunking	424
11.1.2 Inter-Switch Link (ISL)	424
11.1.3 Generic Routing Encapsulation (GRE)	425
11.1.4 IPv6 over IPv4 with Teredo	425
11.1.5 Implications for the Investigator	426
11.2 Tunneling for Confidentiality	427
11.2.1 Internet Protocol Security (IPsec)	427
11.2.2 Transport Layer Security (TLS) and Secure Socket Layer (SSL)	428
11.2.3 Implications for the Investigator	430
11.3 Covert Tunneling	430
11.3.1 Covert Tunneling Strategies	430
11.3.2 TCP Sequence Numbers	430
11.3.3 DNS Tunnels	431
11.3.4 ICMP Tunnels	432
11.3.5 Example: ICMP Tunnel Analysis	434
11.3.6 Implications for the Investigator	438
11.4 Conclusion	439
11.5 Case Study: Ann Tunnels Underground	441
11.5.1 Analysis: Protocol Statistics	442
11.5.2 DNS Analysis	443
11.5.3 Quest for Tunneled IP Packets	446
11.5.4 Tunneled IP Packet Analysis	451
11.5.5 Tunneled TCP Segment Analysis	454
11.5.6 Timeline	456
11.5.7 Theory of the Case	456
11.5.8 Response to Challenge Questions	458
11.5.9 Next Steps	459

Chapter 12 Malware Forensics	461
12.1 Trends in Malware Evolution	462
12.1.1 Botnets	462
12.1.2 Encryption and Obfuscation	463
12.1.3 Distributed Command-and-Control Systems	465
12.1.4 Automatic Self-Updates	469
12.1.5 Metamorphic Network Behavior	472
12.1.6 Blending Network Activity	477
12.1.7 Fast-Flux DNS	479
12.1.8 Advanced Persistent Threat (APT)	480
12.2 Network Behavior of Malware	484
12.2.1 Propagation	485
12.2.2 Command-and-Control Communications	487
12.2.3 Payload Behavior	490
12.3 The Future of Malware and Network Forensics	491
12.4 Case Study: Ann's Aurora	492
12.4.1 Analysis: Intrusion Detection	492
12.4.2 TCP Conversation: 10.10.10.10:4444–10.10.10.70:1036	495
12.4.3 TCP Conversations: 10.10.10.10:4445	502
12.4.4 TCP Conversation: 10.10.10.10:8080–10.10.10.70:1035	508
12.4.5 Timeline	513
12.4.6 Theory of the Case	514
12.4.7 Response to Challenge Questions	515
12.4.8 Next Steps	516
Afterword	519
Index	521

This page intentionally left blank

Foreword

My great-grandfather was a furniture maker. I am writing this on his table, sitting in his chair. His world was one of craft, “the skilled practice of a practical occupation.”¹ He made furniture late in life that was in superficial respects the same as that which he made earlier, but one can see his craft advance.

Cybersecurity’s hallmark is its rate of change, both swift incremental change and the intermittent surprise. In the lingo of mathematics, the cybersecurity workfactor is the integral of a brisk flux of step functions punctuated by impulses. My ancestor refined his craft without having to address a change in walnut or steel or linseed. The refinement of craft in cybersecurity is not so easy.

Forensics might at first seem to be a simple effort to explain the past, and thus an affectation. It is not, and the reason is complexity. Complexity is cumulative and, as the authors say at the outset, enough has accumulated that it is impossible to know everything about even a *de minimus* network. Forensics’ purpose, then, is to discover meaningful facts in and about the network and the infrastructure that were not previously known. Only after those facts are known is there any real opportunity to improve the future.

Forensics is a craft. Diligence can and does improve its practice. The process of forensic discovery is dominated by ruling out potential explanations for the events under study. Like sculpture, where the aim is to chip away all the stone that doesn’t look like an elephant, forensics chips away all the ways in which what was observed didn’t happen. In the terms popularized by EF Schumacher, forensics is a convergent problem where cybersecurity is a divergent one; in other words, as more effort is put into forensics, the solution set tends to converge to one answer, an outcome that does not obtain for the general cybersecurity problem.

Perhaps we should say that forensics is not a security discipline but rather an insecurity discipline. Security is about potential events, consistent with Peter Bernstein’s definition: “Risk is simply that more things can happen than will.” Forensics does not have to induce all the possibilities that accumulated complexity can concoct, but rather to deduce the path by which some part of the observable world came to be as it is. Whereas, in general, cybersecurity the offense has a permanent structural advantage, in forensics it is the defense that has superiority.

That forensics is a craft and that forensics holds an innate strategic advantage are factual generalities. For you, the current or potential practitioner, the challenge is to hone your craft to where that strategic advantage is yours—not just theoretically but in operational reality. For that you need this book.

It is the duty of teachers to be surpassed by their students, but it is also the duty of the student to surpass their teacher. The teachers you have before you are very good; surpassing

1. “WordNet Search—3.1,” Princeton University, 2011. <http://wordnetweb.princeton.edu/perl/webwn>.

them will be nontrivial. In the end, a surpassing craft requires knowing what parts of your then current toolbox are eternal and which are subject to the obsolescence that comes with progress. It is likewise expeditious to know what it is that you don't know. For that, this book's breadth is directly useful.

Because every forensics investigation is, in principle, different, the tools that will be needed for one study may well be a different set from those needed for another study. The best mechanics have all the specialized tools they can need, but may use a few tools far more than others. A collection of tools is only so good as your knowledge of it as a collection of tools, not necessarily that you've used each tool within the last week. Nicholas Taleb described the library of Umberto Eco as an anti-library that "...should contain as much of what you do not know as your financial means, mortgage rates, and the real-estate market allows you to put there."

You, dear reader, hold just such an anti-library of forensics in your hand. Be grateful, and study hard.

Daniel E. Geer, Jr., Sc.D.

Preface

Every day, more bits of data flow across the Internet than there are grains of sand on all the beaches in the world. According to the Cisco Visual Networking Index, the total global IP traffic for 2011 was forecast to be approximately $8.4 * 10^{18}$ bits per day. Meanwhile, mathematicians at the University of Hawaii have estimated the number of grains of sand on all the beaches in the world to be approximately $7.5 * 10^{18}$ grains. According to Cisco, global IP traffic is expected to increase at an annual growth rate of 32% per year, so by the time you read this, the number of bits of data flowing across the Internet *every day* may have *far* exceeded the estimated number of grains of sand on all the beaches in the world.^{2, 3, 4}

Of course, these estimates are very rough, because in both cases the systems involved are far larger and more complex than humanity has the tools to quantify. The Internet has long since passed the point where we can fully analyze and comprehend its workings. We can understand bits and pieces of it and we can make broad generalizations; but the fact is that we humans have already created a monster far more powerful and complex than we can ever fully understand.

In this environment a new, endless field of study has emerged: network forensics. Forensics, in general, is “the application of scientific knowledge to legal problems, especially scientific analysis of physical evidence, as from a crime scene.” Network forensics therefore refers to the scientific study of network-based evidence, commonly applied to legal questions. Of course, network forensics is a field of study independent of any specific legal case, and many of the scientific advances, tools, and techniques developed for the purposes of legal investigation can also be used for social study, historical analysis, and scientific exploration of network environments. In this book, we have endeavored to provide a technical foundation that will be practically useful not just for professional network forensic analysts conducting legal investigations, but also for students, independent researchers, and all those who are curious.

0.1 The Changing Landscape

The Internet is constantly changing. As new features are developed in hardware and software, new protocols are implemented to reflect those changes and old protocols are adapted and revised to suit the latest technology. In the past decade, we have seen the emergence of

2. Cisco estimates the total global IP traffic for 2011 at 28,023 petabytes per month. Dividing this by 30 days in one month, we get approximately $8.4 * 10^{18}$ bits each day.

3. “Networking Solutions,” *Cisco*, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.

4. Howard McAllister, “Grains of Sand on the World’s Beaches,” 21st Century Problem Solving, <http://www.hawaii.edu/suremath/jssand.html>.

protocols for distributed peer-to-peer video chat systems, protocols for conducting surgery on people from thousands of miles away, and protocols for driving robots halfway around the world.

Network forensics can seem daunting to investigators who are familiar with traditional filesystem forensics. There are a relatively small number of filesystem formats in widespread use, compared with hundreds of network protocols. On Windows systems, it is common to find FAT32 or NTFS filesystems. On UNIX/Linux systems, it is common to find ext2, 3, 4, ZFS, or HFS Plus filesystems. In contrast, on any given network you may find Ethernet, 802.11b/g/n, ARP, DHCP, IPv4, IPv6, TCP, UDP, ICMP, TLS/SSL, SMTP, IMAP, DNS, HTTP, SMB, FTP, RTP, and many other protocols.

On the Internet, there is no guarantee that any protocol you encounter will match the documented specifications, or that there are any published specifications in the first place. Furthermore, implementations of protocols change frequently. Manufacturers are not required to adhere to any standards, and so they implement the protocol as suits them best for any particular revision of software or hardware.

Sometimes protocols are developed before their time, and the applications that are built on top of them have not matured to the point where they can support all the features of the protocol. In the interim, the protocol or specific fields within it may go unused, or may be repurposed by vendors, standards committees, or hackers for other uses. Sometimes protocols are replaced because the environment has changed so much that the old protocols no longer work as intended. A perfect example of this is IPv4, which worked well in its original, relatively small environment. IPv4 is designed with 32-bit fields to store the source and destination addresses, which accommodates 2^{32} , or approximately 4.3 billion unique addresses. However, large segments of this address space were allocated to different organizations in the early years of the Internet, when there were relatively few users. Now that over a billion people have connected to the Internet, the 32-bit address space is too limited to accommodate the demand. As a result, IPv6 was developed with a far larger 128-bit address space (2^{128} , or 3.4×10^{38} unique addresses). With it have emerged even more protocols, such as Teredo (which is designed to tunnel IPv6 traffic over IPv4-only networks).

Forensics tools also go through changes and revisions as the protocols change. A tool made in 2010 may not correctly interpret a packet capture from 2002, or vice versa. Sometimes the errors can be very subtle, perhaps even undetectable. It is very important for investigators to understand how forensic tools work, and be capable of going down to the lowest layers to verify findings. Network forensics professionals must be highly skilled, highly motivated, and have a great deal of expertise because you can't always rely on tools that other people have written in order to correctly interpret results and perhaps even testify in court.

Compounding these issues is the overwhelming variety of network devices, including routers, switches, application servers, and more. Each system on any given network may have unique configuration, interface, and capabilities. It is not possible for investigators to be familiar with all network devices—or even a significant percentage—including current and past makes and models. Instead, network investigators must be prepared to research and learn about equipment on the fly, while at the same time managing an investigation and projecting an air of confidence. It's a fine balancing act.

Tracking down devices to examine them in the first place can be difficult or even impossible. Anonymity has been a hallmark of the Internet since its early days. While it may

be possible to track an IP address down to a remote ISP, it is often impossible to wrangle any further identifying details out of a third party—particularly if they are located in a foreign country with lax information security laws. Even when the device is located inside your own organization, tracking it down depends on the quality of network documentation and logs, which are often not granular enough to suit investigative needs. With the rise of mobile networks, tracking down devices often feels like a game of hide-and-seek, where the mobile user (perhaps even unknowingly) has the upper hand.

The point is that the Internet functions as an ecosystem. It is not controlled by central forces or “designed” in the way one designs a car. When you examine network traffic, there is no telling what you may encounter or whether your tools will properly parse the specific versions of the protocols in your packet capture. When you gather evidence from network devices or reconfigure them, you may have to research the specific make and model to properly understand the interfaces and the sources of evidence. When you track down systems, you may have to wander all over creation chasing a mobile device, or call dozens of ISP contacts and law enforcement officials in multiple countries to pinpoint the source.

There is no specification to which manufacturers are universally bound to adhere, no set of rules that users around the globe must follow, and no manual that can tell you precisely how to conduct your investigation.

0.2 Organization

This book is designed to provide you with a broad overview of the most important topics in network forensics. It is divided into four parts: “Foundation,” “Traffic Analysis,” “Network Devices and Servers,” and “Advanced Topics.”

0.2.1 Part I, “Foundation”

Part I, “Foundation,” covers the basic concepts of evidence handling, networking, and evidence acquisition. This provides a foundation for more advanced topics, which we cover later in the book. In addition to the topics in these chapters, we strongly recommend that all readers have a good understanding of TCP/IP networking. *TCP/IP Illustrated* by W. Richard Stevens is a fantastic book that we highly recommend as a reference.

Part I includes the following chapters:

- Chapter 1, “Practical Investigative Strategies,” presents a myriad of challenges faced by network forensic investigators, introduces important concepts in digital evidence, and lays out a methodology for approaching network-based investigations.
- Chapter 2, “Technical Fundamentals,” provides a technical overview of common networking components and their value for the forensic investigator, and presents the concept of protocols and the OSI model in the context of network forensic investigations.
- Chapter 3, “Evidence Acquisition,” dives into passive and active evidence acquisition, including hardware and software used for sniffing traffic, as well as strategies for actively collecting evidence from network devices.

0.2.2 Part II, “Traffic Analysis”

Part II, “Traffic Analysis,” discusses the many ways that investigators can analyze network traffic. We begin with packet analysis, from examination of protocol headers to payload extraction and reconstruction. Since flow record data retention is becoming commonplace, we subsequently devote a full chapter to statistical flow record analysis. This is followed by an in-depth look at wireless networks and the 802.11 protocol suite. Finally, we discuss network intrusion detection and prevention systems, which are designed to analyze traffic in real time, produce alerts, and in some cases capture packets on the fly.

Part II includes the following chapters:

- Chapter 4, “Packet Analysis,” is a comprehensive study of protocols, packets, and flows, and methods for dissecting them.
- Chapter 5, “Statistical Flow Analysis,” presents the increasingly important field of statistical flow record collection, aggregation, and analysis.
- Chapter 6, “Wireless: Network Forensics Unplugged,” discusses evidence collection and analysis of wireless networks, specifically focusing on the IEEE 802.11 protocol suite.
- Chapter 7, “Network Intrusion Detection and Analysis,” is a review of network intrusion prevention and detection systems, which are specifically designed to produce security alerts and supporting evidence.

0.2.3 Part III, “Network Devices and Servers”

Part III, “Network Devices and Servers,” covers evidence acquisition and analysis from all kinds of network devices. We begin by discussing event log collection and examination, including challenges and benefits relating to different types of logging architectures. Next, we specifically talk about forensic investigation of switches, routers, and firewalls, which make up the backbone of our networks. Since web proxies have exploded in popularity, and often contain enormously valuable evidence, we close with a detailed discussion of web proxy evidence collection and analysis.

Part III includes the following chapters:

- Chapter 8, “Event Log Aggregation, Correlation, and Analysis,” discusses collection and analysis of logs from various sources, including operating systems of servers and workstations (such as Windows, Linux, and UNIX), applications, network equipment, and physical devices.
- Chapter 9, “Switches, Routers, and Firewalls,” studies the evidence that can be gathered from different types of network equipment and strategies for collecting it, depending on available interfaces and level of volatility.
- Chapter 10, “Web Proxies,” reviews the explosion of web proxies and how investigators can leverage these devices to collect web surfing histories and even cached copies of web objects.

0.2.4 Part IV, “Advanced Topics”

Part IV, “Advanced Topics,” includes a discussion of two of the most fascinating topics in network forensics: network tunneling and malware. We review legitimate and covert network tunnels and discuss investigative strategies for dealing with different types of tunnels. To close, we review a history of malware and the corresponding impact on forensic analysis, including the evolution of command-and-control channels, botnets, IDS/IPS evasion, and the advanced persistent threat (APT).

Part IV includes the following chapters:

- Chapter 11, “Network Tunneling,” discusses both legitimate and covert network tunnels, methods for recognizing tunnels, and strategies for recovering evidence from tunneled traffic.
- Chapter 12, “Malware Forensics,” is a condensed history of malware development, including the evolution of command-and-control channels, botnets, IDS/IPS evasion, and the advanced persistent threat. Along the way, we discuss how malware has changed—and been changed by—forensic investigations.

0.3 Tools

This book is designed to be accessible to a wide audience, and to teach you the fundamental principles and techniques of network forensics. There are many commercial, point-and-click tools that you can also use to reach the same answers, and we briefly touch on a few of those in this book. However, our focus is on tools that are freely available and that can be used to illustrate fundamental techniques. In this way, we hope to give you the ability to understand how forensic tools work at a low level, verify results gleaned from automated tools, and make educated decisions when selecting tools for your investigations.

0.4 Case Studies

Each of the chapters in Parts II, III, and IV includes a detailed case study designed to showcase the tools and techniques discussed in the chapter. You can download the evidence files and practice dissecting them on your own forensic workstation.

The case study evidence files are located here:

<http://lmgsecurity.com/nf/>

They are freely available for your personal use.

0.5 Errata

In any technical book of this size and density, there are bound to be a few errors. We will maintain a list of errata at:

<http://lmgsecurity.com/nf/errata>

If you find an error, we would like to know about it. Please email us at errata@lmgsecurity.com. Check the web site before emailing to make sure the error is not already listed.

0.6 Final Notes

This book is a labor of love. Each chapter required countless hours of research, discussion, debate, and writing. To create the case studies and corresponding packet captures, we first built a laboratory with the equivalent of a small business-sized network, configured and reconfigured it for each exercise, wrote each scenario, and then ran the scenarios over and over again until we got them all exactly right.

It's impossible to count all the late nights and early mornings, flipped circuit breakers and dead hard drives, warm beers and cold pizzas that went into this book. Even though this book is hundreds of pages, we feel that we have only scratched the surface of the very deep field of network forensics. We learned an enormous amount from all the effort, and we hope you do, too.

Acknowledgments

This book would not exist without the support of two widely respected security professionals: Rob Lee and Ed Skoudis. Three years ago, Rob Lee tapped us to create the network forensics curriculum for the SANS Institute. This was the first time that we pooled our joint knowledge on the subject, and actually gave that body of work a name. Since that time, Rob has continued to act as a mentor and friend, constantly pushing us to improve our work, incorporate feedback, and extend the limits of our knowledge. Rob: Thank you for your high standards, your open and honest feedback, and most of all for believing in us. We could not have accomplished this without you.

Ed, in turn, encouraged us to write this book and kindly took the time to introduce us to his editor. His advice on the process proved invaluable. Thank you, Ed, for all of your help and support. We are forever grateful.

Thanks to all the terrific staff at Pearson who put so much time and effort into producing this book, especially our wonderful editor, Chris Guzikowski, as well as Jessica Goldstein, Raina Chrobak, Julie Nahil, Olivia Basegio, Chris Zahn, Karen Gettman, Chuti Prasertsith, John Fuller, and Elizabeth Ryan. A very special thanks as well to the great team at Laserwords for producing this book, especially Patty Donovan for her patience and kindness.

Many thanks to Jonah Elgart for creating the awesome cover illustration. We deeply appreciate the work of Dr. Dan Geer, who kindly wrote the foreword for this book. We are very grateful to our friends and colleagues who conducted technical reviews of the book: Michael Ford, Erik Hjelmvik, Randy Marchany, Craig Wright, and Joshua Wright. Their perspectives and attention to detail made this book infinitely better.

We would like to give a shout-out to our fantastic crew at LMG Security, particularly: Eric Fulton, Jody Miller, Randi Price, Scott Fretheim, David Harrison, and Diane Byrne. Each of them devoted many hours to helping us with network forensics research and curriculum. Eric Fulton was instrumental in developing several of the ForensicsContest.com puzzles, which formed the basis for some of the case studies (in particular, “HackMe” and “Ann’s Aurora”). Jody Miller became our He-Man when he swooped in and conquered the evil forces of Skeletor—*ahem*, we mean, formatted all of the footnotes for the book (nearly 500!).

Thanks to our friends, colleagues, and mentors who have taught us so much over the years: Shane Vannatta, Marsha and Bill Dahlgren, Pohl Longsine, Gary Longsine, John Strand, Michael P. Kenny, Gary and Pue Williams, the good folks at Modwest, Mike Poor, Kevin Johnson, Alan Ptak, Michael Grinberg, Sarah and Kerry Metlen, Anissa Schroeder, Bradley Coleman, Blake Brasher, Stephanie Henry, Nadia Madden and Jon McElroy, Clay Ward, the MIT Student Information Processing Board (SIPB), Wally Deschene, Steven and Linda Abate, Karl Reinhard, Brad Cool, Nick Lewis, Richard Souza, Paul Asadourian, Larry Pesce, George Bakos, Johannes Ullrich, Paul A. Henry, Rick Smith, Guy Bruneau,

Lenny Zeltser, Eric Cole, Judy Novak, Alan Tu, Fabienne van Cappel, Robert C de Baca, Mark Galassi, and Dan Starr.

We would also like to thank everyone who contributed to ForensicsContest.com, whether by creating tools and writeups, submitting comments, or just playing the games. We have learned so much from all of you!

Special thanks to the staff and faculty of the SANS Institute, in particular: Steven and Kathy Northcutt; Deb Jorgensen; Katherine Webb Calhoon; Lana Emery; Kate Marshall; Velda Lempka; Norris Campbell; and Lynn Lewis.

Thanks to our wonderful families for their encouragement and support while we were writing this book, especially Sheila Temkin Davidoff, E. Martin Davidoff, Philip and Lynda Ham, Barbara and Larry Oltmanns, Laura Davidoff, Michele, Kirk, and Naomi Robertson, Latisha, Mike, Makenna, and Braelyn Monnier, Chad, Amy, and Brady Rempel, Sheryl and Tommy Davidoff, Jonathan and Stefanie Davidoff, Jill and Jake Dinov, Jamie and Adam Levine, Annabelle Temkin, Norman and Eileen Shoefeld, Brian and Marie Shoefeld, and Debbie Shoefeld.

Thanks to our little cat, Shark, who stayed curled up by our side for hundreds of hours as we wrote.



Most important of all: Thanks to our two daughters, Charlie and Violet. This book is for you.

About the Authors



Jonathan Ham specializes in large-scale enterprise security issues, from policy and procedure, to scalable prevention, detection, and response techniques. He's been commissioned to teach NCIS investigators how to use Snort, performed packet analysis from a facility more than 2,000 feet underground, taught intrusion analysis to the NSA, and chartered and trained the CIRT for one of the largest U.S. civilian Federal agencies. Jonathan has helped his clients achieve greater success for over fifteen years. He is a Certified Instructor with the SANS Institute, and regularly teaches courses on network security.

Sherri Davidoff has over a decade of experience as an information security professional, specializing in penetration testing, forensics, social engineering testing, and web application assessments. She has consulted for a wide variety of industries, including banking, insurance, health care, transportation, manufacturing, academia, and government. Sherri is a course author for the SANS Institute and has published many articles on privacy and security. She is a GIAC-certified forensic examiner (GCFA) and penetration tester (GPEN), and holds her degree in computer science and electrical engineering from MIT.

Sherri and Jonathan are principals of the security consulting firm, LMG Security. They are married and live in Montana, where they spend their days cracking jokes about TCP/IP and raising their two beautiful daughters, Charlie and Violet.

Source: Photo by Kristine Paulsen Photography.

This page intentionally left blank

Part I

Foundation

Chapter 1, “Practical Investigative Strategies,” presents a myriad of challenges faced by network forensic investigators, introduces important concepts in digital evidence, and lays out a methodology for approaching network-based investigations.

Chapter 2, “Technical Fundamentals,” provides a technical overview of common networking components and their value for the forensic investigator, and presents the concept of protocols and the OSI model in the context of network forensic investigations.

Chapter 3, “Evidence Acquisition,” dives into passive and active evidence acquisition, including hardware and software used for sniffing traffic, as well as strategies for actively collecting evidence from network devices.

This page intentionally left blank

Chapter 1

Practical Investigative Strategies

“A victorious army first wins and then seeks battle; a defeated army first battles and then seeks victory.” —Sun Tsu, The Art of War¹

Evidence scattered around the world. Not enough time. Not enough staff. Unrealistic expectations. Internal political conflicts. Gross underestimation of costs. Mishandling of evidence. Too many cooks in the kitchen. Network forensic investigations can be tricky. In addition to all the challenges faced by traditional investigators, network forensics investigators often need to work with unfamiliar people in different countries, learn to interact with obscure pieces of equipment, and capture evidence that exists only for fleeting moments. Laws surrounding evidence collection and admissibility are often vague, poorly understood, or nonexistent. Frequently, investigative teams find themselves in situations where it is not clear who is in charge, or what the team can accomplish.

An organized approach is key to a successful investigation. While this book is primarily designed to explore technical topics, in this chapter, we touch on the fundamentals of investigative management. This is particularly important because network forensics investigations often involve coordination between multiple groups and evidence that may be scattered around the globe.

We begin by presenting three cases from different industries in order to give you some examples of how network forensics is used to support investigations in the real world. Next, we explore the fundamentals of evidence collection and distinctions that are made between different types of evidence in many courts. We discuss the challenges specific to network-based evidence, such as locating evidence on a network and questions of admissibility. Finally, we present the OSCAR investigative methodology, which is designed to give you an easy-to-remember framework for approaching digital investigations.

1.1 Real-World Cases

How is network forensics used in real life? In this section, we present three cases:

- Hospital Laptop Goes Missing
- Catching a Corporate Pirate
- Hacked Government Server

1. Sun Tsu (Author), Lionel Giles (Translator), *The Art of War*, El Paso Norte Press, 2005.

These cases have been chosen to provide examples of common IT security incidents and illustrate how network forensics is frequently used. Although these cases are based on real-life experiences, they have been modified to protect the privacy of the organizations and individuals involved.

1.1.1 Hospital Laptop Goes Missing

A doctor reports that her laptop has been stolen from her office in a busy U.S. metropolitan hospital. The computer is password-protected, but the hard drive is not encrypted. Upon initial questioning, the doctor says that the laptop may contain copies of some patient lab results, additional protected health information (PHI) downloaded from email attachments, schedules that include patient names, birth dates, and IDs, notes regarding patient visits, and diagnoses.

1.1.1.1 Potential Ramifications

Since the hospital is regulated by the United States' Health Information Technology for Economic and Clinical Health (HITECH) Act and Health Insurance Portability and Accountability Act (HIPAA), it would be required to notify individuals whose PHI was breached.² If the breach is large enough, it would also be required to notify the media. This could cause significant damage to the hospital's reputation, and also cause substantial financial loss, particularly if the hospital were held liable for any damages caused due to the breach.

1.1.1.2 Questions

Important questions for the investigative team include:

1. Precisely when did the laptop go missing?
2. Can we track down the laptop and recover it?
3. Which patient data was on the laptop?
4. How many individuals' data was affected?
5. Did the thief leverage the doctor's credentials to gain any further access to the hospital network?

1.1.1.3 Technical Approach

Investigators began by working to determine the time when the laptop was stolen, or at least when the doctor last used it. This helped establish an outer bound on what data *could* have been stored on it. Establishing the time that the laptop was last in the doctor's possession also gave the investigative team a starting point for searching physical surveillance footage and access logs. The team also reviewed network access logs to determine whether the laptop was subsequently used to connect to the hospital network after the theft and, if so, the location that it connected from.

2. "HITECH Breach Notification Interim Final Rule," U.S. Department of Health and Human Services, <http://www.hhs.gov/ocr/privacy/hipaa/understanding/coveredentities/breachnotificationifr.html>.

There are several ways investigators could try to determine what time the laptop went missing. First, they could interview the doctor to establish the time that she last used it, and the time that she discovered it was missing. Investigators might also find evidence in wireless access point logs, Dynamic Host Control Protocol (DHCP) lease assignment logs, Active Directory events, web proxy logs (if there is an enterprise web proxy), and of course any sort of laptop tracking software (such as Lojack for Laptops) that might have been in use on the device.

Enterprise wireless access point (WAP) logs can be especially helpful for determining the physical location in the facility where a mobile device was most recently connected, and the last time it was connected. In order to ensure uniform availability of wireless networks, enterprises typically deploy a fleet of WAPs that all participate in the same network. Although they appear to the end user as a single wireless network, network operators can view which mobile devices were connected to specific access points throughout the building. Some enterprises even have commercial software that can graphically represent the movement of wirelessly connected devices as they traverse the physical facility. If the laptop was still connected to the hospital's wireless network as the thief exited the building, investigators might be able to use wireless access point logs to show the path that the thief navigated as he or she exited the building. This might also be correlated with video surveillance logs or door access control logs.

Once investigators established an approximate time of theft, they could narrow down the patient information that might have been stored on the system. Email logs could reveal when the doctor last checked her email, which would place an outer bound on the emails that could have been replicated to her laptop. These logs might also reveal which attachments were downloaded. More importantly, the hospital's email server would have copies of all of the doctor's emails, which would help investigators gather a list of patients likely to have been affected by the breach. Similarly, hospital applications that provide access to lab results and other PHI might contain access logs, which could help investigators compile a list of possible data breach victims.

There might be authentication logs from Active Directory domain controllers, VPN concentrators, and other sources that indicate the laptop was used to access hospital resources even after the theft. If so, this might help investigators track down the thief. Evidence of such activities could also indicate that additional information was compromised, and that the attacker's interests went beyond merely gaining a new laptop.

1.1.1.4 Results

Leveraging wireless access point logs, the investigative team was able to pinpoint the time of the theft and track the laptop through the facility out to a visitor parking garage. Parking garage cameras provided a low-fidelity image of the attacker, a tall man wearing scrubs, and investigators also correlated this with gate video of the car itself as it left the lot with two occupants. The video was handed to the police, who were able to track the license plate. The laptop was eventually recovered amongst a stack of stolen laptops.

The investigative team carefully reviewed VPN logs and operating system logs stored on the central logging server and found no evidence that the doctor's laptop was used to attempt any further access to hospital IT resources. Hard drive analysis of the recovered laptop showed no indication that the system had been turned on after the theft. After

extensive consultation with legal counsel, hospital management concluded that patient data had ultimately not been leaked.

In response to the incident, the hospital implemented full-disk encryption for all laptop hard drives, and deployed physical laptop locking mechanisms.

1.1.2 Catching a Corporate Pirate

GlobalCorp, Inc., has a centrally managed intrusion detection system, which receives alerts from sites around the world. Central security staff notice an alert for peer-to-peer (P2P) file-sharing, and on closer inspection see filename references to movies that are still in theaters. Fearing legal ramifications of inaction, they investigate.

1.1.2.1 Potential Ramifications

Management at GlobalCorp, Inc., were highly concerned that an employee was using the company network for trafficking in pirated intellectual property. If this activity were detected, the owner of the intellectual property might sue the company. This case occurred in 2003, at the height of Digital Millennium Copyright Act (DMCA) fervor, and it was assumed that if an individual within the company was illicitly trading pirated music or movies, then it could place the company at risk of costly legal battles.

1.1.2.2 Questions

Important questions for the investigative team include:

1. Where is the source of the P2P traffic physically located?
2. Which user is initiating the P2P traffic?
3. Precisely what data is being shared?

1.1.2.3 Technical Approach

Using the IP address from the IDS alerts, investigators identified the physical site that was the source of the traffic. In order to specifically identify the client system, its location, and primary user, investigators worked with local network management staff.

Meanwhile, intrusion analysts in the central office began capturing all of the P2P-related packets involving the IP address in question. The local facility confirmed that this IP address was part of a local DHCP pool on the wired local area network (LAN). Intrusion analysts reviewed DHCP lease assignment logs for relevant time periods, and recovered the media access control (MAC) address associated with the suspicious activity. From the MAC address investigators identified the manufacturer of the network card (Dell, in this case).

In order to trace the IP address to a specific office, local networking staff logged into switches and gathered information mapping the IP address to a physical port. The physical port was wired to a cubicle occupied by an email system administrator. Investigators entered his office after hours one evening and recovered his desktop for forensic analysis.

Upon examination, however, it was clear that the confiscated desktop was not the source of the P2P activity. The MAC address of the network card in the confiscated system (a Hewlett-Packard desktop) was not consistent with the MAC address linked to the suspicious

activity. Subsequent analysis of the company's email server produced evidence that the suspect, an email system administrator, had leveraged privileged access to read emails of key networking staff involved in the investigation.

Local networking staff took caution to communicate out-of-band while coordinating the remainder of the investigation. Investigators conducted a thorough search of the premises for a system with the MAC address implicated. The matching desktop was eventually found in the desktop staging area, buried in a pile of systems queued for reimaging.

1.1.2.4 Results

Network forensic analysts examined full packet captures grabbed by the IDS, and were ultimately able to carve out video files and reconstruct playable copyrighted movies that were still in theaters. Hard drive analysis of the correct desktop produced corroborating evidence that the movies in the packet capture had been resident on the hard drive. The hard drive also contained usernames and email addresses linking the hard drive and associated network traffic with the suspect.

Case closed!

1.1.3 Hacked Government Server

During a routine antivirus scan, a government system administrator was alerted to suspicious files on a server. The files appeared to be part of a well-known rootkit. The server did not host any confidential data other than password hashes, but there were several other systems on the local subnet that contained Social Security numbers and financial information of thousands of state residents who had filed for unemployment assistance. The administrative account usernames and passwords were the same for all servers on the local subnet.

1.1.3.1 Potential Ramifications

State laws required the government to notify any individuals whose Social Security numbers were breached. If the servers containing this sensitive information were hacked, the state might be required to spend large amounts of money to send out notifications, set up hotlines for affected individuals, and engage in any resulting lawsuits. In addition, disclosure of a breach might damage the careers of high-ranking elected state officials.

1.1.3.2 Questions

Important questions for the investigative team include:

- Was the server in question truly compromised?
- If so, how was the system exploited?
- Were any other systems on the local network compromised?
- Was any confidential information exported?

1.1.3.3 Technical Approach

The server in question appeared to contain files with names that fit the pattern for a well-known rootkit. Investigators began by examining these files and concluded that they were,

indeed, malicious software. The rootkit files were found in the home directory of an old local administrator account that staff had forgotten even existed.

Investigators found that the local authentication logs had been deleted. Fortunately, all servers on the subnet were configured to send logs to a central logging server, so instead investigators reviewed Secure Shell (SSH) logs from the central logging server that were associated with the account. From the SSH logs, it was clear that the account had been the target of a brute-force password-guessing attack. Investigators used visualization tools to identify the times that there were major spikes in the volume of authentication attempts. A subsequent password audit revealed that the account's password was very weak.

The SSH logs showed that the source of the brute-force attack was a system located in Brazil. This was surprising to IT staff because according to network documentation the perimeter firewall was supposed to be configured to block external access to the SSH port of servers on the subnet under investigation. Investigators gathered copies of the current, active firewall configuration and found that it did not match the documented policy—in practice, the SSH port was directly accessible from the Internet. Subsequently, investigators analyzed firewall logs and found entries that corroborated the findings from the SSH logs.

When one system in the environment is compromised, there is a significant probability that the attacker may use credentials from that system to access other systems. IT staff were concerned that the attacker might have used the stolen account credentials to access other systems on the local subnet.

Fortunately, further analysis of the server hard drive indicated that the attacker's access was short-lived; the antivirus scan had alerted on the suspicious files shortly after they were created. Investigators conducted a detailed analysis of authentication logs for all systems on the local subnet, and found no other instances of suspicious access to the other servers. Furthermore, there were no records of logins using the hacked account on any other servers. Extensive analysis of the firewall logs showed no suspicious data exportation from any servers on the local subnet.

1.1.3.4 Results

Investigators concluded that the server under investigation was compromised but that no other systems on the local subnet had been exploited and no personal confidential information had been breached. To protect against future incidents, the state IT staff corrected the errors in the firewall configuration and implemented a policy in which firewall rules were audited at least twice per year. In addition, staff removed the old administrator account and established a policy of auditing all server accounts (including privileges and password strength) on a quarterly basis.

1.2 Footprints

When conducting network forensics, investigators often work with live systems that cannot be taken offline. These may include routers, switches, and other types of network devices, as well as critical servers. In hard drive forensics, investigators are taught to minimize system

modification when conducting forensics. It is much easier to minimize system modification when working with an offline copy of a write-protected drive than with production network equipment and servers.

In network forensics, investigators also work to minimize system modification due to forensic activity. However, in these cases investigators often do not have the luxury of an offline copy. Moreover, network-based evidence is often highly volatile and must be collected through active means that inherently modify the system hosting the evidence. Even when investigators are able to sniff traffic using port monitoring or tapping a cable, there is always some impact on the environment, however small. This impact can sometimes be minimized through careful selection of acquisition techniques, but it can never be eliminated entirely.

Every interaction that an investigator has with a live system modifies it in some way, just as an investigator in real life modifies a crime scene simply by walking on the floor. We use the term “footprint” throughout this book to refer to the impact that an investigator has on the systems under examination.

You will always leave a footprint. Often, the size of the footprint required must be weighed against the need for expediency in data collection. Take the time to record your activities carefully so that you can demonstrate later that important evidence was not modified. Always be conscious of your footprint and tread lightly.

1.3 Concepts in Digital Evidence

What is evidence? The *Compact Oxford English Dictionary* defines “evidence” as:³

evidence (noun)

1. information or signs indicating whether a belief or proposition is true or valid.
2. information used to establish facts in a legal investigation or admissible as testimony in a law court.

In this book, we are concerned with both of the above definitions. Our goal in many investigations is to compile a body of evidence suitable for presentation in court proceedings (even if we hope never to end up in court!). Both relevance to the case and admissibility are important, but the first goal is to ascertain the facts of the matter and understand truly and correctly what has transpired.

Consequently, we define “evidence” in the broadest sense as any observable and recordable event, or artifact of an event, that can be used to establish a true understanding of the cause and nature of an observed occurrence.

Of course, it’s one thing to be able to reconstruct and understand the events that comprise an occurrence, and yet another to be able to demonstrate that in such a way that

3. “Oxford Dictionaries Online—English Dictionary and Language Reference,” *Oxford Dictionaries*, http://www.askoxford.com/concise_oed/evidence?view=uk.

victims can be justly compensated and perpetrators justly punished within our legal framework. Within this system there are a few categories of evidence that have very specific meanings:

- Real
- Best
- Direct
- Circumstantial
- Hearsay
- Business Records

We'll take each of these in turn and discuss their nature and relative usefulness and importance. Due to the rising popularity of electronic communications systems, we also include the following general categories of evidence:

- Digital
- Network-Based Digital

In this book, our discussion of evidence is based on the United States common law system and the U.S. Federal Rules of Evidence (FRE).⁴ Many of these concepts may be similar in your jurisdiction, although we also recommend that you familiarize yourself with the rules specific to your region of the world.

1.3.1 Real Evidence

What is “real” evidence? “Real evidence” is roughly defined as any physical, tangible object that played a relevant role in an event that is being adjudicated. It is the knife that was pulled from the victim’s body. It is the gun that fired the bullet. It is the physical copy of the contract that was signed by both parties. In our realm it is also the physical hard drive from which data is recovered, and all the rest of the physical computer components involved.

Real evidence usually comprises the physicality of the event, and as such is often the most easily presented and understood element of a crime. Human beings understand tangible objects much more readily than abstract concepts, such as data comprised of ones and zeros (which are themselves comprised of the presence or absence of magnetization on microscopic bits of a spinning platter). Unless the hard drive was used as a blunt object in an assault, and as a consequence is covered in identifiable traces of blood and hair follicles (DNA is real evidence too), the judge or jury may have a difficult time envisioning the process through which the evidence reached its current state and was preserved.

4. Committee on the Judiciary House (US) and US House Committee on the Judiciary, *Federal Rules of Evidence (December 2011)* (Committee on the Judiciary, 2011), <http://judiciary.house.gov/hearings/printers/112th/evidence2011.pdf>.

Examples of “real evidence” can include:

- The murder weapon
- The fingerprint or footprint
- The signed paper contract
- The physical hard drive or USB device
- The computer itself—chassis, keyboard, and all

1.3.2 Best Evidence

“Best evidence” is roughly defined as the best evidence that can be produced in court. The FRE states, “To prove the content of a writing, recording, or photograph, the *original* writing, recording, or photograph is required, except as otherwise provided in these rules or by Act of Congress” [emphasis added].⁵ If the original evidence is not available, then alternate evidence of its contents may be admitted under the “best evidence rule.” For example, if an original signed contract was destroyed but a duplicate exists, then the duplicate may be admissible. However, if the original exists and could be admitted, then the duplicate would not suffice.

Our favorite illustration of the “best evidence rule” comes from Dr. Eric Cole, as presented in his SANS courses: Imagine that a helicopter and a tractor trailer collide on a bridge. Real evidence in this case would be the wreckage, but there is no hope of bringing the real evidence into depositions, much less in front of a jury. In such a case the photographs of the scene comprise the best records that can be brought to court. They will have to suffice, and most often do.

Forensic analysts, lawyers, and jurors have questioned what constitutes “original” evidence in the case of digital evidence. Fortunately, the FRE explicitly addresses this issue, as follows:⁶

An “original” of a writing or recording means the writing or recording itself or any counterpart intended to have the same effect by the person who executed or issued it. For electronically stored information, “original” means any printout—or other output readable by sight—if it accurately reflects the information. An “original” of a photograph includes the negative or a print from it. (e) A “duplicate” means a counterpart produced by a mechanical, photographic, chemical, electronic, or other equivalent process or technique that accurately reproduces the original.

In other words, a printout from a computer hard drive that accurately reflects the data would normally be considered “original” evidence.

With network forensics, the bits and bytes being presented have been recorded and may be treated in the same way as a photograph of an event. It is as though we’ve photographed

5. Committee on the Judiciary House (US) and US House Committee on the Judiciary, *Federal Rules of Evidence*, 25.

6. *Ibid.*

the bullet as it traveled through the air. The difference is that network forensic investigators can often reconstruct a forensically identical copy of the entire bullet from the snapshot, rather than just presenting a grainy photograph from which legal teams hope to divine trajectories, masses, and the sending barrel's rifling.

Examples of “best evidence” include:

- A photo of the crime scene
- A copy of the signed contract
- A file recovered from the hard drive
- A bit-for-bit snapshot of a network transaction

1.3.3 Direct Evidence

“Direct evidence” is the testimony offered by a direct witness of the act or acts in question. There are lots of ways that events can be observed, captured, and recorded in the real world, and our court systems try to accommodate most of these when there is relevant evidence in question. Of course, the oldest method is the reportable observation of a fellow human being. This human testimony is classified as “direct evidence,” and it remains some of the most utilized forms of evidence, even if it is often disputed and unreliable.

Direct evidence is usually admissible, so long as it’s relevant. What other people witnessed can have a great impact on a case.

Examples of “direct evidence” can include:

- “I saw him stab that guy.”
- “She showed me an inappropriate video.”
- “I watched him crack passwords using John the Ripper and a password file he shouldn’t have.”
- “I saw him with that USB device.”

1.3.4 Circumstantial Evidence

In contrast to “direct evidence,” “circumstantial evidence” is evidence that does not directly support a specific conclusion. Rather, circumstantial evidence may be linked together with other evidence and used to deduce a conclusion.

Circumstantial evidence is important for cases involving network forensics because it is “the primary mechanism used to link electronic evidence and its creator.”⁷ Often, circumstantial evidence is used to establish the author of emails, chat logs, or other digital evidence. In turn, authorship verification is necessary to establish authenticity, which is required for evidence to be admissible in court. The DoJ elaborates:⁸

7. Scott M. Giordano, “Electronic Evidence and the Law,” *Information Systems Frontiers* 6, no. 2 (June 1, 2004): 165.

8. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” 203.

[D]istinctive characteristics like email addresses, nicknames, signature blocks, and message contents can prove authorship, at least sufficiently to meet the threshold for authenticity . . . For example, in *United States v. Simpson*, 152 F.3d 1241 (10th Cir. 1998), prosecutors sought to show that the defendant had conversed with an undercover FBI agent in an Internet chat room devoted to child pornography. The government offered a printout of an Internet chat conversation between the agent and an individual identified as ‘Stavron’ and sought to show that ‘Stavron’ was the defendant . . . ‘Stavron’ had told the undercover agent that his real name was ‘B. Simpson,’ gave a home address that matched Simpson’s, and appeared to be accessing the Internet from an account registered to Simpson. Further, the police found records in Simpson’s home that listed the name, address, and phone number that the undercover agent had sent to ‘Stavron.’ Accordingly, the government had provided evidence sufficient to support a finding that the defendant was ‘Stavron,’ and the printout was properly authenticated.

Examples of “circumstantial evidence” can include:

- An email signature
- A file containing password hashes on the defendant’s computer
- The serial number of the USB device

1.3.5 Hearsay

“Hearsay” is the label given to testimony offered second-hand by someone who was not a direct witness of the act or acts in question. It is formally defined by the FRE as “a statement, other than one made by the declarant while testifying at the trial or hearing, offered in evidence to prove the truth of the matter asserted.” This includes the comments of someone who may have direct knowledge of an occurrence, but who is unable or unwilling to deliver them directly to the court. Hearsay is generally not ruled admissible, unless it falls into the category of an exception as listed in the FRE (Rules 803 and 804).

Digital evidence can be classified as hearsay if it contains assertions created by people. The U.S. Department of Justice cites “a personal letter; a memo; bookkeeping records; and records of business transactions inputted by persons” as examples of digital evidence that would be classified as hearsay.⁹

However, digital evidence that is generated by a fully automated process with no human intervention is generally *not* considered heresay. The Department of Justice explains:¹⁰

Computer-generated records that do not contain statements of persons therefore do not implicate the hearsay rules. This principle applies both to records generated by a computer without the involvement of a person (e.g., GPS tracking records) and to computer records that are the result of human conduct other than assertions (e.g., dialing a phone number or punching in a PIN at an ATM).

9. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” 192.

10. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” 193.

In some cases, courts have admitted digital evidence using the “business records” exception of the hearsay rule, which we discuss further in the next section. However, the Department of Justice points out that in these cases, the courts overlooked the question of whether the digital evidence should have been classified as hearsay in the first place. “Increasingly … courts have recognized that many computer records result from a process and are not statements of persons—they are thus not hearsay at all.”¹¹

Examples of “hearsay” can include:

- “The guy told me he did it.”
- “He said he knew who did it, and could testify.”
- “I saw a recording of the whole thing go down.”
- A text file containing a personal letter

1.3.6 Business Records

Business records can include any documentation that an enterprise routinely generates and retains as a result of normal business processes, and that is deemed accurate enough to be used as a basis for managerial decisions. The FRE specifically exempts business records from the rule that hearsay is inadmissible, stating that:¹²

The following are not excluded by the rule against hearsay, regardless of whether the declarant is available as a witness: [...] A record of an act, event, condition, opinion, or diagnosis if ... the record was kept in the course of a regularly conducted activity of a business, organization, occupation, or calling, whether or not for profit ...

This can include everything from email and memos to access logs and intrusion detection system (IDS) reports. There may be legally mandated retention periods for some of this data. Other records may be subject to internal retention and/or destruction policies. The bottom line is that if the records are seen as accurate enough by the enterprise that they are the basis for managerial decision making, then the courts usually deem them reliable enough for a proceeding.

Digital evidence has been admitted under the “business records” exception to hearsay many times, although in some cases this was erroneous. The Department of Justice points out that “courts have mistakenly assumed that computer-generated records are hearsay without recognizing that they do not contain the statement of a person.”

Examples of “business records” can include:

- Contracts and other employment agreements
- Invoices and records of payment received

11. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” 191.

12. Committee on the Judiciary House (US) and US House Committee on the Judiciary, *Federal Rules of Evidence*, 17.

- Routinely kept access logs
- /var/log/messages

1.3.7 Digital Evidence

“Digital evidence” is any documentation that satisfies the requirements of “evidence” in a proceeding, but that exists in electronic digital form. Digital evidence may rest in microscopic spots on spinning platters, magnetized to greater or lesser degrees in a somewhat nonvolatile scheme, but regardless, unintelligible except through multiple layers of abstraction and filesystem protocols. In other cases, digital evidence may be charges held in volatile storage, which dissipate within seconds of a loss of power to the system. Digital evidence may be no more tangible, nor permanent, than pulses of photons, radio frequency waves, or differential levels of voltage on copper wires.

Naturally, digital evidence poses challenges for investigators seeking to preserve it and attorneys seeking to admit it in court. In order for evidence to be admissible in United States federal courts, digital evidence must adhere to the same standards as other types of evidence: it must be deemed relevant to the case and authentic. “The standard for authenticating computer records is the same as for authenticating other records . . .,” wrote the U.S. Department of Justice (DoJ) in 2009. “Importantly, courts have rejected arguments that electronic evidence is inherently unreliable because of its potential for manipulation. As with paper documents, the mere possibility of alteration is not sufficient to exclude electronic evidence. Absent specific evidence of alteration, such possibilities go only to the evidence’s weight, not admissibility.”¹³

Examples of “digital evidence” include:

- Emails and IM sessions
- Invoices and records of payment received
- Routinely kept access logs
- /var/log/messages

1.3.8 Network-Based Digital Evidence

“Network-based digital evidence” is digital evidence that is produced as a result of communications over a network. The primary and secondary storage media of computers (e.g., the RAM and hard drives) tend to be fruitful fodder for forensic analysis. Due to data remanence, persistent storage can retain forensically recoverable and relevant evidence for hours, days, even years beyond file deletion and storage reuse. In contrast, network-based digital evidence can be extremely volatile. Packets flit across the wire in milliseconds, vanish from switches in the blink of an eye. Web sites change depending on from where they’re viewed and when.

The requirements for admissibility of network-based digital evidence are murky. Often, the source that generated the evidence is not obtainable or cannot be identified. When the

13. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations” (Office of Legal Education Executive Office for United States Attorneys, 2009), 198–202, <http://www.justice.gov/criminal/cybercrime/ssmanual/ssmanual2009.pdf>.

evidence is a recording of a chat log, blog posting, or email, the identity of the parties in the conversation (and therefore the authors of the statements) may be difficult to prove. When the evidence is a web site, the litigant may need to provide supporting evidence to demonstrate that the image presented in court is what actually existed at the time and location that it was supposedly viewed. For example, “[s]everal cases have considered what foundation is necessary to authenticate the contents and appearance of a website at a particular time. Print-outs of web pages, even those bearing the URL and date stamp, are not self-authenticating.... Thus, courts typically require the testimony of a person with knowledge of the website’s appearance to authenticate images of that website.”¹⁴

There is little case precedent on the admissibility of network packet captures. Depending on the method of capture and the details of the case, packet captures of network traffic may be treated as *recordings* of events, similar to a taped conversation.

Examples of “network-based digital evidence” can include:

- Emails and IM sessions
- Browser activity, including web-based email
- Routinely kept packet logs
- /var/log/messages

1.4 Challenges Relating to Network Evidence

Network-based evidence poses special challenges in several areas, including acquisition, content, storage, privacy, seizure, and admissibility. We will discuss some common challenges below.

- **Acquisition** It can be difficult to locate specific evidence in a network environment. Networks contain so many possible sources of evidence—from wireless access points to web proxies to central log servers—that sometimes pinpointing the correct location of the evidence is tricky. Even when you do know where a specific piece of evidence resides, you may have difficulty gaining access to it for political or technical reasons.
- **Content** Unlike filesystems, which are designed to contain all the contents of files and their metadata, network devices may or may not store evidence with the level of granularity desired. Network devices often have very limited storage capacity. Usually, only selected metadata about the transaction or data transfer is kept instead of complete records of the data that traversed the network.
- **Storage** Network devices commonly do not employ secondary or persistent storage. As a consequence, the data they contain may be so volatile as to not survive a reset of the device.
- **Privacy** Depending on jurisdiction, there may be legal issues involving personal privacy that are unique to network-based acquisition techniques.

14. H. Marshall Jarrett, Director, EOUSA, “Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations,” 204.

- **Seizure** Seizing a hard drive can inconvenience an individual or organization. Often, however, a clone of the original can be constructed and deployed such that critical operations can continue with limited disruption. Seizing a network device can be much more disruptive. In the most extreme cases, an entire network segment may be brought down indefinitely. Under most circumstances, however, investigators can minimize the impact on network operations.
- **Admissibility** Filesystem-based evidence is now routinely admitted in both criminal and civil proceedings. As long as the filesystem-based evidence is lawfully acquired, properly handled, and relevant to the case, there are clear precedents for authenticating the evidence and admitting it in court. In contrast, network forensics is a newer approach to digital investigations. There are sometimes conflicting or even nonexisting legal precedents for admission of various types of network-based digital evidence. Over time, network-based digital evidence will become more prevalent and case precedents will be set and standardized.

1.5 Network Forensics Investigative Methodology (OSCAR)

Like any other forensic task, recovering and analyzing digital evidence from network sources must be done in such a way that the results are both reproducible and accurate. In order to ensure a useful outcome, forensic investigators should perform our activities within a methodological framework. The overall step-by-step process recommended in this book is as follows:

- Obtain information
- Strategize
- Collect evidence
- Analyze
- Report

We refer to this methodology as “OSCAR,” and walk through each of these steps in the following section.

1.5.1 Obtain Information

Whether you’re law enforcement, internal security staff, or a forensic consultant, you will always need to do two things at the beginning of an investigation: obtain information about the incident itself, and obtain information about the environment.

1.5.1.1 The Incident

Usually you will want to know the following things about the incident:

- Description of what happened (as is currently known)
- Date, time, and method of incident discovery

- Persons involved
- Systems and data involved
- Actions taken since discovery
- Summary of internal discussions
- Incident manager and process
- Legal issues
- Time frame for investigation/recovery/resolution
- Goals

This list is simply a starting point, and you will need to customize it for each incident.

1.5.1.2 The Environment

The information you gather about the environment will depend on your level of familiarity with it. Remember that every environment is constantly changing, and complex social and political dynamics occur during an incident. Even if you are very familiar with an organization, you should always take the time to understand how the organization is responding to this particular incident, and clearly establish who needs to be kept in the loop. Usually you will want to know the following things about the environment:

- Business model
- Legal issues
- Network topology (request a network map, etc. if you do not have one)
- Available sources of network evidence
- Organizational structure (request an organizational chart if you do not have one)
- Incident response management process/procedures (forensic investigators are part of the response process and should be at least basically familiar with it)
- Communications systems (is there a central incident communication system/evidence repository?)
- Resources available (staff, equipment, funding, time)

1.5.2 Strategize

It is crucial that early on you take the time to accurately assess your resources and plan your investigation. While this is important for any investigation, it is especially important for network forensics because there are many potential sources of evidence, some of which are also very volatile. Investigators must work efficiently. You will want to regularly confer with others on the investigative/incident response team while planning and conducting the investigation to ensure that everyone is working in concordance and that important developments are communicated.

Source of Evidence	Likely Value	Effort	Volatility	Priority
Firewall logs	High	Medium	Low	2
Web proxy cache	High	Low	Medium	1
ARP tables	Low	Low	High	3

Figure 1–1. An example of evidence prioritization. In this example, we list potential sources of evidence, the likely value, the likely effort to obtain it, and the expected volatility. These values will be different for every investigation.

Here are some tips for developing an investigative strategy:

- Understand the goals and time frame of the investigation.
- List your resources, including personnel, time, and equipment.
- Identify likely sources of evidence.
- For each source of evidence, estimate the value and cost of obtaining it.
- Prioritize your evidence acquisition.
- Plan the initial acquisition/analysis.
- Decide upon method and times of regular communication/updates.
- Keep in mind that after conducting your initial analysis, you may decide to go back and acquire more evidence. Forensics is an iterative process.

Figure 1–1 shows an example of evidence prioritization. In this example, the organization collects firewall logs but stores them in a distributed manner on systems that are not easily accessed. The organization has a web proxy, which is centrally accessed by key security staff. ARP tables can be gathered from any system on the local LAN.

The table lists potential sources of evidence, the likely value for the investigation, the expected effort required to obtain the evidence, and the expected volatility. All of these values are unique to each investigation; every organization has different system configurations, data retention policies, and access procedures. Furthermore, the network equipment, investigative resources, and goals of each investigation vary widely.

Based on this information, we can create our evidence spreadsheet and prioritize accordingly. Next, we would develop a plan for evidence acquisition based on our available resources.

1.5.3 Collect Evidence

In the previous step, “Strategize,” we prioritized our sources of evidence and came up with an acquisition plan. Based on this plan, we then collect evidence from each source. There are three components you must address every time you acquire evidence:

- **Document**—Make sure to keep a careful log of all systems accessed and all actions taken during evidence collection. Your notes must be stored securely and may be

referenced in court. Even if the investigation does not go to court, your notes will still be very helpful during analysis. Be sure to record the date, time, source, method of acquisition, name of the investigator(s), and chain of custody.

- **Capture**—Capture the evidence itself. This may involve capturing packets and writing them to a hard drive, copying logs to hard drive or CD, or imaging hard drives of web proxies or logging servers.
- **Store/Transport**—Ensure that the evidence is stored securely and maintain the chain of custody. Keep an accurate, signed, verifiable log of the persons who have accessed or possessed the evidence.

Since the admissibility of evidence is dependent upon its relevance and reliability, investigators should carefully track the source, method of acquisition, and chain of custody. It's generally accepted that a bit-for-bit image of a hard drive is acceptable in court. For a lot of network-based evidence, the admissibility is not so clear-cut. When in doubt, take careful notes and consult legal counsel.

As with any evidence gathered in the course of an investigation, proper care must be taken to preserve evidence integrity and to document its use and disposition throughout its life cycle (from the initial acquisition to its return to its rightful owner). As we'll see, in some cases this may mean documenting and maintaining the physical chain of custody of a network device. However, in many cases the original incarnation of the evidence being acquired will never be taken into custody.

1.5.3.1 Tips for Evidence Collection

Best practices for evidence collection include:

- Acquire as soon as possible, and lawfully
- Make cryptographically verifiable copies
- Sequester the originals under restricted custody and access (or your earliest copy, when the originals are not available)
- Analyze only the copies
- Use tools that are reputable and reliable
- Document everything you do!

1.5.4 Analyze

Of course the analysis process is normally nonlinear, but certain elements should be considered essential:

- **Correlation** One of the hallmarks of network forensics is that it involves multiple sources of evidence. Much of this will be timestamped, and so the first consideration should be what data can be compiled, from which sources, and how it can be correlated. Correlation may be a manual process, or it may be possible to use tools to do it for you in an automated fashion. We'll look at such tools later on.

- **Timeline** Once the multiple data sources have been aggregated and correlated, it's time to build a timeline of activities. Understanding who did what, when, and how is the basis for any theory of the case. Recognize that you may have to adjust for time skew between sources!
- **Events of Interest** Certain events will stand out as potentially more relevant than others. You'll need to try to isolate the events that are of greatest interest, and seek to understand how they transpired.
- **Corroboration** Due to the relatively low fidelity of data that characterizes many sources of network logs, there is always the problem of "false positives." The best way to verify events in question is to attempt to corroborate them through multiple sources. This may mean seeking out data that had not previously been compiled, from sources not previously consulted.
- **Recovery of additional evidence** Often the efforts described above lead to a widening net of evidence acquisition and analysis. Be prepared for this, and be prepared to repeat the process until such time as the events of interest are well understood.
- **Interpretation** Throughout the analysis process, you may need to develop working theories of the case. These are educated assessments of the meaning of your evidence, designed to help you identify potential additional sources of evidence, and construct a theory of the events that likely transpired. It is of the utmost importance that you separate your interpretation of the evidence from fact. Your interpretation of the evidence is always a hypothesis, which may be proved or disproved.

1.5.5 Report

Nothing you'll have done to this point, from acquisition through analysis, will matter if you're unable to convey your results to others. From that perspective, reporting might be the most important aspect of the investigation. Most commercial forensic tools handle this aspect for the analyst, but usually not in a way that is maximally useful to a lay audience, which is generally necessary.

The report that you produce must be:

- Understandable by nontechnical laypeople, such as:
 - Legal teams
 - Managers
 - Human Resources personnel
 - Judges
 - Juries
- Defensible in detail
- Factual

In short, you need to be able to explain the results of your investigation in terms that will make sense for nontechnical people, while still maintaining scientific rigor. Executive

summaries and high-level descriptions are key, but they must be backed by details that can easily be defended.

1.6 Conclusion

Network forensic investigations pose a myriad of challenges, from distributed evidence to internal politics to questions of evidence admissibility. To meet these challenges, investigators must carefully assess each investigation and develop a realistic strategy that takes into account both the investigative goals and the available resources.

We began this chapter with a series of case studies designed to illustrate how network forensic techniques are applied in real life. Subsequently, we reviewed the fundamental concepts in digital evidence, as employed in the United States common law system, and touched upon the challenges that relate specifically to network-based digital evidence. Finally, we provided you with a method for approaching network forensics investigations.

As Sun Tsu wrote 2,500 years ago: “A victorious army first wins and then seeks battle; a defeated army first battles and then seeks victory.” Strategize first; then collect your evidence and conduct your analysis. By considering the challenges unique to your investigation up front, you will meet your investigative goals most efficiently and effectively.

Chapter 2

Technical Fundamentals

“If you know the enemy and know yourself, you need not fear the results of a hundred battles.” –Sun Tsu, The Art of War¹

The Internet ecosystem is varied and complex. In any network forensic investigation, there are an enormous number of places where evidence may live, some more accessible than others. Often, network-based evidence exists in places that local networking staff may not have considered, leaving it up to the investigator to review the network diagrams and make suggestions for evidence collection.

On the flip side, many networks are configured for functionality and performance, not for monitoring or auditing—and certainly not for forensic investigations. As a consequence, the specific instrumentation or evidence that an investigator might wish for may not exist. Many vendors do not include rich data recording capabilities into their products, and even when they do, such capacities are often disabled by default. For example, local switches may not be configured to export flow record data, or server logfiles might roll over and overwrite themselves on a daily basis.

From Nortel’s line of enterprise-class networking equipment to Linksys’ ubiquitous WRT54G router series in homes and small businesses, organizations can select from a wide variety of equipment for any networking function. As a result, the experience of the network forensic analyst is sure to be varied, inconsistent, and challenging.

In this chapter, we take a high-level look at classes of networking devices, discuss their value for the forensic investigator, and briefly touch upon how their contents might be retrieved. Subsequently, we review the concept of protocols and the OSI model in the context of network forensic investigations. Finally, we talk about key protocols in the Internet Protocol suite, including IPv4, IPv6, TCP, and UDP. This will provide you with a strong technical and conceptual foundation, which we build upon throughout the remainder of this book.

2.1 Sources of Network-Based Evidence

Every environment is unique. Large financial institutions have very different equipment, staff, and network topologies than local government agencies or small health care offices. Even so, if you walk into any organization you will find similarities in network equipment and common design strategies for network infrastructure.

1. Sun Tsu (Author), Lionel Giles (Translator), *The Art of War*, El Paso Norte Press, 2005.

There are many sources of network-based evidence in any environment, including routers, web proxies, intrusion detection systems, and more. How useful is the evidence on each device? This varies depending on the type of investigation, the configuration of the specific device, and the topology of the environment.

Sometimes data from different sources can overlap, which is useful for correlation. Other times the data sources are merely complementary, each containing snippets of evidence not found elsewhere. In this section, we review the types of network equipment that are commonly found in organizations and discuss the types of evidence that can be recovered from each.

2.1.1 On the Wire

Physical cabling is used to provide connectivity between stations on a LAN and the local switches, as well as between switches and routers. Network cabling typically consists of copper, in the form of either twisted pair (TP) or coaxial cable. Data is signaled on copper when stations on the shared medium independently adjust the voltage. Cabling can also consist of fiber-optic lines, which are made of thin strands of glass. Stations connected via fiber signal data through the presence or absence of photons. Both copper and fiber-optic mediums support digital signaling.

Forensic Value Network forensic investigators can tap into physical cabling to copy and preserve network traffic as it is transmitted across the line. Taps can range from “vampire” taps, which literally puncture the insulation and make contact with copper wires, to surreptitious fiber taps, which bend the cable and cut the sheathing to reveal the light signals as they traverse the glass. Many commercial vendors also manufacture infrastructure taps, which can plug into common cable connectors and are specifically designed to replicate signals to a passive station without degrading the original signal. (Please see Chapter 3, “Evidence Acquisition,” for more details.)

2.1.2 In the Air

An increasingly popular way to transmit station-to-station signals is via “wireless” networking, which consists of radio frequency (RF) and (less commonly) infrared (IR) waves. The wireless medium has made networks very easy to set up. Wireless networks can easily be deployed even without “line-of-sight”—RF waves can and do travel through air, wood, and brick (though the signal will be degraded substantially by particularly dense materials). As a result, enterprises and home users can deploy wireless networks without the expense and hassle of installing cables.

Forensic Value Essentially, wireless access points act as hubs, broadcasting all signals so that any station within range can receive them. As a result, it is often trivial for investigators to gain access to traffic traversing a wireless network. The traffic may be encrypted, in which case investigators would have to either legitimately obtain the encryption key or break the encryption to access the content.

However, investigators can still gather a lot of information from encrypted wireless networks. Although data packets that traverse a wireless network may be encrypted, commonly management and control frames are not. In the clear, wireless access points advertise their names, presence, and capabilities; stations probe for access points; and access points respond to probes. Even when analyzing encrypted wireless traffic, investigators can identify

MAC addresses of legitimate authenticated stations, unauthenticated stations and suspicious stations that may be attacking the wireless network. Investigators can also conduct volume-based statistical traffic analysis and analyze these patterns.

With access to unencrypted or decrypted wireless traffic, of course, investigators can review the full packet captures in detail.

2.1.3 Switches

Switches are the glue that hold our LANs together. They are multiport bridges that physically connect multiple stations or network segments together to form a LAN. In modern deployments, we tend to see switches connected to other switches connected to other switches *ad nauseum* to form a complex switched network environment.

In a typical deployment, organizations have “core” switches, which aggregate traffic from many different segments, as well as “edge” switches,² which aggregate stations on individual segments. Consequently, traffic from one station to another within an enterprise may traverse any number of switches, depending on the physical network topology and the locations of the stations within it.

Forensic Value Switches contain a “content addressable memory” (CAM) table, which stores mappings between physical ports and each network card’s MAC address. Given a specific device’s MAC address, network investigators can examine the switch to determine the corresponding physical port, and potentially trace that to a wall jack and a connected station.

Switches also provide a platform from which investigators can capture and preserve network traffic. With many types of switches, network staff can configure one port to “mirror” (copy) traffic from any or all other ports or VLANs, allowing investigators to capture traffic from the mirroring port with a packet sniffer. Please see Section 3.1.4.1 for more detail.

2.1.4 Routers

Routers connect different subnets or networks together and facilitate transmission of packets between different network segments, even when they have different addressing schemes.

Routers add a layer of abstraction that enables stations on one LAN to send traffic destined for stations on another LAN. Internetworking using routers is what allows us to build campus-wide metropolitan area networks (MANs) or connect remote offices around the globe via wide area networks (WANs). From a certain perspective, the entire global Internet is nothing but a global area network (GAN), connected through a complex multilayer web of routers.

Forensic Value Where switches have CAM tables, routers have routing tables. Routing tables map ports on the router to the networks that they connect. This allows a forensic investigator to trace the path that network traffic takes to traverse multiple networks. (Note that this path can vary dynamically based on network traffic levels and other factors.)

Depending on the specific device’s capabilities, routers may also function as packet filters, denying the transmission of certain types of traffic based on source, destination, or port.

2. “Encyclopedia—PC Magazine,” http://www.pcmag.com/encyclopedia_term/0,2542,t=edge+switch&i=42362,00.asp.

Routers may log denied traffic (or sometimes maintain statistics on allowed traffic). Many enterprise-class routers can be configured to send logs and flow record data to a central logging server, which is extremely helpful for investigators, as this makes it very easy to correlate events from multiple sources. Logs stored on the router itself may be volatile and subject to erasure if the device is rebooted or if the available storage is exceeded.

In short, routers are the most rudimentary network-based intrusion detection systems in our arsenal, and also the most widely deployed.

2.1.5 DHCP Servers

The Dynamic Host Configuration Protocol (DHCP) is widely used as the mechanism for assigning IP addresses to LAN stations, so that they can communicate with other stations on the local network, as well as with systems across internetworked connections. In the early days of networking, administrators had to manually configure individual desktops with static IP addresses. DHCP was developed to provide automated IP address assignments, which could change dynamically as needed, dramatically reducing manual workload for administrators. DHCP service is often provided by the edge devices that perform routing between networks (routers, wireless access points, etc.), but it is not uncommon to find this service provided by infrastructure servers instead.

Forensic Value Frequently, investigations begin with the IP address of a host that is suspected of being involved in some sort of adverse event—whether it was the victim of an attack, the origin, or perhaps both. One of the first tasks the investigator must undertake is to identify and/or physically locate the device based on its IP address.

When DHCP servers assign (or “lease”) IP addresses, they typically create a log of the event, which includes the assigned IP address, the MAC address of the device receiving the IP address, and the time the lease was provided or renewed. Other details, such as the requesting system’s hostname, may be logged as well. Consequently, DHCP logs can show an investigator exactly which physical network card was assigned the IP address in question during the specified time frame.

2.1.6 Name Servers

Just as we need a mechanism to map MAC addresses to IP addresses, we also need a mechanism to map IP addresses to the human-readable names that we assign to systems and networks. To accomplish this, enterprises typically use the Domain Name System (DNS), in which individual hosts query central DNS servers when they need to map an IP address to a hostname, or vice versa. DNS is a recursive hierarchical distributed database; if an enterprise’s local DNS server does not have the information to resolve a requested IP address and hostname, it can query another DNS server for that information.

Forensic Value DNS servers can be configured to log queries for IP address and hostname resolutions. These queries can be very revealing. For example, if a user on an internal desktop browses to a web site, the user’s desktop will make a DNS query to resolve the host and domain names of the web server prior to retrieving the web page. As a result, the DNS server may contain logs that reveal connection attempts from internal to external systems, including web sites, SSH servers, external email servers, and more.

DNS servers can log not only queries, but also the corresponding times. Therefore, forensic investigators can leverage DNS logs to build a timeline of a suspect’s activities.

2.1.7 Authentication Servers

Authentication servers are designed to provide centralized authentication services to users throughout an organization so that user accounts can be managed in one place, rather than on hundreds or thousands of individual computers. This allows enterprises to streamline account provisioning and audit tasks.

Forensic Value Authentication servers typically log successful and/or failed login attempts and other related events. Investigators can analyze authentication logs to identify brute-force password-guessing attacks, account logins at suspicious hours or unusual locations, or unexpected privileged logins, which may indicate questionable activities. Unlike analysis of authentication logs on a single hard drive, a central authentication server can provide authentication event information about all devices within an entire authentication domain, including desktops, servers, network devices, and more.

2.1.8 Network Intrusion Detection/Prevention Systems

Unlike the devices we've discussed so far, network intrusion detection systems (NIDSs) and their newer incarnations, network intrusion prevention systems (NIPSSs), were specifically *designed* to provide security analysts and forensic investigators with information about network security-related events. Using several different methods of operation, NIDS/NIPS devices monitor network traffic in real time for indications of any adverse events as they transpire. When incidents are detected, the NIDS/NIPS can alert security personnel and provide information about the event. NIPSSs may further be configured to block the suspicious traffic as it occurs.

The effectiveness of NIDS/NIPS deployments depends on many factors, including precisely where sensors are placed in the network topology, how many are installed, and whether they have the capacity to inspect the increasing volumes and throughputs of traffic we see in the modern enterprise environment. While NIDS/NIPS may not be able to inspect, or alert on, *every* event of interest, with a well-engineered deployment, they can prove indispensable.

Forensic Value At a high level, the forensic value of a NIDS/NIPS deployment is obvious: They are designed to provide timely data pertaining to adverse events on the network. This includes attacks in progress, command-and-control traffic involving systems already compromised, or even simple misconfigurations of stations.

The value of this data provided by NIDS/NIPS is highly dependent upon the capabilities of the device deployed and its configuration. With many devices it is possible to recover the entire contents of the network packet or packets that triggered an alert. Often, however, the data that is preserved contains little more than the source and destination IP addresses, the TCP/UDP ports, and the time the event occurred. During an ongoing investigation, forensic investigators can request that network staff tune the NIDS to gather more granular data for specific events of interest or specific sources and destinations. (Please see Chapter 7 for more detail.)

2.1.9 Firewalls

Firewalls are specialized routers designed to perform deeper inspection of network traffic in order to make more intelligent decisions as to what traffic should be forwarded and what traffic should be logged or dropped. Unlike most routers, modern firewalls are designed to

make decisions based not only on source and destination IP addresses, but also based on the packet payloads, port numbers, and encapsulated protocols.

These days, nearly every organization has deployed firewalls on their network perimeters—the network boundaries between the enterprise and their upstream provider. In an enterprise environment, firewalls are also commonly deployed within internal networks to partition network segments in order to provide enclaves that are protected from each other. Even home users often have at least rudimentary firewall capabilities built into home routers, which are leased from their ISPs or purchased off-the-shelf.

Forensic Value Originally, event logging was of secondary importance for firewall manufacturers. Firewalls were not initially designed to *alert* security personnel when security violations were taking place, even though they were most definitely designed to implement security policies to prevent violations.

Today, modern firewalls have granular logging capabilities and can function as both infrastructure protection devices and fairly useful IDSs as well. Firewalls can be configured to produce alerts and log allowed or denied traffic, system configuration changes, errors, and a variety of other events. These logs can help operators manage the network and also serve as evidence for forensic analysts.

2.1.10 Web Proxies

Web proxies are commonly used within enterprises for two purposes: first, to improve performance by locally caching web pages and, second, to log, inspect, and filter web surfing traffic. In these deployments, web traffic from local clients is funneled through the web proxy. The web proxy may either allow or deny the web page request (often based on published blacklists of known inappropriate or malicious sites or on keywords in the outbound web traffic).

Once the page is retrieved, the web proxy may inspect the returned content and choose to allow, block, or modify it. For performance, the web proxy may cache the web page and later serve it to other local clients upon request so that there is no need to make multiple requests across the external network for the same web sites within a short period of time. The precise functionality of the web proxy is strongly dependant upon the specific configuration; some organizations choose to limit logging and run their web proxy only for performance reasons, whereas other organizations heavily monitor and filter incoming and outbound web traffic for security reasons.

There are also “anonymizing proxies,” which are set up with the explicit purpose of providing anonymity to clients who deliberately funnel web surfing traffic through them. End clients send their traffic through an anonymous web proxy so that the remote web server only sees the proxy’s IP address rather than the end-user’s IP address. Depending on the precise configuration of the anonymizing proxy, the proxy server may still cache extensive information regarding the end-user’s web surfing behavior.

Forensic Value Web proxies can be a gold mine for forensic investigators, especially when they are configured to retain granular logs for an extended period of time. Whereas forensic analysis of a single hard drive can produce the web surfing history for users of a single device, an enterprise web proxy can literally store the web surfing logs for an entire organization.

There are many commercial and free applications that can interpret web proxy logs and provide visual reports of web surfing patterns according to client IP address or even

username (i.e., when correlated with Active Directory logs). This can help forensic analysts gather lists of users who may have succumbed to a phishing email, investigate a roving user's inappropriate web surfing habits, or identify the source of web-based malware. If the web proxy is configured to cache web pages, it is even possible to retrieve the content that an end-user viewed or carve malware out of a cached web page for further analysis.

2.1.11 Application Servers

Every organization has a variety of application servers, which vary depending on the organization's industry, mission, size, budget, and many other factors. Common types of application servers include:

- Database servers
- Web servers
- Email servers
- Chat servers
- VoIP/voicemail servers

Forensic Value There are far too many kinds of application servers for us to review every one in depth (there have been dozens if not hundreds of books published on each type of application server). However, when you are leading an investigation, keep in mind that there are many possible sources of network-based evidence. Review local network diagrams and application documentation to identify the sources that will be most useful for your investigation.

2.1.12 Central Log Servers

Central log servers aggregate event logs from a wide variety of sources, such as authentication servers, web proxies, firewalls, and more. Individual servers are configured to send logs to the central log server, where they can be timestamped, correlated, and analyzed by automated tools and humans far more easily than if they resided on disparate systems.

The precise contents of a central logging server vary enormously depending on the organization and applicable regulations. Once a luxury that many enterprises did not bother to invest in, deployment of central logging servers has been spurred by regulations and industry standards. As a result, they are much more common today than they were just a few years ago.

Forensic Value Much like intrusion detection systems, central log servers are designed to help security professionals identify and respond to network security incidents. Even if an individual server is compromised, logs originating from it may remain intact on the central log server. Furthermore, devices such as routers, which typically have very limited storage space, may retain logs for very short periods of time, but the same logs may be sent in real time to a central log server and preserved for months or years. Some organizations have installed commercial log analysis products that can provide forensic analysts with complex reports and graphical representations of log data, correlated across a variety of sources.

2.2 Principles of Internetworking

Communication on the Internet involves coordinating hundreds of millions of computers around the world. In order to connect these systems together, we must have standards for everything from physical cabling to application protocols. We must also ensure that our system includes modularity and abstraction so that a change in the agreed-upon voltages sent across a wire would not require that software engineers rewrite web applications. To accomplish this, various software and hardware manufacturers, public standards bodies, and individual hackers have developed protocols for communication. Since the late 1970s, network designers have developed “layered” standards, such as the Open Systems Interconnection (OSI) model, to coordinate the design of networking infrastructure.

2.2.1 Protocols

Protocols take on new meaning when viewed in the context of forensic investigation. Attackers bend and break protocols in order to smuggle covert data, sneak past firewalls, bypass authentication, and conduct widespread denial-of-service (DoS) attacks. While network designers and engineers view protocols as rules for facilitating communication between stations, forensic investigators must view them as guidelines that attackers can leverage to produce unexpected results.

Protocol Mismatch

A Japanese and an American diplomat are scheduled to meet one another and work together on an item of transcontinental importance. Unfortunately, neither has been appropriately briefed about the other’s customary manner of polite greeting. At the precise moment that the Japanese diplomat bows, extremely respectfully and deeply, the American enthusiastically thrusts out his hand. Unhappily, the combined ignorance of the two results in a painful jab in the eye for the Japanese and a figurative black eye for the American. This is clearly no way to begin a useful relationship.

Every computer that is designed to communicate with another computer over a data network must overcome the same problem as our Japanese and American diplomats. They must first be programmed to use some coordinated scheme for exchanging data, from “hello” to “goodbye”—*protocols* for communication. Furthermore, they must have a system for gracefully dealing with ambiguous messages and erroneous conditions because it’s simply not possible to specify a reaction for every conceivable action.

The Free On-line Dictionary of Computing defines “protocol” as:

A set of formal rules describing how to transmit data, especially across a network.

Some protocols are fairly simple. For example, imagine that I dial your number and your phone rings, signaling to you that I wish to communicate. You acknowledge my request

and facilitate communication with me by taking the phone off the hook and saying “Hello, Jonathan!” I acknowledge your signal by saying something like “Hi, Sherri . . .” And so in three easy steps we’ve established a reliable bidirectional conversation.

The ubiquitous Transmission Control Protocol (TCP) similarly uses three distinct steps to establish a reliable bidirectional communication between stations. My computer might send your computer a segment with the TCP “SYN” (“synchronize”) flag set, signaling that I wish to communicate. Your computer acknowledges my request and facilitates communication with me by sending a TCP segment with the “SYN/ACK” (“synchronize”/“acknowledgment”) flags set in return. My computer acknowledges this signal by responding with a segment that has the TCP “ACK” (“acknowledgment”) flag set. In three steps, we’ve conducted a TCP handshake and established our reliable communications channel.

The Internet Engineering Task Force (IETF) and other standards bodies have published thousands upon thousands of protocol standards in minute detail, as we discuss extensively in Chapter 4. However, despite all of these specifications, we still manage to get things wrong and win black eyes on occasion.

What Are Protocols?

- Rules for successful communication between different systems
- Prearranged specifications for actions
- Designed to avoid implementation dependence
- Designed to avoid ambiguity
- Designed to recover gracefully from errors
- A perpetual attack target . . .

2.2.2 Open Systems Interconnection Model

The Open Systems Interconnection (OSI) Model (Figure 2–1) was designed by the International Organization for Standardization (ISO) to provide network architects, software engineers, and equipment manufacturers with a modular and flexible framework for development of communications systems. When data is transmitted across a network, output from one layer is *encapsulated* with data designed for use by the lower layer processes. Conversely, when data is received by the destination host, input from each layer is *demultiplexed* for use by the higher layer processes.³

This is similar to the way that people put letters in envelopes and write the recipient’s address on the outside. With letters in the real world, sometimes encapsulation happens multiple times. Imagine that an employee puts a letter in a yellow envelope for interoffice

3. Hubert Zimmerman, “OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, Vol. COM-28, No. 4, April 1980.

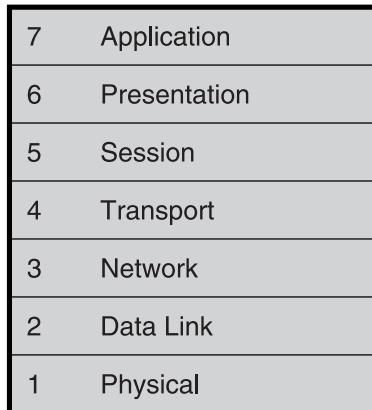


Figure 2–1. Layers in the OSI model. Every network analyst should be fluent in the OSI model labels. Make sure you have memorized the numbers and descriptions of each of the layers so that you can converse about them easily.

mail, specifying the recipient by name and office. The mail room receives this envelope and puts it in a national postal service envelope with a street address of the recipient’s building. The government mail carrier delivers it to the appropriate building, where the destination mail room removes the outer envelope and delivers the yellow envelope to the recipient. Finally, the recipient opens the yellow envelope and removes the letter inside.

Similarly, computers “encapsulate,” or wrap, data from one layer with standard headers and/or footers so that the software at a lower layer can process it. This is how data is transmitted from one server to another across the Internet.

When data is received by the destination server, it is “demultiplexed,” or unwrapped. The receiving process examines the protocol metadata to determine which higher-layer process to hand it to, and then removes the current layer’s protocol metadata before sending it along.

Very complex challenges can often be solved more easily if they are broken down into parts. Using the OSI model, or similar frameworks, designers can partition the big complicated problem of communicating across the Internet into discrete subproblems and solve them each independently. The great benefit of the layered approach is that, even though there may be multiple competing protocols used for any one purpose, there exists a layer of abstraction and modularity that allows us to swap out a solution at one layer without affecting any of the solutions at the other layers.

There are other layered models in use for this type of network communication abstraction, such as the “TCP/IP Model,” which uses four layers instead of seven.⁴ Throughout this book, we use the OSI model (arguably the most popular) as our standard.

4. R. Braden, “Requirements for Internet Hosts—Communication Layers,” IETF, October 1989, <http://www.rfc-editor.org/rfc/rfc1122.txt>.

The OSI Model

The OSI model was designed to provide the following benefits for network designers and engineers:

- **Modularity** Breaks down a complex communication problem into more manageable parts.
- **Flexibility** Supports interchangeable parts within a layer.
- **Abstraction** Describes the functionality of each layer, so that developers don't need to know the specific details of protocol implementations in order to design interoperable processes.

Bad guys (and good guys) can bend this framework and break it in curious and wonderful ways!

2.2.3 Example: Around the World ... and Back

Let's explore a simple web surfing scenario using the OSI model, as shown in Figure 2–2. Imagine that you are an American sitting in front of a Mac OS X laptop. You are sitting in an airport in Salt Lake City, and a small box has popped up indicating that you can get online via a T-Mobile wireless "hotspot." You have no idea what that is, but it sounds nice, so you click "Yes!" Now your laptop is connected to the Internet.

You launch Firefox, your web browser, and tell it that you'd like to ask the Chinese version of Google a question (www.google.cn). Somehow, a page pops up with Hanzi

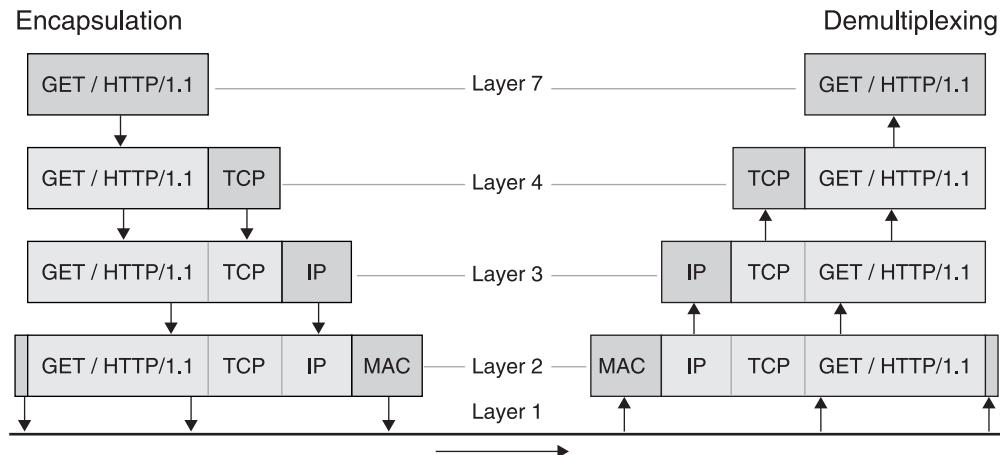


Figure 2–2. An HTTP "GET" request, shown in the framework of the OSI model.

characters. You ask this Google (in Chinese) for “Chinese food in Bozeman, Montana” and get a new page back from somewhere in the universe that tells you there’s a place named Bamboo Garden Asian Grille, which will serve you when you get there. It even includes a phone number and a map.

2.2.3.1 Questions

You stop for a moment and wonder:

- How did your web browser know how to speak Mandarin to some web server somewhere?
- How did your web browser know where to find that faraway web server, in order to speak to it in the first place?
- How did your laptop manage to send it a signal in such a way that the wireless access point sent the request along to the destination?
- How was it that your laptop established itself on the Internet and could communicate with other systems?
- What was the route that your laptop used to talk to that remote web server, and how did it figure that out?
- How did the destination web server know what your request meant and how to format a reply so that your Mac would be able to interpret it (both with Hanzi and in English)?

All you did was point, click, and type a few characters, and within seconds a web server told you in Chinese about a restaurant in Montana. Pretty slick—and pretty complex. Was this all caused by voltage variations in copper cables? Was it perhaps also photons through fiber? Clearly there was some RF too, as your laptop never physically connected to anything at all!

Hard to believe this problem is solved a billion times a second.⁵

2.2.3.2 Analysis

Now, let’s analyze this web surfing scenario using the OSI model. As illustrated in Figure 2–2, we will step through one piece of this example in a simplified manner, using only a few layers but all the same concepts.

Your web browser, an application, operates at Layer 7 of the OSI model. When you type a URI into the address bar and hit Enter, it sends a DNS request to the local DNS server in order to map the search engine’s URI (<http://www.google.cn>) to an IP address. The local DNS server responds with the IP address mapped to the requested hostname.

Once the remote IP address is known, your local computer sends out an HTTP “GET” request (Layer 7), destined for the remote web server’s IP address. In order to get this request to the destination web server, your web browser hands the Layer 7 request to your operating system. The operating system chops this data up into discrete packets and prepends a TCP header (Layer 4), which includes the source port that web client is bound to, the

5. Not an official measure.

destination port (TCP 80) that the server is expected to be listening on, and a variety of other information to support the Layer 4 connection.

Next, the operating system prepends an IP header (Layer 3), which describes the source and destination endpoint IP addresses so that the traffic can be properly routed from network to network across the Internet. Each packet is then framed with 802.11 information for transmission across the wireless LAN (Layer 2). Finally, the entire frame is turned into an RF (Layer 1), and sent out into the world.

The local wireless access point (WAP) receives the RF transmission and strips off the 802.11 header, replacing it with a new Layer 2 header. Next, the WAP transmits the entire frame to the next station over a copper wire (Layer 1). This station is a router that replaces the Layer 2 information and sends the frame over a copper wire (Layer 1) to its next hop. This process continues until the packet finally reaches its destination, the remote web server.

At the destination, the process works in reverse: The destination server is connected via a copper wire (Layer 1) and receives a series of voltage changes. The server's network card interprets this as an Ethernet frame (Layer 2), removes the Layer 2 header, and sends it to the operating system. The operating system sees that the IP packet (Layer 3) contains its own IP address ("Yay, it's for me!"). The operating system removes the IP packet header and notes the TCP header information (Layer 4), which of course includes a destination port, among other information. The operating system then looks to see what process is listening on the specified destination port. Finding a web server, it removes the TCP header and hands the payload to the destination web server (Layer 7).

In this manner, the OSI model can be used to describe the process of sending a web page request across the Internet, by examining the various layers of protocol interaction.

For the forensic analyst, it is very important to be fluent in the numbers and names of the different layers in the OSI model. The layered model both reflects and impacts the design of networking protocols, which we dissect throughout this book.

2.3 Internet Protocol Suite

The Internet Protocol Suite, also known as the TCP/IP protocol suite, is a collection of protocols that are used to implement important functions on the Internet and in many other packet-switched networks. For network forensic investigators, the protocols that make up the Internet Protocol Suite are as important as waves are to surfers or cooking pans are to Julia Child.

Your effectiveness as a network forensic investigator will depend in part on your familiarity with the Internet Protocol Suite, including key protocols and header fields. Nearly every network forensic technique in this book relies on your understanding of these protocols, from flow record analysis to packet analysis to web proxy dissection, and more. To get the most out of this book, we highly recommend that you become proficient at recognizing and recalling the important features and basic structures of key protocols in the Internet Protocol Suite.

In this section, we briefly review the history, specification, and features of IPv4, IPv6, TCP, and UDP. A full treatment of these protocols is beyond the scope of this book. For more details, we highly recommend *TCP/IP Illustrated Volume 1*, by W. Richard Stevens.

2.3.1 Early History and Development of the Internet Protocol Suite

Protocols do not simply fall from the sky, or emerge chiseled in stone—although it can seem that way when you’re tasked with analyzing or implementing them. Excellent network forensics investigators intuitively understand that protocols are not perfect: They change over time, implementations vary greatly, and hackers manage to bend and break them in weird and interesting ways. To help build your intuition, we now briefly review how the most important protocols used on the Internet came to exist.

The most famous protocols in the Internet Protocol Suite, the Transmission Control Protocol (TCP) and the Internet Protocol (IP), were initially developed during the 1970s by Vinton “Vint” Cerf, Jon Postel, and many others. The research was primarily funded by the United States Defense Advanced Research Projects Agency (DARPA). The first published versions of TCP, created by the “Stanford University TCP Project,” actually incorporated functions that have since been split between TCP and IP. We will henceforth refer to the original monolithic protocol as “[TCP/IP],” even though at the time it was actually called simply “TCP,” so as to distinguish it from the modern “TCP” protocol.⁶

The purpose of the original protocol was to allow processes on different computers in different networks to easily communicate with each other. Vint Cerf later recalled that “[t]he initial concept behind [TCP/IP] was very simple. Two processes wishing to communicate had only to know the other’s ‘address.’”⁷

The original 1974 “Specification of Internet Transmission Control Program” stated:⁸

Processes are viewed as the active elements of all HOST computers in a network. Even terminals and files or other I/O media are viewed as communicating through the use of processes. Thus, all network communication is viewed as inter-process communication.

Since a process may need to distinguish among several communication streams between itself and another process [or processes], we imagine that each process may have a number of PORTs through which it communicates with the ports of other processes.

Since port names are selected independently by each operating system, [TCP/IP], or user, they may not be unique. To provide for unique names at each [TCP/IP], we concatenate a NETWORK identifier, and a [TCP/IP] identifier with a port name to create a SOCKET name which will be unique throughout all networks connected together.

Later, the Specification of Internet Transmission Control Program went on to compare the [TCP/IP] protocol to a post office:

The [TCP/IP] acts in many ways like a postal service since it provides a way for processes to exchange letters with each other . . . In addition to acting like a

6. Cerf, Vinton G., “FINAL REPORT OF THE STANFORD UNIVERSITY TCP PROJECT,” Internet Experiment Note #151, April 1, 1980, <http://www.rfc-editor.org/ien/ien151.txt>.

7. *Ibid.*

8. Cerf, Vinton; Dalal, Yogen; Sunshine, Carl. “RFC 675: SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM,” IETF, December 1974, <http://www.rfc-editor.org/rfc/rfc675.txt>.

postal service, the [TCP/IP] insures end-to-end acknowledgment, error correction, duplicate detection, sequencing, and flow control.

By 1977, pressure mounted to split the original [TCP/IP] into two separate protocols. At that time, Jon Postel insightfully wrote:⁹

We are screwing up in our design of internet protocols by violating the principle of layering. Specifically we are trying to use TCP to do two things: serve as a host level end to end protocol, and to serve as an internet packaging and routing protocol. These two things should be provided in a layered and modular way. I suggest that a new distinct internetwork protocol is needed, and that TCP be used strictly as a host level end to end protocol.

Shortly thereafter, work emerged on the Draft Internetwork Protocol Specification,¹⁰ which was eventually published as RFC 791, “Internet Protocol” in September 1981.¹¹ TCP was revised to excise the network layer functions, and was also published in September 1981 as RFC 793, “Transmission Control Program.”¹²

And thus, there was light and darkness, TCP and IP.

2.3.2 Internet Protocol

The Internet Protocol (IP) is designed to handle addressing and routing. It includes a method for uniquely identifying source and destination systems on a network (the “IP address”), and provides support for routing data through networks. To accomplish this, data is prepended with the IP *header*, as shown in Figures 2–3 and 2–4. The combination of the IP header and encapsulated payload together are referred to as an *IP packet*. There is no footer; the length of the IP packet is specified in its header. The packet is sent from the source through any intermediate routers to its destination, where the IP header is removed by the recipient operating system and the encapsulated payload is recovered.

The Internet Protocol (version 4) specification sums it up nicely as follows:¹³

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. The internet protocol can capitalize on the services of its supporting networks to provide various types and qualities of service.

9. Postel, Jon, “Comments on Internet Protocol and TCP,” Internet Experiment Note #2, August 15, 1977, <http://www.rfc-editor.org/ien/ien2.txt>.

10. Postel, Jonathan B., “Draft Internetwork Protocol Specification: Version 2,” Internet Experiment Note #28, February 1978, <http://www.rfc-editor.org/ien/ien28.pdf>.

11. J. Postel, “RFC 791—Internet Protocol,” *Information Sciences Institute*, University of Southern California, September 1981, <http://www.rfc-editor.org/rfc/rfc791.txt>.

12. J. Postel, “RFC 793—Transmission Control Protocol,” *Information Sciences Institute*, University of Southern California, September 1981, <http://www.rfc-editor.org/rfc/rfc793.txt>.

13. J. Postel, “RFC 791—Internet Protocol,” *Information Sciences Institute*, University of Southern California, September 1981, <http://www.rfc-editor.org/rfc/rfc791.txt>.

The TCP Bakeoff

As we've seen, the point of protocols is to ensure that disparate systems can communicate with each other. This is as true for Japanese and American diplomats as it is for processes on different Linux and Windows systems.

Once upon a time, when processes on different operating systems couldn't talk to each other reliably over a network, early developers had to figure out what the protocols were going to be and come up with standards that worked for a wide variety of systems.

After experiencing interoperability problems in 1978, the designers of TCP/IP held an informal "Saturday TCP Bakeoff" to work things out experimentally. The bakeoff was held on January 27, 1979, at the University of Southern California. As described by Jack Haverty on the Internet History mailing list:¹⁴

Jon [Postel] picked Saturday because the ISI offices were mostly deserted, so we all spread out in various offices along a hallway, using the terminals to telnet into our home machines across the Arpanet and try the experiments. We could shout at each other down the corridor. Jon had set up a bunch of goals, like establishing a connection with someone else, and assigned points for each accomplishment. You even got points for doing things like crashing the other guy's system.

I remember at first no one could talk to anyone else, because each implementation seemed to have its own ideas about what was a correct checksum and was rejecting all incoming packets. So we all turned off checksumming, and proceeded to get the basic interactions (SYN, ACK, etc.) to work. Then checksumming was debugged—getting the exact details of which fields were included in each checksum calculation and getting the byte-ordering correct. After that worked, checksumming was turned back on.

Then, documenting what was probably the first instance of a social engineering attack on a TCP/IP network, Jack wrote:

A few hours later, I remember watching Bill Plummer (PDP-10 TCP) with his hand poised over the carriage-return key. Bill yelled down the hall to Dave Clark (Multics TCP)—“Hey Dave, can you turn off your checksumming again for a minute?” A bit later, Dave said “OK!”, and Bill hit the carriage return key. A few seconds later … Dave yelled “Hey, Multics just crashed!” and Bill whooped “Gotcha!”

A discussion of the legality of getting points for crashing a system by subterfuge ensued, but doesn't appear in the minutes.

IP operates at Layer 3 of the OSI model (the network layer). It is a *connectionless* protocol, meaning that it does not explicitly identify individual packets as being part of a related series, and therefore also does not have a method for indicating the initiation or closing of a conversation.

14. Jack Haverty, email message to Internet History mailing list, The Postel Center, University of Southern California, March 31, 2006, <http://mailman.postel.org/pipermail/internet-history/2006-March/000571.html> (accessed December 31, 2011).

IPv4 Packet Header																
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Bytes	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0x00	Version	IHL	DSCP	ECN											Total Length	
0x04			Identification		R	D	M								Fragment Offset	
0x08		Time to Live		Protocol											Header Checksum	
0x0C				Source Address												
0x10				Destination Address												
0x14				Options												

Figure 2–3. The IPv4 packet header.

The IP is also not designed to ensure reliability of transmission (the designers envisioned that this function would be handled by TCP or another higher-layer protocol). The Internet Control Message Protocol (ICMP), which was released in 1981 at the same time as Internet Protocol version 4 (IPv4), is often used in conjunction with IP to “provide feedback about problems in the communication environment.”¹⁵ For more information about ICMP, see Section 11.3.4, “ICMP Tunneling.”

There have been several versions of IP. The most widely deployed version is IPv4, which was officially standardized in 1981 and first deployed on January 1, 1983. During the past thirty years, IPv4 has become globally supported. However, work has continued on IP development, fueled primarily by concerns about address space exhaustion. In 1998, the specification for Internet Protocol version 6 (IPv6) was released, and support for it is increasing, slowly but surely.¹⁶

Characteristics of IP include:

- Support for addressing and routing
- Connectionless
- Unreliable
- Includes a header (no footer)
- Header plus payload is called an *IP packet*

IPv6 Packet Header																
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Bytes	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0x00	Version	Traffic Class													Flow Label	
0x04			Payload Length												Next Header	
0x08															Hop Limit	
0x0C															Source Address (128 bits)	
0x10																
0x14																
0x18																
0x1C															Destination Address (128 bits)	
0x20																
0x24																

Figure 2–4. The IPv6 packet header.

15. J. Postel, “RFC 792—Internet Control Message Protocol,” *IETF*, September 1981, <http://www.rfc-editor.org/rfc/rfc792.txt>.

16. Deering, S.; Hinden, R. “Internet Protocol, Version 6 (IPv6) Specification,” *IETF*, December 1998, <http://www.rfc-editor.org/rfc/rfc2460.txt>.

In the following sections, we briefly summarize the features of IPv4 and IPv6, and discuss key differences between them.

2.3.2.1 IPv4

IPv4 uses a 32-bit address space to identify the source and destination systems. Typically, human-readable IP addresses are divided into four octets, separated by periods, and written in decimal. Each octet has 2^8 possible values, ranging from 0 to 255. For example, here is a 32-bit IP address written in binary and human-readable form:

Binary:

```
00001010.00000001.00110010.10010110
```

Human-readable:

```
10.1.50.150
```

There are 2^{32} , or 4,294,967,296 (approximately 4.3 billion) possible IPv4 addresses. Theoretically, the smallest possible IP address is 0.0.0.0 and the largest is 255.255.255.255. However, many IPv4 addresses are reserved for specific purposes. For example, three ranges of addresses were reserved for private networks by RFC 1918 *Address Allocation for Private Internets*, and are not routed across the Internet.¹⁷ These ranges are:

10.0.0.0	-	10.255.255.255 (10/8 prefix)
172.16.0.0	-	172.31.255.255 (172.16/12 prefix)
192.168.0.0	-	192.168.255.255 (192.168/16 prefix)

Each IPv4 packet header includes a checksum. Routers along the packet's path recalculate the checksum based on the data in the header and, if it is not correct, they discard the packet. IPv4 does not include any built-in support for encryption.

The full structure of the IPv4 packet header is shown in Figure 2–3.

2.3.2.2 IPv6

Although 4.3 billion possible IP addresses may seem like a lot, it's not enough for the long-term needs of the global Internet. In February 2011, the last five IPv4 address ranges were allocated by the Internet Assigned Numbers Authority (IANA) to each of the five Regional Internet Registries (RIRs) around the world.¹⁸

The address space exhaustion occurred in part because of the explosion of mobile devices and the sheer numbers of people connected to the Internet. It is also partly due to inefficient address allocation; for example, in the infancy of the Internet, MIT was allocated an entire “Class A” range (18.*.*.*), with over 16 million possible IP addresses, even though the university uses a fraction of that number.

17. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, “RFC 1918: Address Allocation for Private Internets,” IETF, February 1996, <http://www.rfc-editor.org/rfc/rfc1918.txt>.

18. Lawson, Stephen. “IPv6 Doomsday Won’t Hit in 2012, Experts Say.” *PCWorld*, December 29, 2011, http://www.pcworld.com/businesscenter/article/247091/ipv6_doomsday_wont_hit_in_2012_experts_say.html (accessed December 31, 2011).

In response to the IPv4 address space exhaustion, the IPv6 protocol was developed with 128 bits for each of the source and destination addresses. There are approximately 2^{128} , or 340 undecillion possible IP addresses. Human-readable IPv6 addresses are written in hexadecimal and are divided into 8 groups of 16 bits each.

Here is an example of a human-readable IPv6 address:

```
2001:0db8:0000:0000:0001:0000:0000:0001
```

Here is a shorthand way of writing the same IPv6 address:

```
2001:db8::1:0:0:1
```

Developers of IPv6 also disposed of packet header checksums, as you can see in Figure 2–4. Instead, error detection is left entirely to higher-layer protocols. This helps to improve throughput, since routers no longer have to calculate the packet header checksum at each hop while the packet is in transit. In addition, IPv6 headers are of fixed length, rather than variable length as in IPv4. This eliminates the need for routers to parse the IP header length in order to find and evaluate fields in the embedded protocol (i.e., for packet filtering).

Perhaps most importantly for security purposes, IPv6 is designed to interoperate with IPSEC, providing network-layer confidentiality and integrity.

The key changes introduced in IPv6 compared with IPv4 can be summarized as follows:

- Larger address space (128 bits)
- No packet header checksums
- Fixed-length IP packet header
- Designed to interoperate with IPSEC

2.3.3 Transmission Control Protocol

The Transmission Control Protocol (TCP) is designed to handle multiplexing of process communications on the host, as well as reliability and sequencing. Data is prepended with a TCP header, as shown in Figure 2–5. The combination of the TCP header and the encapsulated payload together is referred to as a *TCP segment*. As with IP, there is no footer. To communicate, a process encapsulates data in a TCP segment header for transmission to another process, possibly on a remote system (in which case the segment would subsequently be encapsulated in an IP packet for transmission across the network and demultiplexed at the receiving end).

TCP Segment Header																		
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7		
Bytes	0	1							2	3								
0x00	Source Port								Destination Port									
0x04					Sequence Number													
0x08					Acknowledgement Number													
0x0C	Length	Reserved	C	E	U	A	P	R	S	F					Window Size			
0x10					Checksum								Urgent Pointer					
0x14	Options																	

Figure 2–5. TCP

Wanna Buy a Class C?

MIT's venerable humor magazine, *Voo Doo*, once ran a tongue-in-cheek article about the ironies of IP address management in their supposedly roomy Class A address space. In a 1995 article, "IP Address Shortage Spurs Black Market," the prolific Alyssa P. Hacker wrote:¹⁹

Although M.I.T. owns one of the few Class A Internet Protocol (IP) address spaces in the world, the now famous "Net 18," there is a campus shortage of available addresses. Not a real shortage, mind you, but an artificial shortage created by Information Systems controlling and rationing the available subnets. I/S claims that proactive measures are prudent and necessary, but critics point out that out of the sixteen million possible addresses of the form 18.*.*.*, there are only about thirteen thousand hosts on MITnet.

... Under ResNet, dormitories and fraternities are assigned Class B networks with over 65,000 available addresses. However, no fraternity we talked to had more than 100 machines running in their house, not even enough to tax a Class C address space.

... House Presidents were uncharacteristically glib about what they were doing with the unused addresses. "We are not using them," said David Conway, President of Chi Phi in Boston. "No further comments." When shown evidence that they are in use, he repeated, "We are not using them."

An anonymous junior at TEP shed some light on the answer, though. "Let's just say the House GPA jumped half a point last term."

Just as with other black markets of the twentieth century, students involved with selling and trading IP addresses, or "dealing IP" as it is called in the dark basements and dangerous streets of M.I.T., turn to other crimes, such as fraud and prostitution. "The temptation is there," states Anne Glavin, Chief of Campus Police. "Anytime a commodity is priced artificially high, a black market appears. We've seen it before at M.I.T. with heroin and DRAM chips."

... "Dealing IP was a gateway crime for me," said Brian Bradley, who asked not to be identified. "When I ran out of IP addresses, I wanted to keep dealing, so I switched to selling cocaine. The switch wasn't difficult, though. I still have the same customers: Media Lab professors and computer science grad students."

The TCP header includes 16-byte fields for "source port" and "destination port." Recall that according to the original "Specification of Internet Transmission Control Program," a process "may have a number of PORTs through which it communicates with the ports of other processes."²⁰ The possible values for TCP ports range from 0 to 65,535.

19. Hacker, Alyssa P. "IP Address Shortage Spurs Black Market." *Voo Doo, MIT Journal of Humour*, 1995. <http://web.mit.edu/voodoo/www/is771/ip.html> (accessed December 31, 2011).

20. Cerf, Vinton; Dalal, Yogen; Sunshine, Carl. "RFC 675: SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM," *IETF*, December 1974, <http://www.rfc-editor.org/rfc/rfc675.txt>.

TCP is a *connection-oriented* protocol. It indicates the state of transmissions using flags in the TCP header. As described earlier in Section 2.2.1, TCP uses three steps to establish a reliable bidirectional communication between stations. First, the initiating station sends a segment with the “SYN” flag set. Next, the responder sends back a segment with the “SYN” and “ACK” flags set. Finally, the initiator responds with a segment that has an “ACK” flag set. This initiation sequence is referred to as the *three-way handshake*. There is also a corresponding two-step sequence for closing a connection using the “FIN” and “ACK” flags.

Key characteristics of the TCP protocol include:

- Reliable
- Connection-oriented
- Handles sequencing
- Port numbers range from 0 to 65535
- Includes a header (no footer)
- Header plus payload is called a *TCP segment*

2.3.4 User Datagram Protocol

The User Datagram Protocol (UDP) is an extremely simple protocol. As you can see in Figure 2–6, it doesn’t do much—it is merely designed to facilitate multiplexing of process communications on the host, without any of the reliability, sequencing, connection setup and teardown, or any of the frills of TCP. This is very useful for some applications, such as real-time data streaming (Voice over IP, music, or videos). For these applications, there is no point in attempting to detect if packets arrive out of order or are dropped because the real-time requirements leave no room for retransmission. It is better to simply drop datagrams when errors occur than to slow down the entire transmission with transport-layer error checking.

Like TCP, the UDP header includes 16-byte fields for “source port” and “destination port.” The possible values for UDP ports range from 0 to 65,535.

Key characteristics of the UDP protocol include:

- Unreliable
- Connectionless
- Port numbers range from 0 to 65535
- Includes a header (no footer)
- Header plus payload is called a *UDP datagram*

UDP Datagram Header																
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Bytes	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0x00	Source Port				Destination Port											
0x04	UDP Length				UDP Checksum											

Figure 2–6. UDP

“Barely Worthy of the Name”

Internet pioneer David P. Reed provided a disenchanted perspective of the development of UDP during a 2006 discussion on the Internet History mailing list:²¹

I recall the spec of UDP (which is barely worthy of the name, it's so simple in concept) being written as part of a cleanup process, pushed by Postel to close some loose ends. Remember the point of UDP was to support such things as packet speech, message exchange protocols like DNS, ... and we were still far from understanding what might need to be in UDP for such things. So a long delay from conception to spec'ing was actually what happened.

I personally focused 99% of my time on TCP and IP issues in this time frame and so did everyone else. UDP was, to me, really a negotiating placeholder, something I was very happy we won in the negotiations leading up to it because my own research interest was in intercomputer message-exchange protocols—a general purpose solution that would be usable by lots of non-virtual-circuit apps. All it needed was demuxing (ports). In fact, I would have preferred (following my more extreme version of the E2E principle or principle of minimum mechanism) if there weren't even a checksum in the spec'ed format, so that error-recovery could have been left to layers on top ... [but] 4 bytes wasted is not a battle I wanted to fight, and you didn't have to actually compute or check the UDP checksum.

2.4 Conclusion

Network investigators must always be ready to learn and adapt. From wires to routers to web proxies to DNS servers, there are innumerable potential sources of network-based evidence. Although most environments are built from the same general types of network devices, there exists a wide variety of equipment, software, and protocols that are constantly changing.

In this chapter, we reviewed common classes of network devices and discussed the types of evidence typically found on each, and the value for forensic investigators. We followed up by discussing the concept of protocols and the OSI model, which form the framework for our studies throughout the remainder of this book. Finally, we discussed the Internet Protocol Suite and highlighted key features of IPv4, IPv6, TCP, and UDP.

With this framework in mind, remember that every environment is different and every investigation is different. Your mileage will vary based on the specific devices and communications protocols involved in each investigation. Always be prepared to encounter something new.

21. David P. Reed, email message to Internet History mailing list, The Postel Center, University of Southern California, March 31, 2006, <http://mailman.postel.org/pipermail/internet-history/2006-March/000569.html> (accessed December 31, 2011).

Chapter 3

Evidence Acquisition

“Some things are hurrying into existence, and others are hurrying out of it; and of that which is coming into existence part is already extinguished . . . In this flowing stream then, on which there is no abiding, what is there of the things which hurry by on which a man would set a high price?”

—The Meditations, by Marcus Aurelius¹

Ideally, we would like to obtain perfect-fidelity evidence, with zero impact on the environment. For copper wires, this would mean only observing changes in voltages without ever modifying them. For fiber cables, this would mean observing the quanta without ever injecting any. For radio frequency, this would mean observing RF waves without ever emitting any. In the real world, this would be equivalent to a murder investigator collecting evidence from a crime scene without leaving any new footprints.

Obviously, we don’t live in a perfect world, and we can never achieve “zero footprint.” Detectives analyzing a murder scene still cannot avoid walking on the same floor as the killer. However, network investigators can minimize the impact.

Network forensic investigators often refer to “passive” versus “active” evidence acquisition. *Passive evidence acquisition* is the practice of gathering forensic-quality evidence from networks without emitting data at Layer 2 and above. Traffic acquisition is often classified as passive evidence acquisition. *Active or interactive evidence acquisition* is the practice of collecting evidence by interacting with stations on the network. This may include logging onto network devices via the console or through a network interface, or even scanning the network ports to determine the current state.

Although the terms “passive” and “active” imply that there is a clear distinction between two categories, in reality, the impact of evidence acquisition on the environment is a continuous spectrum.

In this chapter, we discuss the types of physical media that can be leveraged to passively acquire network-based evidence and delve into popular tools and techniques for acquiring network traffic. Next, we review common interfaces used to interact with network devices. Finally, we discuss strategies for minimizing your footprint when conducting active evidence acquisition.

1. Thomas Bushnell, “The Meditations,” 1994, <http://classics.mit.edu/Antoninus/meditations.mb.txt>.

3.1 Physical Interception

It is possible to obtain network traffic without sending or modifying any data frames on the network. While it is never possible to have absolutely zero impact on the environment, the process of capturing (or *sniffing*) traffic can often be conducted with very little impact.

There are many ways to transmit data over physical media, and just as many ways to intercept it. The simplest case is a station connected to another station over a physical conduit, such as a copper cable. Voltage on copper can easily be amplified and redistributed in a one-to-many configuration. Hubs and switches are designed to extend the physical media in order to share the baseband with additional stations.

Forensic investigators can passively acquire network traffic by intercepting it as it is transmitted across cables, through the air, or through network equipment such as hubs and switches.

Pigeon Sniffing?

IP networks can be built upon a wide variety of physical media. For example, RFC 1149, “Standard for the transmission of IP datagrams on avian carriers” (subsequently extended in RFC 2549), describes the transmission of IP packets over avian carriers, as follows:²

Avian carriers can provide high delay, low throughput, and low altitude service. The connection topology is limited to a single point-to-point path for each carrier, used with standard carriers, but many carriers can be used without significant interference with each other, outside of early spring. [...]

Forensic investigators have not standardized on a method of passively sniffing traffic transmitted over avian carriers. Avian networks are fairly resistant to passive interception. Active interception techniques do exist, and typically involve large packages of breadcrumbs, as well as support staff to transcribe IP packets. The chief difficulty is route determination and interception, since dropped packets may not be recoverable. Consult your local health officials for any areas of avian influenza.

3.1.1 Cables

Cables allow for point-to-point connections between stations. The most common materials for cables are copper and fiber. Each of these can be sniffed, although the equipment and side effects vary based on the physical media.

3.1.1.1 Copper

The two most widely used types of copper cabling used in the modern era are coaxial cable and twisted pair.

- **Coaxial** Coaxial cable, or “coax,” consists of a single copper wire core wrapped in insulation and covered with a copper shield. This package is then sealed with an outer

². D. Waitzman, “RFC 1149—Standard for the Transmission of IP Datagrams on Avian Carriers,” IETF, April 1, 1990, <http://rfc-editor.org/rfc/rfc1149.txt>.

insulation. Since the transmission media is the single copper core, all stations on the network must negotiate the transmission and reception of signals. The benefit is that the copper core is shielded from electromagnetic interference.

In most cases where coax is used, if you can tap the single copper core, you can access the traffic to and from all stations that share the physical medium.

- **Twisted Pair (TP)** TP cables contain multiple pairs of copper wires. Unlike coaxial cable, where the single copper core is shielded against electromagnetic interference by a tubular Faraday cage, in TP each pair of copper wires is twisted together in order to negate electromagnetic interference. TP wires are typically deployed in a star topology. For example, in a large enterprise, end stations are commonly connected via unshielded twisted pair (UTP) (typically CAT5) to an aggregator such as an edge switch. This means that by tapping one pair of TP wires on a switched network, you may receive traffic relating to only one end station.

If you put a commercial TP network tap inline, it can capture all voltages for all twisted pairs in the cable.

3.1.1.2 Optical

Fiber optic cables consist of thin strands of glass (or sometimes plastic) which are bundled together in order to transmit signals across a distance. Light is transmitted into the fiber at one end and travels along an optic fiber, reflecting constantly against the walls until it reaches an optical receiver at the other end. The light naturally degrades during travel, and depending on the length of the fiber optic cable, an optical regenerator may be used to amplify the light signal in transit.³

3.1.1.3 Intercepting Traffic in Cables

There are a variety of tools available for intercepting traffic in cables, including inline network taps, “vampire” taps, induction coils, and fiber optic taps. We discuss each of these in turn.

- **Inline Network Taps** An inline network tap is a Layer 1 device, which can be inserted inline between two physically connected network devices. The network tap will pass along the packets and also physically replicate copies to a separate port (or multiple ports). Network taps commonly have four ports: two connected inline to facilitate normal traffic, and two sniffing ports, which mirror that traffic (one for each direction). Insertion of an inline network tap typically causes a brief disruption, since the cable must be separated in order to connect the network tap inline.

Many network taps use hardware to replicate data, which allows for extremely high-fidelity packet captures. Network taps are commonly designed to require no power for passively passing packets. This reduces the risk of a network outage caused by the tap. It is possible to pass traffic to a monitoring port without using power, although most often power is required for monitoring.

Sophisticated taps exist that can do load-balancing for intrusion detection. These taps tend to be more expensive. Recently, commercial vendors have been offering

3. “Howstuffworks,” <http://communication.howstuffworks.com/fiber-optic-communications/fiber-optic1.htm>.

filtering taps, which can filter traffic prior to replication based on VLAN tag, protocol, port, etc.

Network forensic analysts should keep in mind that every additional break in a cable is a potential point of failure; therefore inline insertion of network taps necessarily increase the risk of network disruption.

- **Vampire Taps** “Vampire taps” are devices that pierce the shielding of copper wires in order to provide access to the signal within. Unlike inline network taps, the cable does not need to be severed in order for a vampire tap to be installed. However, investigators should take caution: As noted by security researcher Erik Hjelmvik, “[i]nserting a vampire tap, even if done correctly, can bring down the link on a TP cable since the characteristics of the required balanced communication will be affected negatively.”⁴

Telecommunications engineers should be familiar with vampire tap technology, as it is standard issue with “butt kits” (telephone circuit testing equipment, typically worn on the toolbelts of telephone repair technicians). These handsets are used by inside-wiring technicians to tap into the punchdown blocks in wiring closets in order to sort through the large numbers of twisted pairs that so often go unnumbered and unlabeled.

Butt kits typically come with a combination of punchdown block adaptors and vampire taps so that you can either test a circuit the way it was wired on the block or just pierce the sheathing of any random set of wires to sample the signal. Get yours at your local hardware store!

- **Induction Coils** All wires conducting voltages emit various electromagnetic signals outside of the intended channel. Such electromagnetic radiation is more pronounced in unshielded wires, such as UTP, due to the lack of shielding that plastic sheathing affords. As a consequence, it is theoretically possible to introduce what is called an “induction coil” alongside such wiring in order to translate the laterally emitted signals into their original digital form.

Induction coils are devices that essentially transform the magnetism of weak signals to induce a much stronger signal in an external system. Such a device could potentially capture the throughput of a cable without the detection of the users, administrators, or owners of the wires. However, such devices are not commercially available in a way that the public can acquire in order to surreptitiously tap Cat5e and Cat6. That’s not to say that serious hobbyists couldn’t build one, or that a dedicated investigator couldn’t acquire one.

This potential attack or surveillance mechanism (take your pick) tends to get more hype than it is probably due.

- **Fiber Optic Taps** Inline network taps work similarly for fiber optic cables and copper cables. To place a network tap inline on a fiber optic cable, network technicians splice the optic cable and connect it to each port of a tap. This causes a network disruption.

Inline optical taps may cause noticeable signal degradation. Network engineers often use tools called optical time-domain reflectometers (OTDR) to analyze and troubleshoot fiber optic cable signals. OTDRs can also be used to locate breaks in the cable,

4. Personal correspondence with Sherri Davidoff and Jonathan Ham.

including splices inserted for taps. With OTDRs, technicians can create a baseline of the normal signal profile of a fiber optic cable, and potentially detect not only when the profile changes but where on the cable the disruption has likely occurred.

It is much more difficult to surreptitiously tap a fiber optic cable than a copper cable. Vampire taps for copper cables can pierce the insulation and physically connect with the copper wires to detect the changes in voltage within. Monitoring light traveling through glass is not so easy. Bend couplers can be used to capture stray photons and gain some information about the signal within a fiber optic cable without cutting the glass. Theoretically, similar devices might exist that can fully reconstruct the optical signal without requiring investigators or technicians to physically splice the optic cable. However, at the time of this writing, noninterference fiber optic taps do not appear to be commercially available to the public.⁵

Undersea Cable Cuts: Truth or Conspiracy Theory?

In early 2008, a series of undersea cable disruptions in the Middle East were reported. These caused Internet and voice outages and slowdowns in India, Pakistan, Egypt, Qatar, Saudi Arabia, and several other countries. After three separate incidents in which five cables were damaged, many people began speculating that the cuts were not a coincidence, but rather deliberately caused to induce economic disruptions, to cause Internet rerouting, or to cover the installation of surreptitious fiber optic network taps.

Security professional Steve Bellovin wrote:

Four failures in less than a week. Coincidence? Or enemy action? If so, who's the enemy, and what are the enemy's goals? You can't have that many failures in one place—especially such a politically sensitive place—without people getting suspicious . . .

Now—the US certainly has the ability to tap undersea cables. After all, they did just that to the Soviets several decades ago.... That said, I don't think it's an NSA or Mossad operation.... Four failures at once will raise suspicions, and that's the last thing you want when you're eavesdropping on people.

If if wasn't a direct attempt at eavesdropping, perhaps it was indirect. Several years ago, a colleague and I wrote about link-cutting attacks. In these, you cut some cables, to force traffic past a link you're monitoring. Link-cutting for such purposes isn't new; at the start of World War I, the British cut Germany's overseas telegraph cable to force them to use easily-monitored links.^{6,7,8}

5. "Undersea Optical Cable Cuts," February 10, 2008, <http://cryptome.info/cable-cuts.htm>.

6. Bruce Schneier, "Schneier on Security: Fourth Undersea Cable Failure in Middle East," February 5, 2008, http://www.schneier.com/blog/archives/2008/02/fourth_undersea.html.

7. "The Internet: Of Cables and Conspiracies," *The Economist*, February 7, 2008, http://www.economist.com/node/10653963?story_id=10653963.

8. Steven M. Bellovin, "Underwater Fiber Cuts in the Middle East," February 4, 2008, <http://www.cs.columbia.edu/smb/blog/2008-02/2008-02-04.html>.

3.1.2 Radio Frequency

Since the late 1990s, radio frequency has become an increasingly popular medium for transmission of packetized data and Internet connectivity. The Institute of Electrical and Electronics Engineers (IEEE) published a series of international standards (“802.11”) for wireless local area network (WLAN) communication. These standards specify protocols for WLAN traffic in the 2.4, 3.7, and 5 GHz frequency ranges. The term “Wi-Fi” is used to refer to certain types of RF traffic, which include the IEEE 802.11 standards.^{9,10,11}

RF waves travel through the air, which is by nature a shared medium. As a result, WLAN traffic cannot be physically segmented in the way that switches segment traffic on a wired LAN. Because of physical media limitations, all WLAN transmissions may be observed and intercepted by all stations within range. Stations can capture the RF traffic, regardless of whether they participate in the link. This attribute makes passive acquisition of WLAN traffic very easy—both for investigators and attackers.

In the United States, the Federal Communications Commission has placed limits on the strength of emissions for stations operating in 802.11 frequency ranges, and the gain of antennae.^{12,13} As a result, there are practical limitations on the distances over which stations can legally capture and receive data over 802.11 networks in the United States. However, directional transceivers can be constructed from off-the-shelf components which can dramatically increase the effective ranges. Such devices may be illegal but are difficult to detect and prevent. If a suspect is already engaged in illicit activities, there is no reason to assume they won’t go further in their criminal activities and eavesdrop or transmit at distances that exceed FCC specifications. (In South America, researcher Ermanno Pietrosemoli announced that his team had successfully transferred data via Wi-Fi over a distance of 238 miles.)^{14,15}

Why does this matter for the investigator? First, investigators should keep in mind that the target of investigation may be able to access the WLAN from a long distance, far outside the physical perimeter of a facility. Second, investigators should remember that when you connect over wireless links, your activity can potentially be monitored from a great distance.

Even when the Wi-Fi traffic is encrypted, there is commonly a single pre-shared key (PSK) for all stations. In this case, anyone who gains access to the encryption key can listen to all traffic relating to all stations (as with physical hubs). For investigators, this is helpful

9. IEEE, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: 36503700 MHz Operation in USA,” *IEEE Standards Association* (June 12, 2007), <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>.

10. IEEE, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: 36503700 MHz Operation in USA” (November 6, 2008), <http://standards.ieee.org/getieee802/download/802.11y-2008.pdf>.

11. “fcc.gov,” http://wireless.fcc.gov/services/index.htm?job=service_home&id=3650_3700.

12. “EIRP Limitations for 80211 WLANs,” <http://www.wi-fiplanet.com/tutorials/article.php/1428941/EIRP-Limitations-for-80211-WLANs.htm>.

13. “CFR,” 2005, http://edocket.access.gpo.gov/cfr_2005/octqtr/47cfr15.249.htm.

14. Michael Kanellos, “New Wi-Fi Distance Record: 382 Kilometers,” *CNET News*, June 18, 2007, http://news.cnet.com/8301-10784_3-9730708-7.html?part=rss&subj=news&tag=2547-1_3-0-5.

15. David Becker, “New Wifi Record: 237 Miles,” *Wired*, June 2007, http://www.wired.com/gadgetlab/2007/06/w_wifi_record_2.

because local IT staff can provide authentication credentials, which facilitate monitoring. Furthermore, there are well-known flaws in common 802.11 encryption algorithms such as Wired Equivalent Privacy (WEP), which can allow investigators to circumvent or crack unknown encryption keys. Complicating matters, wireless access points (WAPs) employ a range of different standards for encryption and authentication. Some devices are even based on draft standards that have not yet been finalized (as of the time of this writing, that includes 802.11n).

It is possible to passively capture encrypted Wi-Fi traffic and decrypt it offline later using the encryption keys. Once an investigator has gained full access to unencrypted 802.11x traffic contents, this data can be analyzed in the same manner as any other unencrypted network traffic.

Regardless of whether or not Wi-Fi traffic is encrypted, investigators can gain a great deal of information by capturing and analyzing 802.11 management traffic. This information commonly includes:

- Broadcast SSIDs (and sometimes even nonbroadcast ones)
- WAP MAC addresses
- Supported encryption/authentication algorithms
- Associated client MAC addresses
- In many cases, the full Layer 3+ packet contents

In order to capture wireless traffic, forensic investigators must first have the necessary hardware. Many standard 802.11 network adapters and drivers do not include support for *monitor mode*, which allows the user to capture all packets on a network, not just packets destined for the host. The network adapter must also support the specific 802.11 protocol in use (i.e., 802.11a/b/g cards do not necessarily support 802.11n). Check your 802.11 network adapter's model and read about the corresponding drivers for your operating system to find out if your adapter can be used for 802.11 passive evidence acquisition.

There are commercially available 802.11 network adapters that are specifically designed for capturing packets. These adapters include very handy features for forensic investigators, such as the ability to operate completely passively (so the investigator does not have to worry about accidentally transmitting data), connectors for extra antennae, and portable form factors such as USB. One popular model is the AirPcap USB adapter, manufactured by Riverbed Technology.¹⁶ Riverbed Technology only supports the AirPcap device on Windows operating systems, though wireless security professional Josh Wright has engineered a modified BackTrack Linux distribution with modified drivers that work with some models. (Please see Chapter 6 for more details.)

3.1.3 Hubs

A network hub is a dumb Layer 1 device that physically connects all stations on a local subnet to one circuit. A hub does not store enough state to track what is connected to it, or how. It maintains no knowledge of what devices are connected to what ports. It is merely a

16. "Riverbed Technology—AirPcap," *Riverbed*, 2010, <http://www.riverbed.com/us/products/cascade/airpcap.php>.

physical device designed to implement baseband (or “shared”) connectivity. In other words, all the devices on the local segment that the hub provides are physically connected and, therefore, share the same physical medium.

When the hub receives a frame, it retransmits it on all other ports. Therefore, every device connected to the hub physically receives all traffic destined to every other device attached to the hub. If a hub exists in the network, then you can connect to it and trivially sniff all of the traffic on the segment. A wireless access point is a special example of a hub, which we discuss in more detail in Chapter 6.

Many devices that are currently labeled as “hubs” by the manufacturer are, in fact, switches. This can be confusing to sort out. There are two indicators that can help you determine whether something labeled as a hub really is a hub. The first way is to examine the lights on the front panel. If there is an LED labeled “collision,” it is a hub. If there is no such light, it’s probably a switch. The collision light indicates that two stations have transmitted at the same time and the physical medium must reset itself. On a busy network, this can happen thousands of times per second.

The more reliable way to determine if a device is actually a hub is to connect a station to it, put the network interface into promiscuous mode, and observe the traffic with `tcpdump` or a similar tool. If all you see is traffic destined for your station and broadcast traffic, then the device is a switch. If it is a hub, you should see all the other traffic.

Investigators must be careful when using hubs as traffic capture devices. The investigator sees all traffic on the segment, but so can everyone else. A compromised system could trivially act as a passive listener and eavesdrop on any data transfers or communications. Any evidence transmitted across the network, or normal traffic sent by the investigator’s operating system, may be trivially captured by anyone else on the local network. It may be appropriate to take advantage of a hub that is already installed on a network, but installing a hub for the purposes of traffic capture can introduce new risks unnecessarily.

Bottom line for the investigator: if you walk into an environment with a hub, you may want to take advantage of that. However, keep in mind that anything you can see, and anything you transmit, can be seen by anyone else on the network as well. Furthermore, the design of hubs varies greatly by manufacturer, and there is no guarantee that you are actually receiving all of the traffic you expect, or that it really is a hub. *Caveat emptor.*

3.1.4 Switches

Switches are the most prevalent Layer 2 device. Like hubs, they also connect multiple stations together to form a LAN. Unlike hubs, switches use software to keep track of which stations are connected to which ports, in its CAM table. When a switch receives a packet, it forwards it only to the destination station’s port. Individual stations do not physically receive each other’s traffic. This means that every port on a switch is its own collision domain. Switches operate at Layer 2 (the data-link layer), and sometimes Layer 3 (the network layer).

Even the simplest switch maintains a CAM table, which stores MAC addresses with corresponding switch ports. A MAC address is an identifier assigned to each station’s network card. The purpose of the CAM table is to allow the switch to isolate traffic on a port-by-port basis so that each individual station only receives traffic that is destined for it, and not traffic destined for other computers.

Switches populate the CAM table by listening to arriving traffic. When a switch receives a frame from a device, it looks at the source MAC address and remembers the port associated with that MAC address. Later, when the switch receives a packet destined for that device, it looks up the MAC address and corresponding port in the CAM table. It then sends the packet only to the appropriate port, encapsulated with the correct Layer 2 Ethernet address.

In this way, a switch segments the traffic endpoint-by-endpoint, even while technically sharing the same physical medium.

3.1.4.1 Obtaining Traffic from Switches

Investigators can, and often do, capture network traffic using switches. Even though by default switches only send traffic to the destination port indicated in the frame, switches with sufficient software capabilities can be configured to replicate traffic from one or more ports to some other port for aggregation and analysis. Different vendors have different terminology for this capability—probably the most common term is Cisco’s Switched Port Analyzer (SPAN) and Remote Switched Port Analyzer (RSPAN). The most vendor-neutral term for this is “port mirroring.”

Switches have varying port mirroring capabilities, depending on their make and model. It is common to encounter a switch that is capable of some port mirroring, but for that capability to be limited by the number of ports that can be mirrored, or by the number of ports that can be used for aggregation and analysis.

Port mirroring is inherently limited by the physical capacity of the switch itself. For example, let’s say you have a 100Mbps switch and you attempt to mirror four ports, which are each passing an average of 50Mbps to a single SPAN port. The total amount of traffic from all four ports adds up to 200Mbps, which is far above the capacity of any one port to receive. The result is “oversubscription,” and packets will be dropped by the switch.

You need administrative access to the switch’s operating system in order to configure port mirroring. Once you have mirrored the ports of interest, you can connect a sniffer to the mirroring port and capture all of the traffic.

If you don’t have administrative access, it is still possible to sniff traffic from a switch. Let’s study the methods that attackers use. In rare cases, such as when the network administrators themselves are not trusted, an investigator may need to use the same techniques as attackers. These are not the safest methods, as they cause the switch to operate outside normal parameters, but they can work.

To sniff traffic from a switch, attackers use one of two common methods. First, the attacker can flood the switch with bogus information for the CAM table by sending it many Ethernet packets with different MAC addresses. This attack is referred to as “MAC flooding.” Once the CAM table is filled, many switches by default will “fail open,” and send all traffic for systems not in the CAM table out to every port.

Second, an attacker can conduct an “ARP spoofing” attack. Normally, the Address Resolution Protocol (ARP) is used by stations on a LAN to dynamically map IP addresses (Layer 3) to corresponding MAC addresses (see Chapter 9 for details). In an ARP spoofing attack, the attacker broadcasts bogus ARP packets, which link the attacker’s MAC address to the victim’s IP address. Other stations on the LAN add this bogus information to their ARP tables, and send traffic for the router’s IP address to the attacker’s MAC address

instead. This causes all IP packets destined for the victim station to be sent instead to the attacker (who can then copy, modify, and/or forward them on to the victim).

To receive outbound communications, the attacker would again use ARP spoofing to link the attacker's MAC address to the IP address used by the victim's gateway. Any packets from the victim destined for the local gateway would be sent instead to the attacker, who could copy or modify them and then forward them along to the gateway.

Note that MAC flooding is an attack on the switch itself, whereas ARP spoofing or poisoning is an attempt to poison the ARP caches of all the systems on the LAN.

Sniffing on Switches

The safest way to obtain traffic from a switch is to coordinate with a network administrator to configure "port mirroring," in which traffic from ports of interest is mirrored to a port that is used by the investigator.

Switches can also be attacked in several ways to try to facilitate sniffing. The most common are:

- MAC flooding (which attacks the switch's CAM table directly)
- ARP spoofing (which attacks the ARP tables of the hosts on the LAN)

It would be hard to argue that either of these methods is really "passive," since they require an attacker to send extensive and continuing traffic on the network. However, these are methods for facilitating traffic capture on switched networks when port mirroring or tapping a cable is not an option.

Configuration of port mirroring with switches is vendor- and device-specific. The following example shows how to configure port mirroring for a Cisco ASA 5500 (IOS 8.3), which has the hostname "ant-fw." In this case, traffic on Ethernet ports 2, 3, 4, 5, and 6 are all copied to port 7.

```
ant-fw(config)# interface ethernet 0/7
ant-fw(config-if)# switchport monitor ethernet 0/6
ant-fw(config-if)# switchport monitor ethernet 0/5
ant-fw(config-if)# switchport monitor ethernet 0/4
ant-fw(config-if)# switchport monitor ethernet 0/3
ant-fw(config-if)# switchport monitor ethernet 0/2
```

Once the SPAN port is set up, an investigator can simply plug a forensic monitoring station into port 7 and sniff traffic copied from ports 2 through 6.

3.2 Traffic Acquisition Software

Once you gain physical access to network traffic, you need software to record it. In this section, we explore the most common software libraries used for recording, parsing, and analyzing captured packet data: libpcap and WinPcap. We also review tools based on these libraries, including tcpdump, Wireshark, and more. We delve into methods for filtering

traffic during and after capture, since the importance of filtering has increased in recent years in proportion to the rising volume of network traffic.

3.2.1 libpcap and WinPcap

Libpcap is a UNIX C library that provides an API for capturing and filtering data link-layer frames from arbitrary network interfaces. It was originally developed at the Lawrence Berkeley National Laboratory (LBNL),¹⁷ and initially released to the public in June 1994.¹⁸ Different UNIX systems have different architectures for processing link-layer frames. Consequently, programmers writing a utility on UNIX to inspect or manipulate link-layer frames originally had to write operating system-specific routines for accessing them. The purpose of libpcap was to provide a layer of abstraction so that programmers could design portable packet capture and analysis tools.

In 1999, the Computer Networks Group (NetGroup) in the Politecnico di Torino published WinPcap, a library based on libpcap that was designed for Windows systems. Since then, many people and companies have contributed to the WinPcap project. The code is now hosted at a site maintained by Riverbed Technology (also the commercial sponsor of Wireshark).

The most popular packet sniffing and analysis tools today are based on the libpcap libraries. These include tcpdump, Wireshark, Snort, nmap, ngrep, and many others. Consequently, these tools are interoperable in the sense that packets captured with one tool can be read and analyzed with another. A quintessential feature of libpcap-based utilities is that they can capture packets at Layer 2 from just about any network interface device and store them in a file for later analysis. Other tools can then read in these “packet capture” or “pcap” files while filtering the traffic based on specific protocol information, perhaps writing out a more refined packet capture for yet further analysis.

Many tools based on libpcap also include specialized functionality, such as the ability to merge packet captures (i.e., mergecap), to split a capture up by TCP streams (i.e., tcpflow), or to conduct regular expression searches on packet contents (i.e., ngrep). There are, in fact, so many libpcap-based tools that we can’t hope to cover them all here. Instead, we focus on the tools that are most commonly used.

Both libpcap and WinPcap are free software released under the “BSD license,” which has been approved by the Open-Source Initiative.¹⁹

3.2.2 The Berkeley Packet Filter (BPF) Language

In the modern era, the volume of data that flows across networks has become so huge that it is very important for investigators to know how to filter it during both capture and analysis.

Libpcap includes an extremely powerful filtering language called the “Berkeley Packet Filter” (BPF) syntax.²⁰ Using BPF filters, you can decide which traffic to capture and inspect and which traffic to ignore. BPF allows you to filter traffic based on value comparisons in fields for Layer 2, 3, and 4 protocols. It includes built-in references called “primitives” for

17. “LBNL’s Network Research Group,” August 2009, <http://ee.lbl.gov/>.

18. “Libpcap,” <http://www.tcpdump.org/release/libpcap-0.5.tar.gz>.

19. “Open Source Licenses—Open Source Initiative,” 2011, <http://www.opensource.org/licenses>.

20. “TCPDUMP/LIBPCAP public repository,” 2010, <http://www.tcpdump.org>.

many commonly used protocol fields. BPF invocations can be extremely simple, constructed from primitives such as “host” and “port” specifications, or very arcane constructions involving specific field values by offset (even down to individual bits). BPF filters can also consist of elaborate conditional chains, nesting logical ANDs and ORs.

BPF syntax is so widely used and supported by traffic capture and analysis tools that every network investigator should be familiar with it. In this section, we review basic elements of the BPF syntax.

3.2.2.1 BPF Primitives

By far, the easiest way to construct a BPF filter is to use BPF *primitives* to refer to specific protocols, protocol elements, or qualities of a packet capture. Primitives, as defined by the “pcap-filter” manual page, “usually consist of an id (name or number) preceded by one or more qualifiers.” The list of available primitives seems to grow with every revision of libpcap/tcpdump. The manual specifies three different kinds of qualifiers:²¹

- **type** qualifiers say what kind of thing the id name or number refers to. Possible types are host, net, port and portrange.
- **dir** qualifiers specify a particular transfer direction to and/or from id. Possible directions are src, dst, src or dst, src and dst, addr1, addr2, addr3, and addr4.
- **proto** qualifiers restrict the match to a particular protocol. Possible protos are ether, fddi, tr, wlan, ip, ip6, arp, rarp, decnet, tcp and udp.

There are many other types of primitives available. Check the manual pages for your version of libpcap or libpcap-based tool.

Perhaps the most commonly used BPF primitive is “host *id*,” which is used to filter traffic involving a host, where *id* is set to an IP address or name. This can be further restricted by directionality using the primitives “dst host *id*” or “src host *id*” to specify that the IP address in question must be either the source or destination, respectively. The same principles apply for the “net,” “ether host,” and “port” classes of primitives. Filtering can also be done based on protocol, using primitives such as “ip” (to filter for IPv4 packets), “ether” (to filter for Ethernet frames), “tcp” (to filter for TCP segments), or “ip proto *protocol*” (to filter for IP packets that encapsulate the specified protocol). (Please see the tcpdump or “pcap-filter” manual page for a full list of options.) Typing something as simple as ‘tcp and host 10.10.10.10’ produces output that only includes TCP traffic to and from 10.10.10.10, with all other frames filtered out.

For instance, suppose we want to see only the traffic in which a computer with the IP address 192.168.0.1 communicates with any other system *except* 10.1.1.1 over ports 138, 139, or 445.

Here’s a BPF filter that would accomplish this:

```
'host 192.168.0.1 and not host 10.1.1.1 and (port 138 or port 139 or port
445) '
```

21. “Man Page for pcap- (freebsd Section 0)—The UNIX and Linux Forums,” January 6, 2008, <http://www.unix.com/man-page/freebsd/7/pcap-filter>.

BPF Primitives

Commonly used BPF primitives include:²²

- host *id*, dst host *id*, src host *id*
- net *id*, dst net *id*, src net *id*
- ether host *id*, ether dst host *id*, ether src host *id*
- port *id*, dst port *id*, src port *id*
- gateway *id*, ip proto *id*, ether proto *id*
- tcp, udp, icmp, arp
- vlan *id*

There are many other BPF primitives. Please see the `tcpdump` or “pcap-filter” manual page for a full list.

3.2.2.2 Filtering Packets by Byte Value

In addition to primitive comparisons, the BPF language can be used to compare the values of any byte-sized (or multibyte-sized) fields within a frame. The BPF language provides syntax to specify the byte offset relative to the beginning of common Layer 2, 3, and 4 protocols. Important: Byte offsets are counted starting from 0! For example, the first byte in a structure is at offset “0,” and is referred to as the zero-byte offset. The seventh byte in a structure is at offset “6,” and is referred to as the sixth-byte offset.

Here are some examples:

- `ip[8] < 64`
This filter would match all packets in which the single byte field starting at the eighth byte offset of the IP header, is less than 64. This field is called the “Time to live,” or “TTL.” The default starting TTL in most Windows systems is 128, so this would probably weed out traffic from Windows systems on the LAN, while matching traffic from Linux systems (which usually have a default starting TTL of 64).
- `ip[9] != 1`
This filter would match frames whose single byte field at the ninth byte offset of the IP header does *not* equal “1.” Since the ninth byte offset of the IP header specifies the embedded protocol, and “1” represents “ICMP,” this would match any packet where the embedded protocol is not ICMP. This expression is equivalent to the primitive “`not icmp`”.
- `icmp[0] = 3 and icmp[1] = 0`
This construct isolates all ICMP traffic where the one-byte field at the 0 byte offset of the ICMP header is equal to 3, and where the one-byte field at the first byte offset

22. “`tcpdump(8)`: dump traffic on network—Linux man page,” 2011, <http://linux.die.net/man/8/tcpdump>.

is equal to 1. In other words, this filter matches only ICMP traffic where the packet is of ICMP type 3 code 0: “Network Unreachable” messages.

- `tcp[0:2] = 31337`

This construct introduces the notation for specifying a multibyte field (2 bytes) at the 0 byte offset of the TCP header (the TCP source port). In this case, it would be equivalent to using the BPF primitive “src port 31337.”

- `ip[12:4] = 0xC0A80101`

Here we see a 4-byte comparison of the contents of the IP header beginning at the 12th byte offset: the source IP address. Notice that the comparison is made in hexadecimal notation. Converted to decimal notation, this is equivalent to 192.168.1.1 ($0x\text{C}0 = 192$, $0xA8 = 168$, $0x01 = 1$). This filter matches all traffic where the source IP address is 192.168.1.1.

3.2.2.3 Filtering Packets by Bit Value

Of course, not all protocol header fields are a byte or more in size, and also begin and end precisely on the byte-boundary. Fortunately, the BPF language includes a way to specify smaller or offset fields (although it may seem complicated). To do this, we cite a specific byte or bytes (as explained above), and then compare them bit-by-bit to some value that we hope to find. This is called “bitmasking.” Essentially, we must identify one or more byte-sized chunks of data that contain the bits we are interested in, and then specify the particular bits of interest using a binary representation known as a “bitmask.” In the bitmask, “1” represents a bit of interest and “0” represents a bit we choose to ignore.

For example, if we’re only interested in the four lowest-order bits of a byte (the lowest-order “nibble”), we can represent this using a bitmask of “00001111” (in binary), which can also be represented as the value “0x0F” (in hexadecimal).

The IP header is a minimum of 20 bytes long, by specification. It is possible to include optional header fields which increase the IP header length. However, IP options are not commonly used in practice (for a while, it was vogue for attackers to use IP options to configure source routing, which facilitated man-in-the-middle attacks).

Let’s suppose that we’d like to filter for packets where IP options are set (in other words, the IP header length is greater than 20 bytes). The low-order nibble of the IP header represents the IP header length, measured in 32-bit “words” (each word is four bytes long). To find all packets where the IP header is greater than 20 bytes in length, we need to match packets where the low-order nibble is greater than five (5 words * 4 bytes per word = 20 bytes). To accomplish this, we create a BPF filter with a bitmask of “00001111” (0x0F), which is logically “AND-ed” with the targeted value. The resulting expression is:

```
ip[0] & 0x0F > 0x05
```

Likewise, if we want to find all of the IP packets where the “Don’t Fragment” flag (a single binary bit at byte offset 6 of the IP header) is set to 1, we can create a binary bitmask of “01000000” (0x40) to denote that we’re only interested in the packets where the second-to-highest-order bit is 1, and then compare this to byte offset 6 of the IP header:

```
ip[6] & 0x40 != 0
```

BPF Language

- libpcap includes the Berkeley Packet Filter (BPF) language
- Almost all libpcap-based utilities leverage this functionality
- Simple filters can be built with *primitives*
- Any byte-sized protocol field can be compared with a numerical value
- Bitwise filters can be built with *bitmasking* and value comparisons
- Complex expressions can be constructed via nested logical ANDs and ORs

3.2.3 tcpdump

Tcpdump is a tool for capturing, filtering, and analyzing network traffic. It was developed at LBNL, and first released to the public in January 1991.²³ Note that although tcpdump currently relies upon libpcap for functionality, its public release actually predates that of libpcap.

Tcpdump was designed as a UNIX tool. In 1999, researchers at the Politecnico di Torino ported it to Windows and released “WinDump,” a Windows version.²⁴ WinDump is now hosted at a site maintained by Riverbed Technology, along with WinPcap. The tcpdump and WinDump utilities are not completely identical, but typically produce comparable results. It can happen that a packet capture produced by WinDump is not subsequently readable by tcpdump, but such an occurrence is infrequent. The command syntax for both tools is very similar. For the purposes of this book, you can consider them interchangeable except when differences are explicitly mentioned.

The basic purpose of tcpdump is to capture network traffic and then print or store the contents for analysis. Tcpdump captures traffic bit-by-bit as it traverses any physical media suitable for conducting link-layer traffic (copper, fiber, or even air). Since tcpdump is based on libpcap, it captures at Layer 2 (the data link layer). (By default, tcpdump’s output only displays Layer 3 and above protocol details, but the “-e” flag can be used to show Layer 2 data as well.)

Beyond merely capturing packets, tcpdump can decode common Layer 2 through 4 protocols (and some higher-layer protocols as well), and then display their information to the user. The decoded packets can be displayed in hexadecimal or in the ASCII equivalents (where the data is textual), or both.

There are two ways that tcpdump is most commonly employed. First, it is used to facilitate on-the-fly analysis for troubleshooting network issues in a tactical way. This typically encompasses capture, filtering, and analysis, all performed simultaneously. However, as

23. “Internet Archive Wayback Machine,” <http://web.archive.org/web/20001001124902/http://www.tcpdump.org/release/tcpdump-3.5.tar.gz>.

24. Riverbed Technology, “WinDump—Change Log,” December 4, 2006, <http://www.winpcap.org/windump/misc/changelog.htm>.

common as this practice is, it tends to be suitable only when a quick glance at the data will suffice.

Tcpdump is also frequently used to capture traffic of interest passing on a target segment over a longer period of time, and store it for offline analysis and perhaps even future correlation with other data. Depending on the throughput and utilization of the network segment and the amount of each packet retained, the volume of data captured can be enormous.

3.2.3.1 Fidelity

One reason that tcpdump is such a powerful tool is that it is capable of capturing traffic with high fidelity, to the degree that the resulting packet capture can constitute evidence admissible in court. However, the quality of the packet capture can be impacted by hardware limitations and configuration constraints.

For example, tcpdump's ability to capture packets may be limited by the clock speed of the processor in the capturing workstation. Capturing packets is a CPU-intensive activity. If the CPU of the capturing station becomes too heavily utilized, either by the tcpdump process itself or by any other running process, tcpdump will “drop packets”—in other words, it will fail to capture them. When this happens, there will be packets flying by that tcpdump simply doesn't have the resources to pick up, and it will ignore them entirely.

If you are merely interested in sampling packets, this may not matter too much. However, if your goal is to obtain a high-fidelity capture of ongoing network traffic, missing nothing, this problem would be absolutely critical.

Especially on high-traffic networks, investigators may also be limited by disk space. If the capturing workstation does not have enough disk space to store the volume of traffic that investigators hope to be able to inspect, it may be necessary to either filter traffic upon capture or provide more disk storage.

One crucial configuration option for capturing packets using tcpdump is the snapshot length, known as “snaplen.” Snaplen represents the number of bytes of each frame that tcpdump will record. It is calculated from the zero-byte offset of the data link-layer frame.

Selecting the correct snaplen for a packet capture is critical. If the chosen snaplen is too short, data will be missing from every frame and can never be recovered. If the snaplen is too long, it may cause performance degradation, limit the volume of traffic that can be stored, and perhaps cause violations of regulations such as the United States Wiretap Act, which prohibit capturing communications contents except in certain circumstances.

Tcpdump's default snaplen varies depending on the version used. As of version 4.1.0, by default tcpdump captures the full frame automatically.²⁵ This has undoubtedly saved many forensic investigators from much pain and heartache. Older versions of tcpdump had a default snaplen of 68, meaning only the first 68 bytes of the frame were actually captured. To capture full packet contents, users had to manually specify a larger value. Woe to the investigator who forgot to specify a longer snaplen, only to discover later that every packet was truncated, the contents never to be recovered!

Once upon a time, many people recommended using a snaplen of 1,514 bytes because the maximum transmission unit (MTU) of Ethernet is 1,500 bytes. Since the Ethernet header itself is 14 bytes long, this meant that the total frame length was 1,514 bytes. Unfortunately,

25. “tcpdump,” <http://www.tcpdump.org/release/tcpdump-4.1.1.tar.gz>.

many people erroneously specified a snaplen of just 1,500 bytes, forgetting to account for the 14-byte Ethernet header. As a result, they accidentally truncated every IP packet by 14 bytes, which rendered full content reconstruction impossible.

Later versions of tcpdump allowed the user to specify “0” for the snaplen, which would tell tcpdump to automatically capture the entire frame, no matter how long it was. Modern versions of tcpdump maintain this syntax for backward compatibility.

For performance and regulatory reasons, investigators may still wish to specify a smaller snaplength. As described in the tcpdump manual, “taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit snaplen to the smallest number that will capture the protocol information you’re interested in.”²⁶

3.2.3.2 Filtering Packets with tcpdump

Filtering during capture is very important because resources such as disk space, CPU cycles, and traffic aggregation capacity are always limited. Filtering indiscriminately, however, can cause loss of evidence, which can never be recaptured. *You only get one chance to capture a frame at the moment it zips past on the wire (or through the air).* Once you miss it, it’s gone forever and will never be part of your analysis.

The ability to filter is also crucial during analysis. When resources permit, it is often useful to “throw a wide net” and sift through the data offline. However, it can be very difficult to find the packets of interest within potentially huge volumes of data.

Fortunately, since tcpdump is a libpcap-based tool, it incorporates the BPF language, which investigators can use to filter traffic during capture and analysis. One good strategy for analysis of large volumes of traffic is to begin by filtering out any types of traffic that are not related to the investigation. For example, imagine that we have a packet capture that contains 75% web traffic (TCP port 80), which is fairly typical for an enterprise network. If our investigation is entirely unrelated to web traffic, we might decide to use a BPF filter of ‘not (tcp and port 80)’ to cut the volume by 75%. Phew!

Here’s an example that shows how tcpdump is used to display traffic from the “eth0” network interface, excluding TCP port 80 traffic:

```
# tcpdump -nni eth0 'not (tcp and port 80)'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
12:49:33.631163 IP 10.30.30.20.123 > 10.30.30.255.123: NTPv4, Broadcast,
    length 48
12:49:38.197072 IP 192.168.30.100.57699 > 192.168.30.30.514: SYSLOG local2.
    notice, length: 1472
12:49:38.197319 IP 192.168.30.100.57699 > 192.168.30.30.514: SYSLOG local2.
    notice, length: 1472
12:49:38.197324 IP 192.168.30.100 > 192.168.30.30: udp
12:49:38.197327 IP 192.168.30.100 > 192.168.30.30: udp
12:49:38.197568 IP 192.168.30.100.57699 > 192.168.30.30.514: SYSLOG local2.
    notice, length: 1472
12:49:38.197819 IP 192.168.30.100.57699 > 192.168.30.30.514: SYSLOG local2.
    notice, length: 1472
12:49:38.197825 IP 192.168.30.100 > 192.168.30.30: udp
```

26. “Manpage of TCPDUMP,” March 5, 2009, http://www.tcpdump.org/tcpdump_man.html.

```

12:49:38.197827 IP 192.168.30.100 > 192.168.30.30: udp
12:49:38.197829 IP 192.168.30.30.39879 > 10.30.30.20.53: 16147+ PTR?
    100.30.168.192.in-addr.arpa. (45)
10 packets captured
10 packets received by filter
0 packets dropped by kernel

```

In Figure 3–1, we have shown a few commonly used tcpdump command-line options. There are many more command-line options for tcpdump than are presented here, but these are some of the most important for capturing traffic and storing it in files.

The -C option, used with -w, allows analysts to specify the maximum size of a packet capture file, in millions of bytes. Once the file has reached the specified size, tcpdump closes it and opens a new file that has the same name with a number appended to it (starting at 1 and incrementing). This allows investigators to keep pcap files limited to a size that is easily transferred and practical for inspection with other analysis tools such as Wireshark. In addition, using the -W option along with the -C option, analysts can specify how many of these output files should exist on the hard drive at any one time. Once the limit has been reached, tcpdump begins to overwrite the oldest file, creating a rotating store of packets that occupies a fixed amount of disk space.

Below are five common invocations of tcpdump, which illustrate some of the basic functionality:

- `tcpdump -i eth0 -w great_big_packet_dump.pcap`

This is the simplest case of listening on interface eth0 and writing all of the packets out to a single monolithic file.

- `tcpdump -i eth0 -s 0 -w biggest_possible_packet_dump.pcap`

This instance is similar to the one above, except that by setting the snaplength to zero, we are telling tcpdump to grab the entire frame regardless of its size (rather than the first 68 bytes only). Note that specifying -s 0 is not necessary for newer versions

tcpdump command-line usage:

```

-i Listen on interface (eth0, en1, 2)
-n Do not resolve addresses to names.
-r Read packets from a pcap file
-w Write packets to a pcap file
-s Change the snapshot length from the default
-C With -w, limit the capture file size, and begin a new file when it is
exceeded
-W With -C, limit the number of capture files created, and begin
overwriting and rotating when necessary
-D List available adapters (WinDump only)

```

Figure 3–1. Handy tcpdump command-line options. See the `tcpdump` manual for more details.²⁷

27. *Ibid.*

of tcpdump, because the command functionality was updated to make this behavior the default.

- `tcpdump -i eth0 -s 0 -w targeted_full_packet_dump.pcap 'host 10.10.10.10'`

Here we introduce a simple BPF filter to grab and store in their entirety only those packets sent to or from the host at the address “10.10.10.10.”

- `tcpdump -i eth0 -s 0 -C 100 -w rolling_split_100MB_dumps.pcap`

Here we abandoned our host-based targeting, and instead we are grabbing all of every frame, but splitting the captures into multiple files no larger than 100MB each.

- `tcpdump -i eth0 -s 0 -w RFC3514_evil_bits.pcap 'ip[6] & 0x80 != 0'`

Finally, we introduce a more complicated BPF filter, in which we target the first byte of the IP fragmentation fields (byte offset 6). We employ a bitmask to narrow our inspection to the single highest order bit, most commonly known as the IP “reserved bit,” and we capture and store the packet only if the reserved bit is nonzero.

The Evil Bit (RFC 791)

The original RFC 791 “Internet Protocol,” published in 1981, specified that the very first, or “high order,” bit of the sixth byte offset would be “reserved”—in other words, unused—and as such it should be zero.²⁸

On April Fool’s Day in 2003, Steve Bellovin published RFC 3514, “The Security Flag in the IPv4 Header,” which proposed that this bit be used as a “security flag.” Bellovin suggested that any packet that had been built for the purpose of benign and normal IP traffic keep the bit unset, but that any packet that had been built for malicious or evil intent must set this bit to one. He reasoned that as a result, firewall vendors and those interested in intrusion detection would have a much easier time detecting which packets to disallow or to alert on.²⁹

The following tcpdump invocation allows us to capture only traffic with the RFC 3514 Evil Bit set, while ignoring all the rest:

```
tcpdump -i eth0 -s 0 -w RFC3514_evil_bits.pcap 'ip[6] & 0x80 != 0'
```

For those attackers who are interested in adhering to specification, Jason Mansfield has created an “evil bit changer.” This handy utility captures traffic or reads it from a file, sets the Evil Bit to “1,” recalculates the IP header checksum, and then forwards the frame along to its intended destination.³⁰

28. Information Sciences Institute, USC, “RFC 791—Internet Protocol: Darpa Internet Program Protocol Specification,” September 1981, <http://www.rfc-editor.org/rfc791.txt>.

29. S. Bellovin, “RFC 3514—The Security Flag in the IPv4 Header,” *IETF*, April 1, 2003, <http://www.rfc-editor.org/rfc/rfc3514.txt>.

30. Jason Mansfield, “evilbitchanger—Set the evil bit on IP packets.—Google Project Hosting,” 2011, <http://code.google.com/p/evilbitchanger/>.

3.2.4 Wireshark

Wireshark is a graphical, open-source tool designed for capturing, filtering, and analyzing traffic. Wireshark's easy-to-use graphical user interface makes it a great first tool for novice network forensics analysts, while its advanced packet filtering capabilities, protocol decoding features, and support for the Packet Details Markup Language (PDML) also make it extremely useful for experienced investigators.

Wireshark (originally named “Ethereal”) was initially released in 1998 by Gerald Combs. (The name was changed in 2006 when Combs moved to CACE Technologies because his previous employer maintained the trademark “Ethereal.” Subsequently, CACE Technologies was acquired by Riverbed Technology.) Over time, Wireshark has continued to mature, and there are now hundreds of contributing authors.³¹

Wireshark allows you to capture packets on any system network interface, assuming you have appropriate permissions to do so and your network card supports sniffing. Wireshark can display packets as they are captured in real time. It is a very powerful protocol analyzer and uses a lot of processing power crunching through protocol data. Recall that in our previous discussion of libpcap that if the CPU load is too high, packets can get dropped. It is worth carefully examining and tuning Wireshark’s options for capturing packets, especially since displaying and filtering packets while capturing can use up CPU power and result in lost packets.

Figure 3–2 shows a screenshot of Wireshark’s “Capture Options” panel.

3.2.5 tshark

Tshark is a command-line network protocol analysis tool that is part of the Wireshark distribution. Like Wireshark, it is libpcap-based, and can read and save files in the same standard formats as Wireshark. In addition to analysis, you can also use tshark to capture packets.³² The example below shows tshark capturing traffic on the network interface “eth0,” filtering out all port 22 traffic, and storing the results in the file “test.pcap.”

```
# tshark -i eth0 -w test.pcap 'not port 22'  
Capturing on eth0  
235
```

Please see Section 4.1.2.3 for more discussion of tshark’s protocol analysis capabilities.

3.2.6 dumpcap

The Wireshark distribution also comes with a command-line tool, “dumpcap,” which is specifically designed to capture packets.³³ If you would like to use the Wireshark distribution to capture packets for an investigation, the dumpcap tool is probably your best choice. Since dumpcap is a specialized tool designed just for capturing packets, it takes up fewer system resources, maximizing your capture capabilities. Furthermore, many operating systems limit traffic sniffing to programs that are run with administrative privileges. Wireshark

31. “Wireshark About,” 2011, <http://www.wireshark.org/about.html>.

32. “tshark—The Wireshark Network Analyzer 1.5.0,” <http://www.wireshark.org/docs/man-pages/tshark.html>.

33. “Wireshark,” http://www.wireshark.org/docs/wsug_html_chunked/AppToolsdumpcap.html.

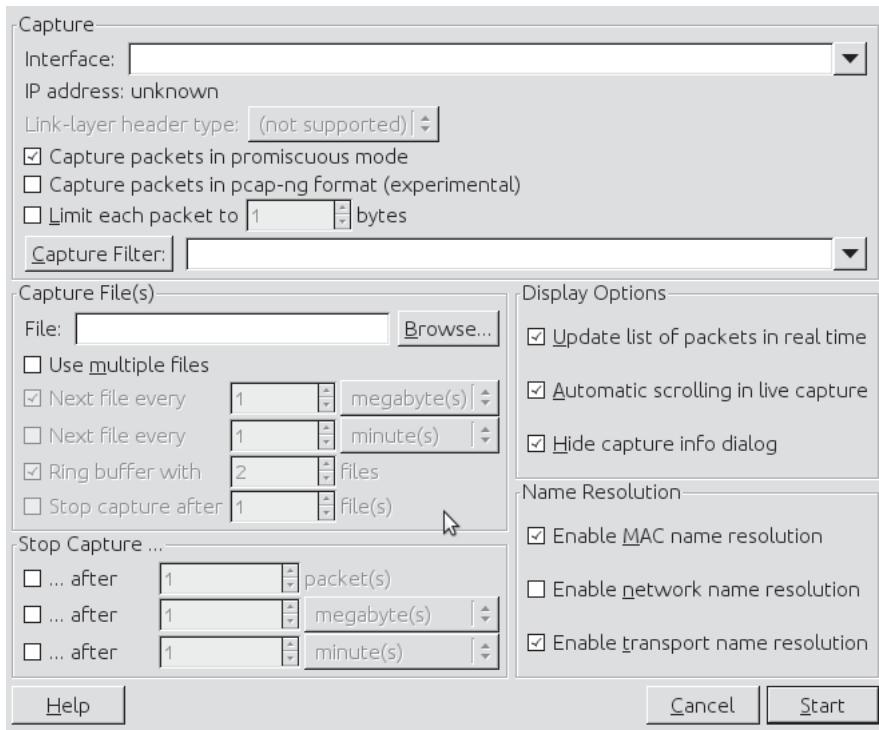


Figure 3–2. Wireshark’s “Capture Options” screen (Wireshark version 1.2.11).

itself is a complex, multifaceted program. From a security perspective, it is safer to limit administrative access to the smaller, simpler dumpcap program.

Dumpcap automatically writes packet captures to a file. Here is an example in which we use dumpcap to capture traffic on the interface eth0, filter out all port 22 traffic, and save the results in the file “test.pcap”:

```
$ dumpcap -i eth0 -w test.pcap 'not port 22'  
File: test.pcap  
Packets: 12  
Packets dropped: 0
```

In Chapter 4, we discuss Wireshark’s packet analysis and data extraction capabilities.

3.3 Active Acquisition

As we discussed in Chapter 2, evidence lives in many places throughout a network. In addition to capturing network traffic as it travels through wires and in the air, we may also choose to gather evidence from network devices, including firewalls, web proxies, logging servers, and more. In many cases, these devices cannot be removed from the production environment

without causing serious damage to business operations. In other cases, the evidence stored is highly volatile and must be collected while the system is still running, perhaps even still on the network. For these reasons and others, network forensic investigators often interact with network devices that are live and on the network.

By definition, active evidence acquisition modifies the environment. Investigators should be highly aware of the various ways in which live acquisition modifies the devices and environment under investigation, and work to minimize the impact.

3.3.1 Common Interfaces

In this section, we review common ways that investigators gain access to live network-based devices, including:

- Console
- Secure Shell (SSH)
- Secure Copy (SCP) and SSH File Transfer Protocol (SFTP)
- Telnet
- Simple Network Management Protocol (SNMP)
- Trivial File Transfer Protocol (TFTP)
- Web and proprietary interfaces

In later chapters, we review evidence acquisition and analysis relating to specific devices, including firewalls, web proxies, central log servers, and more.

3.3.1.1 Console

The console is an input and display system, usually a keyboard and monitor connected to a computer. Many network devices have a serial port that you can use to connect a terminal to the console.

It is possible to connect modern laptops and desktops to the serial console of network devices using USB-to-serial adapters. Figure 3–3 shows an example of a Keyspan serial-to-USB adapter. Once your forensic workstation is connected to the serial port using a



Figure 3–3. A Keyspan serial-to-USB adapter, which can be used to connect a modern laptop or desktop to the serial console of a network device.

serial-to-USB adapter, you can use the Linux “screen” command to connect to the console and log your session:

```
$ screen -L /dev/ttyUSB0
```

Whenever possible, it is best to connect directly to the console of a network device rather than connecting remotely over the network. When you connect to a device over the network, you create additional traffic and often unintentionally change the state of local networking devices (such as CAM tables, log files, etc.). When you connect directly to the console, you can dramatically reduce your footprint.

3.3.1.2 Secure Shell (SSH)

The Secure Shell protocol (SSH) is a common way for investigators to gain remote command-line access to systems containing network-based evidence. Developed as a replacement for the insecure Telnet and rlogin, SSH encrypts authentication credentials and data in transit. This means that even if the SSH traffic is intercepted, an attacker would be unable to recover the username, password, or contents of the communication. OpenSSH is a widely used implementation of SSH, which has been released free and open-source under a BSD license.

Most modern network devices now support SSH as a method for remote command-line interaction. Here is an example in which we use SSH to log into a system remotely on TCP port 4022, using the account “sherri”:

```
$ ssh -p 4022 sherri@remote.lmgsecurity.com
```

You can also use SSH to run commands remotely. In the following example, we execute the command “hostname” to retrieve the hostname of the remote server (which is named “remote”):

```
$ ssh -p 4022 sherri@remote.lmgsecurity.com 'hostname'  
remote
```

3.3.1.3 Secure Copy (SCP) and SFTP

In addition to providing interactive command-line access, SSH implements the Secure Copy Protocol (SCP), which is a command-line utility designed to transfer files between networked systems. Local files can be referred to by their local paths, while remote files are specified by the username, hostname, and path to the file on the remote system, as in the following example:

```
$ scp -P 4022 jonathan@remote.lmgsecurity.com:/etc/passwd .
```

Here we copied the /etc/passwd file from the remote server to the current working directory on our local system, using the account “jonathan.” Note the single dot argument at the end of the line, which specifies that we want to copy in our local directory. Also note the *capital* “P” used to specify the port (in contrast, the “ssh” command uses a lowercase “p” for this purpose, which can be confusing).

The SSH File Transfer Protocol, or SFTP, is an alternative protocol used in conjunction with SSH for secure file transfer and manipulation. It is more portable and offers more capabilities than SCP, but file transfer tends to be slower.

3.3.1.4 Telnet (yes, Telnet)

You may still encounter network devices that can only be accessed remotely using Telnet. Telnet is a command-line remote communications interface, which was originally developed in 1969 and eventually standardized by the IETF in RFCs 854 and 855.^{34,35} It became extremely widespread and is still used to this day for remote access on many network devices. In addition to connecting to Telnet servers, the Telnet client can be used to interact with a wide variety of servers, such as SMTP and HTTP.

As was common for communications protocols developed in the early days of networking, Telnet had only limited security built-in. All transactions are in plain text, so authentication credentials and data are sent unencrypted across the wire. If you are investigating a network that an attacker may be monitoring, be very careful because simply logging into a device via Telnet will expose its credentials to anyone who can capture traffic on the local segment.

Despite its serious security drawbacks, in many cases Telnet is the only option for remote access to a network device, because some devices have such limited hardware or software capacities that they cannot upgrade to more secure remote access tools such as SSH.

Here is an example in which we use Telnet to connect to a remote HTTP server on port 80:

```
$ telnet lmgsecurity.com 80
Trying 204.11.246.1...
Connected to lmgsecurity.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: lmgsecurity.com

HTTP/1.1 200 OK
Date: Sun, 26 Jun 2011 21:39:33 GMT
Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny10 with Suhosin-Patch
    mod_python/3.3.1 Python/2.5.2 mod_ssl/2.2.9 OpenSSL/0.9.8g mod_perl/2.0.4
    Perl/v5.10.0
Last-Modified: Thu, 23 Jun 2011 22:40:55 GMT
ETag: "644284-17da-4a668c728ebc0"
Accept-Ranges: bytes
Content-Length: 6106
Content-Type: text/html
```

3.3.1.5 Simple Network Management Protocol (SNMP)

One of the most commonly used protocols for network device inspection and management is the Simple Network Management Protocol (SNMP). Using SNMP, you can poll networked devices from a central server, or push SNMP information from remote agents to such a central aggregation point. SNMP is frequently used as a medium for communicating and

34. J. Postel and J. Reynolds, “RFC 854—Telnet Protocol Specification,” *IETF*, May 1983, <http://rfc-editor.org/rfc/rfc854.txt>.

35. J. Postel and J. Reynolds, “RFC 855—Telnet Option Specifications,” *IETF*, May 1983, <http://rfc-editor.org/rfc/rfc855.txt>.

aggregating both network management information (often of interest to the forensic analyst) and security event data (usually of *great* interest to forensic analysts). In network forensics, SNMP is commonly used in one of two ways: event-based alerting and configuration queries.

SNMP was designed to be extensible through the definition of the “management information base” (MIB), which basically describes the database of managed information. Unfortunately, this MIB definition is based on ASN.1 notation, which has been found to have multiple flaws both in design and in implementation (parsing languages are notorious for their propensity for errors in input validation). There are also problems with the authentication model, which have been best addressed in the current version (SNMPv3).

SNMP Operations

Here’s a list of the basic SNMP operations:

- Polling: GET, GETNEXT, GETBULK
- Interrupt: TRAP, INFORM
- Control: SET

As flawed as SNMP has historically been, it is still widely used and can be a critical component in network forensic investigations. There are far too many SNMP-based tools in wide deployment to even begin to discuss them all here. However, they all work in roughly the same way, as we now discuss.

In products that poll SNMP agents, there are a few methods available: GET, GETNEXT, and GETBULK. These operations are employed to retrieve information from the managed device, including routing tables, system uptime, hostname, ARP tables, CAM tables, and more. The “GET” operation is used to retrieve one piece of information, whereas the “GETNEXT” and “GETBULK” commands are used to retrieve multiple pieces of information.

Many devices can be configured to emit SNMP traps, which are used to communicate information about events on the device. Rather than polling, the central SNMP console can receive events via the SNMP TRAP operation from many sources as they occur and aggregate them. SNMP traps ensure timely notifications while reducing unnecessary network traffic.

SNMP can also be used to control the configuration of remote devices using the “SET” command.

SNMPv1 and SNMPv2 use “community strings” for authentication, which are sent in plain text across the network. As a result, there is significant risk of credential theft if the community strings are intercepted as they are sent across the network. Commonly, the community string “public” has read-only access to the MIB, and “private” has read-write access. SNMPv3 supports strong encryption algorithms that can be used to encrypt authentication data and packet contents, if these options are selected.

3.3.1.6 Trivial File Transfer Protocol (TFTP)

The Trivial File Transfer Protocol (TFTP) was first published in 1980, and designed as a simple, automated means of transferring files between remote systems. Like Telnet, TFTP was designed before most people were concerned about “bad actors” on the network. Consequently, it fit a useful niche: file transfer without the burden of authentication. The features of TFTP are extremely limited; one of the design goals was to keep the service very small so that it could run on systems with extremely limited storage space and memory. It runs over UDP on port 69.^{36,37}

Despite the lack of security, TFTP remains in widespread use today (generally restricted to internal networks). It has been incorporated into many network devices, from Voice over IP phones to firewalls to desktop BIOSs. TFTP is often used as a means through which distributed devices can download updates from a central server within an organization. On many routers and switches, it is used to back up and restore files. It was also used for payload propagation in both the CodeRed and Nimda outbreaks.

Forensic analysts may need to use TFTP to export files from a router, switch, or other device that does not support SCP or SFTP for such operations.

3.3.1.7 Web and Proprietary Interfaces

These days most commercial network devices, from DSL routers to wireless access points, come with a web-based management interface. Through HTTP or HTTPS, you can access configuration menus, event logs, and other data that the device contains. Web interfaces are popular because they are very portable; they do not require the user to install a special client to access the device.

Typically, web interfaces are available by default as unencrypted HTTP sessions, in which case the login credentials and any data transferred over the connection is unencrypted and easily intercepted. Many vendors also offer SSL/TLS-encrypted web interfaces, although the certificates used with these services often have errors, which cause problems with validation.

Many vendors such as Cisco and netForensics have also developed Java-based cross-platform interfaces or other types of proprietary interfaces for their devices.

For forensic investigators, one of the biggest challenges with GUI interfaces is logging forensic activities. Text-based interfaces can typically be used with “script” or other tools that log commands; with GUIs, it can be harder for an investigator to automatically log activities. Often, the best fallback is screenshots and a good notebook.

3.3.2 Inspection Without Access

In many cases, it is desirable to gain information about a device’s configuration or state without accessing the device at all via an interface. There are also times when the password to user interfaces is not available. It is possible to gather extensive information about a

36. K. R. Sollins, “RFC 783—TFTP Protocol (revision 2),” *IETF*, June 1981, <http://rfc-editor.org/rfc/rfc783.txt>.

37. K. R. Sollins, “RFC 1350—TFTP Protocol (revision 2),” *IETF*, July 1992, <http://www.rfc-editor.org/rfc/rfc1350.txt>.

device's configuration and state through external inspection, using port scanning, vulnerability scanning, and other methods.

3.3.2.1 Port Scanning

Port scanning, using a tool such as nmap, is an effective way to retrieve information about open ports and software versions of a device. Note that port scanning is an active process, meaning that you will generate network traffic and, in the process, modify the state of the targeted device.

3.3.2.2 Vulnerability Scanning

Vulnerability scanning is the next level of active external inspection. In addition to port scanning, vulnerability scanners test target systems for a wide variety of known vulnerabilities. If you are concerned that your target of interest may be compromised, this can sometimes provide strong clues as to how the compromise may have occurred.

Vulnerability scanning generates network traffic and modifies the state of the targeted device. In some cases, it can even crash the targeted device. Be cautious and understand the options you have selected before running a vulnerability scanner against your target of interest.

3.3.3 Strategy

Here are some tips for collecting evidence from network devices. In general, our goal is to preserve the evidence while minimizing our footprints throughout the environment. Of course, your specific strategy will vary for each investigation.

- **Refrain from rebooting or powering down the device.** A lot of network-based evidence exists as volatile data in the memory of network devices. For example, ARP tables on routers change dynamically and typically would not survive a reboot. The current state of network connectivity is similarly volatile. Make sure that you capture all the volatile evidence you need before allowing the device to be rebooted. A reboot may also cause modifications to persistent logfiles stored on disk, particularly if local disk space is limited and log files are overwritten at designated intervals or when space is low.
- **Connect via the console rather than over the network.** It is usually preferable to connect via the system console whenever possible, rather than over the network. Connecting to a device over the network will necessarily generate network traffic and modify the state of the device's network connections. It may also alert an attacker on the network to your presence. Connecting instead directly to the system console will minimize your network footprint.
- **Record the system time.** Always check to see what the time skew is between the network device you are investigating and a reliable source. Even a small time skew can make it very difficult to correlate evidence, unless you are aware of it and can compensate. Unfortunately, most forensic tools do not have easy ways of adjusting specific logs for time skew, so you may need to compare skewed records manually or

using customized scripts. If you have access to a network device for an extended period of time, it is a good idea to collect the time at regular intervals because the time skew may change.

- **Collect evidence according to level of volatility.** This is a general rule of thumb for all digital forensic investigations: when all else is equal, collect the most volatile evidence first and work your way down to more persistent evidence. This will increase your chances of capturing all the evidence you are after. Of course, sometimes other factors may cause you to collect evidence in a different order—for example, if the most volatile evidence is very difficult to obtain or perhaps not as important. In many cases, you simply cannot collect all the evidence you want, and must pick evidence from one system over another.
- **Record your investigative activities.** When connecting via a command-line interface, you can often record your commands using utilities such as “screen” or “script”. These tools can help you clearly delineate what footprints were the result of your own activities and what were those of the existing system. Recording your own commands also helps you stay organized, and will help you become a more efficient investigator because you will be able to refer to your prior work in other investigations.

Many investigators are wary of recording their own commands because they are afraid of messing up and leaving a record. Don’t worry about making mistakes! All investigators fat-finger commands or have to look up proper command syntax from time to time. This is the nature of network forensics; we deal with so many different types of equipment that we must often refer to manuals or rely on local network staff for specific syntax. It’s much better to maintain a record of all your activities, fat-fingered commands and all, than to not maintain any record, or worse, to maintain an *inaccurate* record of your commands that does not match what you actually typed.

Graphical interfaces are challenging because it is more difficult to record your investigative activities. Whenever possible, take screen captures, photos or recordings of your graphical connections.

3.4 Conclusion

As an investigator, you will always leave footprints. The more places you look for evidence, the more footprints you will leave. Fortunately, there are ways that you can minimize your impact on the environment.

In this chapter, we discussed the concepts of passive and active evidence acquisition. We reviewed common methods for gaining access to network traffic and software used for capturing, filtering, and storing packet data. We also discussed interfaces used to actively gather evidence from network devices, ranging from the system console to proprietary graphical interfaces accessed via a remote client. Finally, we talked about strategies for reducing your footprint on the network, while still obtaining the evidence you seek.

Part II

Traffic Analysis

Chapter 4, “Packet Analysis,” is a comprehensive study of protocols, packets, and flows, and methods for dissecting them.

Chapter 5, “Statistical Flow Analysis,” presents the increasingly important field of statistical flow record collection, aggregation, and analysis.

Chapter 6, “Wireless: Network Forensics Unplugged,” discusses evidence collection and analysis of wireless networks, specifically focusing on the IEEE 802.11 protocol suite.

Chapter 7, “Network Intrusion Detection and Analysis,” is a review of network intrusion prevention and detection systems, which are specifically designed to produce security alerts and supporting evidence.

This page intentionally left blank

Chapter 4

Packet Analysis

Twas brillig, and the Protocols
Did USER-SERVER in the wabe.
All mimsey was the FTP,
And the RJE outgrabe,

Beware the ARPANET, my son;
The bits that byte,
the heads that scratch...

—R. Merryman, “ARPAWOCKY” (RFC 527)¹

Once you have captured network traffic, what do you do with it? Depending on the nature of the investigation, you might want to analyze the protocols in use, search for a specific string, or carve out files.

Perhaps you received an alert from an IDS about suspicious traffic from a particular host and you would like to identify the cause. Or perhaps you are concerned that an employee is exporting confidential data and you need to search outbound communications for specific keywords. Or perhaps you’ve discovered traffic on the network that you can’t recognize or interpret and you want to figure out the cause.

In all of these cases, it is useful to understand the fundamentals of protocol analysis, packet analysis, and multipacket stream analysis. During this chapter, we learn how to analyze fields within protocols, protocols within packets, packets within streams, and then reconstruct higher-layer protocol data from streams. Throughout this discussion we provide examples of common tools and techniques that investigators can use to accomplish specific goals.

Investigators face many challenges when conducting packet analysis. It is not always possible to recover all protocol information or contents from a packet. The packet data may be corrupted or truncated, the contents may be encrypted at different layers, or the protocols in use may be undocumented. More and more often, the sheer volume of traffic makes it difficult to find the useful packets in the first place. Fortunately, the tools available for packet analysis are becoming increasingly sophisticated. A well-trained forensic investigator should be familiar with a variety of tools and techniques so that you can select the right tool for the job.

1. R. Merryman, “ARPAWOCKY” (RFC 527), IETF, June 1973, <http://rfc-editor.org/rfc/rfc527.txt>.

4.1 Protocol Analysis

Protocol analysis refers to the art and science of understanding how a particular communications protocol works, what it's used for, how to identify it, and how to dissect it. This may not be as straightforward as you might expect. In an ideal world, all protocols would be neatly cataloged, publicized, and implemented according to specification. In reality, none of this is true. Many protocols are deliberately kept secret by their inventors, either to protect intellectual property, keep out competition, or for the purposes of security and covert communications. Other protocols are simply not well documented because no one has taken the time.

Some protocols are publicly documented, such as the IETF-specified standards (which we discuss in greater detail shortly). However, that does not mean that hardware and software vendors have chosen to properly implement them. Often, manufacturers implement protocols before standards have been formally ratified, or only partially implement them. Engineers and programmers often make mistakes that result in behavior which is not compliant with standards.

Regardless of whether protocol specifications are formally published, you never know what you'll find actually traversing the networks. It is very rare for software developers or equipment manufacturers to completely adhere to every aspect of a published standard (even their own!). This fact is routinely exploited by attackers to bypass intrusion detection systems and firewall rules, smuggle data in strange places, and generally create mayhem.

The bottom line is that protocol analysis is a challenging art.

Definition: Protocol Analysis

Protocol analysis—Examination of one or more fields within a protocol's data structure. Protocol analysis is commonly conducted for the purposes of research (i.e., as in the case of an unpublished protocol specification) or network investigation.

4.1.1 Where to Get Information on Protocols

When you are searching for documentation about a particular protocol, there are many possible places to look. Perhaps the most well known is the IETF's large, public repository of documented protocols. Other standards bodies, vendors, and researchers also maintain public and private caches of protocol documentation.

4.1.1.1 IETF Request for Comments (RFC)

In 1969, with the emergence of the ARPANET, a small group of researchers began distributing “requests for comments” (RFCs), which were initially defined as “any thought, suggestion, etc. related to the HOST software or other aspect of the [ARPANET] network.” Each RFC was numbered and sent to network engineers at different organizations involved

in the creation of the ARPANET. The original distributors wrote that “we hope to promote the exchange and discussion of considerably less than authoritative ideas.”²

Over time, RFCs have emerged as a way to develop, communicate, and define international standards for internetworking. They are developed and distributed by the Internet Engineering Task Force (IETF), a “loosely self-organized group of people who contribute to the engineering and evolution of Internet technologies … the principal body engaged in the development of new Internet standard specifications.”³

A subset of RFCs are approved by the IETF as Internet Standards, labeled the “STD” subseries. Internet Standards RFCs are heavily vetted and tested by the community and subject to stricter documentation requirements than other types of RFCs. RFCs which are “standards-track” documents, have different maturity levels: “Proposed Standard,” “Draft Standard,” and “Internet Standard,” the latter of which is given an STD series number upon publication (Internet Standards also retain their RFC numbers).⁴

The IETF specifies in RFC 4677 that “The IETF makes standards that are often adopted by Internet users, but it does not control, or even patrol, the Internet.” This has important implications for network forensic investigators; although the IETF may publish an Internet Standard, *no one monitors network traffic or inspects networking software to ensure that standards are actually being followed*. Instead, what exists on networks is what individual organizations, companies, users, and attackers create.

Founded in 1992, the Internet Society (ISOC) now sponsors the IETF, and chartered both the Internet Architecture Board (IAB) and the Internet Assigned Numbers Authority (IANA).

The canonical repository of RFCs on the Internet is at “<http://www.rfc-editor.org>” (per RFC 4677).⁵

Rough Consensus and Running Code

Steve Crocker, author of RFC 1, said: “Instead of authority-based decision-making, we relied on a process we called ‘rough consensus and running code.’⁶ Everyone was welcome to propose ideas, and if enough people liked it and used it, the design became a standard.”

“After all, everyone understood there was a practical value in choosing to do the same task in the same way.”⁷

2. S. Crocker, “RFC 10—Documentation Conventions,” *IETF*, July 29, 1969, [http://www.rfc-editor.org/rfc10.txt](http://www.rfc-editor.org/rfc/rfc10.txt).

3. P. Hoffman and S. Harris, “RFC 4677—The Tao of IETF: A Novice’s Guide to the Internet Engineering Task Force,” *IETF*, September 2006, <http://www.ietf.org/rfc/rfc4677.txt>.

4. S. Bradner, “RFC 2026—The Internet Standards Process—Revision 3,” *IETF*, October 1996, <http://rfc-editor.org/rfc/rfc2026.txt>.

5. P. Hoffman and S. Harris, “RFC 4677—The Tao of IETF: A Novice’s Guide to the Internet Engineering Task Force,” <http://www.rfc-editor.org/rfc/rfc4677.txt>

6. *Ibid.*

7. Stephen D. Crocker, “Op-Ed Contributor—How the Internet Got Its Rules,” *New York Times*, April 6, 2009, http://www.nytimes.com/2009/04/07/opinion/07crocker.html?_r=2&em.

4.1.1.2 Other Standards Bodies

The IETF isn't the only standards body, of course. There are other standards organizations that publish communications protocols for networking equipment and software. These include:

- **Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA)** The IEEE Standards Association develops standards for a broad range of electrical engineering topics, ranging from consumer electronics to wired and wireless networking. Some of the most well-known IEEE specifications include those developed by the IEEE LAN/MAN Standards Committee, such as the “802.11” suite of protocols for wireless networking⁸ and 802.3 (Ethernet).⁹
- **International Organization for Standardization (ISO)** The ISO is an international nonprofit organization comprised of representatives from national standards institutes from 163 countries. They publish standards for a variety of industries, including information technology and communications.¹⁰

4.1.1.3 Vendors

Many vendors develop their own proprietary protocols for equipment, software, and communications. Some vendors work with standards bodies to instigate widespread adoption of their standards and help ensure compatibility. For example, Cisco employees worked as part of the IETF to develop RFC 2784, “Generic Routing Encapsulation,”¹¹ which describes an open trunking protocol. Microsoft publishes an online library of their technical specifications, which include communications protocols used by Windows servers and clients.¹²

Other times, vendors work hard to keep their protocols secret, usually to reduce competition. America Online, for example, refused to publish their proprietary instant messaging protocol (Open System for CommunicAtion in Realtime, or OSCAR) for many years, and worked hard to prevent competitors from developing AIM-compatible chat clients. Over time, many vendors and individual engineers reverse-engineered the protocols. Eventually, AOL published information about some aspects of the OSCAR protocol.

4.1.1.4 Researchers

Researchers at many universities, private institutions, and their parents' garages, have analyzed networking protocols and traffic. Often, this is undertaken in an attempt to

8. IEEE, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements,” *IEEE Standards Association* (June 12, 2007), <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>.

9. “IEEE-SA -IEEE Get 802 Program,” *IEEE Standards Association*, 2010, <http://standards.ieee.org/getieee802/802.3.html>.

10. “ISO,” 2011, http://www.iso.org/iso/iso_catalogue.htm.

11. D. Farinacci et al., “RFC 2784—Generic Routing Encapsulation (GRE),” *IETF*, March 2000, <http://www.rfc-editor.org/rfc/rfc2784.txt>.

12. Microsoft Corporation, “Windows Communication Protocols (MCPP),” *MSDN*, 2011, <http://msdn.microsoft.com/en-us/library/cc216513%28v=prot.10%29.aspx>.

reverse-engineer a secret protocol. For example, Russian researcher Alexandr Shutko has published his “Unofficial” OSCAR (ICQ v7/v8/v9) protocol documentation. He conducted the research to “have some fun.”¹³

You can conduct your own protocol research. In many cases, all you need is a laptop, some off-the-shelf networking equipment, and free software tools. To build your own protocol analysis lab, first set up a small network. An older-model, inexpensive hub works very well for sniffing traffic. Then set up endpoints that use the protocol you are interested in analyzing and a system for intercepting the traffic.

In some cases it can be hard to get vendor-specific equipment due to budget restrictions. Depending on your target protocol, you may not be able to replicate the protocol in your lab. However, there are many widely used protocols that are easily accessible and fun to tear apart.

4.1.2 Protocol Analysis Tools

Rather than reinvent the wheel, it’s a good idea to familiarize yourself with tools and languages specifically designed for protocol analysis. Tools such as Wireshark and tshark include built-in protocol dissectors for hundreds of different protocols, using the NetBee PDML and PSML languages as a foundation. These can save you a lot of time and effort.

4.1.2.1 Packet Details Markup Language and Packet Summary Markup Language

The Packet Details Markup Language (PDML) defines a standard for expressing packet details for Layers 2–7 in an XML format. The syntax is essentially a compromise in “readability” between computer and human parsers. Computer software must be programmed to interpret the markup language; humans can learn to read it, or employ other tools to make use of it.¹⁴ The Packet Summary Markup Language (PSML) is a similar XML language for expressing the most important details about a protocol.

PDML and PSML are both part of the NetBee library, which is designed to support packet processing.¹⁵ PDML and PSML were created and remain copyrighted by the Net-Group at the Politecnico di Torino in Italy, where WinPcap was also first developed. These specifications are used by Wireshark and tshark as a foundation for protocol dissection and display.

4.1.2.2 Wireshark

Wireshark is an excellent tool for protocol analysis. It includes built-in protocol dissectors that automatically interpret and display protocol details within individual packets, and allows you to filter on specific fields within supported protocols. You can also write your own packet dissectors for inclusion into the main Wireshark program or to be distributed as plugins.

13. “OSCAR (ICQ v7/v8/v9) Protocol Documentation,” July 2, 2005, <http://iserverd.khstu.ru/oscar/>.

14. “[NetBee] Pdml Specification,” *NetBee*, http://www.nbee.org/doku.php?id=netpdःpdml_specification.

15. “The NetBee Library [NetBee],” *NetBee*, August 13, 2010, <http://www.nbee.org/doku.php>.

By default, Wireshark displays packets in three panels:

- **Packet List** This panel shows packets that have been captured, one per line, with very brief details about them. This typically includes the time the packet was captured, the source and destination IP address, the highest-level protocol in use (according to Wireshark's heuristics for analyzing protocols), and a brief snippet of protocol data.
- **Packet Details** For the packet highlighted in the Packet List View, this shows the details of the protocols in all layers that Wireshark can interpret.
- **Packet Bytes** This shows the hexadecimal and ASCII representation of the packet, including Layer 2 data.

As you can see in Figure 4–1, Wireshark automatically decodes protocols at various layers in the frame and displays the details in the Packet Details window, ordered by layer.

No.	Time	Source	Destination	Protocol	Info
22	13.674786	Vmware_69:e6:2b	Vmware_b0:8d:62	ARP	192.168.1
23	18.870898	192.168.1.158	64.12.24.50	SSL	Continuat
24	18.871477	64.12.24.50	192.168.1.158	TCP	https > 5
25	33.914966	192.168.1.158	64.12.24.50	SSL	Continuat
26	33.915486	64.12.24.50	192.168.1.158	TCP	https > 5
27	34.006599	192.168.1.158	64.12.24.50	SSL	Continuat
28	34.006601	64.12.24.50	192.168.1.158	TCP	..
▶ Frame 24 (60 bytes on wire, 60 bytes captured)					
▶ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: HewlettP_45:a4:bb					
▼ Internet Protocol, Src: 64.12.24.50 (64.12.24.50), Dst: 192.168.1.158 (192.168.1.158)					
Version: 4					
Header length: 20 bytes					
▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)					
Total Length: 40					
Identification: 0xb43d (46141)					
▼ Flags: 0x00					
0.. = Reserved bit: Not Set					
.0. = Don't fragment: Not Set					
..0 = More fragments: Not Set					
Fragment offset: 0					
Time to live: 127					
Protocol: TCP (0x06)					
▶ Header checksum: 0x6d0e [correct]					
Source: 64.12.24.50 (64.12.24.50)					
Destination: 192.168.1.158 (192.168.1.158)					
▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 51128 (51128)					
0000 00 12 79 45 a4 bb 00 0c 29 b0 8d 62 08 00 45 00 ..yE....)..b..E.					
0010 00 28 b4 3d 00 00 7f 06 6d 0e 40 0c 18 32 c0 a8 .(.=.... m.@..2..					
0020 01 9e 01 bb c7 b8 07 e9 60 db 33 6b d2 c9 50 10 `..3k..P.					
0030 fa f0 61 f2 00 00 00 00 00 00 00 00 00 00 00 ..a.....					

Figure 4–1. Screenshot of the main Wireshark panels. The top panel is “Packet List,” the middle panel is “Packet Details,” and the bottom panel is “Packet Bytes.”

In this example, you can see that within Frame 24, Wireshark has identified “Ethernet II,” “Internet Protocol,” and “Transmission Control Protocol.” It shows a summary of important information for each protocol (such as “Src Port” and “Dest Port” for TCP), and also allows you to drill down to see extensive details for each protocol. In the Packet List window, Wireshark automatically displays the name of the highest-layer protocol found in the packet (in this case, TCP).

Wireshark includes hundreds of built-in protocol dissectors.¹⁶ You can view a list of protocol dissectors that are enabled in your version of Wireshark by going to Analyze→Enabled Protocols. You can also modify dissector settings for specific protocols in Edit→Preferences→Protocols. If you are analyzing a protocol that Wireshark doesn’t currently support, you can also write your own dissector. Wireshark plugins are commonly written in C. You can also write Wireshark plugins in Lua, but for performance reasons, Wireshark developers recommend using Lua only for prototyping dissectors.¹⁷

Wireshark automatically decides which protocol dissectors to use for a specific packet. If you have packets that you would like to dissect using a different protocol dissector, you can use Wireshark’s “Analyze→Decode As” function to modify the dissector in use.

4.1.2.3 tshark

Tshark uses Wireshark’s protocol dissection code, and therefore includes much of the same functionality, with a command-line interface.¹⁸ In tshark, the default output displays the information produced by PSML. When the “-V” flag is used, tshark displays the PDML information.

Here are some examples of using tshark for protocol analysis:

- Basic usage for reading from a capture file:

```
$ tshark -r capturefile.pcap
```

- Disable network object name resolution using the -n flag, so you can see actual IP addresses and port numbers. It’s often a good idea to disable network object name resolution because this can slow down display.

```
$ tshark -n -r capturefile.pcap
```

- Select an output format using the T flag. Options include pdml, psml, ps, text, and fields. The default is “text.” In the example below, we print the output in PDML, which provides verbose packet protocol details in XML format.

```
$ tshark -r capturefile.pcap -T pdml
```

16. “ProtocolReference—The Wireshark Wiki,” *Wireshark*, March 7, 2011, <http://wiki.wireshark.org/ProtocolReference>.

17. “Lua—The Wireshark Wiki,” *Wireshark*, May 13, 2011, <http://wiki.wireshark.org/Lua>.

18. “tshark—The Wireshark Network Analyzer 1.5.0,” *Wireshark*, <http://www.wireshark.org/docs/man-pages/tshark.html>.

- To print a specific field (as defined by Wireshark’s protocol dissectors), use the `-e` flag combined with the “`-T` fields” option. The example below is from the “`tshark`” man page, and it prints the Ethernet frame number, IP address, and information about the UDP protocol.

```
$ tshark -r capturefile.pcap -T fields -e frame.number -e ip.addr -e udp
```

- Tshark also includes an option, `-d`, which implements Wireshark’s “Decode As” functionality. This option allows you to manually configure tshark to interpret the indicated traffic as a specific protocol. For example, if you want to configure tshark to treat all traffic in a packet capture on TCP port 29008 as HTTP, you would use the command:

```
$ tshark -r capturefile.pcap -d tcp.port==29008,http
```

- Tshark can be given either Wireshark’s display filters, BPF-filters, or both. Here is a capture file filtered using display filters:

```
$ tshark -r capturefile.pcap -R 'ip.addr == 192.168.1.1'
```

Here is the equivalent command, filtered using BPF:

```
$ tshark -r capturefile 'host 192.168.1.1'
```

4.1.3 Protocol Analysis Techniques

Recall that protocol analysis is the examination of one or more fields within a protocol’s data structure. Protocol analysis is often necessary for packet analysis because investigators must be able to properly interpret the communications structures in order to understand the contents and analyze packets or streams.

Fortunately, much of this analysis has already been done by a worldwide community of developers and analysts, who have created freely available tools such as Wireshark, tshark, tcpdump, and the specification languages upon which they are based. When using these tools, keep in mind that you are standing on the shoulders of giants (and that, occasionally, giants make mistakes).

Network investigators must also be prepared to handle protocols that have not yet been publicly dissected and included in common tools. Furthermore, hackers may sometimes develop new protocols, or extensions to old ones, in order to communicate covertly or add functionality that the original protocol authors never intended.

Protocol analysis is a deep art and we only scratch the surface here. During the next few sections, we describe the fundamentals of protocol identification, decoding, data exportation, and metadata extraction.

4.1.3.1 Protocol Identification

How do you identify which protocols are in use in a packet capture? Here are some common ways that you can identify a protocol:

Scenario: Ann's Bad AIM

Throughout this chapter, we will use the following scenario to illustrate tools and techniques. This scenario is “Puzzle #1: Ann’s Bad AIM,” from the “Network Forensics Puzzle Contest” web site.¹⁹ You can download the original contest materials and the corresponding packet capture from <http://ForensicsContest.com>.

The Case: *Anarchy-R-Us, Inc. suspects that one of their employees, Ann Dercover, is really a secret agent working for their competitor. Ann has access to the company’s prize asset, the secret recipe. Security staff are worried that Ann may try to leak the company’s secret recipe.*

Security staff have been monitoring Ann’s activity for some time, but haven’t found anything suspicious until now. Today an unexpected laptop briefly appeared on the company wireless network. Staff hypothesize it may have been someone in the parking lot because no strangers were seen in the building. Ann’s computer (192.168.1.158) sent IMs over the wireless network to this computer. The rogue laptop disappeared shortly thereafter.

“We have a packet capture of the activity,” said security staff, “but we can’t figure out what’s going on. Can you help?”

- Search for common binary/hexadecimal/ASCII values that are typically associated with a specific protocol
- Leverage information in the encapsulating protocol
- Leverage the TCP/UDP port number, many of which are associated with standard default services
- Analyze the function of the source or destination server (specified by IP address or hostname)
- Test for the presence of recognizable protocol structures

Let’s explore these protocol identification techniques one at a time. Getting back to our scenario, Ann’s Bad AIM, let’s take another look at the packet capture and figure out what protocols are in use. Throughout this discussion and the remainder of this book, remember to count byte offsets starting from 0.

Search for common binary/hexadecimal/ASCII values that are typically associated with a specific protocol Most protocols contain sequences of bits that are commonly, if not always, present in packets associated with that protocol, in predictable places. An excellent example is the hexadecimal sequence “0x4500,” which often marks the beginning of an IPv4

19. Sherri Davidoff, Jonathan Ham, and Eric Fulton, “Network Forensics Puzzle Contest—Puzzle #1: Ann’s Bad AIM,” September 25, 2009, <http://forensicscontest.com/2009/09/25/puzzle-1-annds-bad-aim>.

packet. The following text shows a section of the Ann’s Bad AIM packet capture, produced using tcpdump:

```
$ tcpdump -nn -AX -r evidence01.pcap
22:57:22.022972 IP 64.12.24.50.443 > 192.168.1.158.51128: Flags [.], ack 6,
  win 64240, length 0
  0x0000: 4500 0028 b43d 0000 7f06 6d0e 400c 1832 E..(.=....m.@..2
  0x0010: c0a8 019e 01bb c7b8 07e9 60db 336b d2c9 .....`3k..
  0x0020: 5010faf0 61f2 0000 0000 0000 0000 P....a.....
```

Why does “0x4500” commonly appear at the beginning of IP packets? The high-order nibble of the byte at offset zero of an IP packet represents the IP version. There are only two versions of the IP protocol in common use today: IP version 4 (IPv4), and version 6 (IPv6). IPv4 is still the most widely implemented IP protocol, and so it is common to see the value “4” as the high-order nibble of Byte 0.

The low-order nibble of Byte 0 specifies the number of 32-bit words in the IPv4 header. IPv4 specifies that there are 20 bytes of required fields in the IPv4 header. It is possible to include optional header fields, which would cause the header length to increase. However, there are few legitimate purposes for IP options in normal use, and so few operating systems set them, and few firewalls allow them. Consequently, the low-order nibble of Byte 0 in an IPv4 packet is typically “5” (remember, there are 4 bytes in one 32-bit word, so 5 words is equal to 20 bytes).

Byte 1 of the IPv4 header is really a set of multibit fields, commonly grouped together and referred to as the “Type of Service” field. It is not widely used today, and so most IPv4 packets carry all zeros at Byte 1.²⁰

As a result of these factors, IPv4 packets commonly begin with the value “0x4500.” Experienced network forensic investigators can view raw tcpdump output and manually pick out the beginnings of IPv4 packets without the assistance of automated tools.

Leverage information in the encapsulating protocol Protocols often contain information that indicates the type of encapsulated protocol. Recall that in the OSI model, lower-layer protocol fields typically indicate the higher-layer protocol that may be encapsulated, in order to facilitate proper processing.

Figure 4–2 is a screenshot of the same Ann’s Bad AIM packet displayed within Wireshark. Notice that Wireshark has indeed identified the Layer 3 protocol as IP version 4, just as we guessed in the previous section. Byte 9 of the IP header (highlighted) indicates the protocol encapsulated within the IP packet. In this case, the value at Byte 9 is 0x06, which corresponds with TCP. (The protocol numbers used in the IPv4 “Protocol” field and the IPv6 “Next Header” field are assigned and maintained by IANA.²¹) Based on this information, it is reasonable to assume that the protocol encapsulated within this IP packet is TCP.

Leverage the TCP/UDP port number, many of which are associated with standard default services A simple and common way to identify protocols is by examining the TCP or UDP port number in use. There are 65,535 possible port numbers for each of TCP and

20. Richard W. Stevens, “The Protocols,” in *TCP/IP Illustrated*, vol. 1, Addison-Wesley, 1994.

21. “Protocol Numbers,” May 3, 2011, <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml>.

No.	Time	Source	Destination	Protocol	Info
23	18.070090	192.168.1.150	64.12.24.50	SSL	Continuation Data
24	18.071477	64.12.24.50	192.168.1.158	TCP	https > 51128 [A]
25	33.914966	192.168.1.158	64.12.24.50	SSL	Continuation Data

▶ Frame 24 (60 bytes on wire, 60 bytes captured)

▶ Ethernet II, Src: VMware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: HewlettP_45:a4:bb (00:12:00:45:a4:bb)

▼ Internet Protocol, Src: 64.12.24.50 (64.12.24.50), Dst: 192.168.1.158 (192.168.1.158)

- Version: 4
- Header length: 20 bytes
- ▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
- Total Length: 40
- Identification: 0xb43d (46141)
- ▼ Flags: 0x00
 - 0.. = Reserved bit: Not Set
 - .0. = Don't fragment: Not Set
 - ..0 = More fragments: Not Set
- Fragment offset: 0
- Time to live: 127
- Protocol: TCP (0x06)
- ▶ Header checksum: 0x6d0e [correct]
- Source: 64.12.24.50 (64.12.24.50)
- Destination: 192.168.1.158 (192.168.1.158)
- ▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 51128 (51128), Seq: 1

```
0000 00 12 79 45 a4 bb 00 0c 29 b0 8d 62 08 00 45 00 ..yE.... )..b..E.
0010 00 28 b4 3d 00 00 7f 06 6d 0e 40 0c 18 32 c0 a8 .(=... m.@..2..
0020 01 9e 01 bb c7 b8 07 e9 60 db 33 6b d2 c9 50 10 ..... `..3k..P.
0030 fa fo 61 f2 00 00 00 00 00 00 00 00 00 00 00 ..a..... ....
```

Figure 4–2. IP protocol details displayed within Wireshark. Notice that the IP packet contains information about the encapsulated protocol (in this case, 0x06, or TCP).

UDP. IANA has published a list of TCP/UDP port numbers that correspond with specific higher-layer network services. You can view the canonical list on IANA’s web site,²² and much of the same information is also stored in the /etc/services file on most UNIX/Linux systems.

In Figure 4–3, we can see that Frame 8 of the Ann’s Bad AIM packet capture transmits data over UDP 123. According to IANA, UDP port 123 is assigned to the Network Time Protocol (NTP). This is a service that synchronizes time between systems. Wireshark automatically displays the default service associated with a particular port, if one is assigned, as you can see in the highlighted line of the Packet List pane.

If you look further down in the Packet Details frame, you can also see that Wireshark identified various fields within the higher-layer NTP protocol, such as “Peer Clock Stratum” and “Originate Time Stamp.” This provides corresponding evidence that supports the identification of this protocol as NTP.

Identifying protocols by port number is not always reliable because servers can easily be configured to use nonstandard port numbers for specific services. Take a look at

22. “Port Numbers,” July 6, 2011, <http://www.iana.org/assignments/port-numbers>.

No.	Time	Source	Destination	Protocol	Info
5	0.910234	Vmware_b0:8d:62	Dell_4d:4f:ae	ARP	who has 192.168.1.159? tell 192.168.1.10
6	0.918240	Dell_4d:4f:ae	Vmware_b0:8d:62	ARP	192.168.1.159 is at 00:21:70:4d:4f:ae
7	3.185626	192.168.1.30	192.168.1.10	NTP	NTP client
8	3.186114	192.168.1.10	192.168.1.30	NTP	NTP server
9	4.680216	192.168.1.10	192.168.1.255	NTP	NTP broadcast
10	8.181469	Vmware_b0:e6:2b	Vmware_b0:8d:62	ARP	who has 192.168.1.10? tell 192.168.1.30
				
▶ Frame B (96 bytes on wire, 96 bytes captured)					
▶ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: Vmware_b9:e6:2b (00:0c:29:69:e6:2b)					
▶ Internet Protocol, Src: 192.168.1.10 (192.168.1.10), Dst: 192.168.1.30 (192.168.1.30)					
▶ User Datagram Protocol, Src Port: ntp (123), Dst Port: ntp (123)					
Source port: ntp (123)					
Destination port: ntp (123)					
Length: 56					
▶ Checksum: 0x7146 [validation disabled]					
▶ Network Time Protocol					
Flags: 0x24					
Peer Clock Stratum: secondary reference (3)					
Peer Polling Interval: 7 (128 sec)					
Peer Clock Precision: 0.000001 sec					
Root Delay: 0.0572 sec					
Root Dispersion: 0.0472 sec					
Reference Clock ID: 209.40.204.229					
Reference Clock Update Time: Aug 13, 2009 05:56:42.8583 UTC					
Originate Time Stamp: Aug 13, 2009 05:57:06.3368 UTC					
Receive Time Stamp: Aug 13, 2009 05:57:06.3424 UTC					
Transmit Time Stamp: Aug 13, 2009 05:57:06.3427 UTC					

Figure 4–3. UDP Protocol Details displayed within Wireshark. Notice that Wireshark automatically associates the UDP port, 123, with its IANA-assigned default service, NTP.

Figure 4–4, in which we see a TCP segment (Frame 167 of the Ann’s Bad AIM packet capture) transmitted to destination port 443. According to IANA, TCP port 443 is assigned to the “http protocol over TLS/SSL,” or “https” service. Accordingly, Wireshark has presented and interpreted it as “https” or “Secure Socket Layer.”

However, look carefully at the Packet Bytes window in Figure 4–4. Notice that the payload of this packet is clearly not encrypted! You can see cleartext ASCII (“thanks dude”) as well as HTML tags. Looking back up in the Packet Details window, notice that although Wireshark has identified the highest-layer protocol as “Secure Socket Layer,” there are no protocol details listed in this section. Wireshark identified the protocol in use based solely on the TCP port number (443), but the protocol is NOT actually SSL/TLS. As a result, Wireshark was unable to decode the protocol and display details under the “Secure Socket Layer” heading.

Analyze the function of the source or destination server (specified by IP address or hostname) Often, server hostnames and domains provide clues as to their functions, which can in turn help investigators identify likely protocols in use.

So far we’ve determined that the NTP protocol is likely present within the Ann’s Bad AIM packet capture. We’ve also found that there is unidentified traffic traversing TCP port 443. Although Wireshark identified this traffic as SSL/TLS, we found that this was incorrect. Perhaps the IP addresses or hostnames in use will provide us with clues regarding the highest-layer protocol....

Looking at Figure 4–4, the source IP address in frame 167 is “64.12.24.50.” If you do a WHOIS lookup on this address, you will see that it is owned by “America Online Inc.” (at the time of this writing). Here is a snippet of the WHOIS registration information:

No.	Time	Source	Destination	Protocol	Info
166	69.578661	64.12.25.91	192.168.1.159	TLSv1	Application Data
167	69.578667	64.12.24.50	192.168.1.158	SSL	Continuation Data
168	69.579120	192.168.1.158	64.12.24.50	TCP	51128 > https [ACK]
169	69.682022	64.12.25.91	192.168.1.159	TLSv1	[TCP Retransmission]
170	69.682779	192.168.1.159	64.12.25.91	TCP	sweetware-apps > h...
171	72.200575	Vmware_b0:8d:62	Dell_4d:a4:bb	HTTP	http://192.168.1...
▶ Frame 167 (280 bytes on wire, 280 bytes captured)					
▶ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: HewlettP_45:a4:bb (00:12:79:45:a4:bb)					
▶ Internet Protocol, Src: 64.12.24.50 (64.12.24.50), Dst: 192.168.1.158 (192.168.1.158)					
▼ Transmission Control Protocol, Src Port: https (443), Dst Port: 51128 (51128), Seq: 132					
Source port: https (443)					
Destination port: 51128 (51128)					
[Stream index: 2]					
Sequence number: 132735858					
[Next sequence number: 132736084]					
Acknowledgement number: 862704686					
Header length: 20 bytes					
Flags: 0x18 (PSH, ACK)					
Window size: 64240					
Checksum: 0x2b5a [validation disabled]					
▶ [SEQ/ACK analysis]					
Secure Socket Layer					
.....					
0090	4a 83 a4 85 00 02 00 61 05 01 00 01 01 01 01 00				J.....a
00a0	00 00 00 00 3c 48 54 4d 4c 3e 3c 42 4f 44 59				X.....<HT ML><BODY
00b0	3e 3c 46 4f 4e 54 20 46 41 43 45 3d 22 41 72 69				><FONT F ACE="Ari
00c0	61 6c 22 20 53 49 5a 45 3d 32 20 43 4f 4c 4f 52				al" SIZE =2 COLOR
00d0	3d 23 30 30 30 30 30 3e 74 68 61 6e 6b 73 20				=#000000 >links
00e0	64 75 64 65 3c 2f 46 4f 4e 54 3e 3c 2f 42 4f 44				dude</FO NT></BOD
00f0	59 3e 3c 2f 48 54 4d 4c 3e 00 0d 00 12 00 01 00				Y></HTML >.....
0100	05 2b 00 00 31 6e 00 81 00 05 2b 00 00 14 4f 00				.+.ln.. ..+.0.
0110	0b 00 00 00 13 00 01 03			

Figure 4–4. TCP packet details displayed within Wireshark. Notice that Wireshark automatically associates TCP port 443 with its IANA-assigned default service, HTTPS. However, this interpretation is INCORRECT (as evidenced by the fact that the packet contents are not encrypted, and no protocol details are displayed under the heading “Secure Socket Layer”).

```
$ whois 64.12.24.50
...
NetRange:      64.12.0.0 - 64.12.255.255
CIDR:         64.12.0.0/16
...
OrgName:       America Online, Inc.
OrgId:        AMERIC-158
Address:       10600 Infantry Ridge Road
City:          Manassas
StateProv:     VA
PostalCode:    20109
Country:       US
RegDate:      1999-12-13
Updated:       1999-12-16
Ref:          http://whois.arin.net/rest/org/AMERIC-158
```

It would be reasonable for a network forensic investigator to hypothesize that the traffic associated with this server could be traffic commonly used to support AOL's services, such as HTTP or AIM.

Interlude: AOL Instant Messenger Protocols

The AOL Instant Messenger (AIM) service was created by America Online in 1997, and remains one of the most popular and portable instant messaging services today. The earliest versions were designed for use by AOL employees, before AOL began providing the service to the general public. AIM software is very portable and can be used on Linux, Windows, or Apple systems.

AIM is based on the proprietary, closed-source Open System for Communication in Realtime (OSCAR) protocol.²³ OSCAR is a proprietary protocol developed by AOL and the company has worked to prevent others from building compatible clients. Even so, many have tried to reverse-engineer the protocol, with some success. Published documentation is very limited, but tools such as Wireshark include protocol dissectors, which are helpful as a guide for forensic analysts. AOL has licensed the OSCAR protocol for use in Apple's iChat software as well as AIM.²⁴

To transfer a file, the sender and receiver initially communicate using Inter Client Basic Messages (ICBM) through a third-party messaging server. They use ICBM messages on channel 2 to negotiate the IP address, port, and other details of a direct file transfer. Once the direct connection is set up between sender and recipient on the specified port, OSCAR File Transfer (OFT) is used to transfer the file.

There is a good description of the process in the comments of the source code for Pidgin, a multiplatform chat client (`pidgin-2.5.8/libpurple/protocols/oscar/oft.c`).²⁵

The OFT protocol is relatively simple. To transfer a file between two clients, the sender sends an OFT “prompt” header, with the “Type” set to “0x0101” to indicate it is ready to begin sending. The recipient sends an OFT “acknowledge” header back, with the “Type” set to “0x0202” to indicate that it is ready to receive data. Next, the sender sends the raw data to the recipient. Finally, the recipient responds with an OFT “done” packet, with the “Type” set to “0x0204” to indicate that it has finished receiving data. Figure 4–5 illustrates the OFT protocol in use.

Figure 4–6 is an example of an OFT2 header message. Note that, among other data, the header contains the protocol version (0x4F465432 is “OFT2” in ASCII), the “Type” (which indicates the purpose of the header), the transmitted file size, and the file name.²⁶

23. “AOL Instant Messenger—Wikipedia, the free encyclopedia,” http://en.wikipedia.org/wiki/AOL_Instant_Messenger.

24. “OSCAR Protocol—Wikipedia, the free encyclopedia,” http://en.wikipedia.org/wiki/OSCAR_protocol.

25. “File List,” *SourceArchive*, <http://pidgin.sourceforge.net/documentation/1:2.5.8-1ubuntu2/files.html>.

26. Jonathan Clark, “On Sending Files via OSCAR,” 2005, <http://www.cs.cmu.edu/~jhclark/aim/On%20Sending%20Files%20via%20OSCAR.odt>.

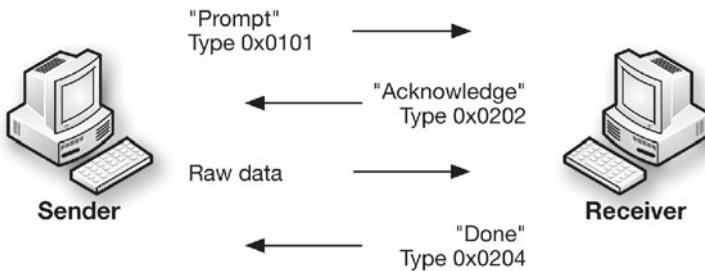


Figure 4–5. Illustration of the OFT protocol used to transfer files through AIM.

OFT2 Message Part Header																																				
Bytes	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																				
0x00	Protocol Version (4 bytes)		Length	Type	Cookie (8 bytes)																															
0x10	Encryption	Compression	Total Files	Files Left	Total Parts	Parts Left																														
0x20	Size		Modification Time		Checksum						RfcvCsum																									
0x30	RfSize		Creation Time		Rfcsum						nRecv																									
0x40	CecvCsum		Identification String (32 bytes)																																	
0x50	Identification String, cont.																																			
0x60	Identification String, cont.		Flags	nOffset sOffset	Dummy (69 bytes)																															
0x70	Dummy, cont.																																			
0x80	Dummy, cont.																																			
0x90	Filename (64 bytes)																																			
0xA0	Macintosh File Information																																			
0xB0	Macintosh File Information, cont. (16 bytes)															Encoding	Subcode																			
0xC0																																				
0xD0																																				
0xE0																																				
0xF0																																				

Figure 4–6. Illustration of the OFT2 header structure.

Test for the presence of recognizable protocol structures You can experimentally test for the presence of a specific protocol in a packet capture if you have some idea of the structure and common header values. For example, Figure 4–7 shows Frame 112 from the Ann’s Bad AIM packet capture. In this frame, we can see that there is traffic from source port 5190 (which Wireshark associates with “aol”), but this traffic is not decoded above Layer 4 (TCP). How can we identify the higher-layer protocol in use?

Given that the traffic is associated with a port used by AOL, and we have prior evidence of AOL-related traffic, we might hypothesize that this unidentified traffic could be a proprietary AOL protocol that does not have a dissector included in our Wireshark distribution, such as OSCAR traffic.

To test this hypothesis, let’s examine the TCP packet payload. As you can see in Wireshark, the payload contents begin with 0x4F465432, or “OFT2” in ASCII. This matches the information that independent researchers have published regarding the OSCAR protocol header, shown in Figure 4–6.

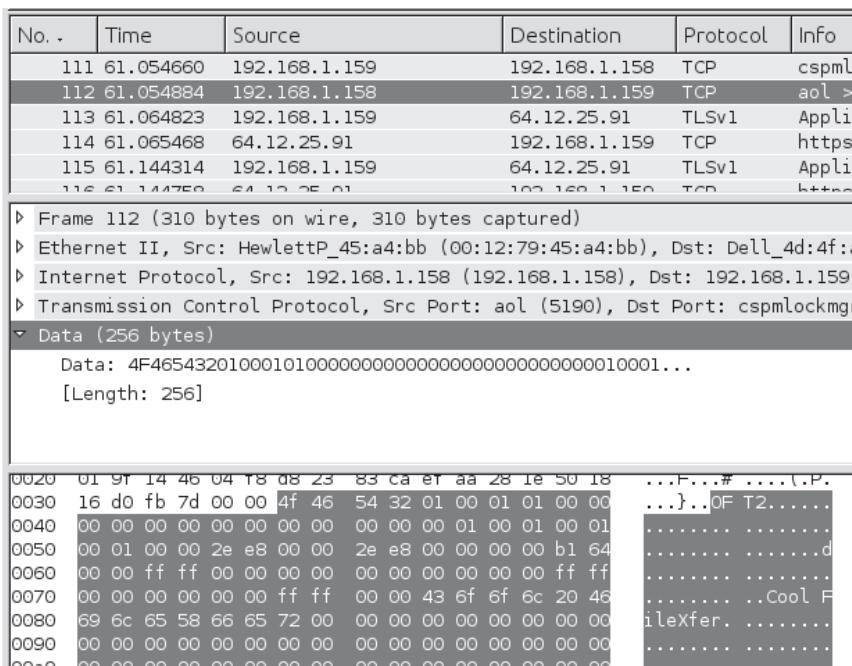


Figure 4–7. Frame #112 from “Ann’s Bad AIM.” Notice that Wireshark does not automatically identify a protocol higher than Layer 4 (TCP). However, analysis of the TCP payload shown in the Packet Bytes window reveals structures that correlate with the OSCAR File Transfer protocol.

According to publicly available documentation, Bytes 6 and 7 of the OFT2 header should indicate the “Type.” In Figure 4–7, we can see that Bytes 6 and 7 of our packet payload are “0x0101,” which do correspond with a valid Type value (“Prompt”). This indicates that the sender is ready to transmit a file.

4.1.3.2 Protocol Decoding

Protocol decoding is the technique of interpreting the data in a frame according to a specific, known structure, which allows you to correctly understand the meaning of each bit in the communication. Some bits in a protocol are used for describing the protocol itself or facilitating the communication. Other bits actually describe an encapsulated higher-layer protocol or its payload. In any case, understanding the purpose of specific bits within a frame and interpreting them according to a known protocol structure is a fundamental network forensics technique.

We rely very heavily on widely available tools for protocol decoding because few investigators have the time or resources to conduct protocol analysis in-depth, and in most cases it is not necessary to reinvent the wheel.

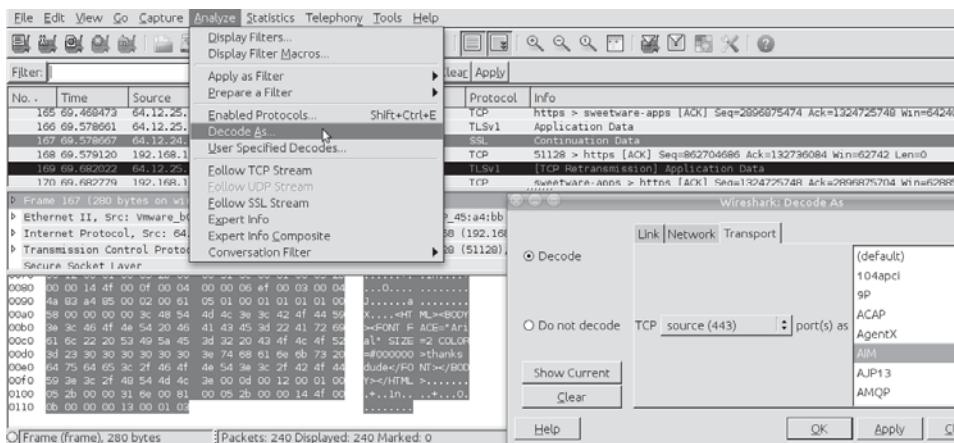


Figure 4–8. Frame #167 from “Ann’s Bad AIM.” Wireshark incorrectly decoded this packet as “Secure Socket Layer.” This screenshot illustrates how to use Wireshark’s “Decode As” function to interpret the data as AIM traffic instead.

To decode network traffic according to a specific protocol specification, you can:

- Leverage publicly available automated decoders and tools
- Refer to publicly available documentation and manually decode the traffic
- Write your own decoder

Let’s return to Frame #167 in our packet capture, which Wireshark mistakenly identified as “Secure Socket Layer” traffic. Based on the source IP address and port number, we have hypothesized that this packet may actually contain traffic relating to an AOL protocol.

Figure 4–8 illustrates how we can use Wireshark to decode this packet as “AIM” traffic. The results are shown in Figure 4–9. Notice that extensive details regarding the “AOL Instant Messenger” protocol are now visible, including Channel ID, Sequence Number, and ICBM Cookie. The presence of this information, and the fact that the values make sense given the context, indicates that we have correctly identified and decoded this as AIM traffic.

Earlier, we manually identified the encapsulated protocol in Packet #109 as OFT2 traffic. However, decoding OFT with Wireshark is significantly more difficult than decoding general AIM traffic because as of the time of this writing, Wireshark does not include a built-in OFT protocol decoder.

Fortunately, the Wireshark suite includes extensive support and documentation for writing new plugins, and many additional plugins have also been freely published online by independent developers. For example, to solve the Ann’s Bad AIM puzzle, independent developer Franck Guénichot wrote his own OFT protocol dissector in Lua.²⁷ (Note that in

27. Franck Guénichot, “Index of /contest02/Finalists/Franck_Guenichot,” December 17, 2009, http://forensicscontest.com/contest01/Finalists/Franck_Guenichot/.

No.	Time	Source	Destination	Protocol	Info
166	69.579661	64.12.25.91	192.168.1.159	TCP	https > sweetware-apps [TCP Retransmission]
167	69.579667	64.12.24.50	192.168.1.158	AIM Messaging	AIM Messaging, Incoming
168	69.579120	192.168.1.150	64.12.24.50	TCP	51128 > https [ACK] Seq=132735858, Ack=8220, Len=20
169	69.602022	64.12.25.91	192.168.1.159	TCP	[TCP Retransmission]
170	69.602779	192.168.1.159	64.12.25.91	TCP	sweetware-apps > https [TCP Retransmission]
171	70.000575	192.168.1.150	64.12.24.50	TCP
▶ Frame 167 (280 bytes on wire, 280 bytes captured)					
▶ Ethernet II, Src: Vmware_b0:8d:62 (00:0c:29:b0:8d:62), Dst: HewlettP_45:a4:bb (00:12:79:45:a4:bb)					
▶ Internet Protocol, Src: 64.12.24.50 (64.12.24.50), Dst: 192.168.1.158 (192.168.1.158)					
▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 51128 (51128), Seq: 132735858, Ack: 8220, Len: 20					
▼ AOL Instant Messenger					
Command Start: 0x2a					
Channel ID: SNAC Data (0x02)					
Sequence Number: 22234					
Data Field Length: 220					
▶ FNAC: Family: AIM Messaging (0x0004), Subtype: Incoming (0x0007)					
▼ AIM Messaging, Incoming					
ICBM Cookie: D41B4ECE87DF1F77					
Message Channel ID: 0x0001					
<pre>0060 00 01 00 02 00 10 00 05 00 04 4a 83 48 06 00 1d J.H... 0070 00 12 00 01 00 05 2b 00 00 31 6e 00 81 00 05 2b+..in....+ 0080 00 00 14 4f 00 00 00 04 00 00 06 ef 00 03 00 040..... 0090 4a 83 a4 85 00 02 00 61 05 01 00 01 01 01 00 J.....a 00a0 58 00 00 00 00 3c 48 54 4d 4c 3e 3c 42 4f 44 59 X....<HT ML><BODY 00b0 3e 3c 46 4f 4e 54 20 46 41 43 45 3d 22 41 /2 69 >Thanks 00e0 64 75 64 65 3c 2f 46 4f 4e 54 3e 3c 2f 42 4f 44 dude</FO NT></BODY 00f0 59 3e 3c 2f 40 54 4d 4c 3e 00 0d 00 12 00 01 00 Y></HTML 0100 05 2b 00 00 31 6c 00 81 00 05 2b 00 00 14 4f 00 ..+..ln.. .+...0. 0110 0b 00 00 00 13 00 01 03</pre>					

Figure 4–9. In this screenshot, Frame 167 has been decoded as AIM traffic. Notice that extensive details regarding the AIM protocol are now visible, indicating that our selection of AIM as the protocol was probably correct.

order to use an external Lua plugin, you first need to enable Lua support in Wireshark’s init.lua configuration file.) Using Franck’s OFT Lua protocol dissector, we can automatically print details of the OFT file transfer within the packet capture. Here is an example from Franck’s OFT dissector in action, used with tshark:

```
$ tshark -r evidence01.pcap -X lua_script:oft-tsk.lua -R "oft" -n -R frame.
number==112
112 61.054884 192.168.1.158 -> 192.168.1.159 OFT OFT2 Type: Prompt
```

As shown above, for frame #112 of Ann’s Bad AIM, tshark identified the higher-layer protocol as an OFT2 “Prompt” message, confirming our manual findings from earlier.

4.1.3.3 Exporting Fields

Once you have identified the protocol in use and determined your method for decoding it, the next step is to extract the values of specific fields of interest. It is trivial to do this graphically using Wireshark if you have the appropriate protocol dissector installed.

Let’s say that you would like to extract the AIM message sent between users in packet #167 of Ann’s Bad AIM. Once you configure Wireshark to decode packet #167 as AIM

No.	Time	Source	Destination	Protocol	Info
165	69.468473	64.12.25.91	192.168.1.159	TCP	https > sweetware-apps [
166	69.578661	64.12.25.91	192.168.1.159	TCP	https > sweetware-apps [
167	69.578667	64.12.24.50	192.168.1.158	AIM Messaging	AIM Messaging, Incoming
168	69.579120	192.168.1.158	64.12.24.50	TCP	51128 > https [ACK] Seq=
169	69.682022	64.12.25.91	192.168.1.159	TCP	[TCP Retransmission] htt
170	69.682770	192.168.1.159	64.12.25.91	TCP	weetware-apps > https [

Block length: 88
 Block Character set: 0x0000
 Block Character subset: 0x0000

Message: <HTML><BODY>thanks dude</BODY></HTML>

▼ TLV: Unknown

Value	Text	Description
0070 00 12 00 01 00 05 2b 00 00 31 6e 00 81 00 05 2b+..in....+	
0080 00 00 14 4f 00 0f 00 04 00 00 06 ef 00 03 00 040.....	
0090 4a 83 a4 85 00 02 00 61 05 01 00 01 01 01 00	J.....a.....	
00a0 58 00 00 00 3c 48 54 4d 4c 3e 3c 42 4f 44 59	X...<HT ML><BODY	
00b0 3e 3c 46 4f 4e 54 20 46 41 43 45 3d 22 41 72 69	><FONT F ACE="Ari	
00c0 61 6c 22 20 53 49 5a 45 3d 32 20 43 4f 4c 5f 52	al" SIZE =2 COLOR	
00d0 3d 23 30 30 30 30 30 3e 74 68 61 6e 6b 73 20	=#000000 >thanks	
00e0 64 75 64 65 3c 2f 46 4f 4e 54 3e 3c 2f 42 4f 44	dude</FO NT></BOD	
00f0 59 3e 3c 2f 48 54 4d 4c 3e 00 0d 00 12 00 01 00	Y></HTML>>.....	
0100 05 2b 00 00 31 6e 00 81 00 05 2b 00 00 14 4f 00	.+.ln...+.+.0.	
0110 0b 00 00 00 13 00 01 03	

Figure 4–10. In this screenshot, Frame 167 has been decoded as AIM traffic. Wireshark displays individual fields such as the Message field, along with corresponding bytes, and allows the user to save the contents of these fields.

traffic (rather than SSL traffic), you can view the “Message” field in the Packet Details window and the corresponding bytes in the Packet Bytes window below. You can also use Wireshark’s “Export Selected Packet Bytes” function to save the contents of the selected field as a file that you can manipulate and analyze with other tools. Figure 4–10 shows the AIM Message field within packet #167 of Ann’s Bad AIM.

You can also use tshark to print any or all fields defined within the protocol dissection. Here is an example using Franck Guénichot’s Lua plugin to export all OFT details from Frame #112, such as the OFT filename, size, and more:

```
$ tshark -r evidence.pcap -X lua_script:oft-tsk.lua -R "oft" -n -R frame.
  number==112 -V
Frame 112 (310 bytes on wire, 310 bytes captured
...
Oscar File Transfer Protocol (256)
  Version: OFT2
  Length: 256
  Type: Prompt (0x0101)
  Cookie: 0000000000000000
  Encryption: None (0)
  Compression: None (0)
  Total File(s): 1
  File(s) Left: 1
  Total Parts: 1
  Parts Left: 1
  Total Size: 12008
  Size: 12008
  Modification Time: 0
  Checksum: 0xb1640000
```

```

Received Resource Fork Checksum: 0xfffff0000
Ressource Fork: 0
Creation Time: 0
Resource Fork Checksum,base: 0xfffff0000
Bytes Received: 0
Received Checksum: 0xfffff0000
Identification String: Cool FileXfer
Flags: 0x00
List Name Offset: 0
List Size Offset: 0
Dummy Block: 00000000000000000000000000000000000000000000000000000000000000...
Mac File Information:
Encoding: ASCII (0x0000)
Encoding Subcode: 0x0000
Filename: recipe.docx

```

The following invocation demonstrates how to use tshark to produce PDML output using the -T option. First, we see general information and Layer 2 frame data. This is followed by Layer 3 IP protocol details, and then higher-layer encapsulated protocol information such as the OSCAR File Transfer Protocol information. The last field shown below is the name of the file that was transferred using OFT. Note that the output below has been abridged to show notable fields.

```

$ tshark -r evidence01.pcap -X lua_script:oft-tsk.lua -R "oft" -n -R frame.
    number==112 -T pdml
<?xml version="1.0"?>
<pdml version="0" creator="wireshark/1.2.11">
<packet>
  <proto name="geninfo" pos="0" showname="General information" size="310">
    <field name="num" pos="0" show="112" showname="Number" value="70" size
      ="310"/>
    <field name="len" pos="0" show="310" showname="Frame Length" value="136"
      size="310"/>
    <field name="caplen" pos="0" show="310" showname="Captured Length" value
      ="136" size="310"/>
    <field name="timestamp" pos="0" show="Aug 12, 2009 23:58:04.206379000"
      showname="Captured Time" value="1250143084.206379000" size="310"/>
  </proto>
  ...
  <proto name="ip" showname="Internet Protocol, Src: 192.168.1.158
    (192.168.1.158), Dst: 192.168.1.159 (192.168.1.159)" size="20" pos="14">
    <field name="ip.version" showname="Version: 4" size="1" pos="14" show="4"
      value="45"/>
    <field name="ip.hdr_len" showname="Header length: 20 bytes" size="1" pos
      ="14" show="20" value="45"/>
  ...
  <proto name="oft" showname="Oscar File Transfer Protocol (256)" size="256"
    pos="54">
    <field name="oft.version" showname="Version: OFT2" size="4" pos="54" show
      ="OFT2" value="4f465432"/>
    <field name="oft.length" showname="Length: 256" size="2" pos="58" show
      ="256" value="0100"/>
    <field name="oft.type" showname="Type: Prompt (0x0101)" size="2" pos="60"
      show="0x0101" value="0101"/>

```

To show only specific fields, such as the filename and total size, you can use tshark's `-T` and `-e` flags:

```
$ tshark -r evidence.pcap -X lua_script:oft-tsk.lua -R "oft" -n -T fields -e  
    "oft.filename" -e oft.totsize -R frame.number==112  
recipe.docx 12008
```

Now we can easily see the filename (“`recipe.docx`”) and the file size (12,008 bytes). This filename is especially interesting given that Anarchy-R-Us is concerned that Ann Dercovor may have leaked their secret recipe.

4.2 Packet Analysis

Packet analysis refers to the art and science of inspecting the protocols within a set of packets. Network analysts and investigators often conduct packet analysis in order to identify packets of interest and understand their structure and relationship to gather evidence and facilitate further analysis.

To identify packets of interest, investigators generally use filtering techniques to isolate packets based on protocol fields or their contents. In addition, investigators may search for strings or patterns in packet contents to identify targets for further inspection, even if the protocol in use is not yet known.

Understanding the packet structure is very important for reconstructing communications, transferred files, or any other flow-based transaction. Careful dissection of a single packet or a small group of packets will often help investigators identify which tools are appropriate for evidence extraction and reconstruction.

Definition: Packet Analysis

Packet Analysis—Examination of contents and/or metadata of one or more packets. Packet analysis is typically conducted in order to identify packets of interest and develop a strategy for flow analysis and content reconstruction.

4.2.1 Packet Analysis Tools

Packet analysis is fun! Using widely available tools, you can dissect packets and extract all kinds of details. In this section, we talk about Wireshark/tshark display filters, which make it easy to isolate packets of interest. We also touch on “ngrep” (one of our favorite tools), which is essentially “grep” for packet captures. Finally, we introduce hex editors in the context of packet analysis, and show how we can use them to manually slice and splice packet data.

4.2.1.1 Wireshark/tshark Display Filters

Wireshark (and its command-line counterpart, tshark) includes a “display filter” language that allows the end user to isolate packets of interest based on protocol fields.²⁸

According to Wireshark, there are over 105,000 display filters that users can choose from—which is to say that there are over 105,000 protocol fields that Wireshark’s parser understands, as it is designed to allow you to filter on any field in any protocol it can parse.²⁹

As more protocol parsers are included in Wireshark, its ability to filter based on those protocols continues to develop. Furthermore, Wireshark provides an open plugin architecture that allows anyone to build a new protocol parser to filter with as well.

In addition to the published documentation released online by Wireshark, the graphical Wireshark tool includes handy references for looking up display filters. You can click on the “Expressions” button in the Filter Toolbar to access a box that will help you build a filter of your choice (see Figure 4-11). Even more intuitively, if you identify a field of interest in the Packet Details windows, you can right-click on it and click “Apply As Filter” to filter using that field and its value. You can also view a list of supported protocols and display filters by going to Help → Supported Protocols.

You can use Wireshark’s display filters from the command line in tshark using the -R option. For example:

```
$ tshark -r capturefile.pcap -R "ip.src==192.168.1.158 && ip.dst==10.1.1.10"
```

28. “Wireshark,” <http://www.wireshark.org/docs/man-pages/wireshark-filter.html>.

29. “Wireshark—Display Filter Reference,” <http://www.wireshark.org/docs/dfrref/>.

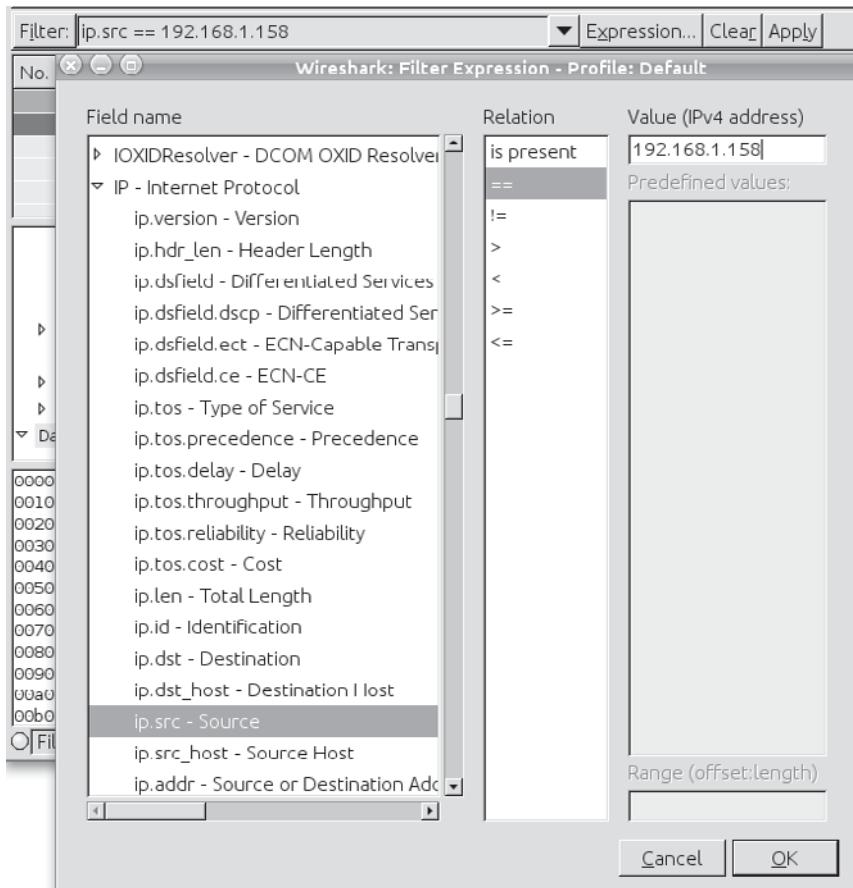


Figure 4-11. Screenshot of the “Filter Expression” box accessed from the Filter toolbar in Wireshark.

4.2.1.2 ngrep

Ngrep is an excellent libpcap-based tool designed for identifying packets of interest based on the presence (or absence) of particular strings, binary sequences, or patterns anywhere within the packet. According to the author, Jordan Ritter, ngrep is designed “to provide most of GNU grep’s common features, applying them to the network layer.”³⁰

Using ngrep, you can search a set of packets for literal ASCII strings, or binary sequences (specified in hex), that appear anywhere in a packet’s payload. It can write out the packets that match to a separate file. It also recognizes common protocols such as IP, TCP, UDP, and ICMP, and prints out summary details such as IP addresses and port numbers for matching packets.

30. Jordan Ritter, “ngrep(8): network grep—Linux man page,” <http://linux.die.net/man/8/ngrep>.

Ngrep doesn't include flow reconstruction capabilities, so it inspects each packet atomically for the specified match criteria. This has two important implications for forensic investigators. First, if the matching content spans packets, ngrep will not detect it. Second, if ngrep finds a match, it will only indicate the matching packet, rather than the matching flow.³¹ Nevertheless, for a first-pass analysis or a quick dirty-word search, ngrep is invaluable; other tools can be used to reconstruct the contents of the flows that contain the matching packets.

Basic usage for analyzing a capture file with ngrep:

```
$ ngrep -I capturefile.pcap "string to search for"
```

The above command will search for the string “string to search for” and print out all packets in ASCII format. To include hexadecimal format, you would add the -x flag.

Many libpcap-based tools have interfaces for specifying BPF filters, and ngrep is no exception. The command below runs ngrep with a BPF filter designed to include only packets with a source IP address of 192.168.1.20 and destination port 80.

```
$ ngrep -I capturefile.pcap "string to search for" 'src host 192.168.1.20 and dst port 80'
```

For more information, see the excellent examples Jordan Ritter provides for the use of his tool.³²

4.2.1.3 Hex Editors

Hex editors allow you to view and modify the raw bits of data including packet capture contents. Although you can use tools such as tcpdump and Wireshark to view the raw bits of specific packets and protocol fields, hex editors are indispensable for analysis and isolation of packet fragments and file carving (which we discuss later in this chapter).

Sometimes tools like Wireshark, Network Miner, and tcpxtract can automatically export files and events for you with little effort, but not always. There are times when the only way to reconstruct data is with manual effort, reading the protocol specification, extracting data, and rebuilding content by hand.

For example, sometimes the higher-layer protocol used for transmission is not recognized by a forensic analysis tool. The tool Loki tunnels data over ICMP, but common protocol dissectors are not designed to recognize the Loki tunneling protocol, correctly interpret the tunneled data, and reconstruct such data streams. Other times, an additional layer of complexity may prevent a tool from automatically extracting evidence. For instance, HTTP servers sometimes compress content using algorithms such as gzip. Forensic analysis tools often cannot “see” inside the compressed data to extract images or content that would have been viewed by the end-user. Instead, the forensic investigator may need to manually recognize that the content is compressed and use an alternate tool to carve and decompress the data.

Find a hex editor that you feel comfortable with (such as the nice GUI tool “Bless”) and become proficient with it.

31. Jordan Ritter, “ngrep—network grep,” November 18, 2006, <http://ngrep.sourceforge.net>.

32. “ngrep—network grep,” February 10, 2005, <http://ngrep.sourceforge.net/usage.html>.

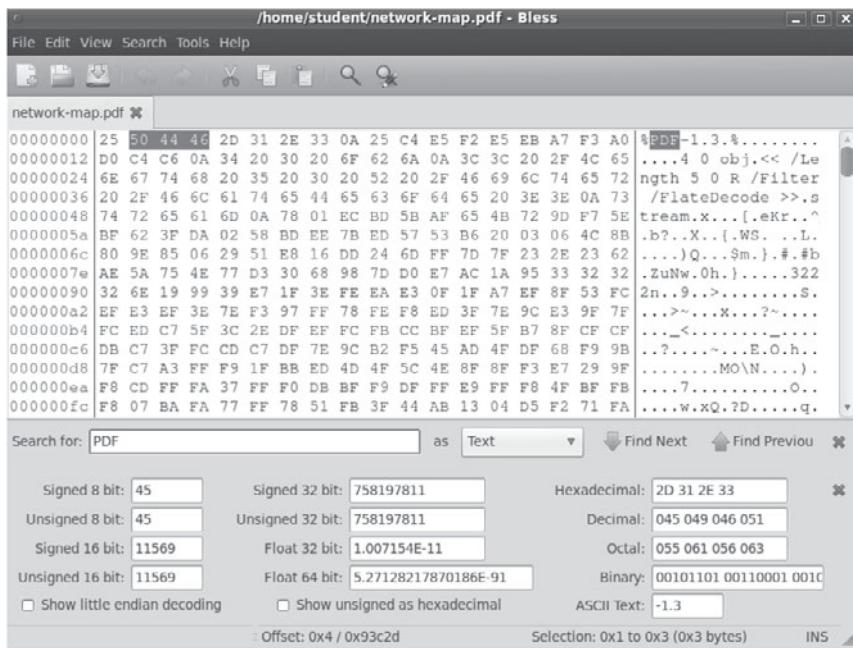


Figure 4–12. Screenshot of the Bless hex editor.

Figure 4–12 shows an example of the hex editor “Bless.” In this example, we’ve used Bless to open a PDF file. Bless is a powerful and easy-to-use hex editor. You can search for binary or strings in the file. Here we searched for the string “PDF,” which we find at byte offset 0x01 through 0x03 (see the bottom right panel). The “magic number” “0x25504446,” or “%PDF” in ASCII, indicates the beginning of a PDF document. Using our hex editor, we can modify, add, or delete sections of the file.

4.2.2 Packet Analysis Techniques

There are three basic techniques that are indispensable when conducting packet analysis:

- **Pattern Matching**—Identify packets of interest by matching specific values within the packet capture.
- **Parsing Protocol Fields**—Extract the contents of protocol fields.
- **Packet Filtering**—Separate packets based on the values of fields in protocol metadata.

While there are certainly other packet analysis techniques, these three are fundamental and incorporated into nearly every investigation that involves packet analysis.

4.2.2.1 Pattern Matching

Is Ann Dercover really a secret agent? Did she export Anarchy-R-Us’s secret recipe? We’ve analyzed the packet capture of her activities and discovered that the AIM protocols were in use, and that a file may have been transferred. This doesn’t bode well for Anarchy-R-Us!

Let's quickly search the packet capture for strings or patterns of interest to us. In hard drive forensics, it is common to do a “dirty word search” when searching for data of interest on the hard drive. A “dirty word list” is a list of strings, names, patterns, etc., that may be related to the suspicious activities under investigation. Network forensic investigators can leverage the concept of a “dirty word list” to identify packets or data of interest within a network traffic capture.

“Ngrep” is one of the best tools for launching a dirty word search on a packet capture. Let's use it to examine Ann's Bad AIM packet capture.

First, we'll create a short list of “dirty words” that are of interest to us. Given the context of the investigation, let's include:

- recipe
- secret
- Ann

Next, let's use these dirty words in a regular expression as the input to ngrep, and print packets that match:

```
$ ngrep -I evidence01.pcap 'secret|recipe|Ann'
input: evidence01.pcap
match: secret|recipe|Ann
#####
T 192.168.1.158:51128 -> 64.12.24.50:443 [AP]
*...a.....E4628778....Sec558user1.....Here's the secret
    recipe... I just downloaded it from the file server. Jus
    t copy to a thumb drive and you're good to go &gt;:-)....
#####
T 192.168.1.158:51128 -> 64.12.24.50:443 [AP]
*...c.z.....G7174647....Sec558user1.....R..7174647..F.CL...."DEST
    .....F.....'.....recipe.docx.
#####
T 192.168.1.158:5190 -> 192.168.1.159:1272 [AP]
OFT2.....d.....Cool
    FileXfer...
    .....recipe.docx
    .....
#####
T 192.168.1.159:1272 -> 192.168.1.158:5190 [AP]
OFT2....7174647.....d.....Cool
    FileXfer...
    .....recipe.docx
    .....
#####
T 192.168.1.159:1272 -> 192.168.1.158:5190 [AP]
OFT2....7174647.....d.....d..Cool
    FileXfer...
    .....recipe.docx
    .....
#####
exit
```

In the output above, we see a snippet of a conversation indicating “Here’s the secret recipe.” Very interesting! The source IP address of the packet containing this snippet of conversation was “192.168.1.158” (known to be Ann’s computer) and the destination was “64.12.24.50.” Subsequently, we see strings associated with an OFT2 file transfer (“recipe.docx”). The destination IP address of the OFT2 file transfer is a system on the local network, 192.168.1.159, but the source is still the same: Ann’s computer.

4.2.2.2 Parsing Protocol Fields

Parsing protocol fields is the practice of extracting the contents of protocol fields within packets of interest. Let’s use tshark to extract all the AIM message data from this packet capture. Perhaps we will see other interesting conversation snippets.

```
$ tshark -r evidence01.pcap -d tcp.port==443,aim -T fields -n -e "aim.messageblock.message"
Here's the secret recipe... I just downloaded it from the file server. Just
copy to a thumb drive and you're good to go &gt;:-)
<HTML><BODY><FONT FACE=\\"Arial\\" SIZE=2 COLOR=#000000>thanks dude</FONT></
BODY></HTML>
<HTML><BODY><FONT FACE=\\"Arial\\" SIZE=2 COLOR=#000000>can't wait to sell it
on ebay</FONT></BODY></HTML>
see you in hawaii!
```

Fascinating! It appears from this conversation that the recipient of the file is planning on selling it on eBay, and that the two are planning a rendezvous in Hawaii.

4.2.2.3 Packet Filtering

Packet filtering is the art of separating packets based on the values of fields in protocol metadata or payload. Commonly, investigators filter packets using either BPF filters or Wireshark display filters. We demonstrate each of these in turn.

Filtering with BPF Earlier we identified two hosts, 192.168.1.58 (Ann’s computer) and 64.12.24.50 (likely an AOL AIM server), that contained an interesting conversation snippet, “Here’s the secret recipe ...”. We subsequently found other conversation snippets in the same packet capture.

To reduce the volume of traffic that we have to inspect, let’s use tcpdump with a BPF filter to dump out all packets between the same source and destination IP address as the first suspicious conversation snippet that we found.

```
$ tcpdump -s 0 -r evidence01.pcap -w evidence01-talkers.pcap 'host
64.12.24.50 and host 192.168.1.158'
Reading from file evidence01.pcap, link-type EN10MB (Ethernet)
```

Wireshark Display Filters Let’s open this filtered packet capture (evidence01-talkers.pcap) in Wireshark and see if we can get more information about the context of this conversation. As shown in Figure 4-13, we only have to browse to Frame 3 to see something we’ve seen before: cleartext traffic on port 443. We can also see that the Packet Bytes window displays the message containing “Here’s the secret recipe ...”. Just as before, we can identify the

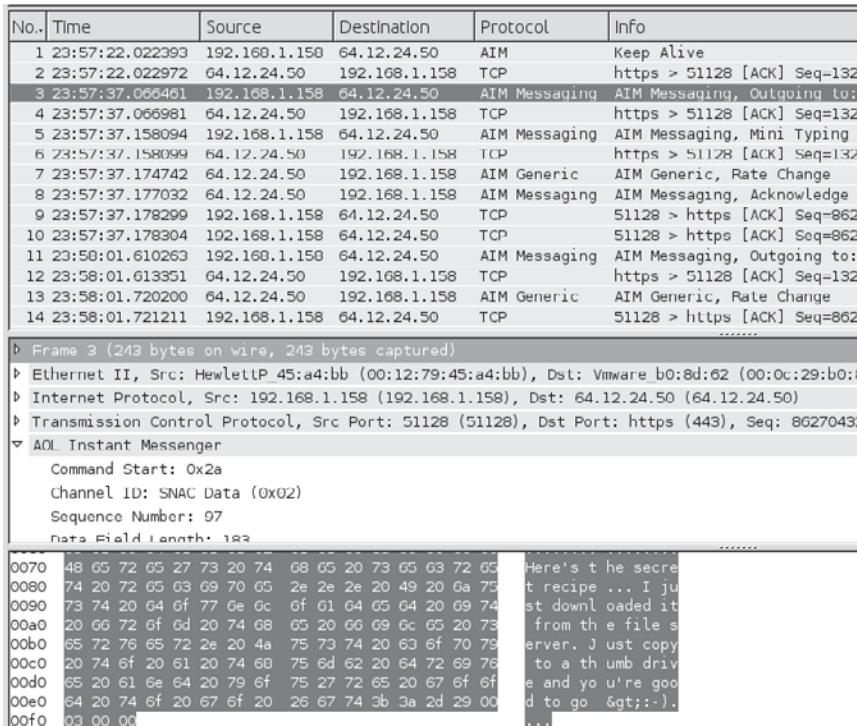


Figure 4–13. The filtered packet capture, evidence01-talkers.png, opened in Wireshark. The traffic relating to TCP port 443 has been decoded as AIM.

protocols in use and manually configure Wireshark to decode this packet as AIM traffic with the “Decode As . . .” feature (see Figure 4–8).

In order to negotiate an OFT file transfer, the sender’s AIM client begins listening for direct connections on a specific port and sends a channel 2 ICBM packet to the AOL server. This packet contains the port and IP address to which the remote peer should connect to receive the file.

Let’s use Wireshark’s display filters to search for channel 2 ICBM packets to see if we can find the packet that contains the file transfer negotiation details. This will allow us to link the conversation snippet, “Here’s the secret recipe . . .” to an actual file transfer between two direct peers.

Figure 4–14 shows the display filter, aim.messaging.channelid==0x0002, which picks out ICBM channel 2 messages. After applying this filter, three packets are displayed. The first packet in the list shows the string “recipe.docx” in the Packet Bytes window. Suspicious!

In Figure 4–15, we scroll down in the Packet Details window to show the inbound IP address (192.168.1.158) and port (TCP port 5190) offered by the initiating client. If the

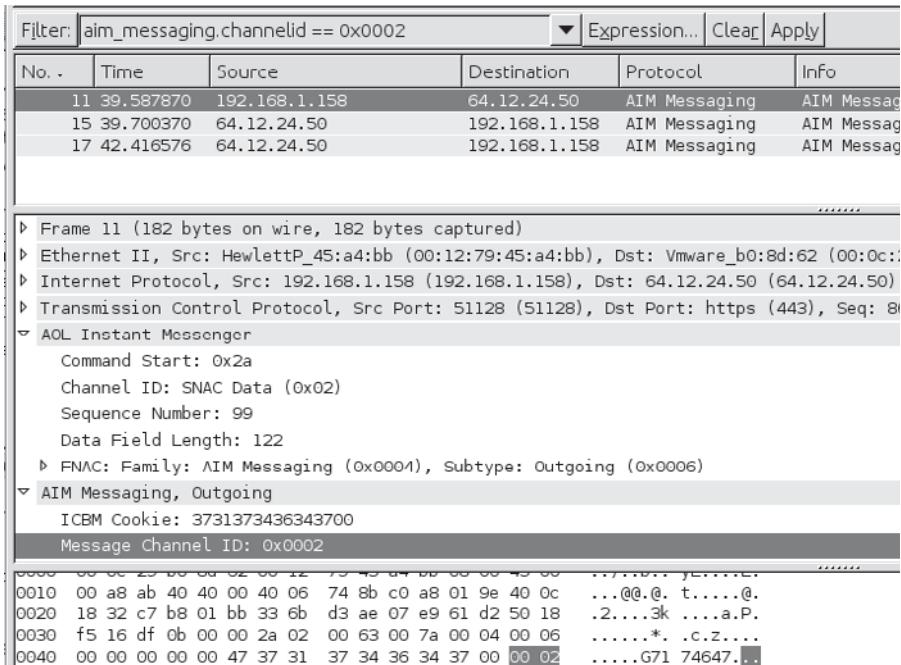


Figure 4–14. A screenshot of Wireshark with a display filter that selects AIM ICBM channel 2 messages.

file transfer is accepted by the remote peer, we should expect to see a subsequent direct connection from the remote peer’s IP address to the IP address and port specified in this packet. (Later, in Section 4.3.2, we will see that this direct connection does exist, and we will analyze it.) *This packet is what links the AIM conversation, transmitted through AOL’s servers, to a subsequent direct OFT file transfer between two peers.*

4.3 Flow Analysis

Flow analysis is the practice of examining related groups of packets in order to identify patterns, analyze higher-layer protocols, or extract data. Once you have identified the protocols in use and of interest and studied them, you will likely want to reconstruct sequences of events and recover original data from them. With some well-known higher-layer protocols, such as HTTP, there are many tools that can do this automatically. With other protocols—especially those that have been bent or broken—you may have to do your own independent analysis, and perhaps even write some code to extract the data.

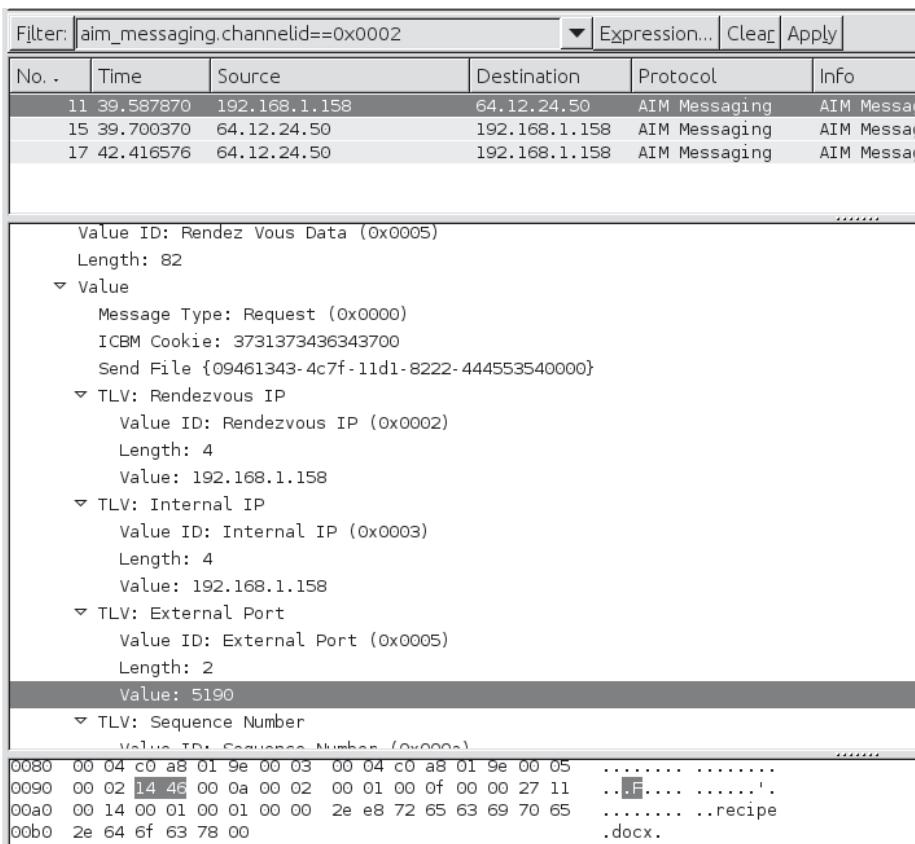


Figure 4–15. A screenshot of Wireshark displaying the IP address and port offered by the sender for an OFT file transfer.

Definition: Flow Analysis

Flow analysis—Examination of sequences of related packets (“flows”). Flow analysis is typically conducted in order to identify traffic patterns, isolate suspicious activity, analyze higher-layer protocols, or extract data.

As the networking industry evolves, so too has the use of the term “flow.” Increasingly, the term “flow analysis” in certain contexts has been used to refer to statistical analysis of flow records or flow metadata. We discuss flow statistics extensively in the next chapter. In this chapter, we focus on the analysis of the full flow content rather than simply the metadata.

The term “stream” is increasingly used interchangeably with “flow” in common parlance, particularly where data segment reassembly and subsequent content analysis are the main

goals (as opposed to the statistical analysis of the sequence of segments or packets involved). In particular, Marty Roesch’s labeling of Snort’s flow reassembly processing module as “Stream” (which originally only processed TCP flows, but which now has evolved to handle flows over UDP as well) led to much of the IDS community referring to such methods as “stream reassembly.” Similarly, Wireshark’s “Follow TCP Stream” function has been augmented by “Follow UDP Stream” and “Follow SSL Stream.” We use the term “stream” in all of these contexts throughout this book.

Definition: Flow

In RFC 3679, a “flow” is defined as “a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow. A flow could consist of all packets in a specific transport connection or a media stream. However, a flow is not necessarily 1:1 mapped to a transport connection.” For practical purposes, identifying all of the TCP segments that comprise a flow can be done strictly within the TCP protocol itself at Layer 4 because the TCP protocol has built-in mechanisms to provide this accounting for the purposes of reliability, flow control, and state maintenance. Flows can be constructed upon other transport-layer protocols, including UDP, which is a connectionless protocol and provides no comparable mechanisms to track or define flows. As implied by RFC 3679, if an endpoint “desires” to define a flow, the flow itself will be defined by some protocol that manages connection establishment, maintenance, and completion. Likewise, an intermediary system (such as a switch, router, or firewall) can be configured to track and define flows.

Regardless of whether traffic is transmitted via TCP, a specific flow is defined by some protocol that establishes the connection, keeps track of the data segments sent or received, and terminates the connection when appropriate.

4.3.1 Flow Analysis Tools

There are several popular tools available that are capable of isolating, reconstructing, and exporting flows. These include Wireshark, tshark, tcpflow, and pcapcat. The tcpxtract tool takes flow reconstruction to the next level, and automatically carves files out of TCP streams using magic numbers. We discuss each of these tools below.

4.3.1.1 Wireshark: Follow TCP Stream

Wireshark has a function called “Follow TCP Stream” (shown in Figure 4–16) which allows you to select any packet that is part of a TCP stream in the Packet List view, and have Wireshark automatically reconstruct the full duplex contents of that stream from beginning to end if it is contained within the packet capture. Those contents can then be saved in different forms. Either side of the conversation can be saved independently. In that way, conversations, transactions, and file transfers that span multiple packets in a stream can be reconstructed in their entirety, as long as the packet capture contains all the necessary data.

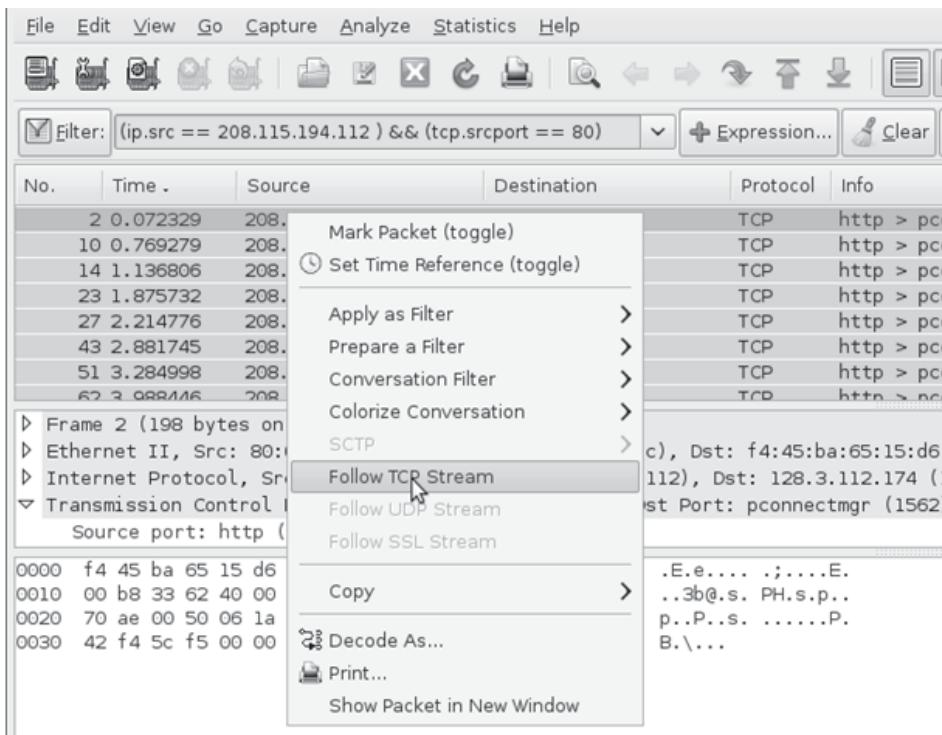


Figure 4–16. An example of Wireshark’s “Follow TCP Stream” function.

Figure 4–17 is a screenshot of Wireshark’s “Follow TCP Stream” feature used to reconstruct and display both sides of a TCP stream (dark gray text represents traffic flowing in one direction, and light gray is the other direction). Notice the long button across the bottom, which allows you to select “Entire Conversation” or only one direction. You can also view in ASCII, Hex, Raw, or other formats.

The “Follow TCP Stream” feature is one of the features of the graphical Wireshark interface that does not currently have a corresponding command-line option in tshark as of the time this was written.

4.3.1.2 Conversations in Wireshark and tshark

Wireshark and tshark define a term, “conversation,” which is similar to a flow. According to the published Wireshark manual, “A network conversation is the traffic between two specific endpoints.”³³

33. “8.4.Conversations,” *Wireshark*, 2011, http://www.wireshark.org/docs/wsug_html_chunked/ChStatConversations.html.

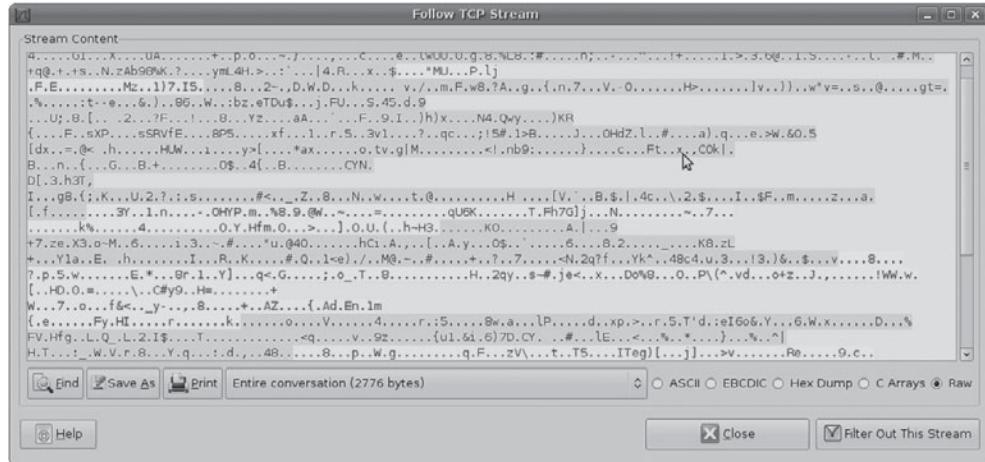


Figure 4–17. Wireshark’s “Follow TCP Stream” function in action. The dark gray text represents traffic flowing in one direction, whereas the light gray text represents traffic flowing in the opposite direction. (In real life these colors were pink and blue, respectively; the image was modified to print in grayscale.)

Both Wireshark and tshark provide support for displaying statistics for different types of conversations in the packet capture, including IP, TCP, and UDP conversations.

4.3.1.3 tcpflow

The quintessential command-line tool for extracting the data from TCP streams is “tcpflow.” Originally released by Jeremy Elson in 1999, tcpflow can parse nonfragmented IP packets and reassemble and extract the payloads of any TCP stream it finds in a libpcap packet capture. In that way its output is similar to Wireshark’s “Follow TCP Stream” function, but it extracts all the streams in one fell swoop, saving their contents to files identified by the quartet of socket elements: source IP and port and destination IP and port.³⁴

4.3.1.4 pcapcat

A more recent and similarly useful tool is pcapcat by Kristinn Guðjónsson. Released in September 2009 as part of his winning entry in the inaugural ForensicsContest.com contest,³⁵ pcapcat is a perl script that by default reads a libpcap capture and lists all of the streams it sees. It allows for interactive analysis and can also dump out individual streams upon selection.³⁶

34. Jeremy Elson, “CircleMUD,” August 7, 2003, <http://www.circlemud.org/jelson/software/tcpflow/>.

35. Kristinn Guðjónsson, “Forensics Contest,” August 14, 2009, <http://forensicscontest.com/contest01/Finalists/Kristinn.Gudjonsson/answer.txt>.

36. Kristinn Guðjónsson, “pcapcat,” 2009, <http://blog.kiddaland.net/dw/pcapcat>.

Magic Numbers

Just as most protocols can be identified by well-known sequences of bytes near the zero offset (as discussed above), almost all file formats have “headers” that lead with a few identifying bytes that can be catalogued. Many have common terminating sequences, or “footers,” as well.

These are often referred to as “magic numbers,” and if they can be catalogued they can be searched for in any arbitrary volume of bytes. The filesystem forensics tool “Foremost” takes exactly this approach. Upon finding an instance of a magic number, it begins carving contiguous data bytes until a preconfigured limit is reached, or the footer for the file (if any is specified) is encountered. This data is saved to a file with the appropriate file extension appended to the name.³⁷

4.3.1.5 tcpxtract

Tcpxtract may be one of the coolest tools for the network forensic analyst. Released in the public domain in 2005 by Nick Harbour, tcpxtract is a libpcap-based tool designed specifically to extract and reconstruct payload data based on file signatures. Analogous to “foremost” for data-layer recovery from filesystem images, tcpxtract inspects reconstructed TCP streams for the beginning sequences of file formats that it knows about and carves the corresponding data out of the packets, to the extent that it is able. It has a configuration file that is very similar to foremost’s (and comes with a script to convert foremost configuration files to the format used by tcpxtract).³⁸

Here’s a snippet of a configuration file:

```
-----
# GRAPHICS FILES
-----
#
# AOL ART files
art(150000, \x4a\x47\x04\x0e, \xcf\xc7\xcb);
art(150000, \x4a\x47\x03\x0e, \xd0\xcb\x00\x00);
# GIF and JPG files (very common)
gif(3000000, \x47\x49\x46\x38\x37\x61, \x00\x3b);
gif(3000000, \x47\x49\x46\x38\x39\x61, \x00\x00\x3b);
jpg(1000000, \xff\xd8\xff\xe0\x00\x10, \xff\xd9);
jpg(1000000, \xff\xd8\xff\xe1);
jpg(1000000, \xff\xd8\xff\xe0);
# PNG (used in web pages)
png(1000000, \x50\x4e\x47\?, \xff\xfc\xfd\xfe);
```

A format variant is described on each line. The first parameter is the number of bytes to carve before giving up on trying to find the end sequence, if one is provided. The second parameter is the hexadecimal sequence that, upon discovery, triggers the carving of what is

37. “Foremost,” March 1, 2010, <http://foremost.sourceforge.net/>.

38. Nick Harbour, “tcpxtract,” October 13, 2005, <http://tcpxtract.sourceforge.net/>.

assumed to be data corresponding to the file type names. The third, optional parameter is the byte sequence that the file format uses to signal “end of file” (EOF), if one is known to exist.

Basic usage for analyzing a capture file:

```
$ tcpextract -f capturefile.pcap
```

This will extract all recognizable files from the packet capture. You can also specify an output directory:

```
$ tcpextract -f capturefile.pcap -o output_dir/
```

4.3.2 Flow Analysis Techniques

Basic flow analysis techniques include:

- **List conversations and flows**—List all conversations and/or flows within a packet capture, or only specific flows based on their characteristics.
- **Export a flow**—Isolate a flow, or multiple flows, and store the flow(s) of interest to disk for further analysis.
- **File and data carving**—Extract files or other data of interest from the reassembled flow.

We discuss each of these, and show examples, in the sections below.

4.3.2.1 List Conversations

Continuing with the Ann’s Bad AIM case, we know from packet analysis that the AIM client at 192.168.1.158 initiated a file transfer on port 5190 in order to transfer the file “recipe.docx”. Did a remote peer ever connect to this port to complete the file transfer?

We can use tshark to quickly view packet capture conversation statistics, as shown below:

```
$ tshark -qn -z conv,tcp -r evidence01.pcap
=====
TCP Conversations
Filter:<No Filter>
          |   <-   |   ->   |     Total    |
          |   Frames  Bytes|   Frames  Bytes|   Frames  Bytes
192.168.1.159:1271  <-> 205.188.13.12:443  31  29717  16  1451  47  31168
192.168.1.159:1221  <-> 64.12.25.91:443  24  4206  16  1799  40  6005
192.168.1.158:51128  <-> 64.12.24.50:443  20  2622  20  1681  40  4303
192.168.1.158:5190  <-> 192.168.1.159:127  9  1042  15  13100  24  14142
192.168.1.159:1273  <-> 64.236.68.246:80  5  1545  5  1964  10  3509
192.168.1.2:54419   <-> 192.168.1.157:80  3  206  4  272  7  478
192.168.1.2:55488   <-> 192.168.1.30:22  2  292  3  246  5  538
=====
```

Scrolling through this list, we can see that there was indeed one conversation that involved 192.168.1.158 on TCP port 5190. The other host involved in the conversation was 192.168.1.159, and 1,042 bytes were transferred.

4.3.2.2 List TCP Flows

Let's identify the specific flows of interest so that we can get ready to extract the higher-layer protocol data. During our earlier packet analysis, we saw that the sender (192.168.1.158) initiated the file transfer during the time frame of the packet capture. The actual file transfer must have occurred subsequently. Hence, we can limit our search to TCP flows that were created during the time of the packet capture (i.e., we can safely ignore any TCP flows that were initiated before our packet capture began).

Coveniently, the default behavior of pcapcat is to list only the TCP flows that were created during the capture time. In the output below, we see that there were only four TCP flows created within the packet capture.

```
$ pcapcat -r evidence01.pcap
[1] TCP 192.168.1.2:54419 -> 192.168.1.157:80
[2] TCP 192.168.1.159:1271 -> 205.188.13.12:443
[3] TCP 192.168.1.159:1272 -> 192.168.1.158:5190
[4] TCP 192.168.1.159:1273 -> 64.236.68.246:80
Enter the index number of the conversation to dump or press enter to quit:
```

Of these, only one flow was related to 192.168.1.158 on port 5190 (flow number 3). The other endpoint of this flow is a host on the local network, 192.168.1.159. This corroborates our findings from the tshark conversation statistics:

```
[3] TCP 192.168.1.159:1272 -> 192.168.1.158:5190
```

4.3.2.3 Export TCP Flow

We've identified the flow that most likely contains the OFT file transfer. Now let's export that flow so that we can attempt to recover any data transferred within it. Using pcapcat, we can export the flow of interest, this time using a BPF filter to quickly narrow down the search:

```
$ pcapcat -r evidence01.pcap -w internal-stream.dump -f 'host 192.168.1.158
and port 5190'
[1] TCP 192.168.1.159:1272 -> 192.168.1.158:5190
Enter the index number of the conversation to dump or press enter to quit: 1
Dumping index value 1
```

No.	Time .	Source	Destination	Protocol	Info
108	61.051429	HewlettP_45:a4:bb	Dell_4d:4f:ae	ARP	192.168.1.158 is at 00:12:79:45:a4:bb
109	61.052925	192.168.1.159	192.168.1.158	TCP	cspmlockmgr > aol [SYN] Seq=0 Win=64240 Len=0 MSS=1460
110	61.052930	192.168.1.159	192.168.1.159	TCP	aol > cspmlockmgr [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
111	61.054660	192.168.1.159	192.168.1.158	TCP	cspmlockmgr > aol [ACK] Seq=1 Ack=1 Win=64240 Len=0
112	61.054684	192.168.1.158	192.168.1.159	TCP	aol > cspmlockmgr [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=256

Frame 109 (62 bytes on wire, 62 bytes captured)
 ▷ Ethernet II, Src: Dell_4d:4f:ae (00:21:70:4d:4f:ae), Dst: HewlettP_45:a4:bb (00:12:79:45:a4:bb)
 ▷ Internet Protocol, Src: 192.168.1.159 (192.168.1.159), Dst: 192.168.1.158 (192.168.1.158)
 ▷ Transmission Control Protocol, Src Port: cspmlockmgr (1272), Dst Port: aol (5190), Seq: 0, Len: 0

Figure 4-18. Frame #109 shown in Wireshark. This frame is part of the identified conversation of interest between 192.168.1.158 and 192.168.1.159 on TCP port 5190.

You can also use the tool “tcpflow” to automatically extract any or all flows from a packet capture, as shown below. Here we will use tcpflow with a BPF filter to extract any TCP flows that related to 192.168.1.158 on port 5190:

```
$ tcpflow -r evidence01.pcap 'host 192.168.1.158 and port 5190'
tcpflow[25586]: tcpflow version 0.21 by Jeremy Elson <jelson@circlemud.org>
tcpflow[25586]: looking for handler for datalink type 1 for interface
  evidence01.pcap
tcpflow[25586]: found max FDs to be 16 using OPEN_MAX
tcpflow[25586]: 192.168.001.159.01272-192.168.001.158.05190: new flow
tcpflow[25586]: 192.168.001.158.05190-192.168.001.159.01272: new flow
tcpflow[25586]: 192.168.001.158.05190-192.168.001.159.01272: opening new
  output file
tcpflow[25586]: 192.168.001.159.01272-192.168.001.158.05190: opening new
  output file

$ ls -l
total 13
-rwx----- 1 student student 12264 2011-01-08 20:53
  192.168.001.158.05190-192.168.001.159.01272
-rwx----- 1 student student 512 2011-01-08 20:53
  192.168.001.159.01272-192.168.001.158.05190
```

Notice that tcpflow extracted two half-duplex flows and saved them to files. As you can see, the filenames indicate the source and destination IP addresses and ports, and also indicate the direction of traffic flow.

You can also export flows manually in Wireshark, although this does not scale as well for large packet captures. In this case, to export the flow of interest using Wireshark, begin by clicking on any packet that is part of the flow of interest. Frame #109, as shown in Figure 4–18, is part of the connection between 192.168.1.158 and 192.168.1.159 on TCP port 5190.

Select packet #109 and right-click on “Follow TCP Stream.” This produces a screen that displays the raw contents of the full-duplex flow. We can select one side of the conversation, as shown in Figure 4–19, to see only the data transferred from 192.168.1.158 (TCP port 5190) to 192.168.1.159. Based on protocol markers such as the “OFT2” header, this half-duplex

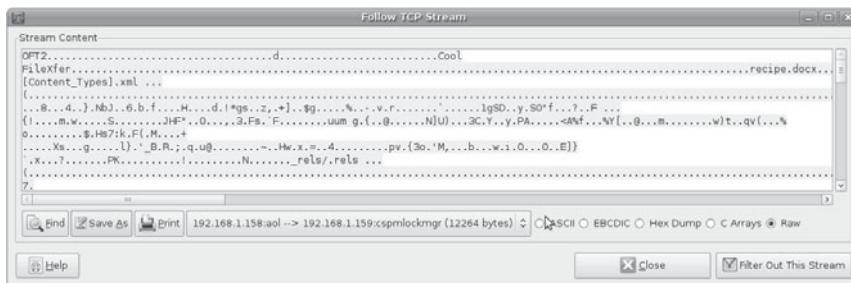


Figure 4–19. Wireshark’s “Follow TCP Stream” function, used in this case to isolate the TCP stream between 192.168.1.158 and 192.168.1.159 on TCP port 5190. We have selected one direction of the stream, which appears to contain an OFT2 file transfer of the file “recipe.docx.”

Figure 4-20. The beginning of our exported flow, internal-stream.dump. Notice the first 4 bytes in the stream are “OFT2,” which mark the beginning of an OFT header. Bytes 6–7 (“Type”) are 0x0101, which indicates that the sender is ready to transfer data.

stream appears to contain one side of an OFT2 file transfer of a file called “recipe.docx.” Click “Save As” to save the half-duplex flow in “raw” format.

4.3.2.4 File and Data Carving

What was the file that Ann's AIM client transferred to 192.168.1.159? Now that we have exported the flow that is most likely to contain this OFT2 file transfer, let's see if we can carve the file itself out of the captured network traffic.

Figure 4–20 shows the exported flow, internal-stream.dump, opened in the Bless hex editor. Notice the first 4 bytes in the stream are “OFT2,” which mark the beginning of an OFT header. Bytes 6–7 (“Type”) are 0x0101, which indicates that the sender is ready to transfer data.

As shown in Figure 4-21, bytes 28–31 represent the “Total Size” of the file that the sender intends to transfer. In this case, the “Total Size” is “0x00002EE8,” or 12,008 bytes in

Figure 4-21. Our exported flow, internal-stream.dump, shown in Bless. Bytes 28–31 represent the “Total Size” of the file that the sender intends to transfer. In this case, the “Total Size” is “0x00002EE8,” or 12,008 bytes in decimal. Further below, we can see the filename “recipe.docx,” which begins at Byte 192 of the OFT header.

000000f0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000100	4F 46 54 32 01 00 02 02 37 31 37 34 36 34 37 00	OFT2...7174647.
00000110	00 00 00 00 00 01 00 01 00 01 00 01 00 00 00 00 2E E8
00000120	00 00 2E E8 00 00 00 00 B1 64 00 00 00 00 00 00 00 00 00d.....
00000130	00 00 00 00 00 00 00 00 FF FF 00 00 00 00 00 00 00 00 00
00000140	00 00 00 00 43 6F 6F 6C 20 46 69 6C 65 58 66 65Cool FileXfe
00000150	72 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	r.....
00000160	00 00 00 00 20 1C 11 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 4–22. Our exported flow, internal-stream.dump, shown in Bless. Here at Byte 256 (0x100) we see a second OFT header with “Type” 0x0202, or “Acknowledge.” OFT “Acknowledge” packets are sent by the recipient to the sender to indicate that the recipient is ready to receive the file transfer.

decimal. The “Filename” string begins at Byte 192 (0xc0), and in this case it is “recipe.docx”, (padded with nulls to 64 bytes).

Scrolling down to Byte 256 (0x100), we see the beginning of another 256-byte OFT2 header (Figure 4–22). This time, the “Type” is “0x0202,” or “Acknowledge.” OFT “Acknowledge” packets are sent by the recipient to the sender to indicate that the recipient is ready to receive the file transfer. Since we are looking at the full-duplex communication, we see OFT messages from both the sender and recipient. If we were looking at only the half-duplex flow that we exported from Wireshark, we would not have seen this message from the recipient.

Now let’s look for the beginning of a transferred .docx file. We can identify the beginning of the .docx file by the “magic number” at the beginning, “0x504B” or “PK” in ASCII. Scrolling down in Bless to Byte 512 (0x200) Figure 4–23, this magic number appears at bytes offset 0x200 to 0x201 (2 bytes). The Bless hex editor displays the byte locations for us when the bytes are highlighted (see the bottom panel for the offset in hexadecimal).

To find the end of the transferred file, we can add the initial byte offset of the file (0x02000) to the expected size (0x2EE8), which means that the file should end right before

000001a0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0	72 65 63 69 70 65 2E 64 6F 63 78 00 00 00 00 00 00 00 00	recipe.docx.....
000001d0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001e0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001f0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200	50 4B 03 04 14 00 06 00 08 00 00 00 21 00 7C 10	PK.....!..
00000210	EE 3D 7F 01 00 00 A4 05 00 00 13 00 08 02 5B 43	=.....[C
00000220	6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D	ontent_Types].xm
00000230	6C 20 A2 04 02 28 A0 00 02 00 00 00 00 00 00 00 00 00	l(.....
00000240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Offset: 01002 / 030747 Selection: 01000 to 01001 (02 bytes) INS

Figure 4–23. Our exported flow, internal-stream.dump, shown in Bless. The magic number for a .docx file, 0x504B, appears at Bytes 512–513 (0x200–0x201).

000030c0	00 00 64 6F 63 50 72 6F 70 73 2F 61 70 70 2E 78	..docProps/app.x
000030d0	6D 6C 50 4B 05 06 00 00 00 00 0E 00 0E 00 94 03	mlPK.....
000030e0	00 00 3E 2B 00 00 00 00 4F 46 54 32 01 00 02 04	..>....OFT2....
000030f0	37 31 37 34 36 34 37 00 00 00 00 00 00 01 00 01	7174647.....
00003100	00 01 00 01 00 00 2E E8 00 00 2E E8 00 00 00 00
00003110	B1 64 00 00 FF FF 00 00 00 00 00 00 00 00 00 00	.d.....

Figure 4–24. Our exported flow, internal-stream.dump, shown in Bless. The .docx file ends just before Byte 0x30E8. We see that it ends with 4 null bytes, and is immediately followed by an OFT2 “Done” header (Type “0x0204”).

byte offset 0x30E8. This matches perfectly with what we see in Figure 4–24; scrolling to the offset in the exported flow data, we see that the next OFT header does in fact begin at byte offset 0x30E8. Notice that the transferred file appears to end with four null bytes: “00 00 00 00.”

Let’s double-check and make sure that the file was transferred fully. Examining the OFT message at byte offset 0x30E8, shown in Figure 4–25, we can see that it has type 0x0204, or “Done.” OFT messages of this type are sent by the server to indicate that it is done sending the file. We can also check the size of the data that has already been sent. This is a 4-byte field at Byte 32 of the OFT header (Byte 0x3108 of the packet capture). The value of this field in the OFT “Done” packet is 0x2EE8, which matches the total size that the sender originally indicated it intended to transfer. From this, we can conclude that the file was fully transferred in one exchange.

Now that we have identified the beginning and end of the transferred file (Bytes 0x200 through 0x30E7), we can easily carve it out using Bless’ “Cut” tool. Figure 4–26 shows how to use Bless to snip off the extra data before the transferred file. Next, we also snip off the extra data after the transferred file, and save the resulting data as “recipe.docx”.

000030e0	00 00 3E 2B 00 00 00 00 4F 46 54 32 01 00 02 04	..>....OFT2....
000030f0	37 31 37 34 36 34 37 00 00 00 00 00 00 01 00 01	7174647.....
00003100	00 01 00 01 00 00 2E E8 00 00 2E E8 00 00 00 00
00003110	B1 64 00 00 FF FF 00 00 00 00 00 00 00 00 00 00	.d.....
00003120	FF FF 00 00 00 00 00 2E E8 B1 64 00 00 43 6F 6Cd..Cool
00003130	20 46 69 6C 65 58 66 65 72 00 00 00 00 00 00 00	FileXfer.....
00003140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 1C 11 00
00003150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031a0	00 00 00 00 00 00 00 00 72 65 63 69 70 65 2E 64recipe.d
000031b0	6F 63 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ocx.....
000031c0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031d0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031e0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 4–25. Our exported flow, internal-stream.dump, shown in Bless. The OFT2 “Done” header begins at byte offset 0x30E8. Note the “Size” 0x2EE8 (12,008 bytes), which indicates the amount of data that has already been transferred. This matches the total file size indicated by the sender, which means our file transfer is complete.

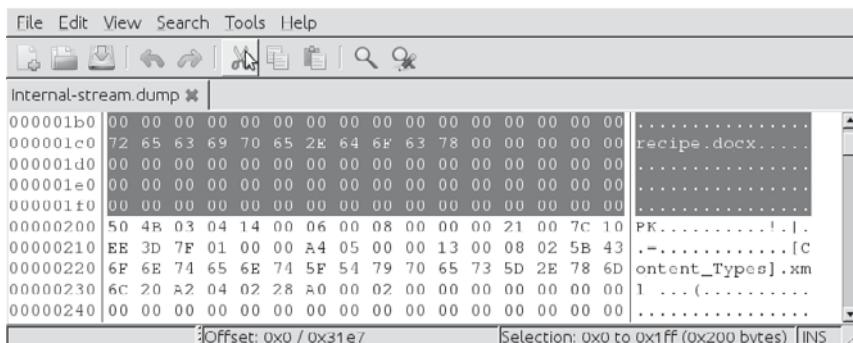


Figure 4–26. In this screenshot, we are using Bless to snip off the extra data before the transferred .docx file.

Let's collect cryptographic hashes of the file so that we can later verify it has not changed since we carved it out:

```
$ sha256sum recipe.docx
f0f74a982a814640aedaa5fd6542ac810e8c5e257552bcc024a5c808343bccf9  recipe.docx

$ md5sum recipe.docx
8350582774e1d4dbe1d61d64c89e0ea1  recipe.docx
```

Double-check the size of the file we just carved:

```
$ ls -l recipe.docx
-rwx----- 1 student student 12008 2010-09-03 16:36 recipe.docx
```

The file size is 12,008 bytes in size, which matches our expectations.

Verify the file type:

```
$ file recipe.docx
/home/student/.magic, 15022: Warning: using regular magic file `/etc/magic'
recipe.docx: Zip archive data, at least v2.0 to extract
```

This result, “Zip archive data,” makes sense, because .docx files are a type of zip file.

In a real investigation, you will want to store this file on media where it cannot easily be modified. As you progress in your forensic analysis, routinely store cryptographically hashed copies of important data (carved-out flows or files) in places where you cannot accidentally modify them.

Finally, we have carved out the file transferred! Let's open a copy of it with a document editor and see what's inside. Remember, programs that you use to view a file may also modify it. Always maintain hashed, write-protected copies of the important data you have extracted during analysis.

In Figure 4–27, we can see Anarchy-R-Us's “Recipe for Disaster.” This is their secret recipe! We have found strong evidence that the user of 192.168.1.158 (Ann) transferred this file to someone at 192.168.1.159 through her AIM client.



Figure 4–27. Here is the file we carved out of the OFT transfer, recipe.docx. The contents are shown in OpenOffice.

Whew! Manually carving out that file was a lot of work. Imagine if you had a much bigger packet capture with *lots* of file transfers. This manual method is good for understanding the underlying system, but in practice it really doesn't scale.

Let's try using tcpextract to automatically carve the transferred file out for us:

```
$ tcpextract -f evidence01.pcap
...
Found file of type "zip" in session [205.188.13.12:47873 ->
 192.168.1.159:63236], exporting to 00000022.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000023.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000024.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000025.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000026.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000027.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000028.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000029.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000030.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000031.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000032.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000033.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000034.zip
Found file of type "zip" in session [192.168.1.158:17940 ->
 192.168.1.159:63492], exporting to 00000035.zip
```

```
Found file of type "zip" in session [192.168.1.158:17940 ->
192.168.1.159:63492], exporting to 00000036.zip

$ ls -l
...
-rwx----- 1 student student 12020 2011-01-08 11:22 00000023.zip
-rwx----- 1 student student 11068 2011-01-08 11:22 00000024.zip
-rwx----- 1 student student 10264 2011-01-08 11:22 00000025.zip
-rwx----- 1 student student 9670 2011-01-08 11:22 00000026.zip
-rwx----- 1 student student 8775 2011-01-08 11:22 00000027.zip
-rwx----- 1 student student 7038 2011-01-08 11:22 00000028.zip
-rwx----- 1 student student 6209 2011-01-08 11:22 00000029.zip
-rwx----- 1 student student 5691 2011-01-08 11:22 00000030.zip
-rwx----- 1 student student 5380 2011-01-08 11:22 00000031.zip
-rwx----- 1 student student 3485 2011-01-08 11:22 00000032.zip
-rwx----- 1 student student 2807 2011-01-08 11:22 00000033.zip
-rwx----- 1 student student 2585 2011-01-08 11:22 00000034.zip
-rwx----- 1 student student 1974 2011-01-08 11:22 00000035.zip
-rwx----- 1 student student 1737 2011-01-08 11:22 00000036.zip
```

If you search the flow of interest for the hex string “0x504B0304,” which tcpextract uses by default to mark the beginning of a zip file, you will see that this string was actually present 14 times within the flow. Only one of these instances corresponds with the “0x504B0304” that marks the beginning of our file transfer for recipe.docx. This was the first instance, which tcpextract carved out as file “00000023.zip.”

Let’s try saving “00000023.zip” as “recipe-tcpextract.docx” and see what happens when we open it in a document editor. Figure 4–28 shows the .docx file carved out by tcpextract, which indeed appears to contain the same content as the recipe.docx file that we carved out manually earlier.

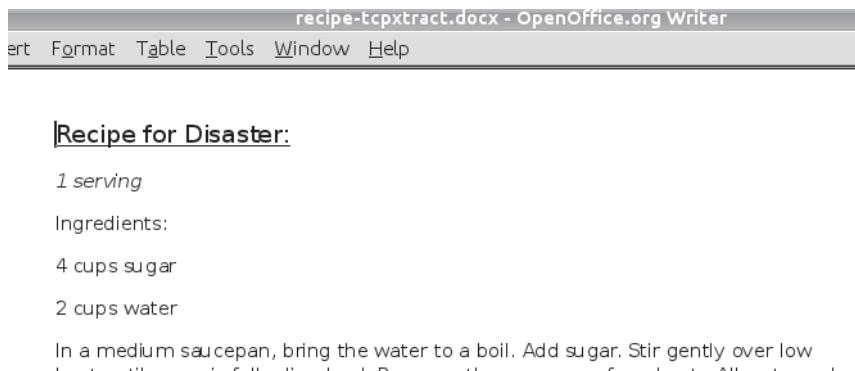


Figure 4–28. Here is a file we carved out of the OFT transfer using tcpextract. The contents are shown in OpenOffice. Note that the contents appear to be the same as the file we carved out earlier manually—but as we will see, they are not quite the same.

But when we take the cryptographic hashes, we see immediately that this file is *not* the same as the one we carved out earlier:

```
$ md5sum recipe-tcpxtract.docx  
a217badfdb530bd55d1dbd2280cb3e2b  recipe-tcpxtract.docx  
  
$ sha256sum recipe-tcpxtract.docx  
3472a1720544098caf9872401932d10a021dfc56ea34cad5db0f80c92080ba82  recipe-  
tcpxtract.docx
```

In fact, the file size is slightly different as well. Here we see that the file carved out by `tcpxtract` is 12,020 bytes, whereas the one we carved earlier was only 12,008 bytes.

```
$ ls -l recipe-tcpextract.docx
-rwx----- 1 student student 12020 2011-01-08 11:58 recipe-tcpextract.docx
```

Which is correct, our manual extraction or tcpxtract's automated extraction?

Let's open both the files up in a hex editor, as shown in Figure 4-29. We can see visually that the beginning of the files are the same, but the file carved by tcpxtract has 12 bytes of extra data at the end. This makes sense, since we know that recipe-tcpxtract.docx was 12,020 bytes in size, compared to recipe.docx, which was only 12,008 bytes in size.

Given that the highest-layer protocol, OFT, reported the expected file size as “0x00002EE8,” or 12,008 bytes, it’s likely that the file we carved out manually is correct. Why, then, did `tcpextract` pad the file with 12 bytes of mysterious trailing nulls?

Carefully examining the relevant TCP stream in Wireshark, we can see that after the end of the file transfer, 192.168.1.158 sent two additional TCP packets to 192.168.1.159 (#134 and #139). Neither of these packets contained any data within the TCP payload. In packet #139 of the evidence file (shown in Figure 4-30), you can see that the “Total Length” of the IP packet is 40 bytes. The IP “Header length” is 20 bytes, and the TCP “Header length” is 20 bytes. Adding up the IP and TCP header lengths, we find that the total length of the IP + TCP headers is 40 bytes. Since that is also the total length of the whole IP packet, there must be zero data in the TCP payload. You can confirm this by looking in the Packet Details window, where the TCP protocol is marked as “Len:0,” to indicate zero payload length.

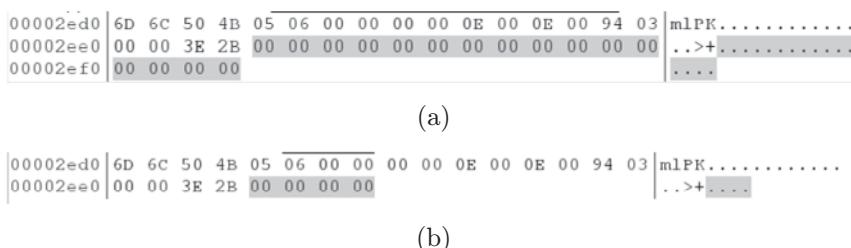


Figure 4–29. This screenshot shows the ends of the two files that we carved out. The top file (a) was carved by `tcpxtract`, and the bottom file (b) was carved manually. Notice that the file carved by `tcpxtract` had 16 nulls at the end, but the manually carved file only had 4.

The minimum payload size of an Ethernet frame is 46 bytes.³⁹ Consequently, the network device driver padded the Ethernet frame's payload with 6 null bytes, since TCP/IP only took up 40 bytes total. You can see these 6 null bytes at the end of Wireshark's Packet Bytes frame in Figure 4–30. Notice how they occur immediately after the TCP header. These days, TCP options are so prevalent that minimum-length 20-byte TCP headers are not as common as they used to be. As a result, padded Ethernet frames are relatively rare.

There are two padded packets in this flow sent from 192.168.1.158 to 192.168.1.159 after the end of the file transfer, which contain a total of 12 null bytes of Ethernet padding. Note that if no footer magic number is present, tcpxtract will continue to carve TCP data until the end of the flow. It is likely that tcpxtract incorrectly included the 6 null bytes of padding from each of these two packets as part of the TCP stream because rather than properly calculating the size of each TCP payload, it simply grabbed all bytes after the TCP header and mistakenly included them as part of the TCP stream.

The moral of the story: Trust, but verify.

Filter: tcp.stream eq 7						Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Info			
133	61.270706	192.168.1.159	192.168.1.158	TCP	cspmlockmgr > aol [PSH, ACK] Seq=4020906270 Ack=4020906271			
134	61.270711	192.168.1.158	192.168.1.159	TCP	aol > cspmlockmgr [ACK] Seq=3626218418 Ack=4020906271			
135	61.296969	192.168.1.159	192.168.1.158	TCP	cspmlockmgr > aol [FIN, ACK] Seq=4020906526 Ack=3626218418			
139	61.337771	192.168.1.158	192.168.1.159	TCP	aol > cspmlockmgr [ACK] Seq=3626218418 Ack=4020906527			
Frame 139 (60 bytes on wire, 60 bytes captured)								
Ethernet II, Src: Hewlett_45:a4:bb (00:12:79:45:a4:bb), Dst: Dell_4d:4f:ae (00:21:70:4d:4f:ae)								
Internet Protocol Version: 4 Header length: 20 bytes								
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)								
Total Length: 40 Identification: 0x7225 (29221)								
Flags: 0x02 (Don't Fragment) Fragment offset: 0								
Time to live: 64 Protocol: TCP (0x06)								
Header checksum: 0x4418 [correct] Source: 192.168.1.158 (192.168.1.158)								
Destination: 192.168.1.159 (192.168.1.159)								
Transmission Control Protocol, Src Port: aol (5190), Dst Port: cspmlockmgr (1272), Seq: 3626218418, Ack: 4020906527, Len: 0								
Source port: aol (5190) Destination port: cspmlockmgr (1272) [Stream index: ?]								
Sequence number: 3626218418 Acknowledgement number: 4020906527 Header length: 20 bytes								
Flags: 0x10 (ACK) Window size: 7504 Checksum: 0x4f18 [validation disabled]								
[SEQ/ACK analysis]								
0000 00 21 70 dd 4f aa 00 12 79 45 aa bb 08 00 45 00 .1pMO... yE...E. 0010 00 28 72 25 40 00 40 06 44 1d c0 a8 01 9e c0 a8 .(F%0,0, D..... 0020 01 0f 14 46 04 18 d8 23 b3 b2 e7 aa 2a 1f 50 16 ...F...#P. 0030 1d 50 4f 18 00 00 00 00 00 00 00 00 00 00 ..P0.....								

Figure 4–30. After the end of the file transfer, 192.168.1.158 sent two additional TCP packets to 192.168.1.159 (#134 and #139). Neither of these packets contains any data within the TCP payload. Each packet is padded with six nulls at the end by the network card, to reach the minimum size for an Ethernet frame (46 bytes).

39. “IEEE-SA -IEEE Get 802 Program,” <http://standards.ieee.org/about/get/802/802.3.html>.

Etherleak

There was a big flap in 2003 when AtStake released a whitepaper detailing how certain improperly written network device drivers were padding Ethernet frames with bytes from previously handled frames or kernel memory. This resulted in potentially sensitive snippets of network traffic and kernel memory leaking to other hosts on the LAN. This vulnerability, which was found across a wide range of device drives from different sources, was dubbed “Etherleak.”⁴⁰

4.4 Higher-Layer Traffic Analysis

If an investigator has been successful in isolating and reconstructing the payloads of a transport-layer flow, it becomes possible to inspect the higher-layer protocols. Common examples of higher-layer protocols include:

- Hypertext Transfer Protocol (HTTP)
- Simple Mail Transfer Protocol (SMTP)
- Domain Name System (DNS)
- Dynamic Host Configuration Protocol (DHCP)

Of course, there are many, *many* others.

4.4.1 A Few Common Higher-Layer Protocols

In this section, we briefly review important details of a few of the most common higher-layer protocols: HTTP, DHCP, SMTP, and DNS. This background will be helpful throughout the remainder of the book.

4.4.1.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) was originally developed in the early 1990s by Tim Berners-Lee at the European Organization for Nuclear Research (CERN).⁴¹ Berners-Lee wanted to help scientists at CERN share information in a flexible, distributed way. In a 1989 proposal, he wrote:⁴²

CERN is a wonderful organisation. It involves several thousand people, many of them very creative, all working toward common goals. Although they are nominally organised into a hierarchical management structure, this does not

40. “Atstake,” 2003, <http://www.atstake.com/research/advisories/2003/a010603-1.txt>.

41. Joshua Quittner, “Network Designer Tim Berners-Lee,”—*Time*, March 29, 1999, <http://www.time.com/time/magazine/article/0,9171,990627,00.html>.

42. Tim Berners-Lee, “Information Management: A Proposal,” March 1989, <http://www.w3.org/History/1989/proposal.html>.

constrain the way people will communicate, and share information, equipment and software across groups.

The actual observed working structure of the organisation is a multiply connected ‘web’ whose interconnections evolve with time.... [I]nformation is constantly being lost. The introduction of the new people demands a fair amount of their time and that of others before they have any idea of what goes on. The technical details of past projects are sometimes lost forever, or only recovered after a detective investigation in an emergency. Often, the information has been recorded, it just cannot be found.

... It was for these reasons that I first made a small linked information system, not realising that a term had already been coined for the idea: ‘hypertext.’

... ‘Hypertext’ is a term coined in the 1950s by Ted Nelson [...], which has become popular for these systems, although it is used to embrace two different ideas. One idea (which is relevant to this problem) is the concept: ‘Hypertext’: Human-readable information linked together in an unconstrained way.

Along with HTTP, Berners-Lee also developed the first web browser and the HyperText Markup Language (HTML), a language used by web browsers to exchange and display information.⁴³

HTTP operates according to a request/response model, where the HTTP client sends a request to a remote server, and the server processes the request and sends a response. By default, HTTP servers operate over TCP port 80, although they are often configured to listen on different ports. A Uniform Resource Identifier (URI) is a string used to specify the location of a resource.⁴⁴

HTTP requests and responses are referred to as *messages*. HTTP messages can include a *message header* (which typically contains fields that store metadata about the transaction) and a *message body* (which normally contains content used in the request or response).

The HTTP protocol uses *methods* to perform operations. For example, the “GET” method is used to retrieve a web page and the “POST” method is used to send data to a server.

There are eight methods defined by the RFC (and the protocol is extensible, so custom methods can be defined). Only the “GET” and “HEAD” methods must be supported by a web server; these are designed simply for information retrieval. Other methods are optional. Methods defined by RFC 2616 include:

- **OPTIONS**—Obtain information about communicating with the remote server.
- **GET**—Retrieve the information identified by the URI.
- **HEAD**—Retrieve the information identified by the URI, without returning a message-body. Helpful for URI validation and debugging.

43. Tim Berners-Lee, “HyperText Transfer Protocol,” June 12, 1992, <http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP.html>.

44. R. Fielding et al., “Hypertext Transfer Protocol—HTTP/1.1,” IETF, June 1999, <http://www.ietf.org/rfc/rfc2616.txt>.

- **POST**—Send data to the resource specified by the URI for processing.
- **PUT**—Upload information to be stored under the specified URI.
- **DELETE**—Delete the resource specified by the URI.
- **TRACE**—Echo a request message back to the client. Helpful for debugging (can also facilitate cross-site scripting attacks).
- **CONNECT**—Reserved “for use with a proxy that can dynamically switch to being a tunnel.”⁴⁵

The HTTP response sent from the server includes a three-digit *status code*, which indicates the status of the request, and *reason phrase*, which provides a human-readable description of the status. Common status codes include 200 (“OK”), which means that the request was successfully processed, and 404 (“Not Found”), which means that the URI requested was not found on the server.

There are five categories for status codes, organized by the first digit:⁴⁶

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

Designed to run on top of a reliable, connection-oriented protocol such as TCP, HTTP does not include built-in mechanisms to track session state. Over time, software engineers have developed ways to track the state of HTTP sessions, such as *cookies*, which are small files on the client system that maintain data related to an HTTP session.

For more details about web page caching and web proxies, please see Chapter 10.

4.4.1.2 Dynamic Host Configuration Protocol (DHCP)

To communicate over modern Ethernet networks, every computer has a network card, which is assigned a 6-byte MAC address by the manufacturer. In order to take advantage of IP-based routing, each computer also needs to be assigned a 32- or 128-bit IP address (for IPv4 or IPv6, respectively). Once upon a time, network administrators configured each computer manually with a *static* IP address that did not change. As large-scale deployments and mobile devices became increasingly common, it became clear that there was a need to *dynamically* assign IP addresses so that computers could be deployed and moved throughout the network without requiring manual IP address configuration changes.

45. “Hypertext Transfer Protocol—HTTP/1.1.”

46. *Ibid.*

IEEE 802.3 Frame Header																
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Bytes	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	1 0 1 0 1 0 1 0 1 0 [...] Preamble (8 bytes)														[...] 1 0 1 0 1 0 1 0 1 1	
0x00	Destination Address (Organizationally Unique ID)								DA (NIC)							
0x04	Destination Address (NIC), cont.								Sending Address (OUI)							
0x08	SA (OUI), cont.								Sending Address (Network Interface Card)							
0x0C	Length								Frame Body							

Figure 4–31. Every network card on an Ethernet network has a 6-byte MAC address, assigned by the manufacturer. Manufacturers have their own 3-byte registered identification number (known as an OUI), which they set as the higher-order 3 bytes of the MAC address.

These days, most organizations use Dynamic Host Configuration Protocol (DHCP) to dynamically assign IP addresses to network cards, rather than configuring static IP addresses on each system. DHCP is a Layer 7 protocol that facilitates automatic configuration of network details, including IP address, gateway, DNS servers, and more. There are separate versions of DHCP for IPv4⁴⁷ and IPv6.⁴⁸ DHCP is designed to operate over UDP, on ports 67 and 68.

For forensic investigators, DHCP records are often the “glue” between a computer’s traffic and the physical realm. DHCP server logs and DHCP communications in packet captures can contain valuable information, including the client MAC address (which can provide clues to the hardware manufacturer), client hostname, routing information, and more.

MAC Addresses As discussed, every network card on an Ethernet network has a 6-byte MAC address, assigned by the manufacturer. Ethernet frames include a 6-byte field for the destination MAC address and another for the source MAC address. Computers on the local subnet typically listen for Ethernet frames destined for their network card’s MAC address (or the broadcast MAC address, ff:ff:ff:ff:ff:ff).

Each manufacturer has their own 3-byte registered identification number, known as an Organizational Unique Identifier (OUI), which they set as the high-order 3 bytes of the MAC address on each card they manufacture, as shown in Figure 4–31. When you encounter a MAC address, you can look up the OUI to determine the likely manufacturer of the network card. For the purposes of forensics, it is often helpful to have evidence that indicates the manufacturer of client hardware. However, the network card’s MAC address can be changed by the end-user from within their operating system, although in practice most people do not.

For example, let’s suppose we have a client with the MAC address “00:12:79:3f:27:e5.” The first 3 bytes represent the manufacturer’s OUI. The IEEE maintains a public listing of OUI assignments. In the IEEE’s public file, the bytes are separated by hyphens rather than colons, so we will search for “00-12-79.” This OUI is assigned to Hewlett-Packard (HP), so

47. R. Droms, “RFC 2131—Dynamic Host Configuration Protocol,” IETF, March 1997, <http://rfc-editor.org/rfc2131.txt>.

48. R. Droms et al., “RFC 3315—Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” IETF, July 2003, <http://rfc-editor.org/rfc3315.txt>.

HP is the likely manufacturer of the network card. Remember, the MAC address can be changed by the end-user, although in practice this is rare.⁴⁹

DHCP When a computer is connected to the network, it broadcasts a DHCP request. The local DHCP server will answer with a unicast reply (at Layer 3). This is much like ARP, where the ARP request is broadcast (at Layer 2) and the reply is unicast. In its response, the DHCP server offers a DHCP lease, which is an IP address assignment for a limited amount of time. That lease includes the IP address assigned, the netmask and gateway address, often DNS server addresses, and always the amount of time that the lease is valid. Most clients will request a lease renewal before the lease expires, thereby maintaining the same IP address for an extended period of time. As a result, in practice DHCP leases are fairly stable, meaning that a computer may be assigned the same IP address for days, weeks, or months. This is not guaranteed, but it is often the case.

When analyzing DHCP traffic, you will frequently see the following exchange:

- **Client:** DHCPDISCOVER (Layer 2 broadcast)
- **Server:** DHCPOFFER
- **Client:** DHCPREQUEST
- **Server:** DHCPACK

This is the process by which a client discovers the DHCP server and receives a DHCP lease. By default, the client will attempt to renew its DHCP lease when half of the lease time has expired.

The following excerpt from the IETF's RFC 2131 "Dynamic Host Configuration Protocol" summarizes the purpose of various DHCP messages:⁵⁰

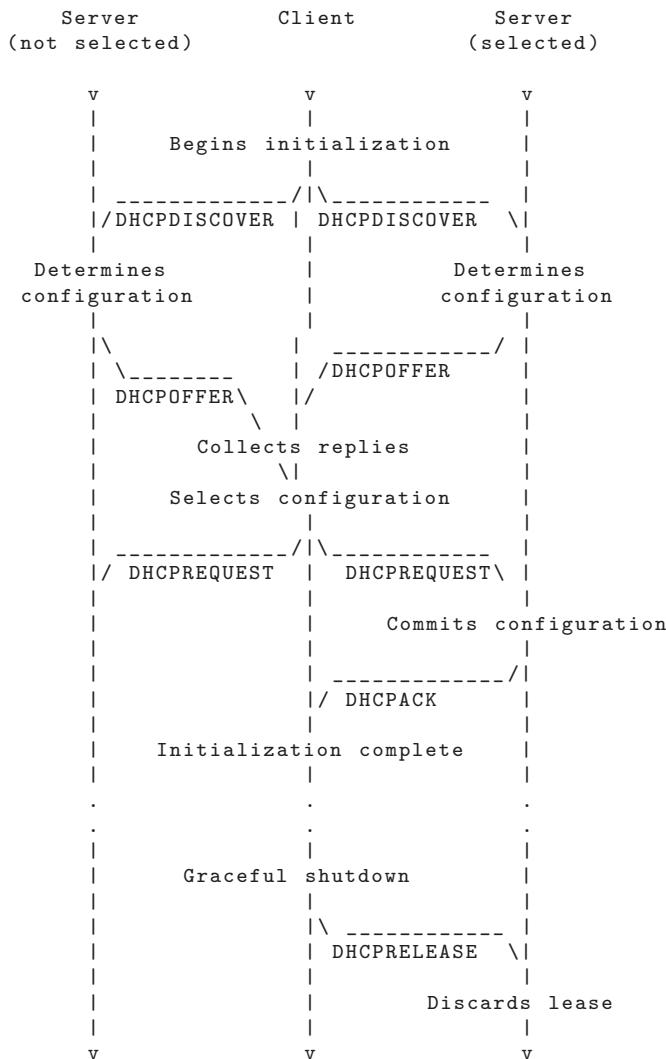
Message	Use
DHCPDISCOVER	- Client broadcast to locate available servers.
DHCPOFFER	- Server to client in response to DHCPDISCOVER with offer of configuration parameters.
DHCPREQUEST	- Client message to servers either (a) requesting offered parameters from one server and implicitly declining offers from all others, (b) confirming correctness of previously allocated address after, e.g., system reboot, or (c) extending the lease on a particular network address.
DHCPACK	- Server to client with configuration parameters, including committed network address.
DCHPNAK	- Server to client indicating client's notion of network address is incorrect (e.g., client has moved to new subnet) or client's lease has expired

49. IEEE, "OUI List," July 7, 2011, <http://standards.ieee.org/develop/regauth/oui/oui.txt>.

50. "RFC 2131—Dynamic Host Configuration Protocol."

- DHCPDECLINE - Client to server indicating network address is already in use.
- DHCPRELEASE - Client to server relinquishing network address and cancelling remaining lease.
- DHCPIINFORM - Client to server, asking only for local configuration parameters; client already has externally configured network address.

The DHCP message exchange for a new network address allocation is illustrated in the table below, from RFC 2131:⁵¹



51. "RFC 2131—Dynamic Host Configuration Protocol."

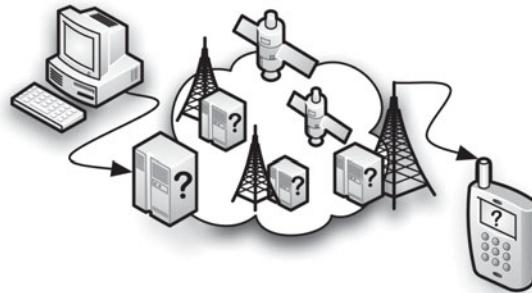


Figure 4–32. As with snail mail, an email message goes through multiple stations on the way to its destination.

4.4.1.3 Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP) was designed as a protocol that clients could speak with servers in order to deliver outbound email, as well as to provide for their delivery hop-by-hop to the end mail transfer agent (MTA). The end MTA either took responsibility for local delivery to another process, or rejected the missive. Originally used on TCP port 25, it is commonly seen on TCP port 587, and used with SMTP Authentication.⁵²

Basic Use As with “snail mail” (see Figure 4–32) email messages go through many different stations on their way between sender and recipient. When you send a physical letter, you might place it in your own mailbox. Then it gets picked up and brought to your local post office, then transferred to a regional post office, then transferred a long distance to another regional post office, then the local post office at the destination town, and finally the recipient’s mailbox. Similarly, when you send an email, the message might travel from your local mail client, to a mail submission agent (MSA) within your organization, to mail transfer agent an MTA, to the mail exchanger (MX) for the destination domain, to the local mail delivery agent (MDA), and finally to the recipient’s mailbox.^{53,54}

SMTP Terminology

Here are definitions for important terms used to describe devices which implement SMTP:

- **Mail User Agent (MUA)**—The end-user’s mail client
- **Mail Submission Agent (MSA)**—Handles local mail submission
- **Mail Transfer Agent (MTA)**—Transfers mail between mail servers
- **Mail eXchanger (MX)**—Accepts incoming messages for a domain
- **Mail Delivery Agent (MDA)**—Handles local mail delivery

52. J. Klensin, “RFC 5321—Simple Mail Transfer Protocol,” *IETF*, October 2008, <http://rfc-editor.org/rfc/rfc5321.txt>.

53. “RFC 5321—Simple Mail Transfer Protocol.”

54. “Simple Mail Transfer Protocol,” *Wikipedia*, July 22, 2011, http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol#Outgoing_mail_SMT_P_server.

Basic commands Each SMTP transaction includes four command sequences:

- **HELO**—Opens the connection
- **MAIL**—Specifies the return address
- **RCPT**—Specifies the recipient address(es)
- **DATA**—The contents of the message. This includes the message header and the message body. The message header contains data such as the “From,” “To,” “Date,” and “Subject” lines, which typically appear in the mail client’s display. The message body contains the actual contents of the message. The message header and message body are separated by an empty line.

At the end of each command sequence, the mail server has the opportunity to accept or reject the values entered by the client.

A simple transcript Here is a simple transcript of an SMTP exchange:

```
$ telnet 10.1.34.114 25
Trying 10.1.34.114...
Connected to 10.1.34.114.
Escape character is '^]'.
220 smtp.lmgsecurity.com
HELO client.example.com
250 smtp.lmgsecurity.com
MAIL FROM: jonathan@lmgsecurity.com
250 2.1.0 Ok
RCPT TO: sherri@lmgsecurity.com
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: Jonathan Ham <jonathan@lmgsecurity.com>
To: Sherri Davidoff <sherri@lmgsecurity.com>
Date: Mon, 06 September 2010 01:20:45 -0700
Subject: nice SMTP example!

That's the best SMTP example I've ever seen! Simply amazing. Two thumbs up.

/jonathan
.
250 2.0.0 Ok: queued as B0F6618C06D
```

SMTP Authentication During the 1990s, email spam became a serious problem. Until then, most mail servers did not require authentication in order to submit messages for relaying, and as a result, spammers worldwide submitted large volumes of messages to be transferred through other people’s mail servers. In the 1990s, the SMTP Authentication standard was first developed to provide a standard protocol for SMTP authentication in order to block illegitimate senders from using third-party mail servers to relay spam. The specification has been improved upon and refined over time. In the example below, note that plain text authentication is used. When this option is selected, the credentials are transmitted as Base64-encoded characters. The string in the example, “dGVzdAB0ZXN0ADEyMzQ=,” decodes as “testtest1234.” (You can use the “base64” command on many Linux systems to decode it.)

Here's a nice example from RFC 4954:⁵⁵

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250-AUTH GSSAPI DIGEST-MD5
S: 250-ENHANCEDSTATUSCODES
S: 250 STARTTLS
C: STARTTLS
S: 220 Ready to start TLS
... TLS negotiation proceeds, further commands
protected by TLS layer ...
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
C: AUTH PLAIN dGVzdABOZXNOADEyMzQ=
S: 235 2.7.0 Authentication successful
```

4.4.1.4 Domain Name System (DNS)

The Domain Name System (DNS) protocol provides a hierarchical distributed database for resolving the names that people prefer to use with the 32-bit IPv4 or 128-bit IPv6 numerical addresses.⁵⁶ For both the original “top-level domains” (TLDs) and the newer country code/vanity TLDs, there are “root” servers established that understand the distribution and delegation of control. Hence querying the “.edu” root server for “computer.mit.edu” will not result in an authoritative answer, but rather a pointer to the delegate nameserver for “mit.edu,” which may know the authoritative answer.

DNS is a query-response protocol. The client typically asks a question within a single UDP packet and the server’s response similarly fits within a single UDP packet. The DNS protocol defines the structure of both the query and response. In the structure, there are fields for many types of data that could be requested and the corresponding answers. Some of these fields can contain arbitrary data, and therefore can be used as the basis for a bidirectional covert channel.

Since the underlying Layer 3 protocol is UDP, it is up to the tunneled protocols to provide reliable delivery, flow control, and so forth. As a consequence, it is common to see TCP traffic tunneled over IP traffic layered over DNS over UDP.

It is possible to route normal DNS traffic over TCP. For example, this commonly occurs when the server’s response to a query is too large to fit within a single UDP packet. In this event, the server, as per the DNS protocol, sets the “truncated” bit in its DNS reply to indicate that the response has been truncated to fit in the UDP-encapsulated reply. Usually the client resubmits the query via a TCP connection so that it can retrieve the entirety of the reply. This often occurs when there is a request for an AXFR record, also known as a DNS “zone transfer.” This query essentially asks the server to tell the client everything it knows about a particular domain. These types of requests provide valuable reconnaissance

55. R. Siemborski et al., “RFC 4954—SMTP Service Extension for Authentication,” *IETF*, July 2007, <http://rfc-editor.org/rfc/rfc4954.txt>.

56. P. Mockapetris, “RFC 1035—Domain Names—Implementation and Specification,” *IETF*, November 1987, <http://rfc-editor.org/rfc/rfc1035.txt>.

information for attackers and are often a precursor to attack. As a result, DNS over TCP port 53 is often blocked at the perimeter or monitored. Therefore, attackers seeking to use DNS for a covert channel are likely to stick with UDP.⁵⁷

DNS Recursion DNS queries that seek to map names to IP addresses (which is the most common case) start out with a client making a query of the server it considers to be most knowledgeable. That server then crawls the hierarchical space for the answer, returns it, and then also caches the result to expedite any subsequent client queries for the same information. To recurse from the root of “.edu” to an authoritative answer about, say, computer.mit.edu, a number of queries might be required.

DNS Queries It can be extremely important to collect name-to-address mapping. This can be accomplished by querying any number of DNS servers, both internal and external. The result will be a snapshot of answers being offered at the time, based on caching parameters. The simplest request can take the form of a command-line query as such:

```
$ dig www.google.com
```

This will result in an address record (A) query from the default nameserver. Specific nameservers can be queried with dig as follows:

```
$ dig @ns.google.com www.google.com
```

Reverse “PTR” records and other records such as nameserver delegations can be obtained as well:

```
$ dig @ns.modwest.com 204.11.246.86 PTR
$ dig lmgsecurity.com NS
```

4.4.2 Higher-Layer Analysis Tools

There are a wide variety of tools available that interpret higher-layer protocols and automatically print important details, carve out files, decode data, or even produce professional-quality forensic reports. Some of these tools are highly specialized for use with one specific higher-layer protocol, whereas others are multipurpose.

In this section, we review a few examples of tools that are designed for higher-layer protocol analysis.

4.4.2.1 oftcat

Written by Kristinn Guðjónsson, oftcat is a dedicated OFT protocol dissector and data-carving tool. It expects as input a data file that contains a single flow’s reassembled transport-layer payload, such as tcpflow or pcapcat, might produce. As output, oftcat produces a protocol summary of all of the OFT activity it can identify. It also recovers any files transferred within the data stream, and saves them as files with their correct names based on the OFT metadata.⁵⁸ You can see this tool in action in Section 4.4.3.1.

57. J. Postel, “RFC 792—Internet Control Message Protocol,” September 1981, <http://www.rfc-editor.org/rfc/rfc792.txt>.

58. Kristinn Guðjónsson, “oftcat,” August 18, 2009, <http://blog.kiddaland.net/dw/oftcat>.

4.4.2.2 smtpdump

The smtpdump tool was written by Franck Guénichot, the co-winner of Puzzle #2, “Ann Skips Bail.”⁵⁹ Smtpdump’s “help” display speaks for itself:

```
$ smtpdump
smtpdump version 0.1,
Copyright (C) 2009 Franck GUENICHOT
smtpdump comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome
to redistribute it under certain conditions.
(GPL v3)

Usage: smtpdump [options] -r <pcap_file>
-A, --auth                         Display SMTP Auth informations (only
    LOGIN method)
-e, --info                           Display E-mail informations
-b, --brief                          Display minimum e-mail informations
-x, --xtract                         Extract e-mail attachments

-m, --md5                            Display extracted attachment MD5 Hash
-s, --save                           Save raw e-mail to file
-f, --flow-index <index>           Filters only given index flow
-r, --read <pcap_file>              Read the given pcap file [REQUIRED]
-v, --version                         Display version information
-h, --help                            Display this screen
```

4.4.2.3 findsmtipinfo.py

Jeremy Rossi, co-winner of Puzzle Contest #2 (“Ann Skips Bail”),⁶⁰ created an SMTP analysis tool that is absolutely simple to use. You specify the pcap file, and it automatically extracts authentication data, decodes Base64-encoded credentials, prints mail header info, extracts each attachment, takes the MD5sum, and produces a report. The report is detailed and well suited for the appendix of a forensic examiner’s findings report.

Here’s a description of findsmtipinfo.py written by Jeremy Rossi for the contest:⁶¹

```
The initial idea was to write the entire process in Python, but after
starting to write the code, I found that tcpflow can handle parsing the
pcap the Python code can be used to present the data and analyze the
output. I called the Python script findsmtipinfo.py.
The script creates a report of the SMTP information, stores any emails in msg
format, stores any attachments from the emails, decompresses them if they
are a compressed format (zip, docx), checks MD5 hashes of all files
including the msg files (and generates MD5 hash of output report).
```

59. Franck Guénichot, “Index of /contest02/Finalists/Franck_Guenichot,” December 17, 2009, http://forensicscontest.com/contest02/Finalists/Franck_Guenichot/.

60. Jeremy Rossi, “Index of /contest02/Finalists/Jeremy_Rossi,” December 17, 2009, http://forensicscontest.com/contest02/Finalists/Jeremy_Rossi/.

61. *Ibid.*

```
So the Python script makes use of three arguments:  
-p|--pcap      This argument specifies the pcap file  
-r|--report    This argument specifies the report output directory [Defaults  
                to ./report]  
-f|--force     If the report directory has files already, this argument is  
                required to allow overwriting of files  
  
findsmtpinfo.py -p evidence02.pcap -r ./report
```

4.4.2.4 NetworkMiner

NetworkMiner, by Erik Hjelmvik, is a fantastic multipurpose traffic analysis tool. It runs on Windows, and has both an intuitive GUI interface and a powerful protocol analysis engine. There is a free, open-source edition of NetworkMiner. You can download it from <http://networkminer.sourceforge.net/>.⁶²

We explore NetworkMiner in more detail in Section 4.4.3.2.

4.4.3 Higher-Layer Analysis Techniques

There are two general strategies to consider when conducting automated analysis of higher-layer protocols:

- Use small, specialized tools that are each specifically designed to analyze a particular higher-layer protocol; or
- Use a multipurpose tool to quickly gather a wide range of information about the traffic.

Each strategy comes with benefits and tradeoffs, as we will discuss momentarily.

4.4.3.1 Small, Specialized Tools

As we have seen, tools such as oftcat and smtpdump are specifically designed for analysis of one higher-layer protocol. They are small and have a limited range of options, but provide more support for analyzing their targeted protocols than most if not all multipurpose tools.

Small and specialized tools are very useful when you have a clear understanding of the higher-layer protocol contained within the packet capture. They are also usually designed to interface easily with other tools so that output can be redirected into another program for further processing.

Let's see what happens when we use "oftcat" to analyze the OFT file transfer in "Ann's Bad AIM." Recall that carving out the transferred file earlier was a painstaking manual process and our attempt to use tcpxtract was unfortunately thwarted due to erroneous TCP stream reconstruction. Perhaps a small tool, specifically designed to analyze the OFT protocol, will help us carve out the transferred file correctly and efficiently.

62. Erik Hjelmvik, "NETRESEC NetworkMiner—The Network Forensics Analysis Tool," 2011, <http://www.netresec.com/?page=NetworkMiner>.

Here we are using oftcat to interpret the half-duplex flow from 192.168.1.158 to 192.168.1.159, which tcpflow carved out:

```
$ oftcat -r 192.168.001.158.05190-192.168.001.159.01272
-----
File name: 192.168.001.158.05190-192.168.001.159.01272

-----
Parsing OFT (Oscar File Transfer) header

Name of file transferred:
Total number of files 1
Files left: 1
Total parts: 1
Parts left: 1
Total size: 12008
Size: 12008
Checksum: 2976120832
ID string 'Cool FileXfer'
Type: PEER_TYPE_GETFILE_RECEIVELISTING, PEER_TYPE_RESUMEACK,
      PEER_TYPE_RESUME, PEER_TYPE_GETFILE_REQUESTLISTING,
      PEER_TYPE_RESUMEACCEPT, PEER_TYPE_GETFILE_ACKLISTING, PEER_TYPE_PROMPT,
Name offset 0
-----
parsing file information
Final header (after file transfer)

File: recipe.docx saved in file recipe.docx
```

As you can see, with the press of a button oftcat automatically identified an OFT message, printed important OFT protocol details to the screen, and carved out the transferred file (“recipe.docx”). We can subsequently check the file size and take the cryptographic hashes:

```
$ sha256sum recipe.docx
f0f74a982a814640aedaa5fd6542ac810e8c5e257552bcc024a5c808343bccf9 recipe.docx

$ md5sum recipe.docx
8350582774e1d4dbe1d61d64c89e0ea1 recipe.docx

$ ls -l recipe.docx
-rwx----- 1 student student 12008 2011-01-08 23:42 recipe.docx
```

The cryptographic hashes and the file size (12,008 bytes) match the values we found earlier when we carved the file out by hand. Notice that the file size of “recipe.docx” (12,008 bytes) also matches the values reported by oftcat as the “Total Size” (“12,008”).

Carving the file out with oftcat was much easier than doing so manually, and it produced the same result. Imagine if you had a packet capture with lots of OFT file transfers! You would want a reliable tool that automates the job, such as oftcat.

4.4.3.2 Multipurpose Tool

Multipurpose tools such as NetworkMiner include support for a wide variety of protocols, including higher-layer protocols such as SMTP, AIM, and others. Multipurpose tools are

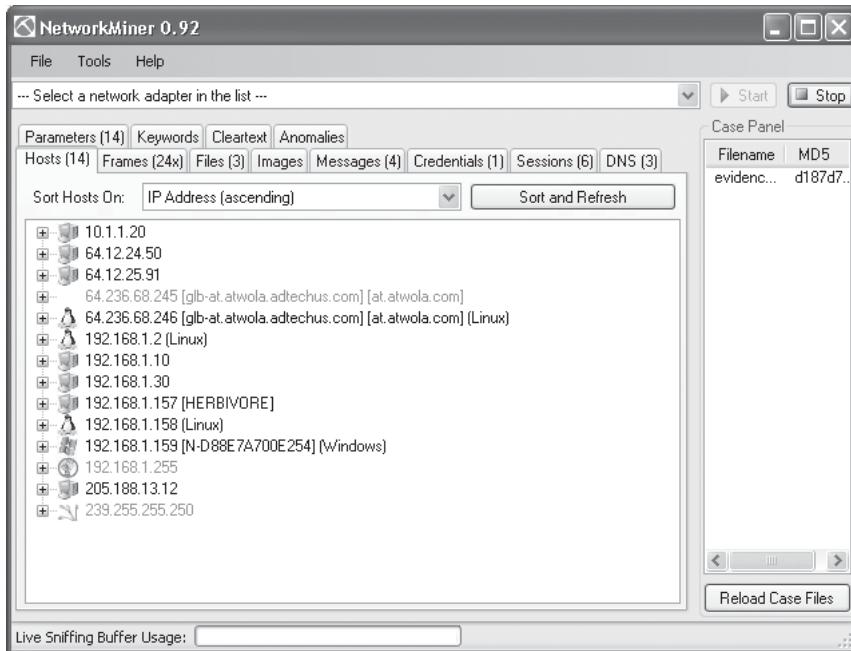


Figure 4–33. NetworkMiner with the “Ann’s Bad AIM” evidence01.pcap loaded.

especially useful when you want to gather a wide variety of information from a packet capture, potentially relating to multiple higher-layer protocols, or if you are not sure exactly what you are looking for. Multipurpose tools are designed to provide a broad spectrum of information about a packet capture.

As an example, let’s use NetworkMiner to analyze the Ann’s Bad AIM evidence file. In Figure 4–33, you can see a screenshot of NetworkMiner in action. We simply dragged and dropped the evidence file (evidence01.pcap) into NetworkMiner and it automatically interpreted the results. The first tab, “Hosts,” lists all of the IP addresses of the hosts that transmitted IP traffic in the packet capture.

In NetworkMiner’s “Files” tab (shown in Figure 4–34), you can see all the files that NetworkMiner carved out of the packet capture, including the “recipe.docx” file. By right-clicking on the filename, you can open it up and view the contents or save it to a folder.

Easy! While a multipurpose tool such as NetworkMiner may not always support the specific higher-layer protocol of interest to you, it is often a good place to start.

4.5 Conclusion

Packet analysis is a complex art. From identifying protocols within packets, to isolating packets of interest, to reconstructing flows and carving out higher-layer protocol data, mastering packet analysis takes patience and keen attention to detail. Many tools exist for packet

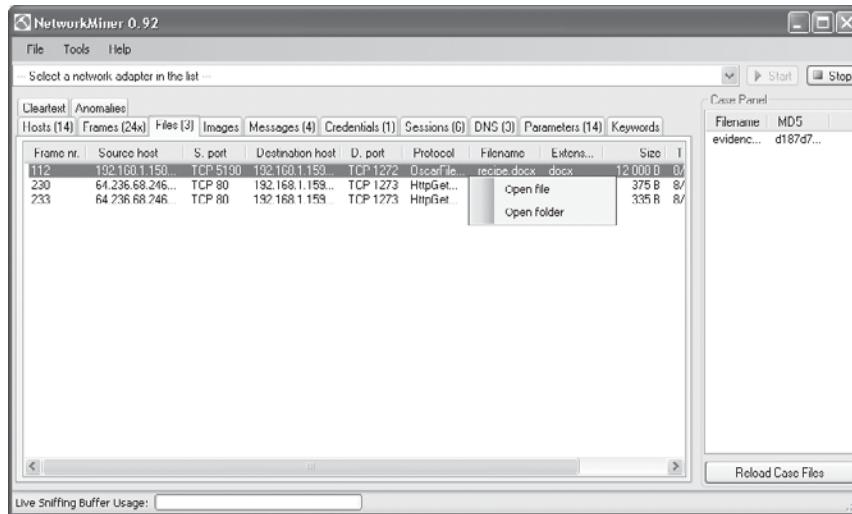


Figure 4–34. NetworkMiner has already extracted recipe.docx.

analysis, but they are not always accurate, especially because the underlying network protocols and specifications are constantly changing.

In this chapter, we began by studying network protocols. We discussed where protocols are defined and documented, with the caveat that there is no guarantee that what is on paper matches reality! We introduced a variety of protocol and packet analysis tools and took an in-depth look at a few important higher-layer protocols, which we will refer to throughout the book. Although our focus is always on teaching you the low-level techniques so that you can understand what your tools are doing, we also introduced multipurpose packet analysis tools, which are very powerful and popular.

4.6 Case Study: Ann's Rendezvous

The Case: *After being released on bail, Ann Dercover disappears! Fortunately, investigators were carefully monitoring her network activity before she skipped town. “We believe Ann may have communicated with her secret lover, Mr. X, before she left,” says the police chief. “The packet capture may contain clues to her whereabouts.”*

Challenge: You are the forensic investigator. Your mission is to analyze the packet capture and gather information about Ann’s activities and plans.

The following questions will help guide your investigation:

- Provide any online aliases or addresses and corresponding account credentials that may be used by the suspect under investigation.
- Who did Ann communicate with? Provide a list of email addresses and any other identifying information.
- Extract any transcripts of Ann’s conversations and present them to investigators.
- If Ann transferred or received any files of interest, recover them.
- Are there any indications of Ann’s physical whereabouts? If so, provide supporting evidence.

Network:

- Internal network: 192.168.30.0/24
- DMZ: 10.30.30.0/24
- The “Internet”: 172.30.1.0/24 [Note that for the purposes of this case study, we are treating the 172.30.1.0/24 subnet as “the Internet.” In real life, this is a reserved non-routable IP address space.]

Evidence: Investigators provide you with a packet capture from Ann’s home network, “evidence-packet-analysis.pcap.” They also inform you that in the course of their monitoring, they have found that Ann’s laptop has the MAC address 00:21:70:4D:4F:AE.

4.6.1 Analysis: Protocol Summary

There are many possible ways to approach this investigation. Let’s begin by taking a quick high-level look at the protocols contained in this packet capture. Using Wireshark, we open the packet capture and go to Statistics→Protocol Hierarchy. As you can see in Figure 4–35, 100% of the traffic in the evidence packet capture is Ethernet (at Layer 2) and IP at Layer 3.

Note the presence of “Bootstrap Protocol,” which is used to transfer DHCP requests and responses. This might allow us to easily link the given MAC address to an assigned IP

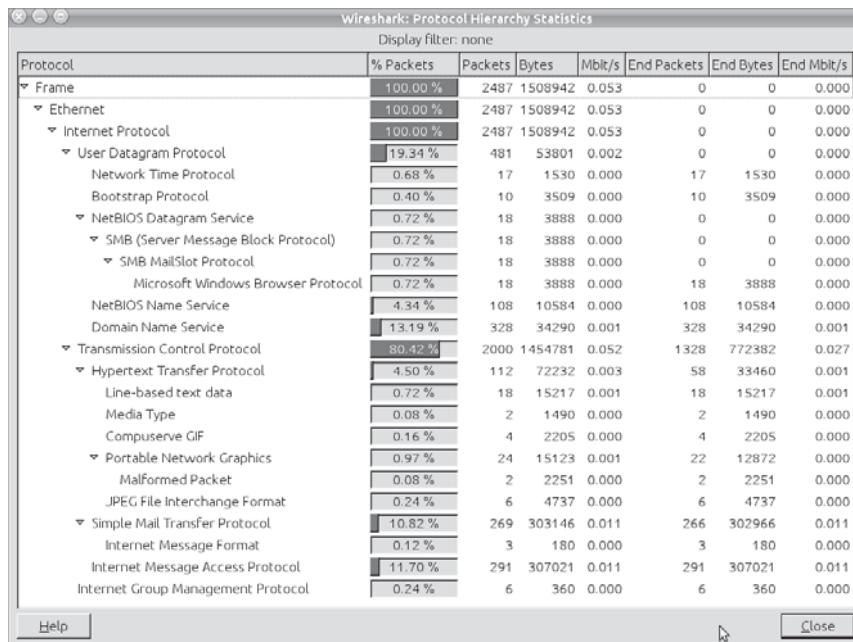


Figure 4–35. This screenshot provides a high-level look at the protocols contained in this packet capture, using Wireshark’s Protocol Hierarchy function.

address and other network configuration information. Since we are looking for communications protocols, it is also worth pointing out that the most voluminous higher-layer protocols used in the packet capture are DNS (13.19% of the packet capture), SMTP (10.82% of the packet capture), and IMAP (11.70% of the packet capture). These three protocols are commonly seen together when email is exchanged over a network.

4.6.2 DHCP Traffic

Now let’s examine the “Bootstrap Protocol” traffic. Using the Wireshark Display Filter “eth.addr == 00:21:70:4d:4f:ae and bootp,” we can narrow down the packet capture so that we are only viewing “Bootstrap Protocol” traffic that relates to the MAC address “00:21:70:4d:4f:ae,” which we refer to as “Ann’s computer” based on the given information.

As you can see in Figure 4–36, Wireshark automatically looks up the registered OUI, “00:21:70,” and displays the corresponding manufacturer, Dell. (We can verify this using the IEEE’s published documentation.) The first message shown is a broadcast DHCP Request from Ann’s computer. In this message, we can see that the Requested IP Address is 192.168.30.108, which is likely the address that Ann’s computer was assigned the last time it was on a network. The hostname is listed as “ann-laptop”—nice evidence to corroborate our theory that this system belongs to Ann.

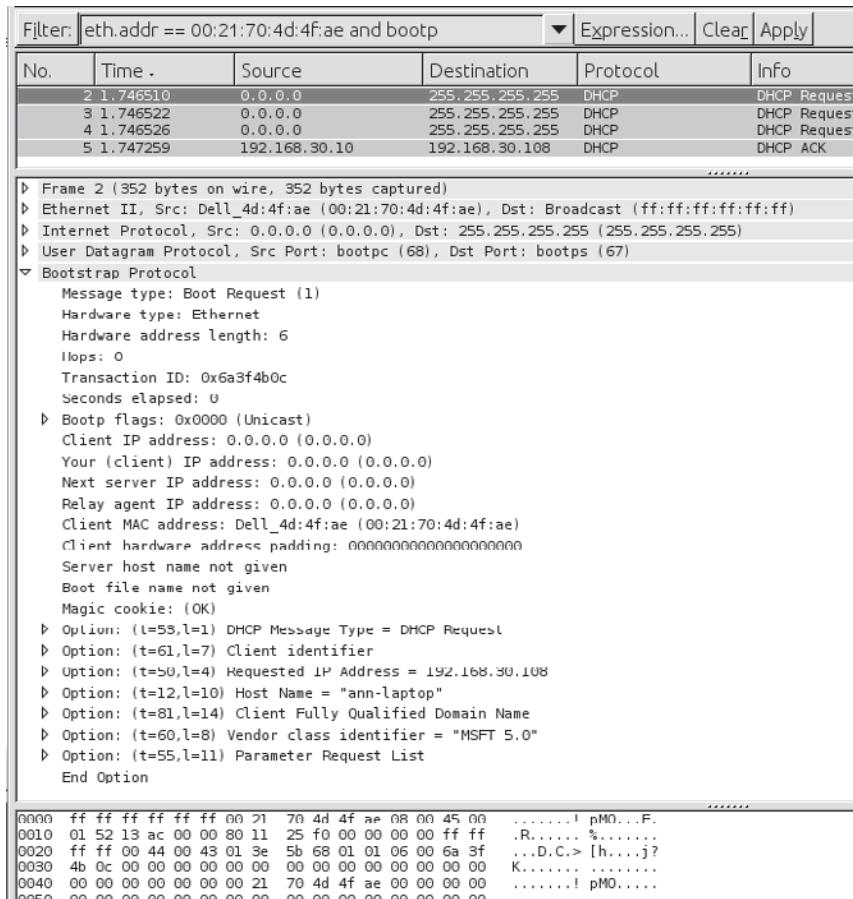


Figure 4–36. A DHCP Request packet from Ann's computer. Notice that Wireshark automatically looks up the registered OUI, “00:21:70,” and displays the corresponding manufacturer, Dell.

After three DHCP Requests, we see a DHCP ACK from 192.168.30.10, as shown in Figure 4–37. Based on the options listed in the DHCP ACK, we can see that the client 00:21:70:4d:4f:ae has indeed been assigned the IP address 192.168.30.108. The DHCP Server Identifier is 192.168.30.10 (which makes sense, given that the DHCP ACK packet originated from 192.168.30.10). The DNS server is listed as 10.30.30.20, so unless Ann's computer has been manually configured to use a different DNS server, we can expect to see her system sending DNS requests to 10.30.30.20. The “Router” is listed as 192.168.30.10, and the Subnet Mask is 255.255.255.0, so again, unless Ann's computer has been manually configured to use a different gateway, her system will send traffic destined for outside the local subnet to 192.168.30.10. The IP Address Lease Time is set to 1 hour. According to RFC we should

Filter: eth.addr == 00:21:70:4d:4f:ae and bootp				Expression...	Clear	Apply
No.	Time .	Source	Destination	Protocol	Info	
2	1.746510	0.0.0.0	255.255.255.255	DHCP	DHCP Request	
3	1.746522	0.0.0.0	255.255.255.255	DHCP	DHCP Request	
4	1.746526	0.0.0.0	255.255.255.255	DHCP	DHCP Request	
5	1.747259	192.168.30.10	192.168.30.108	DHCP	DHCP ACK
Frame 5 (358 bytes on wire, 358 bytes captured) Ethernet II. Src: Cisco_C4:09:94 (d0:d0:fd:c4:09:94), Dst: Dell_4d:4f:ae (00:21:70:4d:4f:ae) Internet Protocol, Src: 192.168.30.10 (192.168.30.10), Dst: 192.168.30.108 (192.168.30.108) User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68) Bootstrap Protocol						
		Message type: Boot Reply (2)				
		Hardware type: Ethernet				
		Hardware address length: 6				
		Hops: 0				
		Transaction ID: 0x6a3f4b0c				
		Seconds elapsed: 0				
		Boot flags: 0x0000 (Unicast)				
		Client IP address: 0.0.0.0 (0.0.0.0)				
		Your (client) IP address: 192.168.30.108 (192.168.30.108)				
		Next server IP address: 0.0.0.0 (0.0.0.0)				
		Relay agent IP address: 0.0.0.0 (0.0.0.0)				
		Client MAC address: Dell_4d:4f:ae (00:21:70:4d:4f:ae)				
		Client hardware address padding: 00000000000000000000				
		Server host name not given				
		Boot file name not given				
		Magic cookie: (OK)				
		Option: (t=53,l=1) DHCP Message Type = DHCP ACK				
		Option: (t=54,l=4) DHCP Server Identifier = 192.168.30.10				
		Option: (t=51,l=4) IP Address Lease Time = 1 hour				
		Option: (t=58,l=4) Renewal Time Value = 30 minutes				
		Option: (t=59,l=4) Rebinding Time Value = 52 minutes, 30 seconds				
		Option: (t=81,l=25) Client Fully Qualified Domain Name				
		Option: (t=1,l=4) Subnet Mask = 255.255.255.0				
		Option: (t=6,l=4) Domain Name Server = 10.30.30.20				
		Option: (t=46,l=1) NetBIOS over TCP/IP Node Type = H node				
		Option: (t=3,l=4) Router = 192.168.30.10				
		End Option			
0110	00 00 00 00 00 00 63 82 53 63 35 01 05 36 04 c0	c. Sc..,6..				
0120	a8 1e 0a 33 04 00 00 0e 10 3a 04 00 00 07 08 3b ...3.... ;.....;					
0130	04 00 00 0c 4e 51 19 00 00 00 61 6e 6e 2d ec b1NQ.. ..ann-la					
0140	70 74 6f 70 2e 65 78 61 6d 70 6c 65 2e 63 6f 6d ptop.ex ample.com					
0150	01 04 ff ff ff 00 06 04 0a 1e 1e 14 2e 01 08 03				

Figure 4–37. A DHCP ACK packet from the DHCP server to Ann's computer. Based on the options listed in the DHCP ACK, we can see that the client 00:21:70:4d:4f:ae has indeed been assigned the IP address 192.168.30.108.

see a DHCP renewal request after half the lease time has expired, as corroborated by the explicit Renewal Time Value of 30 minutes.

4.6.3 Keyword Search

A keyword search, or “dirty word” search, is often an efficient way to identify traffic of interest within a packet capture. In this case, since we are looking for communications relating to a known suspect, we might start by searching the packet capture for references to her name, “Ann Dercov.” (We could have chosen other keywords to begin, such as simply “Ann,” or other known aliases, but short keywords often turn up a lot of extra matches that are not what the forensic investigator intends. It is generally more effective to begin with longer, unique keywords and to execute additional searches as needed based on

the results.) The “ngrep” example below illustrates one way to search for a keyword within a packet capture.

```
$ ngrep "Ann Dercov" -N -t -q -I evidence-packet-analysis.pcap
input: evidence-packet-analysis.pcap
match: Ann Dercov

T(6) 2011/05/17 13:33:07.203874 192.168.30.108:1684 -> 64.12.168.40:587 [A]
Message-ID: <00ab01cc14c9$227de600$6b1ea8c0@annlaptop>..From: "Ann Dercov"
<sneakyg33ky@aol.com>..To: <interOpt1c@aol.com>..Subject: need a favor..Date: Tue, 17 May 2011 13:32:17 -0600..MIME-Version: 1.0..Content-Type: multipart/alternative;...boundary="----=_NextPart_000_00A8_01CC1496.D700DE30"..
X-Priority: 3..X-MSMail-Priority: Normal..X-Mailer: Microsoft Outlook Express 6.00.2900.2180..X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180....This is a multi-part message in MIME format....----=_NextPart_000_00A8_01CC1496.D700DE30..Content-Type: text/plain;...charset="iso-8859-1"..Content-Transfer-Encoding: quoted-printable....Hey, can you hook me up quick with that fake passport you were taking =..about? - Ann....----=_NextPart_000_00A8_01CC1496.D700DE30..Content-Type: text/html;...charset="iso-8859-1"..
Content-Transfer-Encoding: quoted-printable....<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">..<HTML><HEAD>..<META http-equiv=3DContent-Type content=3D"text/html"; =..charset=3Diso-8859-1>..<META content=3D"MSHTML 6.00.2900.2853" name=3DGENERATOR>..<STYLE></STYLE>..</HEAD>..<BODY bgColor=3D#ffffff>..<DIV><FONT face=3DArial size=3D2>Hey, can you hook me up quick with that =..fake=20..passport you were taking about? - Ann</FONT></DIV>..<DIV><FONT face=3DArial size=3D2></FONT>&nbsp;.</DIV></BODY></HTML>...----=_NextPart_000_00A8_01C

T(6) 2011/05/17 13:33:08.648555 192.168.30.108:1685 -> 205.188.58.10:143 [AP]
From: "Ann Dercov" <sneakyg33ky@aol.com>..To: <interOpt1c@aol.com>..Subject: need a favor..Date: Tue, 17 May 2011 13:32:17 -0600..MIME-Version: 1.0..Content-Type: multipart/alternative;...boundary="----=_NextPart_000_00A8_01CC1496.D700DE30"..
X-Priority: 3..X-MSMail-Priority: Normal..X-Mailer: Microsoft Outlook Express 6.00.2900.2180..X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180....This is a multi-part message in MIME format....----=_NextPart_000_00A8_01CC1496.D700DE30..Content-Type: text/plain;...charset="iso-8859-1"..
Content-Transfer-Encoding: quoted-printable....Hey, can you hook me up quick with that fake passport you were taking =..about? - Ann....----=_NextPart_000_00A8_01CC1496.D700DE30..Content-Type: text/html;...charset="iso-8859-1"..
Content-Transfer-Encoding: quoted-printable....<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">..<HTML><HEAD>..<META http-equiv=3DContent-Type content=3D"text/html"; =..charset=3Diso-8859-1>..<META content=3D"MSHTML 6.00.2900.2853" name=3DGENERATOR>..<STYLE></STYLE>..</HEAD>..<BODY bgColor=3D#ffffff>..<DIV><FONT face=3DArial size=3D2>Hey, can you hook me up quick with that =..fake=20..passport you were taking about? - Ann</FONT></DIV>..<DIV><FONT face=3DArial size=3D2></FONT>&nbsp;.</DIV></BODY></HTML>...----=_NextPart_000_00A8_01CC1496.D700DE30--.

T(6) 2011/05/17 13:33:21.259662 205.188.58.10:143 -> 192.168.30.108:1686 [AP]
* 2 FETCH (UID 272 INTERNALDATE "17-May-2011 15:32:17 -0400" RFC822.SIZE 1342 ENVELOPE ("Tue, 17 May 2011 13:32:17 -0600" "need a favor" ((("Ann Dercov" NIL "sneakyg33ky" "aol.com")) ((("Ann Dercov" NIL "sneakyg33ky" "aol.com")) ((("Ann Dercov" NIL "sneakyg33ky" "aol.com")) ((NIL NIL "interOpt1c" "aol.com")) NIL NIL NIL NIL) BODY[HEADER.FIELDS (REFERENCES X-REF X-PRIORITY X-MSMAIL-PRIORITY X-MSOESREC NEWSGROUPS)] {44}..X-Priority: 3..X-MSMail-Priority: Normal.... FLAGS ($Submitted XAOL-SENT \Seen))..p8sr OK UID FETCH completed..
```

T(6) 2011/05/17 13:34:16.481132 192.168.30.108:1687 -> 64.12.168.40:587 [A]
 Message-ID: <00b701cc14c9\$4bc95710\$6b1ea8c0@annlaptop>..From: "Ann Dercov
 " <sneakyg33ky@aol.com>..To: <d4rktangent@gmail.com>..Subject: lunch next w
 eek..Date: Tue, 17 May 2011 13:33:26 -0600..MIME-Version: 1.0..Content-Type
 : multipart/alternative;...boundary="----=_NextPart_000_00B4_01CC1497.004EC
 040"..X-Priority: 3..X-MSMail-Priority: Normal..X-Mailer: Microsoft Outlook
 Express 6.00.2900.2180..X-MimeOLE: Produced By Microsoft MimeOLE V6.00.290
 0.2180....This is a multi-part message in MIME format.....=_NextPart_000_00B4_01CC1497.004EC040..Content-Type: text/plain;...charset="iso-8859-1"
 ..Content-Transfer-Encoding: quoted-printable....Sorry-- I can't do lunch
 next week after all. Heading out of town. =..Another time!....-Ann.....=_
 _NextPart_000_00B4_01CC1497.004EC040..Content-Type: text/html;...charset="i
 so-8859-1"..Content-Transfer-Encoding: quoted-printable....<!DOCTYPE HTML P
 UBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">..<HTML><HEAD>..<META http-equiv=3DContent-Type content=3D"text/html"; ..charset=3Diso-8859-1">..<META content=3D"MSHTML 6.00.2900.2853" name=3DGENERATOR>..<STYLE></STYLE>..</HEAD>..<BODY bgColor=3D#ffffff>..<DIV>Sorry-- I can
 't do lunch next week =..after all.=20..Heading out of town. Another time!<
 /FONT></DIV>..<DIV> .</DIV>..<DIV><F
 ONT face=3DArial size=3D2>-Ann

T(6) 2011/05/17 13:34:18.050493 192.168.30.108:1688 -> 205.188.58.10:143 [A]
 From: "Ann Dercov" <sneakyg33ky@aol.com>..To: <d4rktangent@gmail.com>..Su
 bject: lunch next week..Date: Tue, 17 May 2011 13:33:26 -0600..MIME-Version
 : 1.0..Content-Type: multipart/alternative;...boundary="----=_NextPart_000_00B4_01CC1497.004EC040"..X-Priority: 3..X-MSMail-Priority: Normal..X-Mailer
 : Microsoft Outlook Express 6.00.2900.2180..X-MimeOLE: Produced By Microsof
 t MimeOLE V6.00.2900.2180....This is a multi-part message in MIME format...
=_NextPart_000_00B4_01CC1497.004EC040..Content-Type: text/plain;...
 charset="iso-8859-1"..Content-Transfer-Encoding: quoted-printable....Sorry--
 I can't do lunch next week after all. Heading out of town. =..Another tim
 e!....-Ann.....=_NextPart_000_00B4_01CC1497.004EC040..Content-Type: text
 /html;...charset="iso-8859-1"..Content-Transfer-Encoding: quoted-printable.
 ...<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">..<HTML><H
 EAD>..<META http-equiv=3DContent-Type content=3D"text/html"; ..charset=3D
 iso-8859-1">..<META content=3D"MSHTML 6.00.2900.2853" name=3DGENERATOR>..<STY
 LE></STYLE>..</HEAD>..<BODY bgColor=3D#ffffff>..<DIV>Sorry-- I can't do lunch next week =..after all.=20..Heading out of t
 own. Another time!</DIV>..<DIV>&n
 bsp;.</DIV>..<DIV>-Ann</DIV></BODY></HTML
 >.....=_NextPart_000_00B4

T(6) 2011/05/17 13:35:16.962873 192.168.30.108:1689 -> 64.12.168.40:587 [A]
 Message-ID: <00bc01cc14c9\$6fd1bc60\$6b1ea8c0@annlaptop>..From: "Ann Dercov
 " <sneakyg33ky@aol.com>..To: <mistersekritx@aol.com>..Subject: rendezvous..
 Date: Tue, 17 May 2011 13:34:26 -0600..MIME-Version: 1.0..Content-Type: mul
 tipart/mixed;...boundary="----=_NextPart_000_00B8_01CC1497.244B3EB0"..X-Pri
 ority: 3..X-MSMail-Priority: Normal..X-Mailer: Microsoft Outlook Express 6.
 00.2900.2180..X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180....T
 his is a multi-part message in MIME format.....=_NextPart_000_00B8_01
 CC1497.244B3EB0..Content-Type: multipart/alternative;...boundary="----=_Nex
 tPart_001_00B9_01CC1497.244B3EB0".....=_NextPart_001_00B9_01CC1497.2
 44B3EB0..Content-Type: text/plain;...charset="iso-8859-1"..Content-Transfer
 -Encoding: quoted-printable....Hi sweetheart! Bring your fake passport and
 a bathing suit. Address =..attached. love, Ann.....=_NextPart_001_00B9_0
 1CC1497.244B3EB0..Content-Type: text/html;...charset="iso-8859-1"..Content-
 Transfer-Encoding: quoted-printable....<!DOCTYPE HTML PUBLIC "-//W3C//DTD H

```
TML 4.0 Transitional//EN">>..<HTML><HEAD>..<META http-equiv=3DContent-Type c
ontent=3D"text/html; =..charset=3Diso-8859-1">..<META content=3D"MSHTML 6.0
0.2900.2853" name=3DGENERATOR>..<STYLE></STYLE>..</HEAD>..<BODY bgColor=3D#
ffffff>..<DIV><FONT face=3DArial size=3D2>Hi sweetheart! Bring your fake pa
ssport =..and a=20..bathing su

T(6) 2011/05/17 13:35:24.092584 192.168.30.108:1690 -> 205.188.58.10:143 [A]
From: "Ann Dercov" <sneakyg33ky@aol.com>..To: <mistersekrit@aol.com>..Su
bject: rendezvous..Date: Tue, 17 May 2011 13:34:26 -0600..MIME-Version: 1.0
..Content-Type: multipart/mixed;...boundary="----=_NextPart_000_00B8_01CC14
97.244B3EB0"..X-Priority: 3..X-MSMail-Priority: Normal..X-Mailer: Microsoft
Outlook Express 6.00.2900.2180..X-MimeOLE: Produced By Microsoft MimeOLE V
6.00.2900.2180....This is a multi-part message in MIME format.-----_N
extPart_000_00B8_01CC1497.244B3EB0..Content-Type: multipart/alternative;...
boundary="----=_NextPart_001_00B9_01CC1497.244B3EB0".....----=_NextPart_
001_00B9_01CC1497.244B3EB0..Content-Type: text/plain;...charset="iso-8859-1
".."Content-Transfer-Encoding: quoted-printable....Hi sweetheart! Bring your
fake passport and a bathing suit. Address =..attached. love, Ann.....----=_
NextPart_001_00B9_01CC1497.244B3EB0..Content-Type: text/html;...charset="is
o-8859-1".."Content-Transfer-Encoding: quoted-printable...<!DOCTYPE HTML PU
BLIC "-//W3C//DTD HTML 4.0 Transitional//EN">..<HTML><HEAD>..<META http-equi
v=3DContent-Type content=3D"text/html; =..charset=3Diso-8859-1">..<META co
ntent=3D"MSHTML 6.00.2900.2853" name=3DGENERATOR>..<STYLE></STYLE>..</HEAD>
..<BODY bgColor=3D#ffffff>..<DIV><FONT face=3DArial size=3D2>Hi sweetheart!
Bring your fake passport =..and a=20..bathing suit. Address attached. love
, Ann</FONT></DIV></BODY></HTM
```

Well that looks interesting! We see from the output above that our keyword search for “Ann Dercov” matched seven packets. Three packets were part of a conversation with 64.12.168.40 over TCP port 587. TCP 587 is assigned by IANA as the “submission” port, commonly used by mail clients to submit messages via SMTP to MSAs.⁶³ Four packets were part of a conversation with 205.188.58.10 over TCP port 143. TCP 143 is assigned by IANA to the Internet Message Access Protocol (IMAP) service, which is used for retrieving email.

Each of the packets matched appears to contain email addresses and content.

4.6.4 SMTP Analysis—Wireshark

Next, let’s examine the SMTP traffic of interest in Wireshark. Beginning with the first packet at 2011/05/17 13:33:07.203874, we can use Wireshark’s “Follow TCP Stream” feature to isolate only the packets that were part of the same TCP flow. Conveniently, Wireshark’s “Follow TCP Stream” feature automatically extracts the Layer 7 packet contents from each packet in the stream, reassembles it, and displays it in an easy-to-read format, as shown in Figure 4–38.

In the TCP stream shown in Figure 4–38, we see an authenticated SMTP conversation between a mail client and an MSA. Here we see that Ann authenticated against her SMTP server using plain text:

```
250-AUTH=XAOL-UAS-MB LOGIN PLAIN
```

63. R. Gellens and J. Klensin, “Message Submission,” IETF, December 1998, <http://www.ietf.org/rfc/rfc2476.txt>.



Figure 4-38. An SMTP conversation containing the first packet matched as part of our ngrep search. Wireshark's “Follow TCP Stream” feature automatically extracts the Layer 7 packet contents from each packet in the stream, reassembles it, and displays it in an easy-to-read format.

That means her credentials are only Base-64-encoded, not encrypted. Let's decode them now.

Here's Ann's username:

```
$ echo "c251YWt5ZzMza3k=" | base64 -d
sneakyg33ky
```

Here's her password:

```
$ echo "czAwcGVyczNrcjF0" | base64 -d
s0opers3kr1t
```

The “MAIL FROM” and “RCPT TO” SMTP messages shown below indicate that the email sender was sneakyg33ky@aol.com, and the recipient was inter0pt1c@aol.com. Note that these values can differ from the “From” and “To” headers that are actually *displayed* in the body of the message. The “From” and “To” headers do *not* actually affect where a MTA routes an email message.

```
MAIL FROM: <sneakyg33ky@aol.com>
250 2.1.0 Ok
RCPT TO: <inter0pt1c@aol.com>
```

Finally, we see the body of the email, as shown after the “DATA” SMTP message. In this section, we can infer that the user sneakyg33ky@aol.com is representing herself as “Ann Dercover.” The “Subject” of this email was “need a favor,” and the message contained in the body was short but intriguing: “Hey, can you hook me up quick with that fake passport you were talking = about? - Ann.” Could our suspect have been planning an unexpected, illicit trip abroad?

```
DATA
354 End data with <CR><LF>.<CR><LF>
Message-ID: <00ab01cc14c9$227de600$6b1ea8c0@annlaptop>
From: "Ann Dercover" <sneakyg33ky@aol.com>
To: <interOpt1c@aol.com>
Subject: need a favor
Date: Tue, 17 May 2011 13:32:17 -0600
MIME-Version: 1.0
Content-Type: multipart/alternative;
.boundary="-----_NextPart_000_00A8_01CC1496.D700DE30"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

This is a multi-part message in MIME format.

-----_NextPart_000_00A8_01CC1496.D700DE30
Content-Type: text/plain;
.charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Hey, can you hook me up quick with that fake passport you were talking =
about? - Ann
...

```

In a similar fashion, we can use Wireshark’s “Follow TCP Stream” function to examine the TCP stream containing the second SMTP packet we matched using ngrep, at 2011/05/17 13:34:16.481132. The results are shown in Figure 4–39. In this conversation, we see that the same user, sneakyg33ky@aol.com, sent an email to d4rktangent@gmail.com, with the “Subject” “lunch next week.” In this email, the author writes:

```
Sorry-- I can't do lunch next week after all. Heading out of town. =
Another time!
```

-Ann

Apparently the sender’s plans have changed!

4.6.5 SMTP Analysis—TCPFlow

Finally, let’s examine the third matching SMTP packet from our ngrep results, created at 2011/05/17 13:35:16.962873. We could do this in the same way as the others, using Wireshark’s “Follow TCP Stream,” but instead let’s practice using command-line tools (which typically scale better).

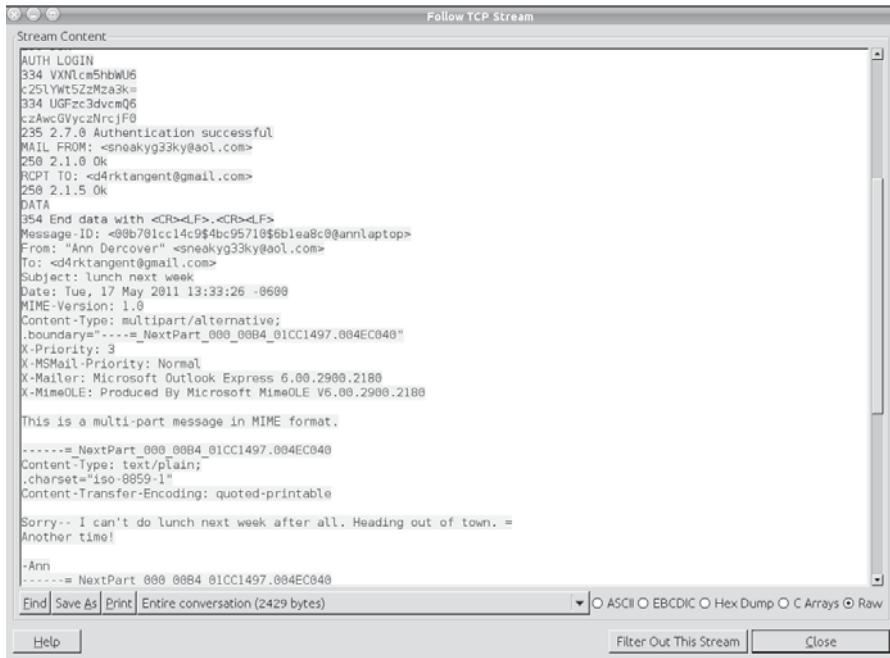


Figure 4–39. An SMTP conversation containing the second packet matched as part of our ngrep search.

Recall from our ngrep output that the matched TCP packet was sent from 192.168.30.108:1689 to 64.12.168.40:587 (the conversation that contained this packet was, of course, bidirectional). In the output below, we use tcpflow with an appropriate BPF filter to list all TCP flows between ports 1689 and 587. As you can see, tcpflow reconstructed two unidirectional data streams, one from 192.168.30.108:1689 to 64.12.168.40, and the other in the opposite direction.

```
$ tcpflow -v -r evidence-packet-analysis.pcap 'port 1689 and port 587'
tcpflow[333]: tcpflow version 0.21 by Jeremy Elson <jelson@circlemud.org>
tcpflow[333]: looking for handler for datalink type 1 for interface evidence-
    packet-analysis.pcap
tcpflow[333]: found max FDs to be 16 using OPEN_MAX
tcpflow[333]: 192.168.030.108.01689-064.012.168.040.00587: new flow
tcpflow[333]: 064.012.168.040.00587-192.168.030.108.01689: new flow
tcpflow[333]: 064.012.168.040.00587-192.168.030.108.01689: opening new output
    file
tcpflow[333]: 192.168.030.108.01689-064.012.168.040.00587: opening new output
    file
```

Let's examine the reconstructed content sent from 192.168.30.108 (Ann's computer) to the remote server 64.12.168.40. If the contents are indeed SMTP data as we expect, then we will be most interested in the outbound content from Ann's computer to the MSA, which

would contain the SMTP “MAIL FROM” and “RCPT TO” headers and authentication data, as well as the body of the email.

Viewing the outbound content reconstructed by tcpflow, we see the following:

```
$ less 192.168.030.108.01689-064.012.168.040.00587
EHLO annlaptop
AUTH LOGIN
c251YWt5ZzMza3k=
czAwcGVyczNrcjF0
MAIL FROM: <sneakyg33ky@aol.com>
RCPT TO: <mistersekritx@aol.com>
DATA
Message-ID: <00bc01cc14c9$6fd1bc60$6b1ea8c0@annlaptop>
From: "Ann Dercovers" <sneakyg33ky@aol.com>
To: <mistersekritx@aol.com>
Subject: rendezvous
Date: Tue, 17 May 2011 13:34:26 -0600
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="====_NextPart_000_00B8_01CC1497.244B3EB0"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

This is a multi-part message in MIME format.

=====_NextPart_000_00B8_01CC1497.244B3EB0
Content-Type: multipart/alternative;
    boundary="====_NextPart_001_00B9_01CC1497.244B3EB0"

=====_NextPart_001_00B9_01CC1497.244B3EB0
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Hi sweetheart! Bring your fake passport and a bathing suit. Address =
attached. love, Ann
=====_NextPart_001_00B9_01CC1497.244B3EB0
Content-Type: text/html;
...
```

In the SMTP conversation shown above, we see the outbound portion of a communication with a remote SMTP server. The “MAIL FROM” and “RCPT TO” SMTP messages shown below indicate that the email sender was sneakyg33ky@aol.com, and the recipient was mistersekritx@aol.com. In the body of the email, we can see that the sender has represented herself once again as “Ann Dercovers.” The “Subject” of this email was “rendezvous.” As we saw above, the message in the body of the email was as follows:

```
Hi sweetheart! Bring your fake passport and a bathing suit. Address =
attached. love, Ann
```

As the message suggests, there is an attachment to the email. Scrolling further in the tcpflow output, we see the following section:

```
-----_NextPart_000_00B8_01CC1497.244B3EB0
Content-Type: application/octet-stream;
    name="secretrendezvous.docx"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
    filename="secretrendezvous.docx"

UEsDBBQABgAIAAAAICht/xGcgEAAFIFAAATAAgCWOVbnR1bnRfVHlwZXNdLnhbtCCiBAIooAAC
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
...
```

Based on the line, “Content-Transfer-Encoding: base64,” it appears that the attachment to the message is Base64-encoded, and the filename of the attachment is “secretrendezvous.docx.”

4.6.6 SMTP Analysis—Attachment File Carving

Now, let’s carve the attachment out of the message body. To do this manually, open the stream dump in the Bless hex editor:

```
$ bless 192.168.030.108.01689-064.012.168.040.00587
```

Cut the SMTP and MIME protocol information off the top and bottom, as shown in Figure 4–40. The email contains multiple parts, as indicated by the mail headers:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="-----_NextPart_000_00B8_01CC1497.244B3EB0"
```

000006f0	65 36 34 0D 0A 43 6F 6E 74 65 6E 74 2D 44 69 73	e64..Content-Di
00000700	70 6F 73 69 74 69 6F 6E 3A 20 61 74 74 61 63 68	position: attach
00000710	6D 65 6E 74 3B 0D 0A 09 66 69 6C 65 6E 61 6D 65	ment;...filename
00000720	3D 22 73 65 63 72 65 74 72 65 6E 64 65 7A 76 6F	= "secretrendezvo
00000730	75 73 2E 64 6E 63 78 22 0D 0A 0D 0A 55 45 73 44	us.docx"....UEsD
00000740	42 42 51 41 42 67 41 49 41 41 41 49 51 43 68	BBQABgAIAAAAICht
00000750	74 2F 78 47 63 67 45 41 41 46 49 46 41 41 41 54	/xGcgEAAFIFAAAT
00000760	41 41 67 43 57 30 4E 76 62 6E 52 6C 62 6E 52 66	AAgCWOVbnR1bnRf
00000770	56 48 6C 77 5A 58 4E 64 4C 6E 68 74 62 43 43 69	VHlwZXNdLnhbtCCi
00000780	42 41 49 6F 41 41 43 0D 0A 41 41 41 41 41 41	BAlOOAAC..AAAAAA

(a)

000450e0	6C 6E 51 42 41 41 44 49 41 67 41 41 45 41 41 41	InQRAADTAGAAKAAA
000450f0	41 41 41 41 41 41 41 41 41 41 41 41 41 43 64	AAAAAAAAAAAAAAACd
00045100	48 41 4D 41 0D 0A 5A 47 39 6A 55 48 4A 76 63 48	HAMN..ZG9jUHJvcCh
00045110	4D 76 59 58 42 77 4C 6E 68 74 62 46 42 4C 42 51	MVYXBwlLnhtbFBtLBQ
00045120	59 41 41 41 41 41 44 41 41 4D 41 41 51 44 41 41	YAAAAADAAAMAODAA
00045130	42 48 48 77 4D 41 41 41 41 3D 0D 0A 0D 0A 2D 2D	BHHWMAAAA-.....
00045140	2D 2D 2D 2D 3D 5F 4E 65 78 74 50 61 72 74 5F 3D	-----_NextPart_0
00045150	30 30 5F 30 30 42 38 5F 30 31 43 43 31 34 39 37	00_00B8_01CC1497
00045160	2E 32 31 34 42 33 45 42 30 2D 2D 0D 0A 0D 0A 2E	.244B3EB0-.....
00045170	0D 0A 51 55 49 54 0D 0A	..QUIT..

(b)

Figure 4–40. This screenshot shows how we carve Ann’s email attachment out of the “Ann Skips Bail” packet capture. The top image (a) shows us cropping the top part of the attachment, and the bottom file (b) shows us cropping the end of the attachment.

The attachment of greatest interest is contained in the part labeled:

```
-----_NextPart_000_00B8_01CC1497.244B3E0
Content-Type: application/octet-stream;
    name="secretrendezvous.docx"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
    filename="secretrendezvous.docx"
```

Immediately after these lines, there are TWO carriage-return/linefeeds (CRLF). (In hexadecimal, a carriage return is “0x0D” and a linefeed is “0x0A.” Please see Section 10.5.2.1 for additional discussion of file carving using CRLF sequences as markers.) At the end of the message part, we see another sequence of TWO CRLFs. To carve out the attachment, we begin carving immediately after the first set of CRLFs and end just before the second set.⁶⁴ Let’s save the carved file as “evidence-packet-analysis-smtp3-attachment.”

The carved attachment contains some line breaks. We need to remove these in order to decode the Base64 encoding. For this purpose, we’ll use the “fromdos” command by Christopher Heng, which is distributed as part of the “tofrodo” Debian package:

```
$ fromdos -b evidence-packet-analysis-smtp3-attachment
```

Now, we can decode the file and recreate the original attachment:

```
$ base64 -d evidence-packet-analysis-smtp3-attachment > secretrendezvous.docx
```

Check the file type:

```
$ file secretrendezvous.docx
secretrendezvous.docx: Zip archive data, at least v2.0 to extract
```

This makes sense, since .docx files are zip archive data. Let’s take the cryptographic checksums:

```
$ md5sum secretrendezvous.docx
9049b6d9e26fe878680eb3f28d72d1d2  secretrendezvous.docx

$ sha256sum secretrendezvous.docx
24601c174587be4ddfff0b9e6d598618c6abfcfadbf16f7dd6dbd7a24aed6fec8
secretrendezvous.docx
```

4.6.7 Viewing the Attachment

Naturally, we want to view the contents of the attachment, which has been labeled with a .docx extension. Always remember that mainstream document-viewing programs often modify the files even when only used to view the file. When viewing a file, make sure that you are working on a *copy* of the evidence, and not the evidence itself. You can also use filesystem permissions, hardware write blockers, and other mechanisms to reduce the risk that the application will modify the source file.

64. N. Borenstein and N. Freed, “MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies,” IETF, June 1992, <http://www.ietf.org/rfc/rfc1341.txt>.

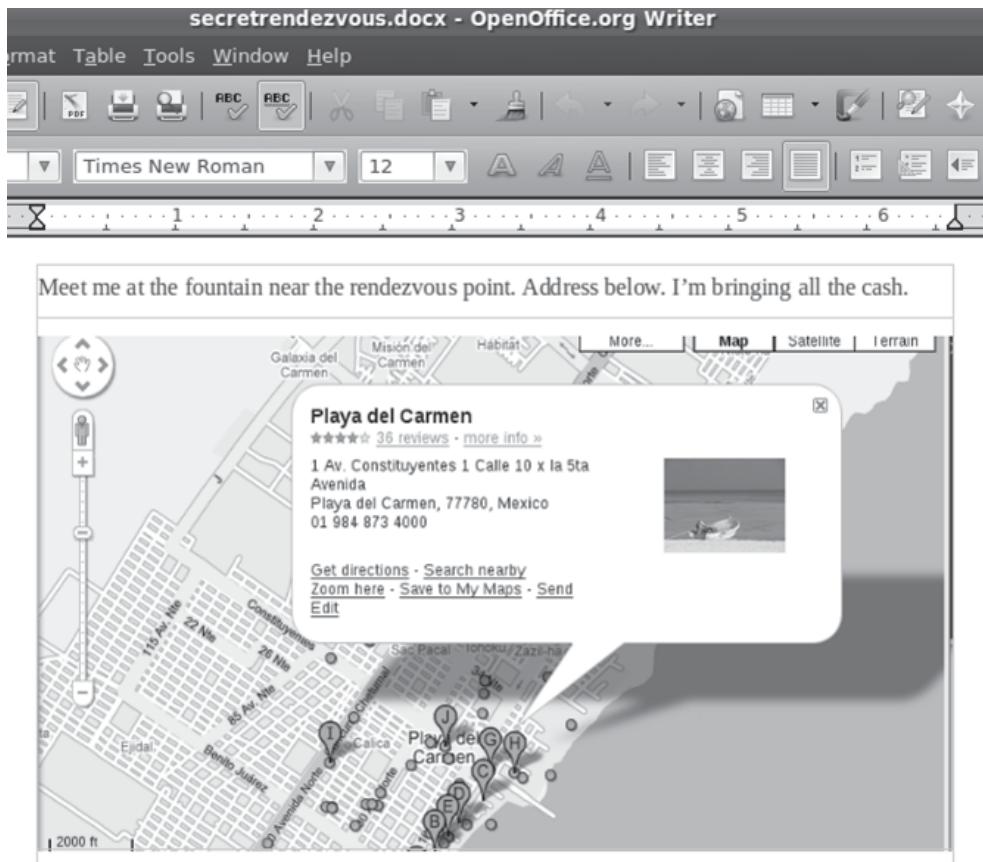


Figure 4–41. The attachment that Ann sent to “mistersecretx@aol.com,” opened using OpenOffice. (Image ©2009 Google—Map data ©2009 INEGI.)

In this case, we open the attachment using OpenOffice, as shown in Figure 4–41. At the top of the document is the text, “Meet me at the fountain near the rendezvous point. Address below. I’m bringing all the cash.” This is followed by an image of what appears to be a map with an address:

Playa del Carmen
1 Av Constituyentes 1 Calle 10 x la 5ta
Avenida
Playa del Carmen, 77780, Mexico
01 984 873 4000

Taking things one step further, we can carve out the image file embedded in the document. This can be useful for correlating with evidence from other sources, such as hard drive analysis. Carving embedded images is easy with .docx files. First, let’s unzip the file:

```
$ unzip secretrendezvous-copy.docx
Archive: secretrendezvous-copy.docx
inflating: [Content_Types].xml
inflating: _rels/.rels
inflating: word/_rels/document.xml.rels
inflating: word/document.xml
extracting: word/media/image1.png
inflating: word/theme/theme1.xml
inflating: word/settings.xml
inflating: word/webSettings.xml
inflating: docProps/core.xml
inflating: word/styles.xml
inflating: word/fontTable.xml
inflating: docProps/app.xml
```

As you can see, there appears to be only one image file in this document, which is labeled with the extension “.png”. Take the cryptographic checksum of the embedded image file:

```
$ md5sum word/media/image1.png
aaeace50997b1ba24b09ac2ef1940b7    word/media/image1.png

$ sha256sum word/media/image1.png
7dc8b5a245a0ba9e7a456e718316c4fb0ab5ae59d34833f415a429a5b8a6b437    word/media/
image1.png
```

Open a copy of the image to confirm that it is the one we are looking for, as shown in Figure 4–42.



Figure 4–42. The image that we retrieved from the unzipped .docx file, which we carved out of SMTP traffic. (Image ©2009 Google—Map data ©2009 INEGI.)

4.6.8 Finding Ann the Easy Way

That was a lot of work! Wouldn't it be nice if there were a program that could analyze SMTP traffic and carve files out for us automatically?

4.6.8.1 NetworkMiner

As we've discussed, NetworkMiner is a great multipurpose tool for analyzing network traffic.⁶⁵ When the "Ann Skips Bail" (Puzzle #2) SMTP contest was released on ForensicsContest.com, Erik entered NetworkMiner as his submission. To solve the puzzle, he extended NetworkMiner to parse SMTP traffic, and also added a new "Messages" tab. This was a great example of how members of the network forensics community can step up and contribute tools to solve challenges that all of us face.

In Figure 4–43, you can see a screenshot of NetworkMiner in action. We simply dragged and dropped the evidence file (evidence-packet-analysis.pcap) into NetworkMiner, and it automatically interpreted the results. The first tab, "Hosts," lists all of the IP addresses of the hosts that transmitted IP traffic in the packet capture.

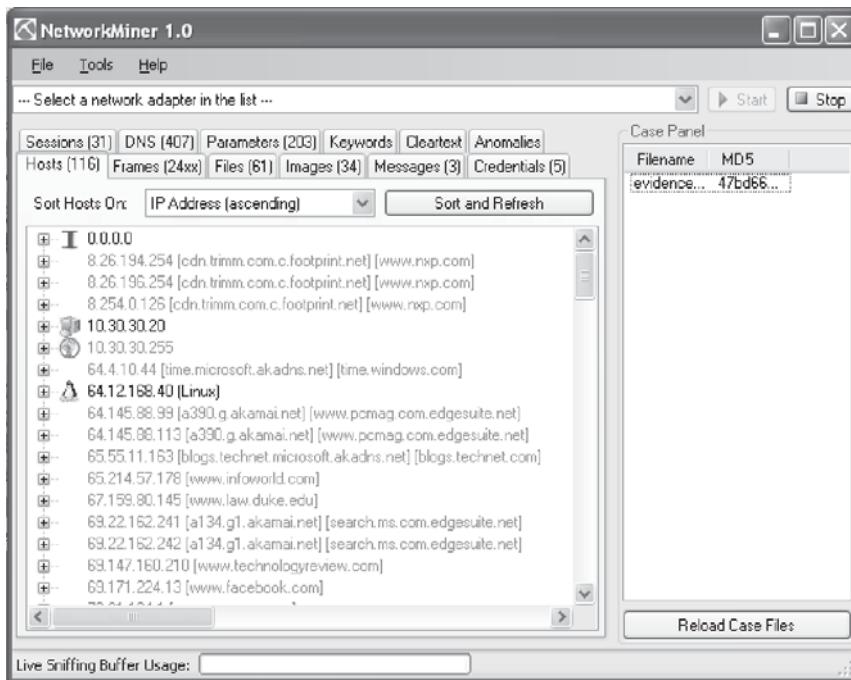


Figure 4–43. Here is a screenshot of NetworkMiner in action. We simply dragged and dropped the evidence file into NetworkMiner, and it automatically interpreted the results.

65. "NETRESEC NetworkMiner—The Network Forensics Analysis Tool."

Messages [3]									
Frame nr.	Source	Destin...	From	To	Subject	Protocol	Timestamp	Attribute	Value
1590	192.168...	64.12.1...	'Ann D...	<interOp...	need a favor	Smtp	5/17/2011 13:30...	Message-ID	<00bc01cc14c9...
1752	192.168...	64.12.1...	'Ann D...	<d4rikta...	lunch next week	Smtp	5/17/2011 13:41...	From	"Ann Dercov..."
2136	192.168...	64.12.1...	'Ann D...	<misters...	rendezvous	Smtp	5/17/2011 13:52...	To	<misterseriktx@...

Hi sweetheart! Bring your fake passport and a bathing suit.
Address attached. love, Ann

Figure 4–44. Erik added the “Messages” tab for the “Ann Skips Bail” SMTP puzzle on ForensicsContest.com. Notice that NetworkMiner automatically parses headers and displays them along with the body of each SMTP message. It can also show other types of messages, such as IMs.

Files [61]											
Frame nr.	Recon...	Source	S. port	Destin...	D. port	Protocol	Filename	Extens...	Size	Timest...	Details
1590	C:\Doc...	192.168...	TCP 1694	64.12.1...	TCP 587	SMTP	needafavor.alternative	alternati...	46 B	5/17/2...	E-mail From: "Ann Dercov..."
1590	C:\Doc...	192.168...	TCP 1694	64.12.1...	TCP 587	SMTP	needafavor.html	html	447 B	5/17/2...	E-mail From: "Ann Dercov..."
1752	C:\Doc...	192.168...	TCP 1697	64.12.1...	TCP 587	SMTP	lunchnextw.alternative	alternati...	46 B	5/17/2...	E-mail From: "Ann Dercov..."
1752	C:\Doc...	192.168...	TCP 1697	64.12.1...	TCP 587	SMTP	lunchnextw.html	html	493 B	5/17/2...	E-mail From: "Ann Dercov..."
2136	C:\Doc...	192.168...	TCP 1699	64.12.1...	TCP 587	SMTP	rendezvous.mixed	mixed	46 B	5/17/2...	E-mail From: "Ann Dercov..."
2136	C:\Doc...	192.168...	TCP 1699	64.12.1...	TCP 587	SMTP	rendezvous.html	html	402 B	5/17/2...	E-mail From: "Ann Dercov..."
2136	C:\Doc...	192.168...	TCP 1699	64.12.1...	TCP 587	SMTP	secretrendezvous.docx	docx	205,409...	5/17/2...	E-mail From: "Ann Dercov..."

Open File
Open Folder

Figure 4–45. The “Files” tab displays files that NetworkMiner automatically carved out of the packet capture.

The “Messages” tab, which Erik added specifically to solve this puzzle (lucky us!), is shown in Figure 4–44. As you can see, it automatically parses SMTP headers and displays them along with the body of each message.

In NetworkMiner’s “Files” tab, you can see all the files that NetworkMiner carved out of the packet capture. In Figure 4–45 you can see that NetworkMiner identified the “secretrendezvous.docx” file, and by right-clicking on it, you can open it up and view the contents.

NetworkMiner even automatically recovers the client’s authentication credentials. As shown in Figure 4–46, It automatically decodes the Base64-encoded SMTP Authentication credentials and lists them in the “Credentials” tab.

4.6.8.2 smtpdump and docxtract

In this example, we use “smtpdump” to automatically analyze the evidence file, print SMTP authentication information, extract the attachments, and print corresponding cryptographic checksums.

Hosts [116]	Frames [24xx]	Files [61]	Images [34]	Messages [3]	Credentials [5]	Sessions [31]	DNS [407]	Parameters [203]	Keywords	Cleartext	Anomalies
Client	Server	Protocol			Username	Password	Valid login			Login timestamp	
192.168.30.108 [ann-laptop]	74.125.224.123 [dart...	HTTP Cookie	test_cookie=C...		N/A	Unknown	5/17/2011 1:32:23 PM				
192.168.30.108 [ann-laptop]	74.125.224.122 [pag...	HTTP Cookie	test_cookie=C...		N/A	Unknown	5/17/2011 1:32:26 PM				
192.168.30.108 [ann-laptop]	74.125.224.122 [pag...	HTTP Cookie	id=2248936b0...		N/A	Unknown	5/17/2011 1:32:26 PM				
192.168.30.108 [ann-laptop]	208.107.213.96 [a12...	HTTP Cookie	UID=21a0e0fa...		N/A	Unknown	5/17/2011 1:32:26 PM				
192.168.30.108 [ann-laptop]	64.12.168.40 [Linux]	SMTP	sneakyg33ky		s0opers3kr1t	Unknown	5/17/2011 1:33:06 PM				

Figure 4–46. NetworkMiner automatically extracts SMTP Authentication credentials, and even Base64 decodes them for you.

The command below tells smtpdump to analyze SMTP flow #3 in the packet capture (-f 3), extract attachments (-x), print the MD5sum (-m), and print authentication data (-A).

```
$ smtpdump -f 3 -x -m -A -r evidence-packet-analysis.pcap
[3] 192.168.30.108:1689 => 64.12.168.40:587
==== Authentication infos ====
  Found LOGIN method
  Username: sneakyg33ky
  Password: s0opers3kr1t

==== Attachments infos ====
  Type: multipart/alternative
  Type: application/octet-stream
  Saving file to disk: secretrendezvous.docx

  File: secretrendezvous.docx (MD5: 0x9049b6d9e26fe878680eb3f28d72d1d2)
```

Subsequently, we can use “docxtract” to extract all images from the carved .docx attachment (-x -i), and print the corresponding cryptopgraphic checksum (-m).

```
$ docxtract -x -i -m secretrendezvous.docx
Extracting: image1.png (194124 bytes)
MD5: aadeace50997b1ba24b09ac2ef1940b7
```

4.6.8.3 findsmtinfo.py

We can use “findsmtinfo.py” to:

- Print SMTP authentication information
- Extract all messages from the packet capture
- Extract all attachments from the messages
- Print the MD5 sums for each of the attachments
- Extract the files embedded within the .docx file
- Print the MD5 sums for each of the embedded files

The “findsmtinfo.py” tool produces these results automatically and creates a series of reports suitable for the appendix of a professional forensics report. This is a nice example of how a small, sharp tool can be used to extract evidence very efficiently, as shown below:

```
$ findsmtinfo.py -p evidence-packet-analysis.pcap
Found SMTP Session data
-----
Report: 192.168.030.108.01687-064.012.168.040.00587
-----
Found SMTP Session data
SMTP AUTH Login: sneakyg33ky
SMTP AUTH Password: s00pers3kr1t
SMTP MAIL FROM: <sneakyg33ky@aol.com>
SMTP RCPT TO: <d4rkta&gent@gmail.com>
Found email Messages
- Writing to file: ./report/messages
  /192.168.030.108.01687-064.012.168.040.00587.msg
- MD5 of msg: 2141fcb61af1fd3985f18c3ca2b985b2
- Found Attachment
  - Writing to filename: ./report/messages/1/part-001.ksh
  - Type of Attachement: text/plain
  - MDS of Attachement: 4bd7b649e5f2b3fd18fd3e3cd40f1fae
- Found Attachment
  - Writing to filename: ./report/messages/1/part-001.html
  - Type of Attachement: text/html
  - MDS of Attachement: 1cf5b25642406fa49b1912cccc93cde2
-----
Report: 192.168.030.108.01688-205.188.058.010.00143
-----
Found SMTP Session data
...
```

The “findsmtinfo.py” tool even automatically unzips the attached .docx file, extracts each of the embedded files, and prints MD5sums for each, as shown below:

```
...
Found SMTP Session data
SMTP AUTH Login: sneakyg33ky
SMTP AUTH Password: s00pers3kr1t
SMTP MAIL FROM: <sneakyg33ky@aol.com>
SMTP RCPT TO: <mistersekritx@aol.com>
Found email Messages
- Writing to file: ./report/messages
  /2/192.168.030.108.01689-064.012.168.040.00587.msg
- MD5 of msg: 94d957fda87d9114016e65e5afc12cef
- Found Attachment
  - Writing to filename: ./report/messages/2/part-001.ksh
  - Type of Attachement: text/plain
  - MDS of Attachement: ba2c98f65f3f678b6a71570adcf362f4
- Found Attachment
  - Writing to filename: ./report/messages/2/part-001.html
  - Type of Attachement: text/html
  - MDS of Attachement: d07c3b721fed36a725c01e4827c1a563
- Found Attachment
  - Writing to filename: ./report/messages/2/secretrendezvous.docx
  - Type of Attachement: application/octet-stream
  - MDS of Attachement: 9049b6d9e26fe878680eb3f28d72d1d2
    - ZIP Archive attachment extracting
      - Found file
        - Writing to filename: ./report/messages/2/secretrendezvous.docx.
          unzipped/[Content_Types].xml
```

```

- Type of file: application/xml
- MDS of File: 8af852cc1775236bac8e8495564a4ef2
- Found file
- Writing to filename: ./report/messages/2/secretrendezvous.docx.
  unzipped/_rels/.rels
- Type of file: None
- MDS of File: 77bf61733a633ea617a4db76ef769a4d
- Found file
- Writing to filename: ./report/messages/2/secretrendezvous.docx.
  unzipped/word/_rels/document.xml.rels
- Type of file: None
- MDS of File: 1445296e2c18e6f42da2f6c4455d6a6c
- Found file
- Writing to filename: ./report/messages/2/secretrendezvous.docx.
  unzipped/word/document.xml
- Type of file: application/xml
- MDS of File: ac21289076d00f81ec885509e27b6d2f
- Found file
- Writing to filename: ./report/messages/2/secretrendezvous.docx.
  unzipped/word/media/image1.png
- Type of file: image/png
- MDS of File: aadeace50997b1ba24b09ac2ef1940b7
...

```

4.6.9 Timeline

Based on our analysis so far, we can create a timeline of events. As always, this is simply a working hypothesis based on the available evidence, and can change with the introduction of new evidence.

The timeline below summarizes the events of interest that we have identified so far. Note that the times are those listed within the packet capture, as recorded by the sniffing system. (The dates and times listed in the bodies of the emails we retrieved under the “Date” header were set by the system under investigation, and should be considered less reliable than our packet capturing system.)

All times listed below occurred on May 17, 2011.

- **13:32:01.419886**—Packet capture begins
- **13:32:03.166396**—First DHCP Request from 00:21:70:4d:4f:ae (Ann’s computer)
- **13:32:03.167145**—DHCP ACK from 192.168.30.10 to Ann’s computer, assigning 00:21:70:4d:4f:ae the IP address 192.168.1.108 with a 1-hour lease time.
- **13:33:05.834649–13:33:07.847758**—First SMTP conversation. Email sent from Ann’s computer with sender sneakyg33ky@aol.com and recipient inter0pt1c@aol.com.
- **13:34:15.110657–13:34:17.204721**—Second SMTP conversation. Email sent from Ann’s computer with sender sneakyg33ky@aol.com and recipient d4rktangent@gmail.com.
- **13:35:15.504697–13:35:23.263802**—Third SMTP conversation. Email sent from Ann’s computer with sender sneakyg33ky@aol.com and recipient mistersekritx@aol.com.
- **13:35:23.263802**—Packet capture ends

4.6.10 Theory of the Case

Now that we have put together a timeline of events, let's summarize our theory of the case. Again, this is a working hypothesis strongly supported by the evidence, references, and experience:

- Ann Dercov connected her laptop ("ann-laptop") to the network on May 17, 2011, at 13:32:03. Her computer was probably manufactured by Dell.
- At 13:33:05, Ann sent email from her AOL account, sneakyg33ky@aol.com, to inter0pt1c@aol.com, asking the recipient, "Hey, can you hook me up quick with that fake passport you were talking about?"
- At 13:34:15, Ann sent email from her AOL account, sneakyg33ky@aol.com, to d4rktangent@gmail.com, informing the recipient, "Sorry—I can't do lunch next week after all. Heading out of town. Another time!"
- At 13:35:15, Ann sent email from her AOL account, sneakyg33ky@aol.com, to mistersekr1t@aol.com, with the message, "Hi sweetheart! Bring your fake passport and a bathing suit. Address attached. love, Ann." The email had a .docx attachment that contained an address and a map.

4.6.11 Response to Challenge Questions

Now, let's answer the investigative questions posed to us at the beginning of the case.

- **Provide any online aliases or addresses and corresponding account credentials that may be used by the suspect under investigation.**

Based on our SMTP analysis, there are indications that Ann Dercov uses the email address sneakyg33ky@aol.com, and that her password is "s00pers3kr1t."

- **Who did Ann communicate with? Provide a list of email addresses and any other identifying information.**

We have seen that sneakyg33ky@aol.com sent emails to the following recipients:

- inter0pt1c@aol.com
- d4rktangent@gmail.com
- mistersekr1t@aol.com

- **Extract any transcripts of Ann's conversations and present them to investigators.**

Here is a quick summary of Ann's conversations, sent via SMTP:

- **SMTP Message #1**

Sender: sneakyg33ky@aol.com

Recipient: inter0pt1c@aol.com

Date [beginning of SMTP conversation]: May 17, 2011 13:33:05

Subject: need a favor

Message [formatting removed]: Hey, can you hook me up quick with that fake passport you were talking about? - Ann

Attachments of interest: None

– SMTP Message #2

Sender: sneakyg33ky@aol.com

Recipient: d4rktangent@gmail.com

Date [beginning of SMTP conversation]: May 17, 2011 13:34:15

Subject: lunch next week

Message [formatting removed]: Sorry—I can't do lunch next week after all. Head-ing out of town. Another time! - Ann

Attachments of interest: None

– SMTP Message #3

Sender: sneakyg33ky@aol.com

Recipient: mistersekritx@aol.com

Date [beginning of SMTP conversation]: May 17, 2011 13:35:15

Subject: rendezvous

Message [formatting removed]: Hi sweetheart! Bring your fake passport and a bathing suit. Address attached. love, Ann

Attachments of interest: secretrendezvous.docx

- If Ann transferred or received any files of interest, recover them.**

We recovered one Office Open XML Document (.docx) file, attached to Ann's email to mistersekritx@aol.com.

The MD5 checksum of the .docx file was:

9049b6d9e26fe878680eb3f28d72d1d2

The SHA256 checksum was:

24601c174587be4ddfff0b9e6d598618c6abfcfad16f7dd6dbd7a24aed6fec8

The document began with the text, "Meet me at the fountain near the rendezvous point. Address below. I'm bringing all the cash." This was followed by a PNG image of a map with an address.

- Are there any indications of Ann's physical whereabouts? If so, provide supporting evidence.**

The document that Ann sent to mistersecretx@aol.com indicates that she would like to meet him at the following address:

Playa del Carmen
1 Av Constituyentes 1 Calle 10 x la 5ta
Avenida
Playa del Carmen, 77780, Mexico
01 984 873 4000

Of course, there is no guarantee that Ann and/or the email recipient ever traveled to this location. Perhaps Ann was trying to throw us off her trail!

4.6.12 Next Steps

By analyzing a short packet capture from May 17, 2011, we've retrieved significant information about Ann's communications, including an email address and password, which she likely uses for personal correspondence, the addresses of three correspondents, and an attachment that may contain clues to her physical whereabouts. What are our next steps? Here are some possibilities:

- **Further Analysis of the Packet Capture** Recall that this packet capture also contains IMAP and HTTP traffic. Each of these protocols may include additional communications. For example, we can analyze the IMAP traffic to recover more email correspondence. The HTTP traffic might contain web-based email account activity, or clues to Ann's interests—perhaps even travel-related information such as searches for hotels, restaurants, or plane fares.
- **Email Account Monitoring** If there is sufficient evidence, and relevant laws and/or ISP policies allow, it may be possible to monitor activity relating to a suspect's email account on an ongoing basis. This may include a history of connections from client IP addresses—very useful information if you are trying to track down the suspect's physical location. It may also include message header information such as recipient, date, and time, or in some cases, the actual contents of the emails. Even if you have the suspect's password, be absolutely sure to consult with legal counsel before using a suspect's credentials to gain access to their account. In general, when possible, it is best to work with the ISP and obtain information through their standard channels.

This page intentionally left blank

Chapter 5

Statistical Flow Analysis

“Often think of the rapidity with which things pass by and disappear. . . . For substance is like a river in a continual flow, and the activities of things are in constant change, and the causes work in infinite varieties; and there is hardly anything which stands still.”

—The Meditations, by Marcus Aurelius¹

If you were capable of watching all of the automobiles driving on the roads within a geographic area, mapping sources and destinations, volumes of people and objects moved, and directions of transfer, you would be able to tell quite a lot about the activities of the people in the area. Not only could you make broad statements about the general habits of all the people, you could also identify areas of congestion on the roads and unusual volumes or types of activity. If you had very granular data, you could track vehicles from specific home addresses, to offices, stores, and back. You could infer the normal work times of individuals, general interests, and hobbies. You could look back in time after a crime had been committed—let’s say a robbery at a store—and make a list of all the people who were likely present during that time frame.

In many ways, network flow record analysis is analogous to analysis of traffic patterns on the roads in real life, except that it is far easier to gather granular information on computer networks.

Every packet that traverses a network can be logged and recorded. Since the early days of networking, it has been common to record basic information about flows,² including source and destination IP address, source and destination port (where applicable), protocol, date, time, and the size of the data transmitted in each packet. Network flow records were originally generated, captured, and analyzed to improve network performance, but gradually it has become clear that the data is extremely valuable for other reasons as well. This information can provide network engineers and investigators with a wealth of information about activity on the network.

Forensic analysts investigating a crime may find that flow records can reveal a detailed portrait of network activity. In network forensic investigations, statistical analysis of flow data can serve a variety of purposes, including:

- **Identifying Compromised Hosts** Often, compromised hosts send out more traffic than normal, transmit or receive traffic on unusual ports, or communicate with other systems that are known to be malicious.

1. Marcus Aurelius, *The Meditations* (The Internet Classics Archive, 2000), <http://classics.mit.edu/Antoninus/meditations.mb.txt>.

2. N. Brownlee, “RFC 2722—Traffic Flow Measurement: Architecture,” *IETF*, October 1999, <http://www.rfc-editor.org/rfc/rfc2722.txt>.

- **Confirming or Disproving Data Leakage** Network investigators often receive flow records with requests to determine whether an unknown attacker exported sensitive information across the network perimeter. If the time frame is bounded, and flow records are comprehensive, it is possible to analyze the volume of exported data and determine whether a leak may have occurred.
- **Individual Profiling** Flow data can provide a surprisingly rich picture of an individual's activity. Flow data from a user's workstation can reveal normal working hours, periods of inactivity, lunch times, break times, sources of entertainment, inappropriate activity, and more. By correlating activity between endpoints, you may determine which users communicate and exchange files or URLs between each other. Profiling can be especially helpful in cases that involve human resources issues.

Statistical flow analysis is becoming increasingly important in the modern world due to the sheer volume of traffic that organizations produce, manage, and inspect. It is easily possible for forensic investigators to be so inundated with data that we lack the processing power to crunch through it all. Consequently, we must be very judicious about which bits of information we collect. While it is possible to record full contents of all packets, this would have a high impact on performance, take up a large amount of storage space, and is generally not necessary. Statistical flow analysis can help investigators identify specific targets for content analysis and further investigation.

Every bit of data that is *not* collected is lost forever. Yet too much useless data can make analysis difficult or impossible.

5.1 Process Overview

In order to gain intelligence about what is flowing on a network, *flow records* must be generated, collected, and analyzed. A “flow record” is simply a summary of information about a flow, which typically includes the source and destination IP address, ports, date and time, and the amount of data transmitted. Other information may be included in a flow record as well, depending on the equipment and software generating the flow record, as well as the local configuration.

Definition: Flow Record

Flow record—A subset of information about a flow. Typically, a flow record includes the source and destination IP address, source and destination port (where applicable), protocol, date, time, and the amount of data transmitted in each flow.

A “sensor” is what we commonly call the device that is used to monitor the flows of traffic on any given segment and extract critical bits of information in a flow record. Network equipment rarely includes much disk space for storing logs of any type, including

network flow data. As a result, sensors are designed to immediately export flow records across the network, once the flow is complete.³ Flow record data is exported from a sensor to a “collector,” which is a server configured to listen on the network for flow record data and store it to a hard drive. An organization may have multiple collectors to capture flow record data from different network segments, or to provide redundancy. Once the flow record data has been exported and stored on a collector or multiple collectors, it can be aggregated and analyzed using a wide variety of commercial, open-source, and homegrown tools.

Flow Record Processing System

Flow record processing systems include the following components:

- **Sensor**—The device that is used to monitor the flows of traffic on any given segment and extract important bits of information to a flow record.
- **Collector**—A server (or multiple servers) configured to listen on the network for flow record data and store it to a hard drive.
- **Aggregator**—When multiple collectors are used, the data is typically aggregated on a central server for analysis.
- **Analysis**—Once the flow record data has been exported and stored, it can be analyzed using a wide variety of commercial, open-source, and homegrown tools.

Forensic investigators interested in analyzing flow record data should start by evaluating whether the flow record data of interest is already being collected. If not, you will need to decide whether it is appropriate and/or possible for you to initiate collection and analysis of flow data, and what strategy will work best for the environment under investigation.

5.2 Sensors

Every organization’s network environment is unique. Some organizations may already have flow sensors installed before an investigation commences, or at least their existing network equipment may be capable of producing flow export data if configured properly. Other organizations may have no such equipment, or the existing sensors may not be suitably placed for generating flow records necessary for the investigation. The architecture of the local network environment and the types of equipment available for use as sensors will strongly influence whether flow record data can be produced within the resources available for the investigation, and will also affect the resulting output format of the flow export data and subsequent analysis techniques.

3. R. Stewart, “RFC 4960—Stream Control Transmission Protocol,” IETF, September 2007, <http://www.rfc-editor.org/rfc/rfc4960.txt>.

5.2.1 Sensor Types

Sensors can be deployed as standalone appliances, or as software processes running on network equipment that serves other purposes. Many common types of network equipment are capable of producing flow record data, although network administrators do not always take advantage of these features. Specific types of network equipment typically support only limited output formats for flow record exportation, and this is one of the primary reasons that setting up a standalone appliance may be preferable.

5.2.1.1 Network Equipment

Some switches support flow record generation and exportation. For example, many popular Cisco Catalyst switches export flow record data. Likewise, current revisions of Cisco routers and firewalls support the ability to produce flow data and export it to a collector, using Cisco's proprietary NetFlow format (which we discuss in detail later in this chapter). Other vendors, such as Sonicwall, now support an open, standardized version of the flow record exportation protocol (IPFIX), in addition to NetFlow.^{4, 5} Be careful because some vendors who advertise support for flow record exportation actually base the exported flow record data on packet *sampling*, rather than a comprehensive inspection of all flows. Sampled data is generally not appropriate for network forensics because it will not provide you with complete information.

5.2.1.2 Standalone Appliances

Although it is common to generate flow records using network equipment that is already installed, in some cases the local network equipment may not be configured to export flow record data when the investigator arrives onsite. Furthermore, the existing network equipment may not generate flow data in a format that local administrators or forensic investigators can use for collection or the equipment may not be capable of generating flow records at all. In these instances, network administrators or forensic investigators may choose to deploy a standalone appliance as a sensor for generating and exporting flow record data.

You can deploy a software-based sensor anywhere on the network where you can capture traffic. Organizations can set up port monitoring or a network tap to send traffic to the standalone sensor, which in turn processes traffic, generates statistics, and sends flow record data to the collector. Since the standalone server does not need to concurrently function as a router or perform other functions, there are usually performance benefits to deploying standalone sensors to export flow record data.

Forensic investigators work to maintain a small footprint within any environment under investigation. For this reason, it is often preferable to set up a separate, standalone sensor rather than modifying existing network infrastructure devices and risk impacting production functionality.

4. "NSA 3500 Network Security Appliance—SonicWALL, Inc.," *SonicWall*, 2011, http://www.sonicwall.com/us/products/NSA_3500.html#tab=specifications.

5. "NetFlow Ninjas—Attention Firewall Vendors: Please Add NetFlow Support!," July 20, 2010, <http://netflowninjas.lancope.com/blog/2010/07/attention-firewall-vendors-please-add-netflow-support-asap.html>.

5.2.2 Sensor Software

Most modern, enterprise-quality routers and switches can be configured to act as sensors and export flow record data across the network to collectors. This includes devices manufactured by widely used vendors such as Cisco, Juniper, SonicWALL, and others. Alternatively, you can set up standalone sensors using free, open-source tools such as Argus, softflowd, or yaf. We now discuss each of these in turn.

5.2.2.1 Argus

Argus stands for “Audit Record Generation and Utilization System.” It is a mature, libpcap-based network flow sensing, collection, and analysis toolkit. Argus was first developed at Carnegie Mellon and released to the public in 1996. It is open-source software, copyrighted by QoSient, LLC and released under the GNU Public License. Partly due to its long history, it has been tested on Mac OS X, Linux, FreeBSD, OpenBSD, NetBSD, Solaris, and has also been ported to many other operating systems (as well as Cygwin).⁶

Argus is distributed as two packages: the Argus server, which reads packets from a network interface or packet capture file; and the Argus client tools, which are used to collect, distribute, process, and analyze Argus data. The Argus server functions as a sensor and can output flow export data in Argus’ compressed format, as files or over the network via UDP. As a libpcap-based tool, Argus supports BPF filtering, which allows the user to finetune filtering and maximize performance. It also supports multiple output streams, which can be directed to different collector processes or hosts.

Argus is one of the few flow record toolkits that specifically includes mention of forensic investigation in official documentation:

“Historical netflow data can be used in forensic investigations several months, or years, after an incident has taken place. Argus’ netflow records offer up to a 10,000:1 ratio from the packet size to the record written to disk, which allows installations to save records for much longer than full packet captures. When network security is very important, non-repudiation becomes a very important requirement that must be provided throughout the network. Argus provides the basic data needed to establish a centralized network activity audit system. If done properly, this system can account for all network activity in and out of an enclave, which can provide the basic system needed to assure that someone can’t deny having done something in the network.”⁷

5.2.2.2 softflowd

“Softflowd” is an open-source flow monitoring tool developed by Damien Miller. It is designed to passively monitor traffic and export flow record data in NetFlow format (versions 1, 5, and 9 are supported, as of the time of this writing).⁸ It can be installed on Linux and OpenBSD operating systems, and is based on libpcap.

6. “ARGUS—Auditing Network Activity—FAQ,” July 1, 2009, <http://www.qosient.com/argus/faq.shtml>.

7. “Argus—NSMWiki,” *Sguil*, 2011, http://nsmwiki.org/index.php?title=Argus#Who_uses_argus.

8. “Softflowd—Fast Software NetFlow Probe,” March 28, 2011, <http://www.mindrot.org/projects/softflowd/>.

The hardware requirements are minimal; you need a network card capable of capturing your target volume of traffic and a second network card for exporting flows from the sensor to a collector. Select disk space based on the volume of flow data you would like to retain. (See Chapter 3, “Evidence Acquisition,” for more details on how to passively gain access to network traffic.)

5.2.2.3 yaf

“Yaf,” stands for “Yet Another Flowmeter.” As the name suggests, it is open-source flow sensor software.⁹ Released in 2006 under the GNU GPL, yaf is based on libpcap and can read packets from live interfaces or packet captures. It exports flow data in the IPFIX format, over SCTP, TCP, or UDP transport-layer protocols. One of the most powerful features of yaf is that it supports BPF filters for the purposes of filtering incoming traffic. This can allow investigators and network engineers to dramatically reduce the required processing capabilities and volume of flow export data.¹⁰ Yaf natively supports encrypted flow export using TLS.¹¹

5.2.3 Sensor Placement

Forensic investigators don’t always have much control over where sensors are placed in an environment under investigation. In the ideal situation, network infrastructures would be designed with flow monitoring in mind, for the purpose of performance tuning, as well as incident response and forensics. However, an investigator may walk into an organization that has little or no flow record collection set up. In some cases, you may have the opportunity to initiate flow record exportation and collection during the course of an investigation.

Network engineers and forensic investigators can work together to design a network that will facilitate flow record collection and analysis. Factors to consider when placing sensors include:

- **Duplication** Minimize duplication of flows collected. Duplication is inefficient, increases the resources needed for analysis, and adds to network congestion during aggregation. If you collect two of every packet because of where your sensors are placed, your processing and storage capacity is cut in half. There are times when it is necessary to place sensors at points where traffic will pass through multiple sensors, and in these cases, you can use filtering to reduce (but usually not eliminate) duplication. Make sure to place sensors in locations where you will obtain your targeted data, of course, while minimizing redundant data collection. To paraphrase Albert Einstein, “Make things as simple as possible, but no simpler.”
- **Time synchronization** Time synchronization between sensors is crucial. If the time on a sensor is not accurate, it is difficult to correlate flows exported by that device

9. Brian Trammell, “YAF—Documentation,” *CERT NetSA Security Suite*, <http://tools.netsa.cert.org/yaf/yaf.html>.

10. Christopher M. Inacio and Brian Trammell, “YAF: Yet Another Flowmeter,” August 23, 2010, http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf.

11. “YAF Documentation.”

with any other device, or evidence gathered from other resources. If you collect flow records from a device that is already set up, note the clock skew between the sensor and an accurate source. If you set up your own sensor, make sure to keep the time synchronized using NTP if possible.

- **Perimeter versus internal traffic** Most enterprises prioritize flow record collection from perimeter devices such as external firewalls. However, if all you do is monitor the perimeter, you'll never see what's going on inside the network. Keep in mind that having visibility *inside* the network is also very valuable. For example, frequently workstations are on a separate subnet from the core data center. Monitoring connections between these two subnets is likely to be of interest in the event of compromise. Often, it is valuable for forensic evaluators to have visibility within a single subnet. Local flow data can help identify compromised workstations that are seeking to propagate to new targets, both internal and external.
- **Resources** Remember that there are as many bits flowing across the Internet every day as there are grains of sand on all the beaches in the world. You can't have them all. Every investigator has limited resources. You have to decide, based on your funding, equipment, staff, and time, which of those grains are most important. Review the local network map carefully and choose choke points that will maximize your collection capacity while still fitting within your budget and your ability to process and analyze.
- **Capacity** Network devices have limited capacity to monitor and process traffic. Enabling flow record processing and exportation can impact the performance of network equipment, especially if it is already at high utilization. Fortunately, most modern, enterprise-class network devices have built-in functionality to help you understand the utilization and capacity in advance. For standalone devices, it is possible to place the sensor in a location where it simply cannot process the volume of traffic passing by. Many switches will allow you to oversubscribe a port while doing mirroring. Try to be aware in advance if capacity will be an issue and, if so, employ more filtering, or partition VLANs to multiple sensing ports and deploy multiple sensors or sensing interfaces.

5.2.4 Modifying the Environment

In most cases, flow record data is not recorded to the extent that an investigator would prefer for the case at hand. It may be appropriate to initiate collection of supplemental flow record data for a specific case, depending on the environment and the nature of the investigation. When an investigative team has decided to gather additional flow record data by modifying the environment, there are a few general options available:

- **Leverage existing equipment** The existing network equipment within the infrastructure may be capable of exporting flow record data. This can include switches, routers, firewalls, and NIDS/NIPS equipment. In this case, investigators may be able to leverage this built-in capability through minor configuration changes to the equipment. However, it is crucial to examine the network device capacity before initiating additional flow record collection to ensure that the network functionality is not negatively impacted and also that the flow record collection will be sufficient. Remember

also to check in advance and make sure that the format of flow record data exported by local equipment is supported by your collection system.

- **Upgrade network equipment** If existing network equipment does not support flow record exportation or cannot handle the increased capacity, investigators may choose to deploy replacement switches or other network devices that can handle the increased capacity. Depending on the type of network equipment, this transition may be straightforward or may require extensive reconfiguration.
- **Deploy additional sensors** Many devices that don't support flow record exportation still support port mirroring. Investigators can use port mirroring to send packets of interest to a standalone sensor (such as Argus or softflowd). Similarly, investigators may choose to deploy a network tap and send data to a standalone sensor for flow record collection.

Make sure that you carefully document any changes that you make to the environment, especially if you are involved in a case that may at some point wind up in court.

5.3 Flow Record Export Protocols

Flow records can be exported in a variety of formats, ranging from proprietary protocols such as Cisco's NetFlow to the IETF's open standard, IPFIX. Different formats can store different types of data, and some formats allow for far more local customization than others. In this section, we highlight a few of the most popular protocols.

Flow record processing is still relatively new (particularly in the context of forensics), and because it is not yet mature, tools and equipment typically only support a small subset of flow record export formats and often are not compatible with each other. Before you begin your evidence collection and analysis, be sure to research the flow record exportation formats supported by the equipment and tools used in your investigation.

5.3.1 NetFlow

NetFlow was first developed in 1996 by Cisco Systems, Inc.¹² for caching and exporting traffic flow information. Originally, NetFlow was designed to cache flow information for the purposes of improving performance. The flow statistics generated proved to be useful for network analysts in monitoring network performance, debugging issues, and responding to security threats.

Many devices can produce NetFlow data, from routers to switches to standalone probes. The sensor maintains a cache that tracks the state of all active flows observed by the device. When a flow is completed or is no longer active, the sensor marks the flow as "expired" and then exports the flow record data in a "NetFlow Export" packet that is transmitted over the network to a collector (or multiple collectors). For NetFlow version 9, the latest release as of this writing, the "NetFlow Export" packet is transport-layer independent and can be

12. Cisco Systems, Inc., "NetFlow Gives Network Managers a Detailed View of Application Flows on the Network," 2004, https://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_case_study0900aecd80311fc2.pdf.

transmitted over UDP, TCP, or even SCTP (earlier versions of NetFlow supported transport only over UDP). Although NetFlow was developed by Cisco, it became so popular that other network device manufacturers included support for the format, including Juniper¹³ and Enterasy.¹⁴

NetFlow version 5 (NetFlow v5) is straightforward and widely used on a wide variety of equipment from different manufacturers. However, NetFlow v5 has some important limitations—for example, it supports only IPv4 (not IPv6) and Export Packets must be transported over UDP, which is an unreliable transport-layer protocol.

NetFlow version 9 (Netflow v9) is the latest release from Cisco and was also selected by the IETF as the basis for the IPFIX standard¹⁵ (see “IPFIX” below). In addition to providing support for IPv6 and transport-layer independent exportation, NetFlow v9 introduced support for flexible template-based specification of flow record fields. This allows users to customize the data cached and exported by a sensor, enabling both greater flexibility for analysis as well as performance improvements.

5.3.2 IPFIX

The IP Flow Information Export (IPFIX) protocol is a successor to Cisco’s NetFlow, specified originally in RFC 5101¹⁶ and based on NetFlow v9. It is a result of the efforts of an IETF Working Group that sought to extend the NetFlow v9 protocol to handle bidirectional flow reporting,¹⁷ to reduce data redundancy when reporting on flows with similar attributes,¹⁸ and to provide for better interoperability as an open standard.¹⁹

The flow record data that IPFIX supports is extensible through data templates. The collector sends the template defining the data to be exported, and the sensor then uses that template to construct the flow data export packets that are sent back to the collector as the flows expire.

5.3.3 sFlow

sFlow is a standard for providing network visibility based on packet sampling. It was developed by InMon Corporation and published as an IETF RFC in 2001²⁰ (the specifications are

13. “Netflow Configuration Juniper,” *Caligare*, May 10, 2006, <http://netflow.caligare.com/configuration-juniper.htm>.

14. “Wireless Access Points, Cloud Based Solutions, Data Center Solutions,” *Enterasys*, July 15, 2010, <http://secure.enterasys.com/support/manuals/hardware/NetFlowFeatGde071510.pdf>.

15. B. Claise, “RFC 3954—Cisco Systems NetFlow Services Export Version 9,” *IETF*, October 2004, <http://www.rfc-editor.org/rfc/rfc3954.txt>.

16. B. Claise, “RFC 5101—Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information,” *IETF*, January 2008, <http://www.ietf.org/rfc/rfc5101.txt>.

17. B. Trammell and E. Boschi, “RFC 5103—Bidirectional Flow Export Using IP Flow Information Export (IPFIX),” *IETF*, January 2008, <http://www.ietf.org/rfc/rfc5103.txt>.

18. E. Boschi, “RFC 5473—Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports,” *IETF*, March 2009, <http://www.ietf.org/rfc/rfc5473.txt>.

19. “IP Flow Information Export (ipfix),” *Datatracker*, 2011, <http://datatracker.ietf.org/wg/ipfix/>.

20. P. Phaal, “RFC 3176—InMon Corporation’s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” *IETF*, September 2001, <http://www.rfc-editor.org/rfc/rfc3176.txt>.

now maintained at sFlow.org).²¹ sFlow is supported by many network device manufacturers (although notably, not Cisco).²²

sFlow is different from Netflow and IPFIX, in that it is designed to conduct statistical packet sampling and does not support recording and processing information about every single packet. The benefit of packet sampling is that it scales well to very large networks, with high throughput. The down side is that packets that are not sampled are not recorded and cannot be analyzed. For this reason, NetFlow and IPFIX are likely to be better options for the purposes of forensic analysis.

Flow Export: Transport-Layer Protocols

Traditionally, flow export data was transmitted over the network via UDP, which is a connectionless, unreliable transport-layer protocol. Unless a higher layer provides reliability (such as the application layer), flow export data transported via UDP may be dropped in transit and never recovered. Modern flow export protocols such as NetFlow v9 and IPFIX are transport-layer independent, allowing network engineers to use reliable transport-layer protocols such as the Stream Control Transmission Protocol (SCTP) for flow export data transmission.

SCTP has built-in support for reliable packet delivery, much like TCP. However, unlike TCP, it is optimized for transport of streaming data. If you consider the way that TCP was first designed, streams of packets could be easily disrupted by any amount of packet loss anywhere in the stream, simply because the entire transfer would have to be held up for the single retransmitted segment (“head-of-line blocking”). TCP has evolved to address this problem to some degree, through features such as the Selective Acknowledgment option.

SCTP was designed from inception to allow for volume transfers, including multiple streams over a single session. This allows different types of data to be transmitted concurrently and reliably on separate streams (for example, NetFlow template information can be sent on a separate stream from flow export data). SCTP is a natural choice for transport of flow export data, since it provides reliability for multiple voluminous streams.

5.4 Collection and Aggregation

Once you have exported flow record data from a sensor, you need a *collector* or multiple collectors to receive data in the export format and store it on disk. The architecture of the collector system is very flexible. A collector can run on a wide variety of hardware and software platforms as long as it supports the flow export format used to export the flow record data of interest.

21. “Specifications for Developers,” *Sflow*, 2011, <http://www.sflow.org/developers/specifications.php>.

22. “sFlow Products: Network Equipment,” 2011, <http://www.sflow.org/products/network.php>.

5.4.1 Collector Placement and Architecture

It's important to carefully consider where to place your collectors within the infrastructure, how to secure the system and the flow record data in transit, and how many collectors will be needed. Factors to consider include:

- **Congestion** Flow record exportation generates network traffic. This can exacerbate network congestion or, conversely, network congestion can cause problems with flow record exportation. When placing collectors, it is wise to choose locations within the network where transit times between sensor and collector are unlikely to be impacted by either routine or incident-based congestion (such as that caused by a worm).
- **Security** Can flow export data be “sniffed” or modified in transit? It's important to consider these questions when setting up flow export collectors, particularly if you suspect the local network may be compromised. While there is no “silver bullet,” forensic investigators have options for implementing reasonable security measures, depending on the environment.

To protect the confidentiality of flow export data, investigators can place collectors on segments where the paths between collector and sensor are access-controlled and well protected. Exporting flow record data on a separate VLAN from other traffic is usually a good idea. Collectors and sensors can even be networked across an isolated physical cable, or run on the same physical system, with sensor data exported across the localhost interface.

It is also possible to encrypt the flow export data as it is transmitted across the network, with protocols such as IPSec or TLS.²³ While flow record data is not commonly encrypted in transit, forensic investigators should strongly consider this as an option when it is supported by sensor and collector software, especially when flow export data is transmitted across insecure or shared network segments.

- **Reliability** Depending on the transport-layer protocol, packets dropped due to network congestion or denial-of-service attacks may be lost forever. Traditionally, flow export data has been transmitted over UDP, a connectionless protocol with no built-in support for reliability. Over the years, flow export formats have evolved to support reliable transport-layer protocols as well, such as TCP and SCTP. It is wise to use transport-layer protocols with built-in reliability whenever possible, especially when flow collection is being conducted for the purposes of forensic investigation.
- **Capacity** One of the main design choices to consider is how many collectors to set up. On the one hand, consolidating collection on a single system reduces hardware and software costs and facilitates aggregation and analysis. On the other hand, it's important to ensure that each collector can keep up with the bandwidth generated by the sensors, based on its network card capabilities, processing power, RAM, and storage space. A single physical system can run multiple collector processes listening

23. S. Leinen, “RFC 3955—Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX),” IETF, October 2004, <http://www.rfc-editor.org/rfc3955.html>.

for input on different ports, which can help system administrators track usage and facilitate remediation in the event of overload. Since hardware requirements for a collector are generally flexible, adding additional collectors is usually fairly easy to accomplish.

- **Strategy for Analysis** Your strategy for analysis will also affect placement, software choices, and configuration of collectors. If you are purchasing a commercial tool which has collector and analysis functionality bundled together, you will have different constraints than if you are planning to have a separate analysis system that can receive flow report data from multiple sources and aggregate it. Consider the architecture of your analysis environment when setting up the collectors.

5.4.2 Collection Systems

There are commercial collectors that often double as analysis tools. These include Cisco NetFlow Collector, Manage Engine’s NetFlow Analyzer, WatchPoint NetFlow Collector, and others.

There are also a wide spectrum of free and open-source collector tools. These are often bundled as part of a larger flow record collection and analysis suite. We discuss four open-source collector tools in this section: flowcap (from the SiLK suite), flow-capture (from the flow-tools suite), nfcapd (from the nfdump suite), and the Argus collection utility.

5.4.2.1 SiLK

The “System for Internet Level Knowledge” (SiLK) is a suite of command-line tools for the collection and storage of flow data that provides a very powerful toolset for the filtering and analysis of flow records.²⁴ One of several projects produced by the Network Situational Awareness (NetSA) group at CERT,²⁵ SiLK can process NetFlow versions 5 and 9 and IPFIX data and produce statistical data in a variety of output formats, configurable to a fine level of granularity. Of course, as with most extremely powerful and flexible toolsets, there is a cost in the form of a learning curve.

SiLK includes two collector-specific tools, flowcap and rwflowpack. The “rwflowpack” tool is designed to collect flow record data from a network socket or files and export it to the compressed SiLK Flow format. It supports NetFlow version 5, NetFlow version 9, and IPFIX, although IPFIX and NetFlow v9 are only supported if SiLK is compiled with support for NetSA’s libfixbuf²⁶ software library.²⁷

Rwflowpack can also accept input over a TCP socket from the SiLK flowcap tool. Flowcap similarly listens for flow records on a network socket, temporarily stores flow data to disk or RAM, and forwards the compressed stream to a client program such as rwflowpack.²⁸ This

24. “SiLK,” *CERT NetSA Security Suite*, 2011, <http://tools.netsa.cert.org/silk>.

25. “Featured Projects,” *CERT NetSA Security Suite*, 2011, <http://tools.netsa.cert.org>.

26. “Fixbuf,” *CERT NetSA Security Suite*, 2011, <http://tools.netsa.cert.org/fixbuf/libfixbuf/index.html>.

27. “SiLK sensor.conf,” *CERT NetSA Security Suite*, 2011, <http://tools.netsa.cert.org/silk/sensor.conf.html>.

28. “Flowcap—Capture, Store, and Forward Flow Data,” 2006, <http://silktools.sourceforge.net/flowcap.pub.html>.

architecture allows flow records to be collected close to the sensor and then is transferred via TCP to rwflowpack for processing. The SiLK collector tools support TCP and UDP for IPFIX flow record collection and UDP for NetFlow flow record collection.

5.4.2.2 flow-tools

The flow-tools suite is an open-source collection of flow export data collection and analysis tools. It is modular and easily extensible. The flow-capture utility is designed to collect NetFlow traffic exported by sensors via UDP. As of the time of this writing, flow-capture supports NetFlow versions 1, 5, 6, 7, and 8.1-14. However, NetFlow version 9 and IPFIX are not supported.²⁹ This is a serious limitation, especially for forensic investigators, since earlier versions of NetFlow do not provide support for reliable transport protocols. The flow-capture tool only accepts input on UDP ports and does not natively support flow export data collection via TCP or SCTP.

5.4.2.3 nfdump/NfSen

The nfdump suite includes tools for the collection, display, and analysis of network flow record data. The collector daemon, nfcapd, supports NetFlow versions 5, 7, and 9. It can read flow export data from a UDP network socket or pcap files, and it allows for extensive customization of the data files stored on disk. As with flow-tools, the transport-layer limitation is important for forensic investigators since forensics typically requires a higher degree of reliability than performance analysis. Nfcapd is designed to be integrated with the nfdump and NfSen display and analysis tools.³⁰ The nfdump and NfSen tools are released under the open-source BSD license.³¹

5.4.2.4 Argus

The Argus client tools suite also includes a collector. All of the Argus client programs accept input both in the Argus format and NetFlow versions 1–8.³² Unfortunately, NetFlow version 9 and IPFIX are not yet supported, although Qosient has indicated that support for NetFlow 9 is in development. The Argus “radium” (“Argus Record multiplexor”) client program is particularly useful for collection and distribution of flow export data because it can accept Argus or NetFlow input from multiple network or filesystem sources and sends Argus output to up to 128 client programs via the network or filesystem. As with other Argus client programs, radium supports the use of BPF-style filter expressions.

29. Mark Fullmer, “flow-capture(1)—Linux man page,” 2011, <http://linux.die.net/man/1/flow-capture>.

30. “NFDUMP,” *SourceForge*, July 28, 2010, <http://nfdump.sourceforge.net/>.

31. Peter Haag, “Watch your Flows with NfSen and NFDUMP” (Switch: The Swiss Education and Research Network, May 3, 2005), <http://www.ripe.net/ripe/meetings/ripe-50/presentations/ripe50-plenary-tue-nfSEN-nfdump.pdf>.

32. “Auditing Network Activity—Argus and Netflow,” March 26, 2010, <http://www.qosient.com/argus/argusnetflow.shtml>.

5.5 Analysis

Flow record data lends itself very well to statistical analysis. Although each flow record contains only a very limited set of information about each flow, aggregated over time, this information can be used to create a detailed picture of activity on the network, predict normal behavior, and identify anomalies. The costs of collecting and aggregating network flow record data are remarkably low when compared to the effort it takes to collect and aggregate similarly descriptive data about real-world activities such as highway traffic or pedestrian movements. As network forensic analysts, we are lucky to have access to so much high-fidelity data regarding our environment. The next question is, What can we do with it?

Definition: Statistics

Statistics—“The science which has to do with the collection, classification, and analysis of facts of a numerical nature regarding any topic.” (*The Collaborative International Dictionary of English* v.0.48)

In this section we discuss common analysis techniques that are useful for investigators with access to flow data and also provide you with an overview of some popular tools.

5.5.1 Flow Record Analysis Techniques

The purpose of flow record collection is to store a summary of information about the traffic flowing across the network. As a result, forensic analysis techniques such as carving content out of traffic flows do not apply to flow record data. That said, forensic analysis of flow record data is exceptionally powerful because the limited scope of data stored for each flow allows for a huge number of flows to be analyzed in ways that would simply exceed the scope of available storage and processing capabilities if full packet captures were used for the same purpose.

The precise analysis techniques that you employ will vary on a case-by-case basis, depending on your investigative goals and resources. As we discuss in the next section, the tools you choose will also impact your analysis (and vice versa). There are some analysis techniques, primarily statistical in nature, which are particularly well suited for flow record forensics. Keep in mind that this field of study is still nascent and flow record analysis techniques will quickly advance in the years to come.

Flow record analysis is typically an iterative process. Important highlights include:

1. **Goals and Resources** As always, the goals of your investigation and available resources shape your analysis. Are you trying to identify compromised systems? Evaluate whether a data breach occurred? Investigate an HR violation?

Make sure to assess your available time, staff, equipment, and tools. How much flow record data can you process? Given the format of the data and your goals, what tools are most appropriate for use? As your investigation begins, strategize and identify the analysis techniques that will best fit with your investigative goals and available resources.

2. Starting Indicators Every investigation begins with some triggering event. Investigators are normally provided with some pieces of evidence that can be used to begin analysis (“starting indicators”). Starting indicators may include:

- IP address of a compromised or malicious system
- Time frame in which malicious activity is suspected, even if the participants are unknown
- Known ports on which a worm was suspected to be active
- Specific flow records that indicate abnormal or unexplained activity

You may be given the IP address of a compromised system, or the domain name of a suspicious web site, or a description of malware behavior to search for in flow record data. Each of these “starting indicators” is useful for different analysis techniques. The volume and type of data you begin with will shape your investigation.

3. Analysis Techniques

- (a) **Filtering** Filtering is a general term for narrowing down a large pool of evidence to a subset—or groups of subsets—that are of interest. This is a fundamental technique in nearly every investigation that involves flow record data, since normally investigators are provided with far more flow record data than is relevant for the case at hand and must select only small percentages for detailed analysis and presentation.
- (b) **Baselining** Using flow record data, network administrators can build a profile of “normal” network activity. Forensic investigators can compare flow record traffic to the baseline profile in order to identify anomalies.
- (c) **“Dirty Values”** A common strategy in digital forensics of all types is to compile a list of suspicious keywords or other values to use when searching for evidence relevant to a case. For flow record data specifically, “dirty values” might include IP addresses, ports, or protocols.
- (d) **Activity Pattern Matching** Every activity leaves a fingerprint on the network. Simple patterns, such as large unidirectional volumes of traffic flow, can indicate specific activities that may be suspicious given the context (such as the presence of a high-volume, Internet-accessible file server on a subnet that should only contain company workstations). More complex patterns may match the behaviors of known worms or viruses. A trained investigator can pick out suspicious activity that manifests as complex and subtle variations in flow record evidence.

In this section, we briefly discuss how the forensic analysis techniques of filtering, baselining, “dirty values” search, and activity pattern matching apply to flow record analysis.

5.5.1.1 Filtering

Filtering is fundamental to flow record analysis. As a forensic investigator, your job is to remove extraneous details and identify events, hosts, ports, and activities of interest. For example, if your starting indicator was a specific IP address, you might start by isolating

all activity relating to that IP address and picking out subsets of flows for analysis until you have gathered the evidence you need and built your case. Alternatively, you might filter for flows that match particular patterns of activity indicative of worm behavior, data exfiltration, port scanning, or other suspicious behavior, depending on your investigative goals.

In rare instances, you may be provided with only the specific flows of interest in a case, but the vast majority of the time you will have access to a large volume of flow record data and will need to pick out the flows relevant to your case.

Most analysis techniques used for forensic analysis of flow records are based on filtering.

5.5.1.2 Baselining

One of the advantages of collecting flow record data (as opposed to full traffic captures) is that due to the dramatically smaller size of flow record data, longer historical periods can be kept accessible and perhaps even archived indefinitely.

Network traffic analysts can build, maintain, and reference a baseline of traffic and identify trends and patterns of activity that are considered “normal” for the environment (or at least understood and explainable). By aggregating a large volume of flow information, it’s possible to build a large, detailed set of benchmarks of what should be considered normal over any span of time and across business cycles. Forensic investigators can reference these baselines in order to analyze suspicious activity.

Network baselines By looking at general trends over time for monitored segments, the traffic seen can be understood even if specific source and destination IP addresses must be abstracted or generalized.

Host baselines Likewise, when a particular host becomes of interest, investigators can build or refer to a historical baseline of a specific host’s activities in order to identify or investigate anomalous behavior.

In most cases, the network flow patterns of a host change dramatically when it is compromised or under attack. By referring to a baseline of activity over time, you can often pick out approximate times of compromise or other important events relating to the investigation.

5.5.1.3 “Dirty Values”

In hard drive analysis, it’s common to compile a list of “dirty words”—keywords that relate to the investigation—and then search the hard drive for ASCII or Unicode strings that contain any of those keywords. This allows the investigator to pick out relevant sectors of the hard drive, even if the filesystem or partition table has been mangled, and find needles in the haystack.

Similarly, network forensic analysts can compile a list of “dirty values” and search flow record data to pick out relevant entries. When conducting statistical flow analysis, “dirty values” aren’t usually *words*—they are more likely to be suspicious IP addresses, ports, dates, and times. However, the concept is the same. As you conduct your analysis, you will often find that it is helpful to maintain an updated list of suspicious values that you collect as you move forward.

5.5.1.4 Activity Pattern Matching

Network flow record data, when aggregated, represents complex behavior that is often predictable and can be analyzed mathematically. We have previously compared network flows to traffic on physical roads. As with physical-world phenomena network flows contain patterns that can be empirically measured, mathematically described, and logically analyzed.

The biggest challenge for forensic investigators and anyone who analyzes flow record data is the absence of large publicly available data sets for research and comparison. However, within an organization, it is entirely possible to empirically analyze day-to-day traffic and build statistical models of normal behavior.

In this section, we briefly describe the basic elements of flow record analysis and present a high-level discussion of how to identify and analyze activity patterns that are commonly useful for network forensic investigators.

Elements Flow record data can be customized, but common elements include source and destination IP address, source and destination ports, protocol and flags, directionality of flow, and volume of data transferred.

- **IP address** The source and destination IP addresses are great clues that reveal a lot about the cause and purpose of a flow. Consider whether the addresses are on an internal network or Internet-exposed, countries of origin, what companies they are registered to, and other factors.
- **Ports** Although ports are not required to correspond with particular services, much of the time they do correspond with assigned or well-known ports linked to specific applications or services. Port numbers can also indicate whether a system is port scanning or being scanned and help you identify malicious activity.
- **Protocol and Flags** Layer 3 and 4 protocols are often tracked in flow record data. These can indicate whether connections were completed and help you tell the difference between connection attempts that were denied by firewalls, successful port scans, successful data transfers, and more. They can also help you make educated guesses as to the content and purpose of a flow.
- **Directionality** The directionality of the flows are crucial. For example, directionality can indicate whether proprietary data has been leaked or a malicious program was downloaded. Taken in aggregate, the directionality of data transfers can allow you to tell the difference between *web-surfing* activity and *web-serving* activity.
- **Volume of data transferred** The volume of data transferred can help indicate the type of activity and whether or not higher-layer data transfer attempts were successful. For example, many small TCP packets may be indicative of port scanning, whereas larger packets can indicate file exportation. The distribution of the data transferred over time matters; a large volume of data transferred in a very short period of time is usually caused by something different than the same amount of data transferred over a very long period of time.

Simple Patterns Even novice forensic investigators can identify simple patterns by eye in flow traffic and often logically discern their meanings. Here are a few examples of simple patterns relating to source and destination IP addresses:

- **“Many-to-one” IP addresses** If you see many IP addresses sending large volumes of traffic to one destination IP address, this may be indicative of a:
 - Distributed denial-of-service attack against the destination IP address
 - Syslog server
 - “Drop box” data repository on the destination IP address
 - Email server (at destination)
- **One to many IP addresses** If you see one IP address sending large volumes of traffic to many IP addresses, this may be indicative of a:
 - Web server
 - Email server (at source)
 - SPAM bot
 - Warez server
 - Network port scanning
- **Many to many IP addresses** Many IP addresses sending distributed traffic to many destinations can be indicative of:
 - Peer-to-peer filesharing traffic
 - Widespread virus infection
- **One to one IP address** Traffic from one IP address targeted at one other IP address can indicate:
 - Targeted attack against a specific system
 - Routine server communications

Of course, context is key. Specific ports, protocols, timing information, and other elements can help to narrow down the meaning of the flow record data even further.

Complex Patterns “Fingerprinting” is the process of matching complex flow record patterns to specific activities. When fingerprinting traffic, we examine multiple elements and their context and develop a hypothesis of the cause of the behavior.

For example, a TCP SYN port scan might have the following characteristics:

- One source IP address
- One or more destination IP addresses
- Destination port numbers increasing incrementally
- Volume of packets surpassing a specified value within a period of time

- TCP protocol
- Outbound protocol flags set to “SYN” (no full TCP connections)

It’s even possible to identify specifically which tool created the port scan, using flow record fingerprinting, since different port scanning tools have different algorithms for selecting source and destination port numbers, TCP window sizes, timing of packets, and other characteristics.

Statistical flow record analysis is still in its infancy. Over time, as this field matures, analysts will develop better tools and mechanisms for identifying the causes of network activity. In the meantime, forensic investigators should at the very least be familiar with the basic elements and simple patterns common in flow record analysis and refer to any baselines that exist within the environment under investigation.

5.5.2 Flow Record Analysis Tools

Many flow record analysis tools have emerged over the years, designed for different purposes and with varying levels of maturity. In this section, we provide you with a brief overview of the analysis capabilities included in selected widely used flow record analysis tools: flow-tools, SiLK, Argus, FlowTraq, nfdump/NfSen. There are many other tools out there, and more are released every day. We encourage you to do your own research to identify tools that are most appropriate for your customized investigative toolkit.

Notably, some of the most popular tools from the past decade, such as flow-tools, do not natively support the IPFIX/NetFlow v9 format, which limits both collection and analysis. Furthermore, some versions of tools do not produce accurate output when data generated using proprietary formats, such as Cisco NetFlow v9, is used as input.

Throughout this book, we generally present open-source tools that are widely accessible. However, developers of open-source tools often have limited access to expensive commercial network equipment that generates flow record data in proprietary formats. In some cases, such as nfdump, specific branches of development have forked from the main project in order to incorporate input from vendors. As a result, if you are analyzing flow record data that is stored in a proprietary format such as Cisco’s NetFlow v9, be sure to carefully research whether your analysis tools, and the specific version you are using, fully support the format you wish to analyze.

5.5.2.1 SiLK

The SiLK suite includes powerful flow export data analysis capabilities. Here is a quick summary of some of the more important analysis tools in the SiLK suite.

- **rwfilter** One of SiLK’s core tools, rwfilter, is designed to extract flows of interest from a particular data repository, filter them by time and category, and then “partition” them further by protocol attributes. It supports a rich syntax that is generally as functional as the BPF, though different.³³

33. “The SiLK Reference Guide (SiLK-2.4.5),” February 25, 2011, <http://tools.netsa.cert.org/silk/reference-guide.html>.

Initial flow selection typically includes date and time ranges, sensor class (by default all sensors), and a flow type. The SiLK toolkit defines six basic “types” of traffic, although not so much by protocol or layer: in, inweb, innull, out, outweb, and outnull, which correspond to successful versus unsuccessful flow attempts in either direction, further sorted by TCP port 80 versus everything else. (Of course the “successful vs. unsuccessful” part presumes that the reporting sensor is itself a filtering device, which is not uncommon.)

Rwfilter can be configured to output statistical summaries, or simply output detailed results to “stdout,” suitable for input to most of the rest of the filtering and manipulation tools in the SiLK suite.

- **rwstats, rwcount, rwcutf, rwuniq, et al.** SiLK includes an arsenal of basic flow export data manipulation utilities. Rwstats produces statistical aggregations based on the protocol fields specified. Rwcount counts packets and bytes. Rwcutf selects the fields that rwuniq can help you sort on.³⁴
- **rwidssqlery** This is a real gem. Let’s suppose there is a Snort sensor somewhere in the environment, most likely on a critical segment. Perhaps some unusual alerts have been detected on that heavily instrumented segment, and there’s concern the activity may have spread to other internal segments where only flow record data is being collected (as opposed to full packet captures). Rwidssqlery can be fed either a Snort rule file, or an alert file, and figures out what flows from its input would match the rule or alert, and writes an rwfilter invocation to produce the flows that match.
- **rwpmatch** This is essentially a libpcap-based program that reads in SiLK-formatted flow metadata and an input packet source and saves out just the packets that match the flow metadata. So if you just *happen* to have a full packet capture in addition to flow records, and your flow metadata turns up something interesting, you can extract those packets with precision for further inspection.
- **Advanced SiLK** In addition to the chainable command-line suite, a Python interpreter, “PySiLK,” is available that implements the SiLK functionality by exposing it through a Python API.³⁵ The nice folks at NetSA have provided a “Toolips” wiki so the user community can share experience and grow better faster.³⁶

5.5.2.2 flow-tools

The flow-tools suite includes a variety of flow export data collection, storage, processing, and sending tools, including a few tools that are particularly useful for forensic analysis. The “flow-report” tool creates ASCII-readable text reports based on stored flow data. Report contents can be customized by the user through the configuration file, and then sent as

34. Timothy Shimeall, “Using SiLK for Network Traffic Analysis” (Software Engineering Institute: Carnegie Mellon, September 2010), <http://tools.netsa.cert.org/silk/analysis-handbook.pdf>.

35. “PySiLK Reference Guide (SiLK-2.4.5),” February 25, 2011, <http://tools.netsa.cert.org/silk/pysilk.pdf>.

36. Peter Rush, “NetSA Tools Wiki,” CERT, April 9, 2008, <https://tools.netsa.cert.org/confluence/display/tt/Toolips>.

input to graphing or analysis programs. The “flow-nfilter” program allows users to filter flow export data based on “primitives,” which are specific to flow-tools.

For forensic investigators, “flow-dscan” is a particularly useful utility, designed to identify suspicious traffic based on flow export data. It includes features for identifying port scanning, host scanning, and denial-of-service attacks.

5.5.2.3 Argus Client Tools

The Argus suite includes a variety of specialized utilities with powerful analysis capabilities. These include, among others:³⁷

- **ra**—Argus’ basic tool for reading, filtering, and printing Argus data. As with all other Argus client programs, ra can read input from the network or the filesystem, in Argus or Netflow 1-8 format. It includes support for BPF-style filter expressions. Ra allows the user to specify fields for printing, select specific records for processing, match regular expressions, and more.
- **racluster**—Cluster flow export data based on user-specified criteria. This is very helpful for printing summaries of flow record data.
- **rasort**—Sort flow data based on user-specified criteria, such as source or destination IP address, time, TTL, flow duration, and more.
- **ragrep**—Powerful regular expression and pattern matching, based on the GNU “grep” utility.
- **rahisto**—Generate frequency distribution tables for user-selected metrics such as flow record duration, source and destination port number, bytes transferred, packet count, bits per second, and more.³⁸
- **ragraph**—Create visual plots based on user-specified fields of interest, such as bytes, packet counts, average duration, IP address, ports, and more. Ragraph includes a variety of tools that allow users to customize the graph appearance.

5.5.2.4 FlowTraq

FlowTraq is a commercial flow record analysis tool developed by ProQueSys. Although there are many commercial tools out there for processing flow record data, FlowTraq is a nice option for several reasons. First, it supports a very wide variety of input formats, including NetFlow v9, IPFIX, JFlow, and many others. It can also sniff traffic directly and generate flow records. Once collected, FlowTraq allows users to filter, search, sort, and produce reports based on flow records. In addition, you can specify patterns to generate alerts. FlowTraq supports a variety of operating systems (Windows, Linux, Mac, and Solaris), and is designed

37. “Argus—NSMWiki,” *Sguil*, 2011, http://nsmwiki.org/index.php?title=Argus#St.C3.A9phane_Peters.27s_Cheat_sheet.

38. Carter Bullard, “Argus Development Mailing List—Real Time Flow Monitor (RTFM) for Comprehensive IP Network Traffic Auditing,” November 21, 2010, <http://blog.gmane.org/gmane.network.argus/month=20061121>.

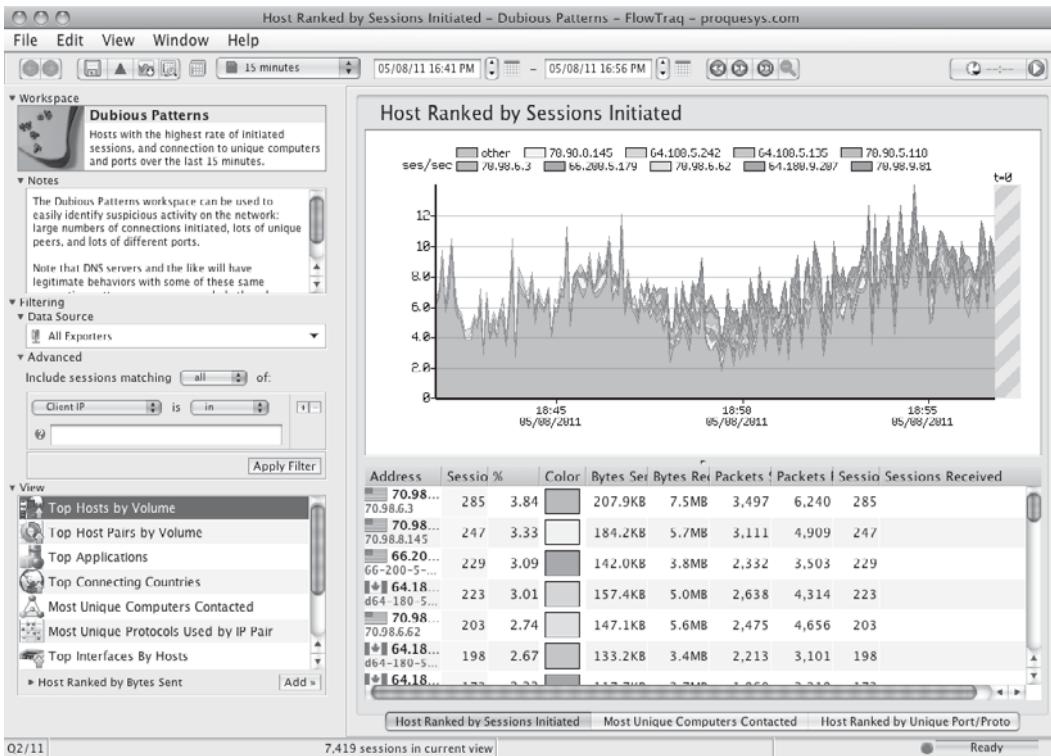


Figure 5–1. A screenshot of the FlowTraq commercial flow record analysis GUI, created using ProQueSys' online demo.³⁹

and marketed for forensics and incident response (among other purposes).⁴⁰ Figure 5–1 shows an example screenshot of FlowTraq, using data from their online demo.⁴¹

5.5.2.5 nfdump/NfSen

The “nfdump” utility (part of the nfdump suite) is designed to read flow record data, analyze it, and produce customized output. It offers users powerful analysis features, including:⁴²

- Aggregate flow record fields by specific fields
- Limit by time range
- Generate statistics about IP addresses, interfaces, ports, and much more
- Anonymize IP addresses

39. “Online Demo,” ProQueSys, 2010, <http://www.proquesys.com/corporate/flowtraq-demo>.

40. “Features,” ProQueSys, 2010, <http://www.proquesys.com/corporate/product/features>.

41. “Online Demo,” ProQueSys, 2010, <http://www.proquesys.com/corporate/flowtraq-demo>.

42. “NFDUMP.”

- Customize output format
- BPF-style filters

“NfSen” (“Netflow Sensor”) provides a graphical, web-based interface for the nfdump suite. It is an open-source tool written in Perl and PHP, designed to run on Linux and POSIX-based operating systems.⁴³

5.5.2.6 EtherApe

EtherApe is an open-source, libpcap-based graphical tool that visually displays network activity in real time. It reads packet data directly from a network interface or pcap file. (Note that EtherApe does not take flow records as input. We are including it here because it provides a nice high-level visualization of traffic patterns, and therefore may be of interest to the reader.) As shown in Figure 5–2, EtherApe visually displays source and destination IP addresses/hostnames, with corresponding volumes of traffic. The GUI provides an intuitive, high-level view of network activity. Colors indicate traffic protocols such as HTTP, SMB, ICMP, IMAPS, and more. EtherApe also provides sortable traffic summaries, as shown in Figure 5–3 and Figure 5–4.⁴⁴

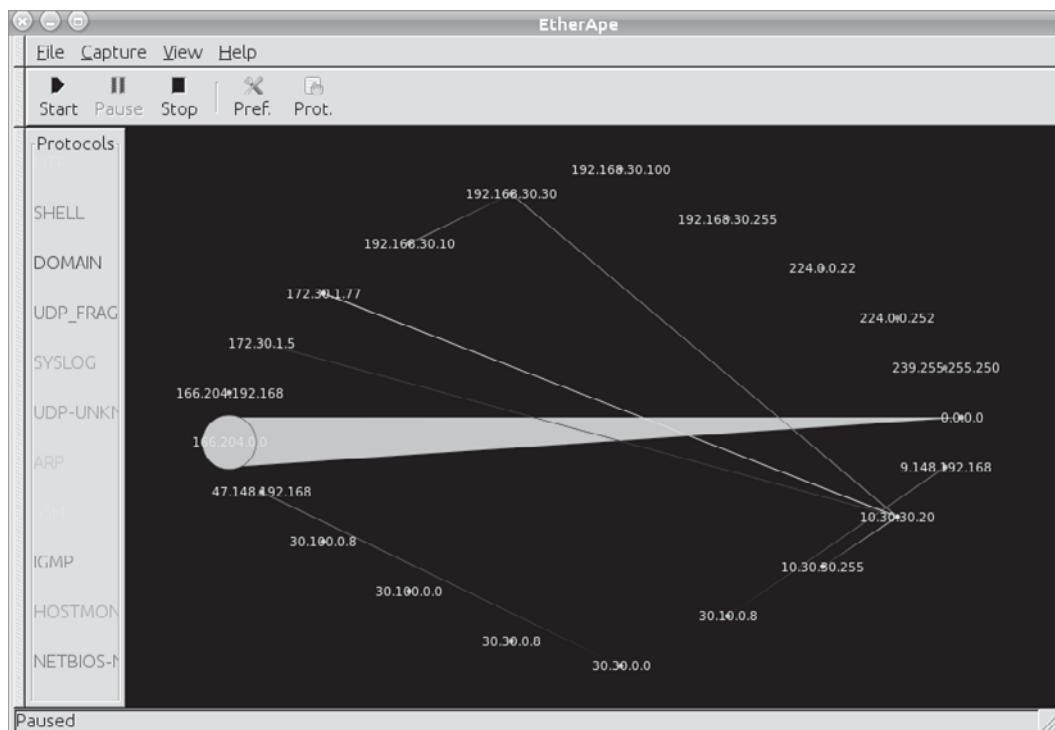


Figure 5–2. The main EtherApe screen.

43. Peter Haag, “nfsen,” <http://sourceforge.net/projects/nfsen/>.

44. Riccardo Ghetta and Juan Toledo, “EtherApe, A Graphical Network Monitor,” May 31, 2011, <http://etherape.sourceforge.net/>.

EtherApe: Protocols						
Protocol	Port	Inst Traffic	Accum Traffic	Last Heard	Packets	
ARP	-	0 bps	6.094 Kbytes	10" ago	104	
BOOTPS	67	0 bps	5.368 Kbytes	20" ago	16	
DOMAIN	53	10.258 Kbps	62.395 Kbytes	0" ago	454	
HOSTMON	5355	0 bps	9.246 Kbytes	1" ago	144	
IGMP	-	0 bps	5.631 Kbytes	9" ago	96	
NETBIOS-NS	137	11.556 Kbps	19.621 Kbytes	0" ago	209	
NTP	123	0 bps	450 bytes	1'39" ago	5	
SHELL	514	23.948 Kbps	78.920 Kbytes	0" ago	594	
SSH	22	80.135 Kbps	126.131 Kbytes	0" ago	827	
SYSLOG	514	0 bps	565.164 Kbytes	1" ago	2292	
UDP-UNKNOWN	-	0 bps	208.497 Kbytes	1" ago	193	
UDP_FRAGMENT	-	0 bps	10.018 Kbytes	24" ago	46	
WWW	80	0 bps	960 bytes	17" ago	9	

Figure 5–3. The EtherApe “Protocols” window.

Nodes						
Name	Address	Inst Traffic	Accum Traffic	Last Heard	Packets	
0.0.0.0	0.0.0.0	0 bps	4.078 Kbytes	23" ago	27	
10.30.30.20	10.30.30.20	114.341 Kbps	254.281 Kbytes	0" ago	1778	
10.30.30.255	10.30.30.255	0 bps	180 bytes	1'39" ago	2	
166.204.0.0	166.204.0.0	0 bps	1.055 Kbytes	23" ago	18	
166.204.192.168	166.204.192.168	0 bps	2.227 Kbytes	10" ago	38	
169.254.255.255	169.254.255.255	0 bps	5.812 Kbytes	49" ago	60	
169.254.37.184	169.254.37.184	0 bps	8.836 Kbytes	48" ago	108	
172.30.1.5	172.30.1.5	11.478 Kbps	22.480 Kbytes	0" ago	158	
172.30.1.77	172.30.1.77	80.135 Kbps	126.131 Kbytes	0" ago	827	
192.168.30.10	192.168.30.10	0 bps	609.253 Kbytes	1" ago	2244	
192.168.30.100	192.168.30.100	20.118 Kbps	197.947 Kbytes	0" ago	648	
192.168.30.101	192.168.30.101	0 bps	1.072 Kbytes	1'2" ago	2	
192.168.30.102	192.168.30.102	0 bps	1.092 Kbytes	1'1" ago	2	
192.168.30.255	192.168.30.255	20.118 Kbps	13.809 Kbytes	0" ago	149	

Figure 5–4. The EtherApe “Nodes” window.

5.6 Conclusion

Statistical flow record analysis is becoming increasingly important for forensic analysis, as the rate of network traffic generated and the volume of data stored far outpaces the rate at which investigators can process and analyze full packet captures. Although flow records were originally generated for the purposes of monitoring and improving network performance, in recent years investigators have begun to recognize that they are excellent sources of network-based forensic evidence as well.

A variety of sensor, collector, aggregation, and analysis tools exist, ranging from proprietary to free and open-source tools. One of the biggest challenges forensic investigators face is ensuring that the formats used by sensors and collectors are compatible with the analysis tools chosen for the investigation. As the industry matures, there have been movements to standardize flow record exportation formats, but support for flow record export formats is still fragmented and many older tools exist that rely on older protocols. Forensic investigators also need to consider the placement and type of flow sensors and collectors throughout the network environment in order to determine whether the evidence needed is already being collected, or whether it is appropriate to configure additional flow record collection.

Ultimately, statistical flow record analysis is a very powerful field of study that will only grow in importance over the next decades. As flow record processing tools and techniques mature, investigators can expect to have increasingly powerful flow record analysis options at our fingertips.

5.7 Case Study: The Curious Mr. X

The Case: While a fugitive in Mexico, Mr. X remotely infiltrates the Arctic Nuclear Fusion Research Facility (ANFRF) lab network over the Internet. Sadly, Mr. X is not yet very stealthy.

Meanwhile ... Unfortunately for Mr. X, the ANFRF network is instrumented to capture flow record data. Security staff notice port scanning from his external IP address, 172.30.1.77, beginning at 2011-04-27 12:51:46 in the Cisco ASA flow record logs. His activities are discovered and analyzed ... by you!

Challenge: You are the forensic investigator. Your mission is to:

- Identify any compromised systems
- Determine what the attacker found out about the network architecture
- Evaluate the risk of data exfiltration

Since the Arctic Nuclear Fusion Research Facility stores a lot of confidential information, management is highly concerned about the risk of data exfiltration. If you find suspicious traffic, provide an analysis of the risk that Secret Information was compromised. Be sure to carefully justify your conclusions.

Network: The Arctic Nuclear Fusion Research Facility network consists of three segments:

- Internal network: 192.168.30.0/24
- DMZ: 10.30.30.0/24
- The “Internet”: 172.30.1.0/24 [Note that for the purposes of this case study, we are treating the 172.30.1.0/24 subnet as “the Internet.” In real life, this is a reserved nonroutable IP address space.]

Evidence: Security staff at ANFRF collect network flow data from a Cisco ASA switch/firewall that connects all three subnets at the perimeter. The flow record data is exported in Cisco’s NetFlow v9 format to a collector running nfcapd. (Note that to collect data in Cisco’s proprietary NetFlow v9 format, a specific fork of the nfdump suite, nfdump-1.5.8-NSEL, was used for collection and analysis.)

In addition, the Cisco ASA is also configured with a SPAN port that monitors the Internal and DMZ subnets. There is an Argus listener connected to the SPAN port, which retains flow record data in Argus format from the two subnets (192.168.30.0/24 and 10.30.30.0/24).

You are provided with two files containing data to analyze:

- **cisco-asa-nfcapd.zip**—A zip archive containing flow records from the perimeter Cisco ASA, stored by the nfdump collector utility (nfcapd) in 5-minute increments.

- **argus-collector.ra**—An Argus archive containing flow record data collected from the Internal and DMZ subnets via a SPAN port.

IMPORTANT NOTES: As you will see in the flow record data, there is a **time skew of approximately 8 seconds** between the Cisco ASA and the Argus listener. In addition, be aware that Network Address Translation (NAT) is used on this network. The DMZ IP address 10.30.30.20 translates to the external address 172.30.1.231, and the internal IP address 192.168.30.101 translates to the external address 172.30.1.227. Please note that the command output shown in the analysis had been modified of fit the page (in some cases, extraneous columns have been removed for brevity).

5.7.1 Analysis: First Steps

Let's begin by using nfdump to browse flow records relating to our known attacker system, 172.30.1.77, beginning at the time of interest (2011-04-27 12:51:46.130):

```
$ nfdump -R cisco-asa-nfcapd/ 'host 172.30.1.77'  
Date flow start      Proto  Src IP Addr:Port X-Src IP Addr:Port      Dst IP  
      Addr:Port   Event   XEvent Bytes  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:135 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:139 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:256 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:5900 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:3389 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:80  DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:995 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:143 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:1720 DENIED   I-ACL   0  
2011-04-27 12:51:46.130 TCP    172.30.1.77:44197  172.30.1.77:44197  ->  
      172.30.1.231:1025 DENIED   I-ACL   0
```

Notice that the source port on the attacker system remains the same, even as it makes fast connection attempts to different destination ports on the targeted system, 172.30.1.231. This is common behavior for an automated port scanner.

Using nfdump, we can determine which ports the attacker may have found open on the targeted system, 172.30.1.231. Notice that the firewall “DENIED” traffic to most targeted ports. The flows that were not DENIED by the firewall actually reached the targeted system and may have resulted in a response. In this case, the only port that did not result in a DENIED response from the firewall was TCP port 22 on 172.30.1.231, as shown below:

```
$ nfdump -R cisco-asa-nfcapd/ 'host 172.30.1.77'
Date flow start      Proto  Src IP Addr:Port X-Src IP Addr:Port      Dst IP
    Addr:Port   Event   XEvent Bytes
...
2011-04-27 12:51:47.330 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:443 DENIED I-ACL 0
2011-04-27 12:51:47.330 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:8080 DENIED I-ACL 0
2011-04-27 12:51:47.330 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:22 CREATE Ignore 0
2011-04-27 12:51:47.340 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:22 DELETE Deleted 0
2011-04-27 12:51:47.340 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:554 DENIED I-ACL 0
2011-04-27 12:51:47.340 TCP 172.30.1.77:44197 172.30.1.77:44197 ->
    172.30.1.231:21 DENIED I-ACL 0
...
...
```

5.7.2 External Attacker and Port 22 Traffic

Was there any more traffic on port 22 (TCP) relating to the attacker? In the nfdump command below, we filter the flow data to pull out only records pertaining to the attacker system and TCP port 22. Beginning at 2011-04-27 12:52:30.111, we see a series of connection attempts on port 22 of size 3,755 bytes. These occur in rapid succession (roughly every 6 seconds) until 2011-04-27 13:00:45.452.

```
$ nfdump -R cisco-asa-nfcapd/ 'host 172.30.1.77 and port 22'
Date flow start      Proto  Src IP Addr:Port X-Src IP Addr:Port      Dst IP
    Addr:Port   Event   XEvent Bytes
...
2011-04-27 13:00:16.672 TCP 172.30.1.77:56356 172.30.1.77:56356 ->
    172.30.1.231:22 DELETE Deleted 3755
2011-04-27 13:00:22.722 TCP 172.30.1.77:56358 172.30.1.77:56358 ->
    172.30.1.231:22 CREATE Ignore 0
2011-04-27 13:00:22.732 TCP 172.30.1.77:56357 172.30.1.77:56357 ->
    172.30.1.231:22 DELETE Deleted 3755
2011-04-27 13:00:29.532 TCP 172.30.1.77:56359 172.30.1.77:56359 ->
    172.30.1.231:22 CREATE Ignore 0
2011-04-27 13:00:29.532 TCP 172.30.1.77:56358 172.30.1.77:56358 ->
    172.30.1.231:22 DELETE Deleted 3755
2011-04-27 13:00:35.612 TCP 172.30.1.77:56360 172.30.1.77:56360 ->
    172.30.1.231:22 CREATE Ignore 0
2011-04-27 13:00:35.612 TCP 172.30.1.77:56359 172.30.1.77:56359 ->
    172.30.1.231:22 DELETE Deleted 3755
2011-04-27 13:00:41.962 TCP 172.30.1.77:56361 172.30.1.77:56361 ->
    172.30.1.231:22 CREATE Ignore 0
2011-04-27 13:00:41.962 TCP 172.30.1.77:56360 172.30.1.77:56360 ->
    172.30.1.231:22 DELETE Deleted 3603
2011-04-27 13:00:45.452 TCP 172.30.1.77:56361 172.30.1.77:56361 ->
    172.30.1.231:22 DELETE Deleted 3335
2011-04-27 13:01:00.133 TCP 172.30.1.77:56362 172.30.1.77:56362 ->
    172.30.1.231:22 CREATE Ignore 0
```

What does this pattern mean? First of all, by default TCP port 22 is assigned to SSH protocol by IANA.⁴⁵ SSH is a remote access service that is commonly targeted for brute-force

45. “Port Numbers,” July 7, 2011, <http://www.iana.org/assignments/port-numbers>.

password-guessing attacks when attackers find it exposed. Brute-force password-guessing attacks are typically automated and produce connection attempts at regular intervals with approximately the same, small amount of data transferred each time. While it is possible to set up the SSH server on a different port (and many people do, to obfuscate the attack surface), the pattern of this traffic (both frequency and port numbers) supports the hypothesis that this is a brute-force password-guessing attack against an externally accessible SSH service.

As you can see, at 2011-04-27 13:00:41.962, the bytes transferred suddenly changed. This was followed by one more very quick connection, and then at 2011-04-27 13:01:00.133 a flow was created between the attacker (172.30.1.77) and the target (172.30.1.231) on port TCP 22, which did not expire within the time frame of the recorded evidence. The length of this flow implies that the connection was probably successful. Of course, we will want to follow up by examining the internal flow record data and the authentication logs, if available.

From the timing, port number, and data transfer sizes of the flows, we hypothesize that these flow records indicate an automated brute-force password-guessing attack on an SSH server running on the targeted system, 172.30.1.231. After approximately 8 minutes, this attack probably resulted in a successful SSH login.

Now let's turn our attention to the internal Argus flow record data. Searching for traffic relating to the attacker, 172.30.1.77 and port 22, we see:

```
$ ra -z -nn -r argus-collector.ra - 'src host 172.30.1.77 and port 22'
  StartTime Proto SrcAddr Sport Dir DstAddr Dport TotPkts State
04-27-11 12:51:55 6 172.30.1.77.44197 -> 10.30.30.20.22 3 sSR
04-27-11 12:51:57 6 172.30.1.77.44208 -> 10.30.30.20.22 3 sSR
04-27-11 12:51:58 6 172.30.1.77.44209 -> 10.30.30.20.22 3 sSR
04-27-11 12:52:38 6 172.30.1.77.54348 -> 10.30.30.20.22 43 sSEfF
04-27-11 12:52:44 6 172.30.1.77.54349 -> 10.30.30.20.22 42 sSEfF
04-27-11 12:52:50 6 172.30.1.77.54350 -> 10.30.30.20.22 42 sSEfF
04-27-11 12:52:56 6 172.30.1.77.54351 -> 10.30.30.20.22 42 sSEfF
04-27-11 12:53:02 6 172.30.1.77.54352 -> 10.30.30.20.22 42 sSEfF
04-27-11 12:53:09 6 172.30.1.77.54353 -> 10.30.30.20.22 42 sSEfF
04-27-11 12:53:15 6 172.30.1.77.54354 -> 10.30.30.20.22 42 sSEfF
...
...
```

Recall that the internal NAT-ed address 10.30.30.20 corresponds with the external address 172.30.1.231 (they are the same server). Accounting for 8 seconds of time skew between the Cisco ASA and the Argus server, the Argus flow records show the same attempts to connect to TCP port 22 on the same server during the same time frame—correlating evidence. Using the `ra` utility's `-z` flag, we can also see the TCP state changes in the flow records, as described below:

```
$ man ra
...
-z  Modify status field to represent TCP state changes. The values
of the status field when this is enabled are:
's' - Syn Transmitted
'S' - Syn Acknowledged
'E' - TCP Established
'f' - Fin Transmitted (FIN Wait State 1)
'F' - Fin Acknowledged (FIN Wait State 2)
'R' - TCP Reset
```

From this information, we can see that the attacker first initiated a connection to 10.30.30.20:22 three times (sent a TCP SYN, received a SYN-ACK, sent a RST). The connection was aborted by the attacker before the TCP handshake was completed. This matches the behavior of a port scanner testing to see if port 22 is open using a TCP SYN scan (notably, the default nmap behavior when run with administrative privileges). Having received a TCP SYN/ACK back from the server, the attacker has verified that indeed the port is open:

```
$ ra -z -nn -r argus-collector.ra - 'src host 172.30.1.77 and port 22'
      StartTime Proto SrcAddr Sport Dir DstAddr Dport TotPkts State
04-27-11 12:51:55 6 172.30.1.77.44197 -> 10.30.30.20.22 3 sSR
04-27-11 12:51:57 6 172.30.1.77.44208 -> 10.30.30.20.22 3 sSR
04-27-11 12:51:58 6 172.30.1.77.44209 -> 10.30.30.20.22 3 sSR
```

Next, we see the series of short connections to port 22 approximately every 6 seconds. In the flows below, the attacker sent a TCP SYN packet, received a SYN ACK packet, established a full TCP handshake, sent a FIN packet, and received a FIN packet. This is a full, successful Layer 4 TCP communication, over which application data could have been sent. (Note that Argus, unlike nfdump, displays the number of bytes transferred *including* the lower-layer Ethernet frame and IP headers.)

```
$ ra -z -nn -r argus-collector.ra - 'src host 172.30.1.77 and port 22'
      StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
...
04-27-11 12:52:38 6 172.30.1.77.54348 -> 10.30.30.20.22 6609 sSEfF
04-27-11 12:52:44 6 172.30.1.77.54349 -> 10.30.30.20.22 6543 sSEfF
04-27-11 12:52:50 6 172.30.1.77.54350 -> 10.30.30.20.22 6543 sSEfF
04-27-11 12:52:56 6 172.30.1.77.54351 -> 10.30.30.20.22 6543 sSEfF
04-27-11 12:53:02 6 172.30.1.77.54352 -> 10.30.30.20.22 6543 sSEfF
04-27-11 12:53:09 6 172.30.1.77.54353 -> 10.30.30.20.22 6543 sSEfF
04-27-11 12:53:15 6 172.30.1.77.54354 -> 10.30.30.20.22 6543 sSEfF
...
...
```

Finally, starting at 04-27-11 13:00:44 (Argus time), we see two smaller, short, completed connections followed by a series of longer established TCP connections on port 22, with no FIN packets logged as part of the flow. These established TCP connections between 172.30.1.77 and 10.30.30.20:22 continue from 04-27-11 13:01:08 through 04-27-11 13:15:55, more than 15 minutes:

```
$ ra -z -n -r argus-collector.ra - 'src host 172.30.1.77 and port 22'
      StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
...
04-27-11 13:00:31 tcp 172.30.1.77.56358 -> 10.30.30.20.22 6543 sSEfF
04-27-11 13:00:38 tcp 172.30.1.77.56359 -> 10.30.30.20.22 6543 sSEfF
04-27-11 13:00:44 tcp 172.30.1.77.56360 -> 10.30.30.20.22 6127 sSEfF
04-27-11 13:00:50 tcp 172.30.1.77.56361 -> 10.30.30.20.22 5661 sSEfF
04-27-11 13:01:08 tcp 172.30.1.77.56362 -> 10.30.30.20.22 147296 sSE
04-27-11 13:02:11 tcp 172.30.1.77.56362 -> 10.30.30.20.22 77440 sSE
04-27-11 13:03:11 tcp 172.30.1.77.56362 -> 10.30.30.20.22 24508 sSE
04-27-11 13:04:14 tcp 172.30.1.77.56362 -> 10.30.30.20.22 447368 sSE
04-27-11 13:05:24 tcp 172.30.1.77.56362 -> 10.30.30.20.22 228032 sSE
04-27-11 13:06:24 tcp 172.30.1.77.56362 -> 10.30.30.20.22 47906 sSE
04-27-11 13:07:55 tcp 172.30.1.77.56362 -> 10.30.30.20.22 1108 sSE
```

```
$ ra -z -n -r argus-collector.ra - 'src host 172.30.1.77 and port 22'
  StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
...
04-27-11 13:08:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1124  sSE
04-27-11 13:09:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1092  sSE
04-27-11 13:10:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22     80742  sSE
04-27-11 13:11:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1124  sSE
04-27-11 13:12:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1124  sSE
04-27-11 13:13:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1092  sSE
04-27-11 13:14:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1108  sSE
04-27-11 13:15:55  tcp 172.30.1.77.56362 -> 10.30.30.20.22      1108  sSE
```

The sudden halt of short, regular connection attempts, followed by larger, longer flow records with more data associated, fits with the pattern of a brute-force password-guessing attack that was successful and was subsequently followed by an extended SSH login through at least 13:15:55. This pattern is illustrated using the Argus utility “ragraph” in Figure 5–5. The following command was used to generate the graph in Figure 5–5:

```
$ ragraph bytes -M 1s -r argus-collector.ra -t 2011/04/27.12:52+10m \
-ttitle "Traffic Snippet from 172.30.1.77 on Port 22 (TCP)" \
-'src host 172.30.1.77 and dst host 10.30.30.20 and port 22'
```

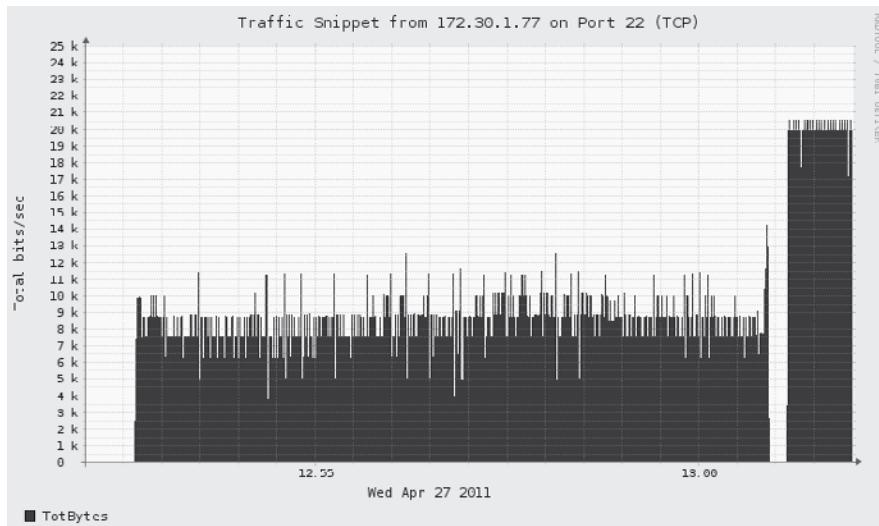


Figure 5–5. A graph of Argus flow record data from the attacker (172.30.1.77) to a victim server in the DMZ (10.30.30.20) on port 22. Notice the 8-minute automated brute-force password-guessing attack, followed by a short break, and then followed by a higher-bandwidth exchange of data (implying that the attack was probably successful).

5.7.3 The DMZ Victim—10.30.30.20 (aka 172.30.1.231)

Now let's get more information about the activities of 10.30.30.20, the targeted DMZ system.

The first noticeably different SSH connection from 172.30.1.77 to 10.30.30.20 occurred at 04-27-11 13:01:08 (in the Argus flow record data). The size of the data transferred

during this flow was substantially larger than previous attempts, likely indicating a successful connection. More than a minute later, we see another connection on port 22. The slower timing of these connections also implies a possible manual connection after the automated brute-force tool completed.

Subsequently, we see 10.30.30.20 communicating over port 80 to an external IP address, 91.189.92.166.80. Then, at 13:03:31, it began sending TCP SYN packets to internal systems on ports 80 and 443. At 13:03:44, these packets were targeted at sequential, incrementing IP addresses—very common for an automated port *sweep*. Notice the sequential destination IP addresses all on the same ports (80 and later 443). The system 10.30.30.20 continued to send TCP SYN packets to ports 80 and 443 on the internal networks until 13:03:49. In most cases, the targeted systems did not respond with a TCP SYN/ACK packet—or any packet at all—as shown in the “State” column:

```
$ ra -z -nnr argus-collector.ra - 'host 10.30.30.20 and not udp and not port
514 and not port 53'
   StartTime Proto  SrcAddr  Sport  Dir  DstAddr  Dport  TotPkts  State
04-27-11 13:02:57 6 10.30.30.20.36339 -> 91.189.92.166.80      508  SEF
04-27-11 13:02:58 6 10.30.30.20.35194 -> 91.189.92.161.80     1221  sSEF
04-27-11 13:03:11 6 172.30.1.77.56362 -> 10.30.30.20.22      214  sSE
04-27-11 13:03:31 6 10.30.30.20.60833 -> 10.30.30.10.80      1    s
04-27-11 13:03:33 6 10.30.30.20.32907 -> 10.30.30.10.80      1    s
04-27-11 13:03:33 6 10.30.30.20.53778 -> 10.30.30.10.443     1    s
04-27-11 13:03:33 6 10.30.30.20.53808 -> 10.30.30.10.443     1    s
04-27-11 13:03:44 6 10.30.30.20.41339 -> 192.168.30.1.80      1    s
04-27-11 13:03:44 6 10.30.30.20.42486 -> 192.168.30.2.80      1    s
04-27-11 13:03:44 6 10.30.30.20.48181 -> 192.168.30.3.80      1    s
04-27-11 13:03:44 6 10.30.30.20.56116 -> 192.168.30.4.80      1    s
04-27-11 13:03:44 6 10.30.30.20.58905 -> 192.168.30.5.80      1    s
04-27-11 13:03:44 6 10.30.30.20.37560 -> 192.168.30.6.80      1    s
04-27-11 13:03:44 6 10.30.30.20.36524 -> 192.168.30.7.80      1    s
04-27-11 13:03:44 6 10.30.30.20.35544 -> 192.168.30.8.80      1    s
04-27-11 13:03:44 6 10.30.30.20.43568 -> 192.168.30.9.80      1    s
...
04-27-11 13:03:49 6 10.30.30.20.48728 -> 192.168.30.86.443     1    s
04-27-11 13:03:49 6 10.30.30.20.55470 -> 192.168.30.116.443     1    s
04-27-11 13:03:49 6 10.30.30.20.48601 -> 192.168.30.119.443     1    s
04-27-11 13:03:49 6 10.30.30.20.58087 -> 192.168.30.87.443     1    s
```

Which systems *did* respond to the port sweep? We can easily pick these needles out of the haystack using a filter that searches for flow records where the number of packets sent from the *targeted* systems back to the port scanner was greater than zero:

```
$ ra -z -t 2011/04/27.13:03:31+18s -nnr argus-collector.ra -s stime saddr
sport dir daddr dport spkts dpkts state - 'host 10.30.30.20 and port (80
or 443) and tcp and dst pkts gt 0'
   StartTime  SrcAddr  Sport Dir  DstAddr  Dport SrcPkt DstPkt State
04-27-11 13:03:47 10.30.30.20.39307 -> 192.168.30.30.80  2      2    SR
04-27-11 13:03:47 10.30.30.20.53512 -> 192.168.30.90.80  3      3    SR
```

From this, we see that the two systems 192.168.30.30 and 192.168.30.90 sent TCP RST packets back to 10.30.30.20. This indicates that port 80 was not open on those systems, but the systems were in fact alive.

At 04-27-11 13:03:49, 10.30.30.20 changed behavior and began sending SYN packets exclusively to a range of ports on the two systems that had responded, 192.168.30.30 and 192.168.30.90. The pattern lasted only through 13:03:50 (2 seconds). This behavior is indicative of a *port scan* against the two hosts, 192.168.30.30 and 192.168.30.90, as shown below.

```
$ ra -z -nnr argus-collector.ra - 'host 10.30.30.20'
...
      StartTime Proto SrcAddr  Sport  Dir   DstAddr  Dport TotBytes
      State
04-27-11 13:03:49 6 10.30.30.20.52140 -> 192.168.30.90.143    262  sR
04-27-11 13:03:49 6 10.30.30.20.45717 -> 192.168.30.30.143    268  sR
04-27-11 13:03:49 6 10.30.30.20.60813 -> 192.168.30.90.1723   262  sR
04-27-11 13:03:49 6 10.30.30.20.39655 -> 192.168.30.30.1723   268  sR
04-27-11 13:03:49 6 10.30.30.20.35499 -> 192.168.30.90.5900   262  sR
04-27-11 13:03:49 6 10.30.30.20.55132 -> 192.168.30.30.5900   268  sR
04-27-11 13:03:49 6 10.30.30.20.43935 -> 192.168.30.90.8888   262  sR
04-27-11 13:03:49 6 10.30.30.20.49207 -> 192.168.30.30.8888   268  sR
04-27-11 13:03:49 6 10.30.30.20.59231 -> 192.168.30.90.8080   262  sR
04-27-11 13:03:49 6 10.30.30.20.58646 -> 192.168.30.30.8080   268  sR
04-27-11 13:03:49 6 10.30.30.20.58594 -> 192.168.30.90.3389   262  sR
04-27-11 13:03:49 6 10.30.30.20.35855 -> 192.168.30.30.3389   268  sR
...
04-27-11 13:03:50 6 10.30.30.20.45320 -> 192.168.30.90.24800  262  sR
04-27-11 13:03:50 6 10.30.30.20.46080 -> 192.168.30.30.24800  268  sR
04-27-11 13:03:50 6 10.30.30.20.53337 -> 192.168.30.90.9207  262  sR
04-27-11 13:03:50 6 10.30.30.20.58434 -> 192.168.30.30.9207  268  sR
...

```

We can use a quick Bash shell command to sort and count the destination ports targeted. As you can see below, 10.30.30.20 targeted exactly 1,000 different port numbers. (Experienced penetration testers will recall that by default Nmap, the world's most popular port scanner, “scans the most common 1,000 ports for each protocol.”⁴⁶)

```
$ ra -z -t 2011/04/27.13:03:49+2s -nnr argus-collector.ra -s dport - 'host
10.30.30.20 and dst host (192.168.30.30 or 192.168.30.90)' | grep -v Dport
| sort -u | wc -l
1000
```

Which ports did the attacker find open on the targeted internal systems, 192.168.30.30 and 192.168.30.90? By filtering for TCP SYN/ACK packets, we can see that the attacker found 192.168.30.90:22 (TCP), 192.168.30.30:22 (TCP), and 192.168.30.30:514 (TCP) open, as shown below:

```
$ ra -z -t 2011/04/27.13:03:49+2s -nnr argus-collector.ra - 'host
10.30.30.20 and dst host (192.168.30.30 or 192.168.30.90) and synack'
      StartTime Proto SrcAddr  Sport  Dir   DstAddr  Dport TotBytes State
04-27-11 13:03:49 6 10.30.30.20.46692 -> 192.168.30.90.22      560  sSER
04-27-11 13:03:49 6 10.30.30.20.36307 -> 192.168.30.30.22      560  sSER
04-27-11 13:03:49 6 10.30.30.20.39370 -> 192.168.30.30.514     560  sSER
```

46. Gordon “Fyodor” Lyon, “Port Specification and Scan Order,” NMAP, 2011, <http://nmap.org/book/man-port-specification.html>.

Next, between 13:04:09 through 13:04:14, 10.30.30.20 began sending TCP SYN packets to sequential IP addresses on the 192.168.30.1/24 network, but only targeted at TCP port 3389, as shown below:

	StartTime	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	TotBytes	State
04-27-11	13:04:09	6	10.30.30.20.56212		->	192.168.30.1.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.56255		->	192.168.30.2.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.52168		->	192.168.30.3.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.45057		->	192.168.30.4.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.32898		->	192.168.30.5.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.55453		->	192.168.30.6.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.48122		->	192.168.30.7.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.39936		->	192.168.30.8.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.51199		->	192.168.30.9.3389		74	s
04-27-11	13:04:09	6	10.30.30.20.40066		->	192.168.30.10.3389		74	s
...									
04-27-11	13:04:14	6	10.30.30.20.47058		->	192.168.30.221.3389		74	s
04-27-11	13:04:14	6	10.30.30.20.47002		->	192.168.30.232.3389		74	s
04-27-11	13:04:14	6	10.30.30.20.40560		->	192.168.30.226.3389		74	s

This is indicative of a port sweep targeting TCP port 3389 (commonly associated with Microsoft's Remote Desktop Protocol). We can see which systems responded by again searching for TCP SYN/ACK responses within that time frame:

```
$ ra -z -t 2011/04/27.13:04:09+6s -nnr argus-collector.ra - 'host
10.30.30.20 and net 192.168.30.0/24 and synack'
StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
04-27-11 13:04:13 6 10.30.30.20.41371 -> 192.168.30.100.3389 560 sSER
04-27-11 13:04:13 6 10.30.30.20.33786 -> 192.168.30.101.3389 568 sSER
04-27-11 13:04:13 6 10.30.30.20.57473 -> 192.168.30.102.3389 568 sSER
```

The attacker would have found that three systems on the 192.168.30.0/24 network have TCP port 3389 open: 192.168.30.100, 192.168.30.101, and 192.168.30.102.

After the port sweep of port 3389 is completed, we see a series of flows from the DMZ system 10.30.30.20 to 192.168.30.101 on port 3389 (RDP), spanning more than 11 minutes:

```
$ ra -z -nnr argus-collector.ra - 'host 10.30.30.20 and port 3389'
StartTime Proto SrcAddr Sport Dir DstAddr Dport TotBytes State
...
04-27-11 13:04:32 6 10.30.30.20.34187 -> 192.168.30.101.3389 568 sSER
04-27-11 13:04:32 6 10.30.30.20.34188 -> 192.168.30.101.3389 1260 sSEFF
04-27-11 13:04:54 6 10.30.30.20.34189 -> 192.168.30.101.3389 852450 SSE
04-27-11 13:05:54 6 10.30.30.20.34189 -> 192.168.30.101.3389 285408 SSE
04-27-11 13:06:54 6 10.30.30.20.34189 -> 192.168.30.101.3389 10032 SSE
04-27-11 13:07:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2128 SSE
04-27-11 13:08:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2170 SSE
04-27-11 13:09:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2116 SSE
04-27-11 13:10:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 126806 SSE
04-27-11 13:11:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2180 SSE
04-27-11 13:12:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2164 SSE
04-27-11 13:13:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2112 SSE
04-27-11 13:14:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2126 SSE
04-27-11 13:15:55 6 10.30.30.20.34189 -> 192.168.30.101.3389 2140 SSE
```

Recall that during this same time frame, there were also likely active SSH connections between the external attacker, 172.30.1.77, and the DMZ victim, 10.30.30.20. Based on the flow record data, we have seen that while this was occurring, the DMZ victim system, 10.30.30.20, conducted port scans and port sweeps of the internal ANFRF network, and then subsequently connected to port 3389 (RDP) on an internal system, 192.168.30.101.

5.7.4 The Internal Victim—192.30.1.101

Now let's filter for traffic relating to 192.168.30.101. In the output below, we again see the internal port scanning traffic, followed by the port 3389 connection. Then at 13:05:33 we see an outbound connection directly from 192.168.30.101 to our attacker system 172.30.1.77 on TCP port 21 (FTP)!

```
$ ra -z -nn -r argus-collector.ra - 'host 192.168.30.101'
      StartTime Proto      SrcAddr   Sport Dir    DstAddr   Dport TotBytes State
...
04-27-11 13:03:47 6     10.30.30.20.47537 -> 192.168.30.101.80       148     s
04-27-11 13:03:47 6     10.30.30.20.57498 -> 192.168.30.101.443      148     s
04-27-11 13:03:49 6     10.30.30.20.57963 -> 192.168.30.101.443      148     s
04-27-11 13:03:49 6     10.30.30.20.48054 -> 192.168.30.101.80       148     s
04-27-11 13:04:04 17    192.168.30.101.1600 -> 192.168.30.30.514     13610   INT
04-27-11 13:04:13 6     10.30.30.20.33786 -> 192.168.30.101.3389     568    sSER
04-27-11 13:04:32 6     10.30.30.20.34187 -> 192.168.30.101.3389     568    sSER
04-27-11 13:04:32 6     10.30.30.20.34188 -> 192.168.30.101.3389     1260   sSEFF
04-27-11 13:04:54 6     10.30.30.20.34189 -> 192.168.30.101.3389     852450  SSE
04-27-11 13:05:06 17    192.168.30.101.1600 -> 192.168.30.30.514     4108   INT
04-27-11 13:05:33 6     192.168.30.101.1603 -> 172.30.1.77.21        1221   SSE
04-27-11 13:05:54 6     10.30.30.20.34189 -> 192.168.30.101.3389     285408  SSE
04-27-11 13:06:50 6     192.168.30.101.1603 -> 172.30.1.77.21        1254   SSE
04-27-11 13:06:50 6     172.30.1.77.20 -> 192.168.30.101.5002     558    sSEFF
...
...
```

File Transfer Protocol (FTP) is used for transferring files between systems. This traffic may indicate that data was transferred from our internal system, 192.168.30.101, to the attacker, 172.30.1.77. Filtering for FTP-related traffic (default ports TCP 20/21), we see:

```
$ ra -z -n -r argus-collector.ra -s stime saddr sport dir daddr dport sbytes
  dbytes state - 'host 192.168.30.101 and port (20 or 21) and tcp'
      StartTime      SrcAddr   Sport Dir    DstAddr   Dport SBytes DBBytes State
04-27-11 13:05:33 192.168.30.101.1603 ->          172.30.1.77.21  673    548   SSE
04-27-11 13:06:50 192.168.30.101.1603 ->          172.30.1.77.21  630    624   SSE
04-27-11 13:06:50          172.30.1.77.20 -> 192.168.30.101.5002  348    210   sSEFF
04-27-11 13:07:03          172.30.1.77.20 -> 192.168.30.101.5003  998   16874  sSEFF
04-27-11 13:11:14 192.168.30.101.1603 <?>          172.30.1.77.21  180    128   EfFR
```

In the port 20 flow at 13:07:03, there were 16,874 bytes of data exported from the internal system, 192.168.30.101, to the external attacker. This is certainly large enough for a decent size file to be exported. What is the risk that data was exfiltrated? High!

Let's see if we can correlate the internal Argus data relating to FTP with the Cisco ASA flow records. Let's go back to our original nfdump data. (Remember that the internal address 192.168.30.101 is NAT-ed and corresponds with the external address, 172.30.1.227. These are the same system!) Adjusting for our 8-second time skew, we can indeed see the same connections in the output below. Note that the flow record data from the Cisco ASA

shows the Layer 4 payload size (TCP payload size only rather than Ethernet frame size), and so it reports less data transferred than the Argus utility, which includes lower-layer frame and packet headers in its reported data transferred.

It is likely that the last flow record on port 20 indicates an outbound file transfer from 172.30.1.227 (192.168.30.101) to the attacker (172.30.1.77) of a file with a size of 15,872 bytes (or multiple files adding up to that size).

```
$ nfdump -R cisco-asa-nfcapd/ 'host 172.30.1.77 and (port 20 or port 21)'
Date flow start      Proto   Src IP Addr:Port X-Src IP Addr:Port     Dst IP
    Addr:Port      Event   XEvent Bytes
...
2011-04-27 13:05:24.739 TCP 192.168.30.101:1603 172.30.1.227:1603 ->
    172.30.1.77:21 CREATE Ignore 0
2011-04-27 13:06:42.070 TCP 172.30.1.77:20 172.30.1.77:20 ->
    172.30.1.227:5002 CREATE Ignore 0
2011-04-27 13:06:42.081 TCP 172.30.1.77:20 172.30.1.77:20 ->
    172.30.1.227:5002 DELETE Deleted 10
2011-04-27 13:06:55.291 TCP 172.30.1.77:20 172.30.1.77:20 ->
    172.30.1.227:5003 CREATE Ignore 0
2011-04-27 13:06:55.631 TCP 172.30.1.77:20 172.30.1.77:20 ->
    172.30.1.227:5003 DELETE Deleted 15872
2011-04-27 13:11:05.767 TCP 192.168.30.101:1603 172.30.1.227:1603 ->
    172.30.1.77:21 DELETE Deleted 631
```

5.7.5 Timeline

Based on our flow record analysis, we can create a working hypothesis of what likely transpired. Of course, we must make some educated guesses in order to interpret the activity we have seen. However, our analysis is strongly supported by the evidence, references, and experience. With that in mind, here is a timeline of events related to the incident at ANFRF April 27, 2011:

[Please note that the times are adjusted to match the internal Argus flow record data, which is 8 seconds ahead of the Cisco ASA.]

- **12:49:33**—Flow captures begin.
- **12:51:54**—Port scanning begins from 172.30.1.77 (attacker) against 172.30.1.231 (DMZ victim). The attacker likely found that port 22 (TCP) was open on the DMZ victim system.
- **12:52:38**—172.30.1.77 begins likely brute-force password-guessing attack against an SSH server on DMZ victim.
- **13:00:45**—172.30.1.77 ends likely brute-force password-guessing attack.
- **13:01:08**—172.30.1.77 begins extended connection to SSH port on DMZ victim.
- **13:03:31**—DMZ victim begins port sweep of internal and DMZ networks on TCP ports 80 and 443. Two systems on the internal network responded: 192.168.30.30 and 192.168.30.90.
- **13:03:49**—DMZ victim ends port sweep of internal and DMZ networks on TCP ports 80 and 443.

- **13:03:49**—DMZ victim begins port scan of 192.168.30.30 and 192.168.30.90. 1,000 ports were targeted. The attacker found 192.168.30.90:22 (TCP), 192.168.30.30:22 (TCP), and 192.168.30.30:514 (TCP) open.
- **13:03:50**—DMZ victim ends port scan of 192.168.30.30 and 192.168.30.90.
- **13:04:09**—DMZ victim begins port sweep of internal network, 192.168.30.0/24, on port 3389. Three systems on the 192.168.30.0/24 network appeared to have TCP port 3389 open: 192.168.30.100, 192.168.30.101, and 192.168.30.102.
- **13:04:14**—DMZ victim ends port sweep of internal network targeting port 3389.
- **13:04:32**—DMZ victim begins a series of connections to 192.168.30.101 on port 3389 (TCP). This port is commonly associated with RDP, a remote connection protocol commonly used on Microsoft Windows systems.
- **13:05:33**—192.168.30.101 begins outbound connections on port 21/TCP (FTP) to the attacker, 172.30.1.77.
- **13:07:03**—192.168.30.101 conducts a particularly large outbound data transfer to 172.30.1.77, with a likely file size of 15,872 bytes.
- **13:15:55**—Last flow record from 172.30.1.77 to DMZ victim (port 22/TCP). Connection still active.
- **13:15:55**—Last flow record from DMZ victim to 192.168.30.101 (port 3389/TCP). Connection still active.
- **13:16:09**—Flow captures end.

5.7.6 Theory of the Case

Now that we have put together a timeline of events, let's summarize our theory of the case. Again, this is a working hypothesis strongly supported by the evidence, references, and experience:

- The attacker (172.30.1.77) conducted a port scan of the DMZ victim 172.30.1.231 (10.30.30.20).
- The attacker found that TCP port 22 (SSH) was exposed on the targeted DMZ victim 172.30.1.231.
- The attacker (172.30.1.77) conducted a brute-force password-guessing SSH attack on the DMZ victim, 172.30.1.231 (10.30.30.20). After approximately 8 minutes, this attack was successful.
- The attacker logged into the DMZ victim 172.30.1.231 (10.30.30.20) using SSH and conducted a port scan of the internal network.
- Two systems, 192.168.30.30 and 192.168.30.90, were responsive and had port 22/TCP open.

- From the DMZ victim 172.30.1.231 (10.30.30.20), the attacker also conducted a port sweep of the internal network for open port 3389 (RDP).
- Three systems had port 3389 open: 192.168.30.100, 192.168.30.101, and 192.168.30.102.
- The attacker, pivoting through the DMZ victim 172.30.1.231 (10.30.30.20), logged into 192.168.30.101 via RDP.
- On 192.168.30.101 (172.30.1.227), the attacker used FTP to connect outbound to 172.30.1.77.
- The attacker transferred a file from the internal system 192.168.30.101 (172.30.1.227) to the attacker's system, 172.30.1.77.

5.7.7 Response to Challenge Questions

Now, let's answer the investigative questions posed to us at the beginning of the case.

- **Identify any compromised systems**

The evidence suggests that at least two systems were compromised: the DMZ victim (10.30.30.20) and an internal system (192.168.30.101). The attacker compromised the DMZ victim through a brute-force SSH password-guessing attack, and was able to leverage access to the DMZ in order to access a system on the internal network. The internal system, 192.168.30.101, did not appear to be subjected to a password-guessing attack before the RDP connection, so it may be the case that the attacker was able to reuse the same password in order to log into both systems remotely.

- **Determine what the attacker found out about the network architecture**

Based on the port scanning activity and responses that we have seen, we can expect that the attacker found out that:

- Firewall rules allow connections via TCP port 22 to the DMZ victim, 172.30.1.231 (10.30.30.20).
- The ANFRF has at least two subnets: 10.30.30.0/24 and 192.168.30.0/24.
- Firewall rules also allow access from the DMZ to various systems on the internal network for TCP ports 22, 80, 443, 514, and 3389.
- FTP traffic (TCP port 20/21) is allowed outbound from the internal network.

- **Evaluate the risk of data exfiltration**

Based on our analysis, the risk of data exfiltration is high. The flow record data includes strong evidence that an outbound FTP connection was made from an internal system to the external attacker, and that a significant amount of data was transferred over this connection.

5.7.8 Next Steps

Using only flow record data, we have learned a significant amount about the events that likely transpired on April 27, 2011. As forensic investigators, where do we go from here? Although this depends on the resources and goals of the organization initiating the investigation,

often the next steps involve a two-pronged attack of gathering additional evidence and containing/eradicating any compromise. In this case, common next steps might include:

- **Containment/Eradication** What can ANFRF staff do to contain the damage and prevent further compromise? Here are a few options:
 - Change all passwords that may have been compromised. This includes any passwords related to the DMZ victim, 10.30.30.20, and the internal system 192.168.30.101. Organizations that “play it safe” may choose to reset *all* passwords.
 - Rebuild the two compromised systems, 10.30.30.20 and 192.168.30.101 (after gathering evidence from them as needed).
 - Tighten firewall rules to more strictly limit access from the DMZ to the internal network. Is there really a need for DMZ systems to access internal systems? Limit this access to the greatest extent possible.
 - Block outbound connections on TCP ports 20/21 (FTP), and any other ports that are not needed.
 - Remove or restrict access to the externally exposed SSH service, if possible.
 - Consider using two-factor authentication for external access to the network. Single-factor authentication is risky and leaves the system at much higher risk of compromise due to brute-force password guessing, as we have seen.
- **Additional Sources of Evidence** Here are some high-priority potential sources of additional evidence that might be useful in the case:
 - Central Logging Server—Based on the flow record data, there may be syslog traffic (port 514) sent over the network. (Although IANA has only assigned UDP 514 to syslog, many organizations have switched to using TCP for transport of syslog data, and frequently choose to use TCP 514 for syslog.) It would be wise to inquire with local system administrators to determine if there is a central logging server. If so, logs from this system can be gathered and analyzed. In particular, authentication logs from the two compromised systems (10.30.30.20 and 192.168.30.101) would be of great interest.
 - Firewall logs—The firewall logs may provide more granularity regarding denied and allowed network activity relating to the incident.
 - Hard drives of compromised systems—Forensic analysis of the compromised system hard drive may reveal detailed information about the attacker’s activities, or at the very least, allow for an inventory of confidential information that may have been compromised.

This page intentionally left blank

Chapter 6

Wireless: Network Forensics Unplugged

“You are only coming through in waves . . .”
—Pink Floyd¹

Wireless networks are everywhere. They exist in enterprises, homes, subways, buses, cafes, anywhere people go. For an investigator, wireless networks can be a boon or a wretched headache. In the best scenarios, wireless networks provide easy access to client traffic, a convenient entry point into the network, and extensive access logs.

On the other hand, many wireless devices and access points do not retain access logs, especially when deployed with default configurations. Client systems tend to be connected transiently, and their physical locations are often unknown or difficult to pinpoint. Employees or attackers can set up rogue wireless networks, unbeknownst to central IT staff, for the purposes of convenience or covert data exfiltration. Wireless devices in the enterprise can pose a major challenge for forensic investigators—especially since these same devices are often accessible from the parking lot of a facility and can represent a major loophole in organizational security defenses.

Wireless devices have exploded in popularity during the past decade. It is impossible to enumerate all the wireless devices that exist in even a single person’s house, let alone a large corporate environment. Individuals often walk around carrying multiple wireless devices, such as cell phones, iPads, laptops, Bluetooth headsets, and GPS tracking devices.

Common types of wireless devices and networks include:

- AM/FM radios
- Cordless phones
- Cell phones
- Bluetooth headsets
- Infrared devices, such as TV remotes

1. R. Waters and D. Gilmour, “Comfortably Numb,” *The Wall* (EMI, 1979).

- Wireless doorbells
- Zigbee devices, such as HVAC, thermostat, lighting, and electrical controls
- Wi-Fi (802.11)—LAN networking over RF
- WiMAX (802.16)—“last-mile” broadband²

Why investigate wireless networks? Here are some examples of cases involving wireless networks:

- Recover a stolen laptop by tracking it on the wireless network.
- Identify rogue wireless access points that have been installed by insiders for convenience or to bypass enterprise security.
- Investigate malicious or inappropriate activity that occurred via a wireless network.
- Investigate attacks on the wireless network itself, including denial-of-service, encryption-cracking, and authentication bypass attacks.

In addition, you may find yourself capturing wireless packets, investigating wireless routers and switches, or conducting a forensic analysis relating to any other topic in this book *where the physical layer happens to be air rather than a cable or fiber*.

Wireless networks, particularly those based on IEEE 802.11 (“Wi-Fi”) standards, have proliferated in the last decade. These days almost no enterprise is without them. As a result, every network forensic investigator should be prepared to handle wireless evidence.

With computing devices getting smaller and more mobile, we are increasingly reliant upon wireless connectivity. The proliferation of mobile devices is putting pressure on government and enterprises alike to expand the availability of network access. In the United States, the Federal Communications Commission (FCC) has been instructed to facilitate ubiquitous broadband access.

If you calculate the expense of a network deployment versus the number of potential endpoints that can be supported, wireless networks are clearly the economical option. In general, it is cheaper to deploy a wireless network than run all the cables necessary for a wired infrastructure. This is especially important in sparsely populated areas where cables might have to be run long distances or over rough terrain, and also in high-density areas where older buildings were not designed with modern wiring needs in mind. As a result, wireless networks are being deployed in nearly every type of environment, often replacing or expanding existing network access.

In this chapter, we focus our attention on 802.11 “Wi-Fi” networks specifically. This is because Wi-Fi networks are extremely common both in the enterprise and at home, and we can leverage many of our previously discussed forensic techniques on 802.11 Wi-Fi networks. Many of the concepts we cover in this chapter are also applicable to other types of wireless devices and networks.

2. Ian Mansfield, “WiMAX Companies Receive \$504 Million in Funding for Last Mile Broadband Projects,” *Cellular-News*, October 20, 2010, <http://www.cellular-news.com/story/45995.php>.

6.1 The IEEE Layer 2 Protocol Series

As previously discussed in Chapter 3, “Evidence Acquisition,” the IEEE has published a series of international standards (“802.11”) for Wireless Local Area Network (WLAN) communication. These standards specify protocols for WLAN traffic in the 2.4, 3.7, and 5 GHz frequency ranges. The term “Wi-Fi” is used to refer to certain types of RF traffic, which include the IEEE 802.11 standards. For more details about passive evidence acquisition of 802.11 traffic, please see Section 3.1.2, “Radio Frequency” in Chapter 3.

As analysts of network traffic, it is easy to get in the habit of assuming that all of the protocols that we care about are described in an IETF RFC. It is worth remembering that not all standards are sponsored by the IETF (see discussions in Chapter 4). Many of the core network protocols, especially at Layer 2, were instead proposed and published by IEEE standards groups—particularly the 802 series, which covers Ethernet (802.3), trunking (802.1q), LAN-based authentication (802.1X), and various aspects of Wi-Fi (802.11) including both WEP- and WPA-based encryption standards.

6.1.1 Why So Many Layer 2 Protocols?

You may be wondering why we need so many different protocols for Layer 2 functionality. Often, network forensic students ask, “*Why didn’t we just use existing Ethernet standards over RF just like we did over copper?*” Radio frequency and copper simply don’t have the same physical characteristics, and so signals sent across them don’t behave the same way! In order to insulate Layer 3 protocols (chiefly IP) from those differences in physical properties, we need intermediary protocols to allow a uniform interface to the network layer, regardless of the physical media.

We’ve been talking about how a wireless access point is essentially just a Layer 2 hub, but it’s a little bit more complicated than that. A physical copper hub can be relied upon to deliver signals from each station to every other station.

For forensic investigators, it is important to realize that if you are capturing traffic from a wireless network, there may well be stations actively participating in the network that you cannot overhear from your vantage point, due to signal strength (unlike on wired media, where voltages propagate much more reliably through copper or fiber cables). This simple fact has far-reaching effects on both data link-layer protocols themselves and forensic analysis of the wireless evidence.

While Ethernet (802.3) is designed to use the “carrier sense multiple access with collision detection” (CSMA/CD) method, the 802.11 wireless protocols we discuss use “carrier sense multiple access with collision avoidance” (CSMA/CA). We briefly review these concepts in the next sections.

6.1.1.1 CSMA/CD

Ethernet, designed for wired networks, is a protocol based on CSMA/CD. What this means is that all stations on the network share the same medium—typically a star-shaped but essentially physically contiguous piece of copper. A single conductor can only transmit one single signal at a time, by raising the voltage on the line (a “one”) or reducing the voltage

on the line (“a zero”). In other words, only one station can be using the “multiple access” medium at the same time. (There may be other more sophisticated and multiplexing ways to encode signals with electrons on copper, but that’s how Ethernet does it.) Meanwhile, it must be *all* stations’ responsibility to make sure that the wire is live and in use (the “carrier sense” part of CSMA) and to detect if any two stations are trying to use it at the same time (the “collision detection” part).

This isn’t always easy, as electrons don’t propagate down copper instantaneously, so two stations on the farthest ends of the circuit could commence sending at roughly the same time without realizing that someone else down the wire was trying to talk too. The specified result is that the first station that detects overlapping signals should send out a special “jamming” signal to all stations, essentially informing everyone that they need to stop talking and try again, but only after selecting at random a time increment to wait (so that they don’t all try at once again).

6.1.1.2 CSMA/CA

In wireless LANs, collisions cannot be reliably detected by the sender. On a wire, the high and low voltages will eventually propagate to every station, so long as they remain physically connected. In contrast, on a wireless network, it is entirely common to have multiple stations sharing the same frequency and channel, even if each station is not necessarily capable of detecting all signals from its peers. As long as a station can reliably send and receive signals to and from the access point, it can participate in the network. A station that can communicate with the access point but not other stations is referred to as a “hidden node.”

Since collisions cannot be reliably detected over radio frequency, 802.11 wireless networks are designed to use collision *avoidance* techniques by decreasing the likelihood that two stations will attempt to communicate at the same time. As described in the IEEE 802.11-2007 standard, before transmission, a station on the wireless network listens to determine if the transmission medium is idle. If it is busy, the station waits until the end of the current transmission, and then it waits an additional random amount of time before attempting to send traffic. This helps to reduce the likelihood that all stations will transmit at once. In addition, stations can also exchange control frames of message type “request-to-send” and “clear-to-send” to help avoid collisions, as described further in Section 6.1.2.1, “802.11 Frame Types.”³

6.1.2 The 802.11 Protocol Suite

The IEEE developed the 802.11 protocol suite as a standard for data link-layer transmission over wireless physical media, including the radio and infrared frequency spectra. 802.11 is designed to incorporate CSMA/CA, in keeping with the needs of the wireless physical medium.

3. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” (June 12, 2007): 251, <http://standards.ieee.org/getieee802/download/802.11-2007.pdf> (accessed December 31, 2011).

6.1.2.1 802.11 Frame Types

The 802.11 protocol suite defines different types of frames. For forensic investigators, different types of frames contain different types of evidence, as we will see.

There are three types of 802.11 frames:

- **Management Frames**—Govern communications between stations, except flow control;
- **Control Frames**—Support flow control over a variably available medium (such as RF);
- **Data Frames**—Encapsulate the Layer 3+ data that moves between stations actively engaged in communication on a wireless network.

802.11 is a complicated Layer 2 protocol in many respects. Figure 6–1 is a chart that shows the 802.11 frame’s data structure. You’d be forgiven for looking at the chart in Figure 6–1 and wondering to yourself what the fields meant—especially since there are several 6-byte locations for frame addresses. The purpose of each “address” field is dependent upon the frame’s type and subtype.⁴

Management Frames Management frames (type 0) are designed to coordinate communication on any wireless LAN, from infrastructure networks to individual stations sending out probe requests. These frames are the glue that keeps the stations together through a single access point, or on an extended series of bridged access points. Management frame subtypes include Association Requests, Association Responses, Probes, Beacons, and others.

For forensic investigators, management frames are important for several reasons. First and foremost, they are not encrypted. As a result, these clear-text frames provide a wealth of information as to which stations are trying to communicate, in which ways, and with whom. At a minimum, type 0 frames are interesting because their subtypes indicate basic

IEEE 802.11 Frame Header																											
Bits	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7			
Bytes	0				1				2				3														
0x00	Ver.	Type	Subtype	DS	F	R	P	M	W	O	Duration/ID																
0x04	Address 1																										
0x08	Address 1, cont.													Address 2													
0x0C	Address 2, cont.													Address 3													
0x10	Address 3, cont.													Sequence Control													
0x14	Address 4, cont.													Address 4													
0x18	Frame Body																										
0x1C																											

Figure 6–1. The 802.11 frame’s data structure, which includes four fields for 6-byte MAC addresses per frame. How each is used is dependent upon the frame’s type and subtype.

4. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” (June 12, 2007): 59–87.

associative activities. In addition, with a bit of statistical analysis (as we'll soon see), they can provide a treasure trove of forensic data. From the information in a management frame, you can enumerate station MAC addresses, infer likely manufacturers, identify access point Basic Service Set Identification (BSSID) and Service Set Identifiers (SSIDs), investigate successful and failed authentication attempts, and more.

Management frames are often the target of manipulation or the vector of attacks against an access point. As we will see, common attacks such as WEP cracking and Evil Twin attacks are often facilitated through manipulation of management frames, or simply careful attention to the details broadcast within them.⁵

Management frame subtypes include:

- 0x0 — Association Request
- 0x1 — Association Response
 - Status Code: 0x0000 — Successful
- 0x2 — Reassociation Request
- 0x3 — Reassociation Response
- 0x4 — Probe Request
- 0x5 — Probe Response
- 0x6 — Reserved
- 0x7 — Reserved
- 0x8 — Beacon frame
- 0x9 — Announcement Traffic Indication Map (ATIM)
- 0xA — Disassociation
- 0xB — Authentication
- 0xC — Deauthentication
- 0xD — Action
- 0xE — Reserved
- 0xF — Reserved

Control Frames Control frames (type 1) are designed to manage the flow of traffic across a wireless network. One of the main challenges wireless protocol designers faced is the “hidden node” problem. Recall that while on a wired medium, it can be presumed that every station will eventually be able to see all nodes (as voltages propagate to all stations). With RF, it is entirely possible that some stations cannot “see” each other, while still

5. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 79–87.

being networked using the same notional physical media (since they can each “see” the WAP). One way of addressing this problem is to have individual stations send “request-to-send” control frames, and for all stations to watch for corresponding “clear-to-send” and “acknowledgment” frames. That way availability of transmission and reception can be tested prior to the establishment of data circuits.

There are three control frame subtypes (as displayed by Wireshark, with the high-order nibble set to “1” for the “Control” type):

- 0x1B—Request-to-send (RTS)
- 0x1C—Clear-to-send (CTS)
- 0x1D—Acknowledgment

Control frames are relatively sparse, but can provide an investigator with information about timing and station MAC addresses.

Data Frames Data frames (type 2) contain the actual data transmitted across the wireless network, including encapsulated higher-layer protocols. For instance, every IP packet that flows across the wireless 802.11 network is part of the payload of an 802.11 data frame.

There are many different data frame subtypes, including the Null function (subtype 4), indicating no data. However, the most interesting is likely to be those where the subtype is 0 (Data).⁶

As a forensic investigator, if the wireless network is not encrypted, or if you have access to the encryption key and can gain access to unencrypted data frames, then you can capture and analyze the wireless traffic at Layer 3 and above. Even encrypted data frames can still be analyzed using statistical flow analysis techniques to reveal volumes and directionality of traffic and stations involved.

6.1.2.2 802.11 Frame Analysis

When you analyze captured network traffic, how do you know which bits correspond with which protocol fields? This may seem like a trivial question at first, but stations can be configured to transmit the bits in different orders, depending on the data link-layer protocol used. As long as the sending and receiving station are both synchronized to interpret and reassemble the bits according to the same protocol, that is all that matters.

The order that bits are transmitted in the 802.11 protocol suite is not straightforward. This can cause forensic analysts to produce incorrect results if you are not careful. To fully understand how the bits we capture correspond with protocol charts and field descriptions, let’s first explore the concept of “endianness.”

Endianness The term “endianness” emerged from the tale *Gulliver’s Travels*, in which Gulliver encounters two groups of people: the Lilliput and the Blefuscus. The Lilliput people insist on cracking open their eggs at the little end (and hence, are “little-endian”), whereas

6. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 77–79.

the Blefuscu are convinced that everyone should crack their eggs open at the big end (“big-endian”).⁷

This idea was eventually applied to the field of computer science. Both humans and computers can be programmed to read, write, and transmit numbers in different orders. We refer to this distinction as “endianness.” When the most-significant digit is ordered first, the system is called “big-endian.” When the least-significant digit is ordered first, the system is called “little-endian.”

As trivial as it may seem, the concept of “endianness” is deeply important for the purposes of computer science and network implementation. While it doesn’t really matter which order we choose to transmit bits, it *is* important that we agree on standards so that bits can be transmitted and properly interpreted at the other end.

Most of the time when we refer to “big-endian” we mean that the most significant byte is represented, stored, or transmitted first (though we can use other data boundaries than bytes—commonly including 16-bit or 32-bit words). When we refer to “little-endian” we usually mean that the least significant byte is represented, stored, or transmitted first (though again we could be referring to byte order within a 16- or 32-bit word, or the word order itself).⁸

Figure 6–2 shows an example of how the concept of “endianness” applies in the familiar example of Hindu-Arabic numerals using base-10 math. “Base 10” means that we use 10 different characters (in this case, 0 through 9) to represent any arbitrary numerical value. In Figure 6–2, the left-most digit represents the number of thousands, the next digit to the right represents the number of hundreds, the next digit to the right represents the number of tens, and finally the right-most digit represents the number of ones. As a result, we can refer to the right-most digit as “least significant” (its value is the smallest) and the left-most digit as “most significant” (its value is the largest).

In English, we read Hindu-Arabic numerals from left to right. That means that we have been trained to read the number in Figure 6–2 beginning with the “most significant digit” on the left. Hence, Hindu-Arabic numerals read by English-speakers are considered “big-endian.”

10^3	10^2	10^1	10^0
1000s	100s	10s	1s
1	0	2	4
$= 1,024$			

Figure 6–2. Base10: From left to right, 1000s, 100s, 10s, 1s. Since we normally read Hindu-Arabic numerals from left to right, beginning with the “most significant bit,” this numbering system is “big-endian.”

7. Jonathan Swift, *Gulliver’s Travels* (London: Penguin, 2010).

8. IEEE, “IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Medium Access Control (MAC) Security Enhancements” (July 23, 2004), <http://standards.ieee.org/getieee802/download/802.11i-2004.pdf>.

6.1.2.3 Network-Byte Order (TCP/IP, but NOT 802.11)

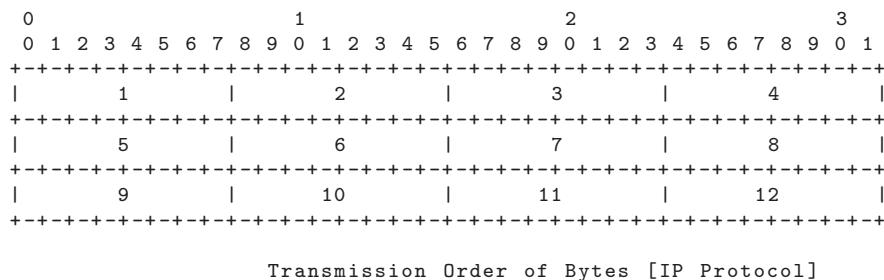
Network Protocols: Big-Endian or Little-Endian?

With respect to network protocols, if the most significant value is transmitted first, the system is “big-endian.” If the least significant value is transmitted first, the system is “little-endian.” If the system is a combination of these methods, it is considered “mixed-endian” or “middle-endian.”

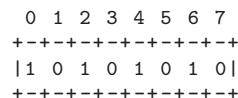
Network forensic analysts are used to viewing captured bits in big-endian form. The IP protocol specifies the order the bits are transmitted across the network as big-endian.⁹ This is often referred to as “network-byte order.”

As described in RFC 791, "Internet Protocol—Appendix B: Data Transmission Order":

The order of transmission of the header and data described in this document is resolved to the octet level. Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram the octets are transmitted in the order they are numbered.



Whenever an octet represents a numeric quantity the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).



Significance of Bits [IP Protocol}

Similarly, whenever a multi-octet field represents a numeric quantity the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

9. Jon Postel, "RFC 791—Internet Protocol, Darpa Internet Program Protocol Specification," Information Sciences Institute, University of Southern California, September 1981, <http://rfc-editor.org/rfc/rfc791.txt>.

6.1.2.4 802.11 Endianness

The IEEE 802.11 specification transmits bits in a different order from the TCP/IP protocol suite, which most network forensic analysts are familiar with. Hence, the need for this discussion!

Mixed-endian? 802.11 is neither big-endian nor little-endian, but is best described as “mixed-endian.” While the bit ordering within each individual data field is big-endian, the fields themselves are transmitted in reverse order, within the byte-boundaries.

The top diagram in Figure 6–3 shows the written protocol specification for the first two bytes of a typical 802.11 Data Frame (type 2, subtype 0). You can see that the payload is encrypted—note that the privacy bit, marked with a “W,” is set to 1 (the “W” originally meant “WEP-protected,” though now it could be protected with TKIP or AES-CCMP instead). The bit ordering depicted in the top diagram shows what we would expect to capture if the 802.11 frame were transmitted in network-byte order like IP, which it most certainly is *not*.

When the first byte of the frame is actually transmitted, the least significant field (“Subtype”) is sent first! The Type goes second, and the protocol version goes third. When libpcap receives the bytes, they arrive bit-for-bit as shown in the lower chart in Figure 6–3. Notice that in the second byte, all seven fields were sent in reverse order, though the bit ordering *within* the 2-bit DS field remains intact. Similarly, the bit ordering within the Version, Type and Subtype fields remains intact.

This has an immediate impact on the values you see displayed in libpcap-based tools such as Wireshark and tcpdump.

Binary sequences (series of 1s and 0s) can be encoded easily in hexadecimal to facilitate reading and interpretation by human analysts. For example, Wireshark displays each frame’s data in the order received, in the “Packet Bytes” frame at the bottom of its window.

The bits are displayed, nibble by nibble, in hexadecimal representation. When interpreting the data, libpcap treats the bit received *first* as the *most significant bit* in the nibble. As

First 2 Bytes of the IEEE 802.11 Frame Header											
Some example bit values, and their hexadecimal representations:											
Field	Version	Type	Subtype	DS	F	R	P	M	W	O	
Data Bits	0	0	1	0	0	0	0	0	0	1	0
Hex Bytes	version/type/subtype = 0x20										flags = 0x42

First 2 Bytes of the IEEE 802.11 Bit Transmission Order											
Here we see the same bit values, with fields reversed on the byte boundaries:											
Field	Subtype	Type	Version	O	W	M	P	R	F	DS	
Data Bits	0	0	0	0	1	0	0	0	0	0	1
Hex Bytes	version/type/subtype = 0x08										flags = 0x41

Figure 6–3. 802.11 traffic: Comparison of field orders within the byte-boundaries. The top diagram represents the protocol specification, whereas the bottom diagram represents the order of bits captured via libpcap.

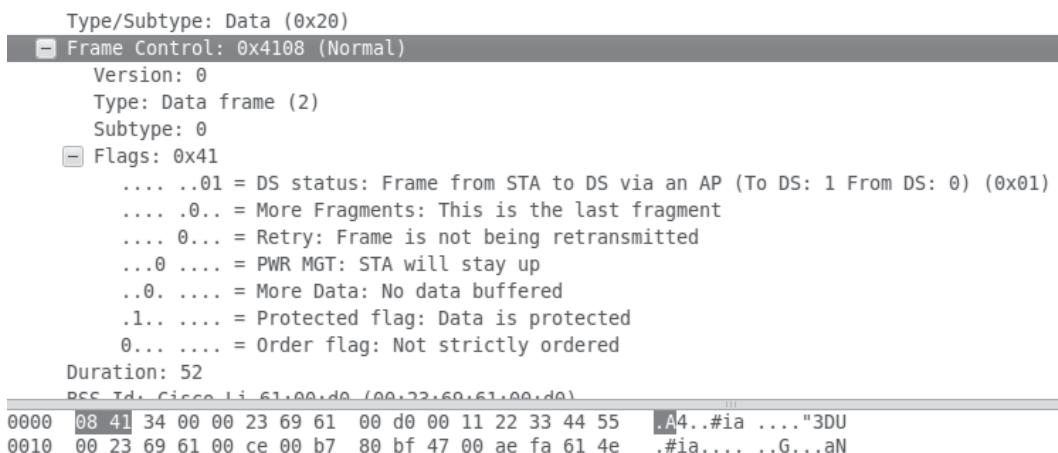


Figure 6–4. In the Packet Details panel, Wireshark interprets the 802.11 frame correctly. In the Packet Bytes panel below, you can see the order that the bits were actually transmitted across the wire (shown as hexadecimal). The 802.11 transmission was mixed-endian.

a result, the hexadecimal representation of a byte in the Packet Bytes window may differ from the protocol-aware interpretation of the meaning of the bits.

Figure 6–4 shows an example of an 802.11 data frame in Wireshark. As you can see, in Wireshark’s Packet Details panel, Wireshark has correctly *interpreted* the first byte of the 802.11 frame (“Version/Type/Subtype”) as 0x20 (0b00100000). However, when you view the *raw traffic* in the Packet Bytes panel below, it appears as 0x08, since the bits were actually received in that order (0b00001000).

Let’s look at the second byte of the 802.11 frame, the “Flags.” This contains six single-bit frame control flags and the 2-bit DS field. These are transmitted in reverse order, from least to most significant, field-by-field! Again, while Wireshark correctly interpreted the flags in the second byte (as seen in the Packet Details panel), due to the order in which the bits were received, the Packet Bytes panel displays 0x41 (0b01000001), and *not* 0x42 (0b01000010). See Figure 6–3 for details.

This can get a bit confusing to say the least!

6.1.2.5 Wired Equivalent Privacy (WEP)

Wired Equivalent Privacy (WEP) is part of the 802.11 standards, published by the IEEE. It was proposed as a way to enable a WAP to provide a “private” network, similar to the environment that a wired hub could provide due to natural limitations of the physical media.

To gain access to a WEP-encrypted wireless network, users need knowledge of a “shared secret” key to gain access to the wireless hub’s service at Layer 2. (Recall that the WAP’s physical layer—radio frequencies—is by definition a physically broadcast medium. Any station with a suitable antenna in range and tuned to the appropriate frequency and channel may observe and transmit data over the Layer 1 shared physical medium.)

Problems in WEP-land The WEP protocol has a couple of significant problems, which has led to its being deprecated in favor of newer wireless privacy protocols such as WPA2. Most importantly, the WEP protocol has a commonly known flaw that allows any eavesdropper

(any station with sufficient RF gain) to harvest unprotected key material, which can then be used to expedite a brute-force attack and recover the shared encryption key. Widely published tools such as “aircrack-ng” make it easy even for beginners to crack WEP keys and access “protected” wireless networks. We discuss this vulnerability in greater detail in Section 6.4.4, “WEP Cracking.”

WEP (and other protocols used with shared secret keys) also pose a logistical Layer 8 (human) problem. The most obvious is that a “shared secret” is not really a secret at all. If the network administrator has to give the WEP key to many people, Layer 8 failures should be anticipated, expected, and compensated for. Adding to that problem is the logistical difficulty in changing the shared secret, which means that even compromised shared secrets often remain unchanged for long periods of time.

Forensic investigators should assume that WEP-protected segments are at high risk of compromise, and may be a likely vector for unauthorized network intrusions. On the plus side, investigators who are (legally) conducting covert investigations without the knowledge of local IT staff may find that WEP-protected networks are a convenient point of covert entry to the network.

If WEP is so broken, why are we still studying it? Although WEP has widely known flaws, it is also still widely implemented. Happily, most commodity Wi-Fi hardware now supports WPA2 (see below). However, forensic investigators still often encounter WEP in use today, for a variety of reasons, such as:

- *Legacy equipment*—Sometimes WAPs are deployed and forgotten, or never maintained, or passwords are never rotated. Lack of resources may prevent equipment upgrades.
- *Modern equipment deployed to support legacy equipment*—Modern WAPs may be deployed in an environment where other legacy devices persist, and therefore must be configured to support the lowest common denominator with respect to key length and cryptosystem.

Often, if a WAP is working, system administrators simply never touch it. Unfortunately, the model “if it ain’t broke, don’t fix it” doesn’t really apply when a device’s functionality continues, even as the security model completely breaks down.

How can you tell if a packet capture or stream is encrypted? 802.11 management frames have a “Privacy” subfield that is set to “1” when “data confidentiality is required for all data type frames exchanged within the BSS.”¹⁰ In other words, when the access point is configured to use WEP, WPA, or WPA2, the “Privacy” subfield in WAP management frames is set to 1.

802.11 data frames include a “Protected” bit within the first byte offset of the 802.11 MAC framing, which indicates whether encryption is in use. (Remember, when specifying offsets, we begin counting from zero!) The “Protected” bit is set to “1” when WEP, WPA,

10. IEEE, “IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Medium Access Control (MAC) Security Enhancements,” (July 23, 2004), <http://standards.ieee.org/getieee802/download/802.11i-2004.pdf>.

or WPA2 is enabled. This bit-sized field was previously labeled the “WEP” field, but this name was changed when the 802.11i standard was updated in 2004.¹¹

6.1.2.6 TKIP, AES, WPA, and WPA2

Clearly, WEP did not provide the level of protection that its designers had intended. The 802.11 working group had to come up with something better to replace it, but they were also constrained because the replacement protocol had to be compatible with existing networking hardware.

In response, the IEEE 802.11 working group came up with a new protocol that the Wi-Fi Alliance dubbed “Wi-Fi Protected Access” (WPA). WPA was a stop-gap measure designed to deal with some of the weaknesses of WEP, such as key rotation. This was done through the introduction of the Temporal Key Integrity Protocol (TKIP). In 2008 flaws were discovered in TKIP as well, and as a result there are widely available tools that exist to crack WPA pre-shared keys (PSKs).

Happily, the working group also drafted specifications for the use of the “Counter Mode with CBC-MAC Protocol” (CCMP) mode of the Advanced Encryption Standard (AES). This is at the core of the improvements that the Wi-Fi Alliance has termed “WPA2,” which is far more secure (although it requires equipment upgrades over the original equipment). Properly implemented, WPA2 is not known to be broken.

In order for stations to communicate their ability to conform to the new specifications, a new category of functionality was created called robust security networks (RSNs). Stations configured to support RSN associations (RSNAs) such as WPA and WPA2 include RSN information within the tagged parameters of management frames, including Beacons, Association Request, Reassociation Request, and Probe Response. Figure 6–5 shows RSN information displayed in an 802.11 management frame. Forensic investigators can therefore filter traffic to isolate packets that employ specific types of encryption algorithms, or none at all.¹²

In summary:

- WPA is:
 - Designed to work with legacy 802.11 hardware.
 - Intended to compensate for keying weaknesses (using TKIP).
 - Already broken.
- WPA2 is:
 - Required next generation hardware.
 - Built to utilize AES-CCMP (real crypto!).
 - Currently considered secure when used with strong preshared keys or in combination with strong 802.1X authentication mechanisms.

11. *Ibid.*

12. *Ibid.*

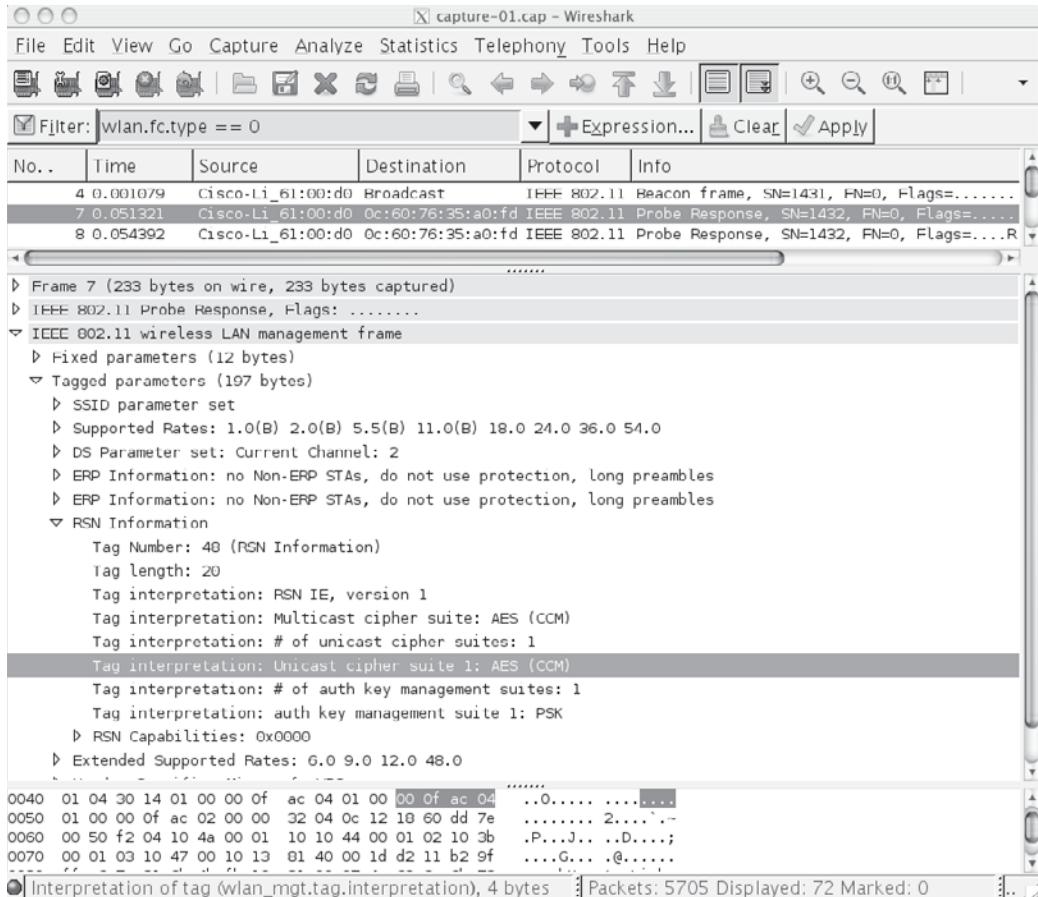


Figure 6–5. An 802.11 packet capture displayed in Wireshark. Having filtered down to only management frames, a probe response is selected. The Packet Details panel shows the RSN Information within the tagged parameters, indicating that AES-CCMP is used as the cipher. Note that this is a clear indication that WPA2 is used, and not WPA, since WPA doesn't support AES-CCMP.

6.1.3 802.1X

802.1X was designed to provide a modular, extensible authentication framework for LANs (regardless of physical medium). It can be used over wired or wireless networks, and it is designed to control access to the LAN. Forensic investigators should be aware of 802.1X when it is used in the environment under investigation because it limits access to the network and requires a back-end authentication system, that typically stores access logs.

802.1X is the IEEE's standard for implementing the IETF's Extensible Authentication Protocol (EAP) over LANs.¹³ EAP was intended as an improvement to the Point-to-Point

13. B. Aboba, “RFC 3748—Extensible Authentication Protocol (EAP),” *IETF*, June 2004, <http://rfc-editor.org/rfc/rfc3748.txt>.

Protocol (PPP) methods for authentication. Though initially deployed predominantly as a data link-layer protocol over asynchronous dial-up links, the abandonment of the 56kbps modulator-demodulator (modem) hasn't spelled the end for PPP. In its many incarnations it remains perhaps the most widely used WAN protocol. PPP over Ethernet (PPPoE)¹⁴ is commonly used for DSL circuits. The Evolution-Data Optimized (EV-DO) wireless network—used widely for both smartphones and laptops in the United States—uses PPP over its wireless links as well. As a component of PPP, the Challenge Handshake Authentication Protocol (CHAP) is still widely deployed, as is—regrettably—the Password Authentication Protocol (PAP), which transmits passwords in cleartext.

RFC 3748 and the IEEE's 802.1X standards provide a better framework for low-layer authentication. Support for EAP by any Layer 2 device (physically switched or wireless) allows for authentication not by knowledge of a “shared secret” but rather based on a central authentication store. This can be performed over a dial-up link, 802.3 Ethernet, or 802.11 Wi-Fi, among others. The authentication is abstracted, and 802.1X can therefore support a wide range of back-end authentication methods, including “EAP-Transport Layer Security” (EAP-TLS), the “Protected Extensible Authentication Protocol” (PEAP), and the “Lightweight Extensible Authentication Protocol” (LEAP).

6.1.3.1 Impact on Wireless Networks

EAP was not widely used until the weakness of WEP became well known, and there was a scramble to deploy stronger access control mechanisms. Some enterprises responded by deploying 802.1X with Cisco's LEAP protocol on their wireless networks (and sometimes wired networks as well). LEAP was based on Microsoft's version of CHAP (MS-CHAP), which does not send authentication credentials in the clear, but which has flaws that allow passwords to be trivially brute-forced.¹⁵ Josh Wright's “asleap”¹⁶ tool makes breaking LEAP passwords an academic exercise.

PEAP, another proprietary protocol cooperatively designed by Cisco, Microsoft, and RSA Security, tunnels EAP traffic via TLS in order to provide server-to-client authentication, confidentiality, and integrity.

6.1.3.2 Implications for the Investigator

When you encounter an 802.1X implementation, the back-end authentication system in use—whether a RADIUS system or an Active Directory/LDAP server—is much more likely to be producing an audit trail than a stand-alone WAP with a static key shared by all. However, if LEAP is being used for authentication, network compromise due to use of a weak authentication mechanism should be considered a very real possibility.

14. L. Mamakos, “RFC 2516—A Method for Transmitting PPP Over Ethernet (PPPoE),” IETF, February 1999, <http://rfc-editor.org/rfc/rfc2516.txt>.

15. B. Schneier and D. Wagner, “Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2),” *Secure Networking—CQRE [Secure]’99* (1999): 782782.

16. Joshua Wright, “asleap—exploiting cisco leap,” May 2, 2009, <http://www.willhackforsushi.com/Asleap.html>.

6.2 Wireless Access Points (WAPs)

A wireless access point (WAP) is a Layer 2 device that aggregates endpoint stations into a LAN. Instead of voltages across a wire, the physical medium is RF through air (or buildings, trees, etc). For WAPs, the physical layer is a shared medium, and as a result all stations have access to all signals traveling across the LAN. This facilitates traffic sniffing in much the same way as with wired hubs. Anyone within range to receive the RF signal can intercept the traffic.

Unlike simple hubs, WAPs have a wide range of configuration options and logging capabilities. Hubs are dumb devices; all they do is replicate traffic. WAPs perform the same function, but are much smarter devices.

Configuration and logging capabilities vary widely across brands and models. Even low-end WAPs generally have built-in web management interfaces, basic logging capabilities, MAC address filtering, and DHCP servers. Higher-end models also function as routers and support syslog and SNMP.

In the enterprise, you will often find a centrally managed wireless network with many distributed WAPs. These systems typically include corresponding software that aggregates logs from all WAPs, which can allow you to gather the connection history for a given MAC address and sometimes view the physical location of the connections on a map. In some cases, you can even reconstruct the route that a wireless client (such as a tablet) took through a building or campus.

On the one hand, we must treat WAPs as a class of hubs whose physical access is nearly unlimited. This makes them a special case. On the other hand, unlike standard hubs, most WAPs bundle functionality that transcends Layer 2. This can include Layer 3 routing, DHCP functionality, Layer 4 NATing, and various sophisticated management capabilities. To complicate matters, WAPs employ a range of different standards for encryption and authentication. Some devices are even based on draft standards that have not yet been finalized (such as 802.11aa).

6.2.1 Why Investigate Wireless Access Points?

Wireless access points are typically involved in forensic investigations for one of a few reasons:

- Wireless access points may contain locally stored logs of connection attempts, authentication successes and failures, and other local WAP activity.
- WAP logs can help you track the physical movements of a wireless client throughout a building or campus.
- The WAP configuration may provide insight regarding how an attacker gained access to the network.
- The WAP configuration may have been modified by an unauthorized party as part of an attack.
- The WAP itself may be compromised.

6.2.2 Types of Wireless Access Points

There are a wide variety of wireless access points available, each with different interfaces and functionality. The sheer variety of WAPs make them challenging for investigators to assess and examine. Remember, any device with an 802.11 card can potentially be made to function as a WAP, and so investigators must be familiar with a wide variety of platforms. General classes of WAPs include enterprise and consumer devices.

6.2.2.1 Enterprise

Enterprise facilities typically span a much wider geographic range than home offices or small businesses. Wireless networks are commonly set up to provide seamless, ubiquitous coverage throughout the environment. Since enterprise IT staff usually deploy many wireless access points, centralized management and integration are important. In the enterprise environment, wireless networks are commonly set up to use central authentication systems. They also include central management consoles that can be used to monitor network state, user activity, or even track wireless clients throughout the physical environment.

Common features of enterprise-class WAPs include:

- Support for IEEE 802.11a/b/g/n
- Physical media is RF waves
- Layer 3+ functionality, including:
 - Support for routing protocols
 - DHCP
 - Network address translation
 - Packet filtering
- Centralized authentication
- Auditable access logs (local and central)
- Station location tracking
- Performance monitoring capabilities
- Power over Ethernet (PoE)
- Indoor and outdoor options

The interface options for enterprise-class wireless access points frequently include:^{17,18}

- Console (command-line interface (CLI))
- Remote console (SSH/Telnet)

17. “Cisco Aironet 1250 Series Access Point Data Sheet [Cisco Aironet 1250 Series],” *Cisco Systems*, 2011, http://www.cisco.com/en/US/prod/collateral/wireless/ps5678/ps6973/ps8382/product_data_sheet0900aecd806b7c5c.html.

18. “Products | Aruba Networks,” 2011, <http://www.arubanetworks.com/products/>.

- SNMP
- Web interface
- Proprietary interface
- Central management interface

Examples of enterprise wireless access points include the Cisco 3600 AP and Aruba's AP-135 WAPs for high-density environments.

6.2.2.2 Consumer

Small businesses and home users often deploy consumer-class WAPs in their home and office environments. These devices are inexpensive and easy to configure for simple use.

There are two different kinds of consumer-class wireless access points that an investigator might encounter and need to be concerned with. First, consumers may purchase their own WAPs (or routers with built-in wireless capabilities) from retail stores. These low-end WAPs typically have all the features that you need to easily build a home network or small office to extend or replace wired infrastructure.

Second, WAP functionality is commonly built into DSL modems and cable modems leased from an ISP. In these cases, you may not have legal or technical access to the device without support from the local ISP. Furthermore, it is important to keep in mind that these devices are often deployed by ISPs with trivial or default passwords.

Features of consumer-class wireless access points typically include:

- Support for IEEE 802.11a/b/g/n
- Physical media is RF waves
- Often contain Layer 3+ functionality, including:
 - Limited routing
 - DHCP service
 - NATing
 - Limited filtering
- Logging (locally and sometimes remotely)

The configuration interface options for consumer-class WAPs are typically limited to a web interface or proprietary application. Examples of consumer-class WAPs are the ubiquitous Linksys WRT54G and the Apple Airport.

Apple Airport Express Now let's look at the higher-layer functionality bundled in common consumer WAPs. The first example will be an Apple Airport Express. This is not merely an access point; it also includes DHCP and NAT technology, as well as sophisticated traffic statistics and management capabilities. It may not be the most widely deployed device, but it is a good example of a proprietary interface with many common features.

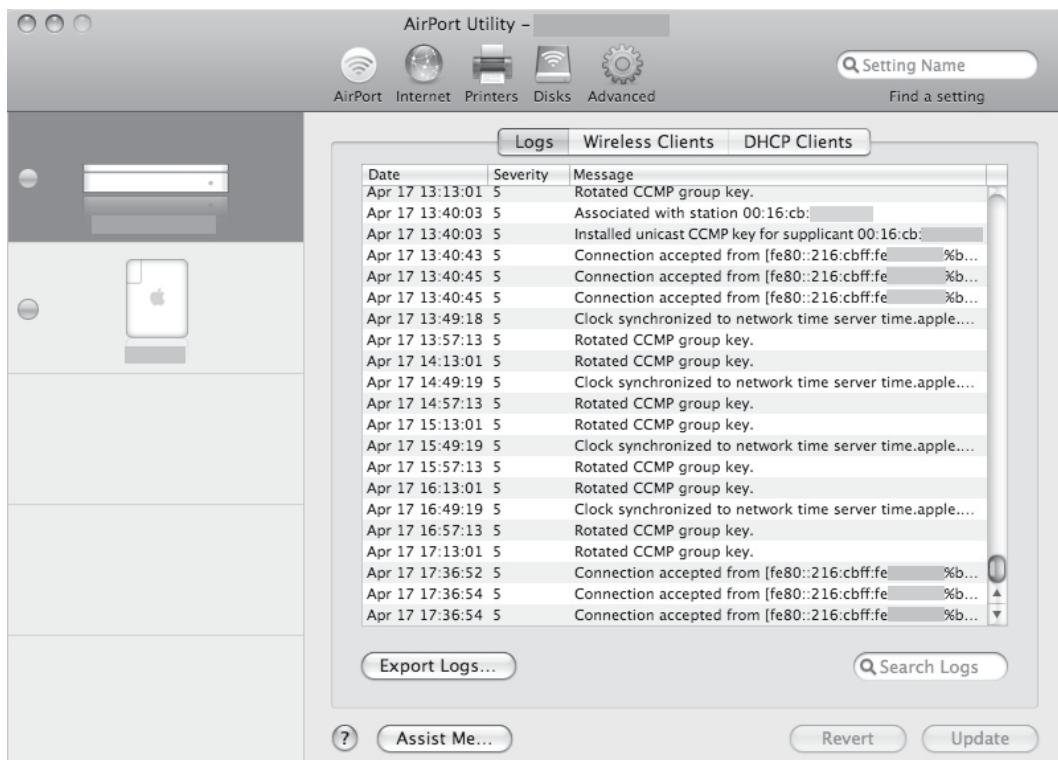


Figure 6–6. Log entries saved locally on an Apple Airport, accessed through the native, proprietary Airport Utility. These logs can also be exported via Syslog or SNMP.

Figure 6–6 is a screenshot that displays local logs from an Apple Airport Express. (This is from a real production device, so the client addresses have been partially blocked out.)

Linksys WRT54G The Linksys WRT54G series WAP (Figure 6–7) is one of the most commonly deployed low-cost COTS WAPs, retailing at less than US\$60. It is a fairly full-featured router/switch/WAP with a web-based configuration interface that addresses basic setup, wireless setup (including security settings), HTTP URL blocking ability, port-based restrictions, and the ability to backup configurations to a local file.

However, the web-based interface is not terribly intuitive for laymen, and all the security features that are most critical are disabled by default (hence the proliferation of “linksys” SSIDs with no authentication or encryption enabled). Furthermore, the logging ability of the native firmware is extremely limited in that no remote logging is possible.

One of the things that has made this particular make and its various models so popular is that the firmware is easily flashable. This has opened the devices up to be reprogrammable, and many projects exist that offer replacement software systems for the relatively cheap and capable hardware. Many of these are based on Linux, and so offer much of the capabilities that such distributions have natively, including syslog, packet capturing, robust filtering, and network intrusion detection systems (NIDS).

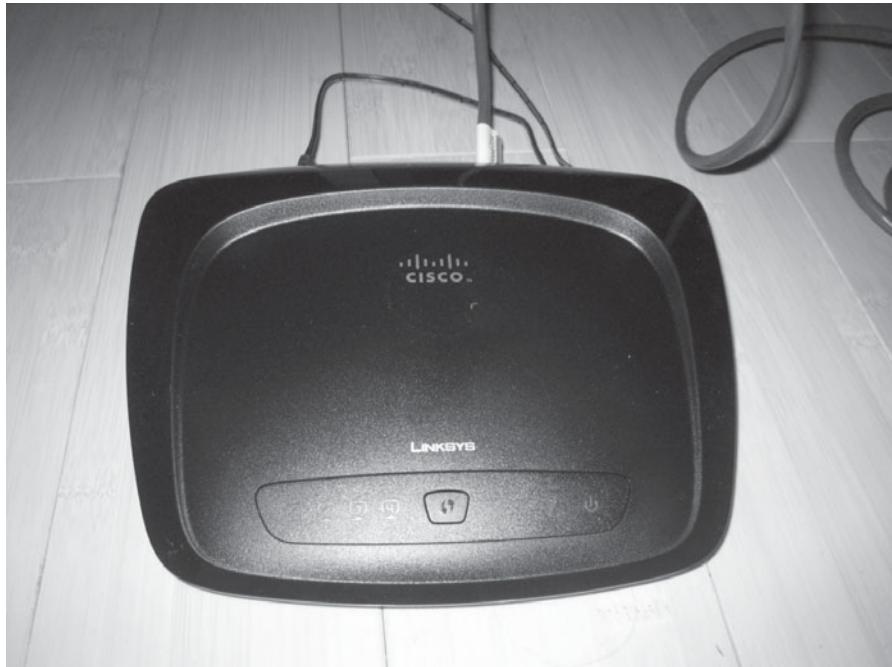


Figure 6–7. This is one of the second-generation WRT54G wireless routers from Linksys (now Cisco of course). This device is an extremely common sight in homes and small offices, though its slim lines and low profile make it relatively inconspicuous.

6.2.3 WAP Evidence

Wireless access points contain both volatile and nonvolatile evidence, although due to their persistent storage capabilities tend to be very limited. WAPs can also send logs over the network to a remote repository.

6.2.3.1 Volatile

As with switches and routers, most of the evidence on WAPs tends to be quite volatile. Enterprise-class WAPs tend to include the same functionality and range of evidence as wired routers, with the addition of wireless-specific capabilities. Evidence that you may find on wireless access points includes:

- History of connections by MAC address
- List of IPs associated with MACs
- Historical logs of wireless events (access requests, key rotation, etc.)

- History of client signal strength (can help identify geographic location)
- Routing tables
- Stored packets before they are forwarded
- Packet counts and statistics
- ARP table (MAC address to IP address mappings)
- DHCP lease assignments
- Access control lists
- I/O memory
- Running configuration
- Processor memory
- Flow data and related statistics

6.2.3.2 Persistent

Again, like wired routers and switches, WAPs are not designed to include much local persistent storage space. The WAP operating system and startup configuration files are maintained in persistent storage by necessity. Persistent evidence you may find on a WAP includes:

- Operating system image
- Boot loader
- Startup configuration files

6.2.3.3 Off-System

Wireless access points can be configured to send event logs to remote systems for off-site aggregation and storage. Syslog and SNMP are commonly supported. Enterprise-class devices may include other options, often proprietary. Check the documentation for the model you are investigating and review local configuration to locate devices that may contain off-system WAP logs.

6.3 Wireless Traffic Capture and Analysis

Capturing and analyzing wireless traffic often provides valuable evidence in an investigation, for the same reasons we discussed in Chapter 3. However, there are some additional complexities involved in capturing wireless traffic, as opposed to sniffing traffic on the wire. In this section, we review some important notes for capturing and analyzing wireless traffic. For further discussion of passive evidence acquisition and analysis, please see Chapter 3, “Evidence Acquisition.”

6.3.1 Spectrum Analysis

There are, literally, an infinite number of frequencies over which data can be transmitted through the air. Sometimes the most challenging part of an investigator's job is simply identifying the wireless traffic in the first place.

For Wi-Fi traffic, the IEEE utilizes three frequency ranges:

- 2.4 GHz (802.11b/g/n)¹⁹
- 3.6 GHz (802.11y)²⁰
- 5 GHz (802.11a/h/j/n)²¹

Each of these frequency ranges is divided into distinct channels, which are smaller frequency bands (for example, the IEEE has specified 14 channels in the 2.4 GHz range). Although the IEEE has set globally recognized frequency boundaries for 802.11 protocols, individual countries typically allow only a subset of these frequency ranges.

The precise frequencies in use vary by country. For example, the United States only allows WiFi devices to communicate over channels 1–11 in the 2.4 GHz range, while Japan allows transmission over all 14 channels. As a result, WiFi equipment manufactured for use in the United States is generally not capable of transmitting or receiving traffic on all of the channels used in Japan. This has important consequences for forensic investigators. For example, an attacker can purchase a Japanese WAP that supports Channel 14 and plug it into a corporate network in the United States, and U.S. wireless clients will not “see” the access point.

Wireless security researcher Joshua Wright has also published articles about the use of 802.11n in “Greenfield” (GF) mode. 802.11n devices operating in Greenfield mode are not visible to 802.11a/b/g devices. As a result, investigators scanning for wireless devices using 802.11a/b/g cards will not detect the 802.11n network. Please see Section 6.4.2, “Rogue Wireless Access Points,” for more details.

When monitoring for the presence of wireless traffic, make sure that you fully understand the capabilities of your monitoring device, as well as the potential for devices that operate outside your range of detection.

19. IEEE, “IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput” (October 29, 2009): Annex J, <http://standards.ieee.org/getieee802/download/802.11n-2009.pdf> (accessed December 31, 2011).

20. IEEE, “IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: 3650-3700 MHz Operation in USA” (November 6, 2008): Annex J, <http://standards.ieee.org/getieee802/download/802.11y-2009.pdf> (accessed December 31, 2011).

21. IEEE, “IEEE Standard for Information Technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput” (October 29, 2009): Annex J, <http://standards.ieee.org/getieee802/download/802.11n-2009.pdf> (accessed December 31, 2011).

Spectrum analyzers are designed to monitor RF frequencies and report on usage. They can be very helpful for identifying stealthy rogue wireless devices and WiFi channels in use. MetaGeek's Wi-Spy product line supports the 2.4 GHz and 5 GHz frequency bands (as well as 900 MHz), and range in price from \$100 to \$1,000. AirMagnet (owned by Fluke Networks) also produces a popular wireless spectrum analyzer that can "identify, name and find: Bluetooth devices, 2.4G cordless phones, microwave ovens, RF Jammers, analog video cameras, etc."²²

6.3.2 Wireless Passive Evidence Acquisition

In order to capture wireless traffic, investigators need an 802.11 wireless card capable of running in Monitor mode. Many wireless cards do not support this capability. Furthermore, in order to ensure totally passive monitoring, it is preferable to use a special-purpose WiFi monitoring card that can be configured to operate completely passively.

Riverbed Technology offers the AirPcap USB adapters, that are designed for exactly this task. The AirPcap USB adapter plugs into a USB port and can monitor Layer 2 WiFi traffic (one channel at a time). AirPcap software runs on Windows, integrates with Wireshark, and can be configured to automatically decrypt WEP-encrypted frames. The AirPcap "Classic" and "Tx" models support the 2.4 GHz 802.11b/g band, while the "Nx" model additionally supports 802.11n. The "Nx" model also includes an external antenna connector.²³ Figure 6-8 shows an example of the AirPcap USB dongle.



Figure 6-8. The AirPcap USB adapter from Riverbed Technology (previously CACE Technologies).

22. "WLAN Design, Security and Analysis," *Fluke Networks*, 2011, <http://www.airmagnet.com/products/spectrum-analyzer/>.

23. "Riverbed Technology—AirPcap," 2011, <http://www.cacetech.com/products/airpcap.html>.

For Linux users, the AirPcap USB adapter can be used via a modified driver (although the AirPcap software is still Windows-only). Josh Wright provides a patch for the zd1211rw wireless driver, which supports sniffing using the AirPcap dongle.²⁴

Once you have the ability to monitor Layer 2 802.11 traffic, you can use standard tools such as tcpdump, Wireshark, and tshark to capture and analyze it.

Regardless of whether or not a WAP's traffic is encrypted, investigators can gain a great deal of information by capturing and analyzing 802.11 management traffic. This information commonly includes:

- Broadcast SSIDs (and sometimes even nonbroadcast ones)
- WAP MAC addresses
- Supported encryption/authentication algorithms
- Associated client MAC addresses

Even when the WAP traffic is encrypted, there is a single shared key for all stations. This means that anyone who gains access to the encryption key can listen to all traffic relating to all stations (as with physical hubs). For investigators, this is helpful because local IT staff can provide authentication credentials, which facilitate monitoring of all WAP traffic. Furthermore, there are well-known flaws in common WAP encryption algorithms such as WEP, which can allow investigators to circumvent or crack unknown encryption keys.

Once an investigator has gained full access to unencrypted 802.11 traffic contents, this data can be analyzed in the same manner as any other unencrypted network traffic.

6.3.3 Analyzing 802.11 Efficiently

So, you have some 802.11 frames. During the course of an investigation, you may search for the answers to questions such as:

- Are there any beacons in the wireless traffic?
- Are there any probe responses?
- Can you find all the BSSIDs/SSIDs from authenticated/associated traffic?
- Can you find malicious traffic? What does that look like?
- Is the captured traffic encrypted using WEP/WPA? Is anyone trying to break the encryption?

6.3.3.1 tcpdump and tshark

It's certainly true that you could use Wireshark to sort out the endianness problem for you, and you could use the graphical interface to try to zero in on the answers to any of the above questions. However, for large packet captures in particular, tcpdump and tshark tend to be more efficient and scalable.

24. <http://www.willhackforsushi.com/code/zd1211rw-airpcap-linux-2.6.31.diff>. (Accessed Jan. 6, 2012.)

With nothing but a powerful filtering language and an understanding of how 802.11 is structured—and how it transmits the bits—you can very quickly hone in on important wireless traffic. The following discussion presents useful BPF filters and display filters that can be used to filter 802.11 traffic.

Find the WAPs: Finding Beacon frames with tcpdump and BPF filters is straightforward, as shown below. Recall from Section 6.1.2.1 that Beacon frames are a type of management frame (type 0) with subtype 0x08. With a “Version” field of 0b00, the 0-byte offset of the 802.11 frame header (referred to as “wlan[0]”) is 0b00001000. In order of transmission (remember that 802.11 is “mixed-endian”) that becomes 0b10000000, or 0x80.

```
'wlan[0] = 0x80'
```

The 802.11 specification includes a 1-bit field called “ESS capabilities,” which has a Wireshark field name of “wlan_mgt.fixed.capabilities.ess.” According to the IEEE’s 802.11 specification, “WAPs set the ESS subfield to 1 and the IBSS subfield to 0 within transmitted Beacon or Probe Response management frames.”²⁵ Let’s use tshark to search for Beacon or Probe Response frames where the ESS subfield is set to 1 and the IBSS subfield is set to 0, as shown below:

```
$ tshark -nn -r wlan.pcap -R '((wlan.fc.type_subtype == 0x08 || wlan.fc.type_subtype == 0x05) && (wlan_mgt.fixed.capabilities.ess == 1) && (wlan_mgt.fixed.capabilities.ibss == 0))'
1  0.000000 00:23:69:61:00:d0 -> ff:ff:ff:ff:ff:ff 802.11 105 Beacon frame
   , SN=3583, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
265 20.409086 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
   Response, SN=3801, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
270 20.597504 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
   Response, SN=3804, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
335 23.318463 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
   Response, SN=3837, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
412 26.317951 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
   Response, SN=3873, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
[...]
```

Find the Encrypted Data Frames: Similarly, how can we filter quickly down to encrypted data frames? Just for fun, let’s use a BPF filter to accomplish this. 802.11 data frames are version 0, type 2, subtype 0 (in binary 0b00100000). In order of transmission, the first byte (“wlan[0]”) is 0b00001000, which in hexadecimal is 0x08.

As discussed earlier, the “Protected” bit indicates whether the frame is encrypted using WEP, TKIP, or AES-CCMP. The Protected bit is located at bit 6 of the 1-byte offset of the 802.11 frame (refer to Figures 6–1 and 6–3). With fields reversed within the byte for transmission, the Protected bit is the second bit received in the 1-byte offset (“wlan[1]”). Consequently, we have to construct a bitmask of 0b01000000 (0x40 in hexadecimal) to test whether the Protected bit is set.

25. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” (June 12, 2007): 251, <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>. (accessed December 31, 2011).

The combination of the two tests, shown below, produces all of the encrypted data packets in a given capture!²⁶

```
'wlan[0] = 0x08 and wlan[1] & 0x40 = 0x40'
```

6.4 Common Attacks

Often, investigators suspect that a wireless network has been or is currently under attack. Common attacks on wireless networks include:

- **Sniffing** An attacker eavesdrops on the network
- **Rogue Wireless Access Points** Unauthorized wireless devices that extend the local network, often for an end-user's convenience
- **The Evil Twin Attack** An attacker sets up a WAP with the same SSID as a legitimate WLAN
- **WEP Cracking** An attacker attempts to recover the WEP encryption key to gain unauthorized access to a WEP-encrypted network.

It is important for network forensic investigators to recognize the signs of common attacks. We discuss each of these in detail below.

6.4.1 Sniffing

Eavesdropping on wireless traffic is extremely common, in part because it is so easy to do! From script kiddies in coffeeshops to professional surveillance teams, wireless traffic monitoring is, frankly, popular. Even where it is completely illegal, the risk of detection is exceptionally low, and the information gained can be very valuable. Both forensic investigators and attackers alike know how to passively monitor wireless traffic and use this technique to their advantage.

Wireless LANs, by virtue of their physical medium, can be accessed over great distances. Although WLANs can be designed to serve a specific geographic range, it is challenging for network administrators to limit the signal to that area and prevent leakage.

The FCC stipulates rules that govern the effective range of 802.11 transmissions. Based on these rules, theoretically the distance from which a station can interact with a wireless access point is limited to roughly 200 feet or 61 meters.²⁷ However, directional antennae can be constructed from off-the-shelf components that can dramatically increase the effective

26. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” (June 12, 2007): 60–64.

27. “Title 47 CFR Part 15: Low Power Broadcast Radio Stations, Audio Division (FCC) USA,” 2011, <http://www.fcc.gov/mb/audio/lowpwr.html>.

ranges. (As we discussed in Section 3.1.2, one research team claimed a successful data transfer of 3Mbps over a distance of 238 miles!²⁸)

Eavesdropping on telecommunications (including those transmitted over RF) is a violation of wiretap statutes in many jurisdictions. Remember that even stations that are not associated with a wireless network can capture and analyze WAP traffic. Forensic investigators should be aware that an attacker may have access to the network via a WAP, and that they may be able to monitor local traffic or communicate on the LAN from a location far outside what is considered normal range, a great distance away.

6.4.2 Rogue Wireless Access Points

For \$40, anyone can purchase a cheap WAP and plug it into the company network. Often, employees do this simply for the sake of convenience, not realizing that it opens the company to attack. Criminals also deliberately plant wireless access points that allow them to bypass the pesky firewall and remotely access the network later on. These days, disgruntled employees can easily hide a WAP behind the file cabinet before cleaning out their desks and then access the company network months later from the parking lot.

Many companies conduct regular “war-walking” scans to detect rogue access points (i.e., using Kismet or NetStumbler) or invest in commercial wireless intrusion detection systems (WIDSs). However, there are sneaky ways to bypass traditional war-walking and WIDSs.

Forensic investigators should be aware of the methods that attackers can use to place rogue access points and evade detection. Rogue access points can be used to covertly extend the range of an internal network, facilitating access from far outside the physical bounds that network administrators might expect. Rogue access points may also allow for untracked LAN access, and act as a pivot point for attacks.

Conversely, in certain situations a forensic investigator may be charged with monitoring a network in which the network administrators are hostile or unaware of the investigation. In these circumstances, where law and ethics allow, it may be the forensic investigator employing these same techniques for the purposes of covert monitoring and evidence acquisition.

6.4.2.1 Changing the Channel

In the United States, the FCC has licensed 11 channels for 802.11b/g/n, which have center frequencies between 2.412 GHz to 2.462 GHz. However, most of Europe allows 13 channels (up to 2.472 GHz) and Japan allows 802.11b all the way up to channel 14, or 2.484 GHz.²⁹

Cards manufactured for the United States often don’t support channel 14, since it’s illegal to transmit on that frequency. There’s overlap between the channels, but at 2.484 GHz, channel 14 is far enough away from channel 11 that network cards are unlikely to pick up much signal on channel 11. If an attacker were to configure a WAP to illegally transmit on channel 14 and export data at 2.484 GHz, security teams monitoring U.S. channels would probably never detect it.

28. Michael Kanellos, “Ermanno Pietrosemoli has set a new record for the longest communication Wi-Fi link,” *Historia de Internet en Amrica Latina y el Caribe*, June 2007, <http://interred.wordpress.com/2007/06/18/ermanno-pietrosemoli-has-set-a-new-record-for-the-longest-communication-wi-fi-link/>.

29. “List of WLAN channels—Wikipedia, the free encyclopedia.”

Similar tactics are effective in other countries, when attackers use frequencies outside the bounds of normal wireless device operation.

6.4.2.2 802.11n Greenfield Mode

The IEEE's 802.11n ("MIMO"-based) specification is designed to allow much greater throughput than 802.11a/b/g (100Mbps or more).³⁰ The 802.11n standard specifies two modes:³¹

- "Mixed mode," which allows it to work with legacy 802.11a/b/g networks,
- "Greenfield" (GF) or "high-throughput-only" mode, which takes full advantage of the enhanced throughput but is not visible to 802.11a/b/g devices. Older devices will see GF-mode traffic only as noise.

Not visible to 802.11a/b/g devices? That means if you're war-walking with an 802.11a/b/g card, you can't see 802.11n devices operating in Greenfield (GF) mode. Even before the specification was finalized, 802.11n devices were already available for as little as \$50—easy to buy, easy to plug into the company's network. However, many companies have not yet purchased 802.11n-compatible equipment and hence cannot detect GF-mode 802.11n rogue WAPs.

Josh Wright submitted a vulnerability report explaining this, in which he wrote: "With the inability to decode GF mode traffic, an attacker can position a malicious rogue WAP on a victim network using the GF mode preamble. This would allow an attacker to evade wireless intrusion detection systems (WIDS) based on non-HT devices. This includes all WIDS devices based on 802.11a/b/g wireless cards."³²

6.4.2.3 Bluetooth Access Point

When you think about Bluetooth, you probably envision your tiny little headset that crackles and hisses every time you walk too far away from your phone. That's because your Bluetooth headset is designed for a Class 2 Bluetooth network, which is fairly low-power (2.5mW) and has a maximum range of about 9m.³³

However, there's more to Bluetooth than your rinky-dink headset. Bluetooth Class 1 devices are much more powerful, with ranges similar to 802.11b WAPs. A Bluetooth Class 1 device can transmit up to 100mW, with a typical range of up to about 91m (or possibly

30. Joshua Wright, "Wireless Ethical Hacking, Penetration Testing, and Defense: Wireless Architecture & Analysis," The SANS Institute, 2008.

31. IEEE, "IEEE Standard for Information technology—Telecommunications and information exchange between systems— Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 5: Enhancements for Higher Throughput" (October 29, 2009), <http://standards.ieee.org/getieee802/download/802.11n-2009.pdf> (accessed December 31, 2011).

32. Joshua Wright, "GF Mode WIDS Rogue AP Evasion," *Wireless Vulnerabilities and Exploits*, November 13, 2006, <http://www.wirelessve.org/entries/show/WVE-2008-0005>.

33. Karen Scarfone and John Padgette, "Guide to Bluetooth Security: Recommendations of the National Institute of Standards and Technology," Special Publication 800-121, National Institute of Standards and Technology, September 2008, <http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf> (accessed December 31, 2011).

miles, if the receiver has a directional antenna). You can buy a Class 1 Bluetooth WAP for \$100–\$200.³⁴

Can you discover Bluetooth WAPs while war-walking? Not if you’re just using an 802.11 card. Even if you’re using a spectrum analyzer like WiSpy, you may not notice it. Bluetooth uses Frequency Hopping Spread Spectrum,³⁵ and hops 1,600–3,200 times a second across 79 channels throughout the 2.4–2.4835 GHz band. Because it’s spread out across the spectrum, it can be hard to notice and easily mistaken for noise by the untrained eye. Most Wireless IDS systems and security teams simply don’t look for it (yet).^{36, 37}

6.4.2.4 Wireless Port Knocking

Remember port knocking? Instead of installing a backdoor to listen on a particular port (where it might be noticed), l33t h4x0rs installed rootkits that would wait for a particular sequence of ports to be scanned, at which point the knocker’s IP address would be granted access. With wireless knocking, a rogue WAP sits on the network in monitor mode, listening for probe requests. When the rogue WAP receives a packet (or sequence of packets) with the preconfigured SSID, it awakens and switches to master mode. The program “WKnock” is designed for this purpose,³⁸ and it can be installed on any WAP supported by the OpenWRT framework. During times when the rogue WAP isn’t active, it is silent and can’t be detected using common wireless scanning tools. Sneaky!³⁹

6.4.3 Evil Twin

The “Evil Twin” attack is when an attacker sets up a WAP with the same SSID as one that is used in the local environment, usually in order to conduct a man-in-the-middle attack on an 802.11 client’s traffic.

By default, commercial 802.11 clients associate with the SSID that their operators tell them to. If there is more than one WAP with the same SSID, as will be the case with most centrally-managed wireless network (either corporate or in a Wi-Fi “hotspot”) then the client will associate with the WAP providing the strongest signal. When the Evil Twin’s signal strength is stronger than the “real” WAP, 802.11 clients will associate with the Evil Twin.

It is trivial for any 802.11 device to masquerade as the closest infrastructure WAP for any given SSID. Any 802.11 device can be made to advertise itself as an available peer. These advertisements can be of two kinds: ad-hoc and infrastructure. By default, commercial

34. *Ibid.*

35. Sherri Davidoff, “Philosecurity, Blog Archive: Off the Grid,” July 28, 2008, <http://philosecurity.org/2008/07/28/off-the-grid>.

36. Joshua Wright, “Wireless Ethical Hacking, Penetration Testing, and Defense: Wireless Security Exposed, Part 4,” The SANS Institute, 2008.

37. Karen Scarfone and John Padgette, “Guide to Bluetooth Security: Recommendations of the National Institute of Standards and Technology,” Special Publication 800-121, National Institute of Standards and Technology, September 2008, <http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf> (accessed December 31, 2011).

38. “rstack: wknock,” 2011, <http://rstack.org/oudot/wknock>.

39. Oudot Laurent, “WLAN and Stealth Issues,” 2005, http://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Oudot/BH_EU_05-Oudot.pdf.

WAPs are infrastructure devices and, by default, most commercial operating systems that support 802.11 networking devices allow them to advertise as ad-hoc networks for peer-to-peer purposes. However, it is not difficult to switch an 802.11 interface on a desktop or laptop into infrastructure mode. With Linux, it's as easy as a single "iwconfig" command.

The "Evil Twin" ruse allows any sufficiently strong 802.11 broadcaster to become a "man-in-the-middle" between the unwitting client and every other system that it communicates with. Sufficiently strong broadcasting can be accomplished over surprisingly wide geographic areas.

Once a client has connected to the "Evil Twin," the attacker can intercept traffic, replace images or words on the fly, conduct SSL-stripping attacks, harvest credentials, and more.

6.4.4 WEP Cracking

Security professionals often joke that "WEP" stands for "Weak Encryption Protocol." This isn't far off the mark (although "WEP" really stands for "Wired Equivalent Privacy," as we discussed earlier). Due to flaws in the protocol, there are tools that can help attackers "crack" WEP keys in minutes, and thereby gain access to any WEP-protected network or packet capture.

WEP is designed to encrypt the payload of data frames on a wireless network using a shared key. The key, once selected, is distributed to all stations as a "pre-shared key" (PSK). The PSK itself is never exposed on the network, and so it is expected to be shared in some out-of-band way between the stations that need it.

Each station encrypts the payload of all data frames with the PSK and a randomly selected initialization vector (IV) so that the encryption key changes for every frame. The problem with using an IV in a reversible, symmetric encryption algorithm, such as RC4, is that stations have to supply the IV in plain text. Each station adds a cleartext 24-bit IV to each frame, but 24 bits is actually quite small when you consider the number of frames that can be transmitted across a WLAN. With only 24 bits of IV, the randomized values are bound to repeat at some point, given enough traffic. (This is guaranteed to happen after 2^{24} —or 16,777,216—frames. With a "maximum transmit unit" (MTU) of 1,500 bytes, that's less than 24GB of network data.)

As it turns out, however, after only a few thousand packets you can reliably guess that at least some of those packets have been encrypted with the same IV, but have different plain text input and ciphertext output. This enables attackers to leverage the "related-key attack", based on the knowledge of some of the bits of the key material.

An attacker's ability to leverage the related key attack depends on the volume of IVs exposed. On a quiet network, it may take weeks to capture enough IVs to crack the key. Fortunately for the attacker (unfortunately for the rest of us), there are weaknesses in WAP behaviors and implementations that allow attackers to force stations on a WLAN to generate large volumes of IVs. Using widely published tools, attackers can force the generation of enough IVs to crack a WEP key within minutes—even on an unused WLAN.

If you see anomalous behavior from an unknown station on a WEP-encrypted WAP, it could be that the station is attempting to crack the WEP key in order to gain access to the network. Commonly, WEP-cracking tools used on relatively quiet networks are designed to force local stations to generate unnecessary packets, with lots of IVs to speed cracking.

6.5 Locating Wireless Devices

Perhaps the single most challenging aspect of wireless networks for the investigator is the inherent difficulty in physically locating devices of interest. A compromised laptop may physically move throughout an enterprise’s network; a rogue wireless access point may be hidden in crafty places like under ceiling tiles.

Strategies for locating wireless devices include:

1. Gather station descriptors, such as MAC addresses, which can help provide a physical description so that you know what to look for;
2. For clients, identify the WAP that the station is associated with (by SSID);
3. Leverage commercial enterprise wireless mapping software;
4. Poll the device’s signal strength; and
5. Triangulate on the signal.

Of course all of this takes time, and is far more challenging if the device sought for is mobile and only transiently on the network—exactly the sort of thing that Wi-Fi networks were designed to accommodate.

6.5.1 Gather Station Descriptors

You can learn a lot about what a wireless device probably looks like from its network traffic. For example, recall our earlier discussion from Chapter 4, “Packet Analysis,” in which we learned that every network card is assigned a unique OUI by the manufacturer. The 802.11 frame indicates the source and destination station MAC addresses. (For wireless access points, the “BSSID” field in the 802.11 header is also the MAC address of the WAP’s network card.) Although MAC addresses can be changed, in most cases, no one bothers to change them. Hence, from sniffing Layer 2 network traffic and examining the MAC addresses in 802.11 frames of interest, you can make an educated guess as to the manufacturer of the device generating the traffic. Figure 6–9 shows the 802.11 frame of traffic between an Apple device and a Cisco WRT54G wireless router. Note that Wireshark automatically translates the OUI into a manufacturer description.

The content of wireless traffic itself can provide a surprising amount of insight regarding the physical description of a device. In Figure 6–10, we were able to crack the WEP key of the wireless traffic and decrypt the contents of the data frames. Now, we can see the contents of communications between the Apple device and its Layer 3 endpoint (routed through the Cisco WAP, of course). The traffic includes HTTP data, which contains User-Agent headers sent by the Apple device. The frame highlighted in Figure 6–10 reveals a User-Agent string “iTunes-iPad/3.2.1 (16GB).” That’s handy! Now we know that we’re most likely looking for a 16GB iPad, running OS version 3.2.1. This evidence correlates nicely with the Apple MAC address we examined moments ago.

6.5.2 Identify Nearby Wireless Access Points

Your strategy for locating a wireless device will depend in part on the function of the device. For example, you may be searching for a rogue wireless access point or a roving endpoint

No.	Time	Source	Destination	Protocol	Info
342242	479.620098	Cisco-Li_b3:cc:ee	Apple_3b:4e:52	IEEE 802.11	QoS Data, SN=232, FN=0, Flags=.p....F.
342243	479.620064		Cisco-Li_b3:cc: IEEE 002.11		Acknowledgement, Flags=.....
342244	479.620072		Apple_3b:4e:52	IEEE 802.11	Clear-to-send, Flags=.....
342245	479.620575	Apple_3b:4e:52	Cisco-Li_b3:cc: IEEE 802.11		QoS Data, SN=1920, FN=0, Flags=.p....T
342246	479.620611		Cisco-Li_b3:cc: IEEE 002.11		Clear-to-send, Flags=.....
342247	479.621634	Cisco-Li_b3:cc:ee	Apple_3b:4e:52	IEEE 802.11	QoS Data, SN=233, FN=0, Flags=.p....F.

Protocols in frame.wlan0.dat0

- IEEE 802.11 QoS Data, Flags: .p....T
- Type/Subtype: QoS Data (0x28)
- Frame Control: 0x4100 (Normal)
- Duration: 44
- BSS Id: Cisco-Li_b3:cc:f0 (00:1c:10:b3:cc:f0)
- Source address: Apple_3b:4e:52 (d8:a2:5e:3b:4e:52)
- Destination address: Cisco-Li_b3:cc:ee (00:1c:10:b3:cc:ee)

Figure 6–9. An 802.11 frame from an Apple device to a Cisco wireless router. Note that Wireshark automatically translates the OUI into a human-readable manufacturer description.

No.	Time	Source	Destination	Protocol	Info
144533	149.018452	204.0.59.58	10.5.5.113	HTTP	HTTP/1.1 200 OK (text/html)
144539	149.053233	10.5.5.113	204.0.59.40	HTTP	POST /WebObjects/MZSoftwreUpdate.w...
144550	149.400049	10.5.5.113	66.235.139.54	HTTP	GET /b/ss/applesuperglobal/1/G.6.-N...
144557	149.522225	10.5.5.113	204.0.59.35	HTTP	GET /htmlResources/C6DA/k2-storefr...

Frame 144550 (736 bytes on wire, 736 bytes captured)

Ethernet II, Src: Apple_3b:4e:52 (d8:a2:5e:3b:4e:52), Dst: Cisco-Li_b3:cc:ee (00:1c:10:b3:cc:ee)

Internet Protocol, Src: 10.5.5.113 (10.5.5.113), Dst: 66.235.139.54 (66.235.139.54)

Transmission Control Protocol, Src Port: 50231 (50231), Dst Port: http (80), Seq: 2280646653, Ack: 712155169,

Hypertext Transfer Protocol

[truncated] GET /b/ss/applesuperglobal/1/G.6.-NS?h5=appleitmssnapmb%2Appleitmssusapmb&pccr=true&pageName=Ap...

Host: metrics.apple.com\r\n

Cookie: Pod=8; s_v=[CS]v1|2623DAFF05013E32-60000109200003794[CE]\r\n

User-Agent: iTunes-iPad/3.2.1 (16GB)\r\n

Accept-Language: en;q=1.0,fr;q=0.9,de;q=0.8,ja;q=0.7,nl;q=0.6,it;q=0.5,es;q=0.4,zh-Hans;q=0.3,ru;q=0.2\r\n

X-Apple-Store-Front: 143441-1,9\r\n

X-Apple-Partner: origin.0\r\n

X-Apple-Connection-Type: WiFi\r\n

X-Dsid: 1320246249\r\n

0170 53 5d 76 31 7c 32 36 32 33 34 41 41 46 46 30 35 30 SJV1 262 3DFFF050
0180 31 33 45 33 32 2d 36 30 30 30 30 31 30 39 32 30 13E32-60 00010920
0190 30 30 33 37 39 34 5b 43 45 5d 0d 0a 55 73 65 72 003/94[C E]..User
01a0 2d 41 67 65 6e 74 3a 20 69 54 75 6e 65 73 2d 69 -Agent: iTunes-i
01b0 50 61 64 2f 33 2e 32 2e 31 20 28 31 36 47 42 29 Pad/3.2.1 (16GB)
01c0 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 ..Accept -Language

Figure 6–10. With the packet capture WEP-decrypted, we can see the User-Agent client-side HTTP header, which seems to confirm that the device is indeed an Apple.

station. In the case where you are searching for a client station that is actively associating with other WAPs, it is often helpful to identify which WAPs the station is associating with. Generally (although not always), endpoint stations associate with a wireless access point that is physically close by. In the case of a wireless bridged network, clients typically associate with the WAP in the bridged network that has the strongest signal, which is also often physically closest.

There are basically two ways to find out which WAPs a rogue endpoint client is associated with or attempting to associate with: WAP logs and traffic monitoring.

If you are lucky enough to be in an environment that captures wireless authentication attempts on a central logging system, you may be able to watch station association requests and responses by examining logs on the central server.

Otherwise, you can passively monitor the wireless traffic for association requests, responses, and other Layer 2 traffic related to the MAC address of interest. Generally, this requires that either you know the general vicinity of the rogue device already and can sniff traffic in that area, or that you have access to a wireless intrusion detection system with sensors distributed around a wide area.

In this way, you can track the station as it moves from device to device and locate the client using locations of known WAPs.

6.5.3 Signal Strength

There are many tools such as NetStumbler or Kismet that will list the nearby wireless access points and show you their relative signal strengths. Often, you can locate a mysterious wireless device simply by viewing the signal strengths using one of these applications and walking in the direction of increasing signal strength. This works well in situations where the station of interest is not mobile.

6.5.3.1 Received Signal Strength Indication (RSSI)

It is sometimes possible to see both the IEEE 802.11 Received Signal Strength Indication (RSSI) and the Transmit (Tx) Rate information when viewing a packet capture—but only if the tool that captured the packets supplies that data in its own additional framing. The 802.11 specification simply doesn't include such information in the data link-layer header.

If available, per-frame RSSI and Tx Rate information can be added manually to Wireshark's Packet List pane by editing user preferences.⁴⁰

6.5.3.2 NetStumbler

NetStumbler⁴¹ is a Windows tool designed to discover 802.11 networks. Though it is extremely popular for blackhats and whitehats alike, it is not totally passive, which means that its presence and activities can be detected by other wireless auditing tools. Like most tools of its kind, it supports GPS integration for the mapping of signals to physical locations, making it useful for “wardriving” or “warwalking.” NetStumbler is free for download, though not open source.

Due to considerable architectural differences between XP and Vista/Windows 7, NetStumbler does not work on the latter. Vistumbler is a similar tool designed to run on Vista, though provided by different authors, and so it has a different user interface and functionality.^{42, 43} A more popular replacement for all three platforms is inSSIDer.⁴⁴

40. A. Orebaugh et al., *Wireshark & Ethereal: Network Protocol Analyzer Toolkit* (Syngress, 2006).

41. Mariusm, “stumbler dot net,” February 16, 2010, <http://www.stumbler.net/>.

42. *Ibid.*

43. “Vistumbler,” December 12, 2010, <http://vistumbler.sourceforge.net/>.

44. “inSSIDer,” <http://www.metageek.net/products/inssider/>.

6.5.3.3 Kismet

Kismet is a widely used libpcap-based 802.11 “wireless network detector, sniffer, and intrusion detection system” for Linux/UNIX operating systems. It is both free and open source, and also capable of being totally passive. In addition to providing support for many attacks against authentication and encryption, it boasts the following features:⁴⁵

- Wireshark- and tcpdump-compatible data logging
- Network IP range detection
- Hidden network SSID decloaking
- Graphical mapping of networks
- Manufacturer and model identification of access points and clients
- Detection of known default access point configurations
- Runtime decoding of WEP packets for known networks
- Named pipe output for integration with other tools, such as a Layer 3 IDS like Snort
- Distributed remote drone sniffing
- XML output
- Over 20 supported card types

6.5.3.4 KisMAC

KisMAC is a free, open source 802.11 discovery tool for Mac OS X, which is open source and supports the same essential features as NetStumbler for Windows. However, KisMAC supports totally passive scanning, meaning that it can merely observe RF traffic without emitting any RF itself. KisMAC boasts the following features and attack strategies:⁴⁶

Features:

- Reveals hidden/cloaked/closed SSIDs
- Shows logged-in clients (with corresponding MAC addresses, IP addresses, and signal strengths)
- Mapping and GPS support
- Can draw area maps of network coverage
- PCAP import and export
- Support for 802.11b/g
- Different attacks against encrypted networks

45. “Kismet,” 2011, <http://www.kismetwireless.net/>.

46. “kismacng,” January 2011, <http://trac.kismac-ng.org/>.

- Deauthentication attacks
- AppleScript-able
- Kismet drone support (capture from a Kismet drone)

Crypto attack support:

- Bruteforce attacks against LEAP, WPA, and WEP
- Weak scheduling attack against WEP
- Newsham 21-bit attack against WEP

6.5.4 Commercial Enterprise Tools

Enterprises that deploy campus-wide wireless LANs often install central management consoles, which include mapping and station tracking capabilities. Vendors such as Aruba and Cisco offer specialized wireless tracking and WIDS software for use in these environments.

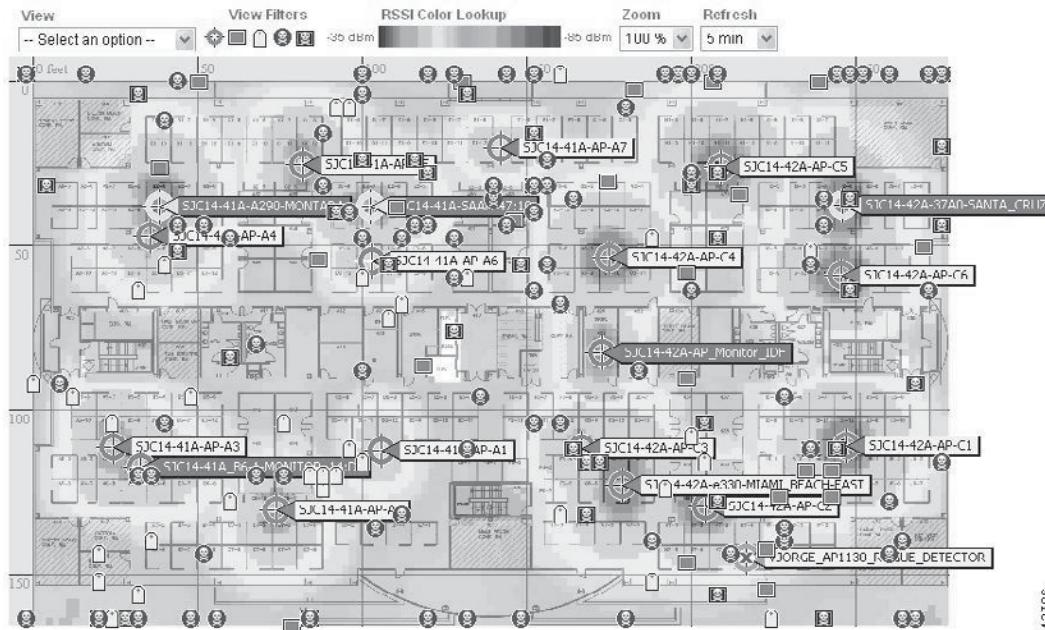
For example, organizations can deploy a mesh of access points throughout their facilities using Cisco products. The access points are controlled by Cisco Wireless LAN Controllers, which in turn can be centrally managed through the Cisco Wireless Control System (WCS). The WCS can display the location of a wireless device on floor maps that have been uploaded to the system and preconfigured by network administrators. The Cisco Wireless Location Appliance (WLA) has even greater wireless device tracking and searching capabilities. The WLA receives data from the Wireless LAN Controllers, and uses this to provide network administrators with a central console for tracking wireless devices throughout the enterprise.⁴⁷ Figure 6–11 shows a screenshot of devices located on an enterprise floor map. Known devices are marked with a box, while “rogue” devices are labeled with a skull.

6.5.5 Skyhook

Skyhook Wireless Positioning System (WPS) is a proprietary location tracking service provided by Skyhook Wireless. It is an extremely popular alternative to GPS, especially because it works well indoors and can provide results with 10–30m of accuracy in urban environments where GPS is less effective. For years, the company has employed hundreds of “data specialists” to wardrive and collect BSSIDs for beaconing WAPs and cell tower IDs, along with corresponding GPS data.⁴⁸ Using this information, they have compiled an intensive database. When a client queries the Skyhook API in order to determine its own location, it sends information about surrounding wireless devices and cell towers to Skyhook’s Location Server. The Location Server searches the database for matches and provides the client with the corresponding GPS coordinates.

47. “Cisco 2700 Series Wireless Location Appliance Deployment Guide [Cisco Wireless Location Appliance],” Cisco Systems, 2011, <http://www.cisco.com/en/US/docs/wireless/technology/location/deployment/guide/depgd.html>.

48. “Skyhook: How It Works,” <http://www.skyhookwireless.com/howitworks/>.



142708

Figure 6–11. A screenshot of Cisco’s Wireless Location Appliance, which displays devices located on an enterprise floor map and allows system administrators to search and sort. This is WLA’s main view, listing stations detected, BSSID, signal strength, and more. Known devices are marked with a box, while “rogue” devices are labeled with a skull. Courtesy of Cisco Systems, Inc. Unauthorized use not permitted.⁴⁹

Popular products that use the Skyhook WPS include Apple iPhones, which have a “Locate Me” feature, and Eye-Fi SD cards, which tag photos taken by digital cameras with their GPS coordinates.

In 2008, several researchers published articles describing how to leverage the Skyhook system in order to get GPS coordinates of arbitrary MAC addresses.⁵⁰ With knowledge of a WAP’s MAC address, third parties can query the Skyhook system to obtain a location from Skyhook’s highly granular database.⁵¹ This may or may not be legal, depending on your purposes and the laws in your area. However, you may be involved in an investigation where there is a legitimate need and legal way to query the Skyhook API for an arbitrary MAC address.

49. “Cisco 2700 Series Wireless Location Appliance Deployment Guide [Cisco Wireless Location Appliance],” Cisco Systems, 2011, <http://www.cisco.com/en/US/docs/wireless/technology/location/deployment/guide/depgd.html#wp37848>.

50. thebmxr, “Don’t Locate Me,” 2011, http://thebmxr.googlepages.com/Don_t_Locate_me.pdf.

51. Coderrr, “Get the Physical Location of Wireless Router from Its MAC Address (BSSID),” September 10, 2008, <http://coderrr.wordpress.com/2008/09/10/get-the-physical-location-of-wireless-router-from-its-mac-address-bssid/>.

6.6 Conclusion

With the explosion of mobile devices and wireless deployments in enterprises and homes, forensic investigators are increasingly tasked with examining wireless traffic and devices. Compared with wired Ethernet networks, wireless networks pose special challenges. For example, the 802.11 data link-layer protocol suite is fundamentally different from Ethernet, and even details such as the order that bits are transmitted may surprise network forensics investigators used to Ethernet traffic analysis.

In this chapter, we studied the IEEE's 802.11 protocol series, including encryption algorithms and their flaws. We talked about the types of evidence that you can gather from wireless access points, and touched on wireless traffic capture and analysis. We reviewed common attacks on wireless networks that investigators should be familiar with so that you can recognize them in the field. Finally, we discussed one of the most common hurdles facing wireless network forensic investigators: locating wireless devices. The very feature that makes wireless networks so popular—client mobility—is perhaps our biggest investigative challenge.

6.7 Case Study: HackMe, Inc.

The Case: *September 17th, 2010: InterOptic is on the lam and is pinned down. The area is crawling with cops, and so he must stay put. But he also desperately needs to be able to get a message out to Ann and Mr. X. Lucky for him, he detects a wireless access point (WAP) in the building next door that he might be able to use. But it is using encryption, and there are no other opportunities available. What is InterOptic to do?*

Meanwhile ... *Next door, Joe is a sysadmin at HackMe, Inc. He runs the technical infrastructure for a small company, including a WAP that is used pretty much exclusively by him. He's trying to use it now, and has discovered that he's begun to get dropped. He captures some traffic, but he really has no idea how to interpret it. Suddenly he discovers he can't even login to administer his WAP at all!*

The Challenge: You are the forensic investigator. Your team got a tip that InterOptic might be hunkered down in the area. Can you figure out what's going on and track the attacker's activities?

The following questions will help guide your investigation:

- What are the BSSID and SSID of the WAP of interest?
- Is the WAP of interest using encryption?
- What stations are interacting with the WAP and/or other stations on the WLAN?
- Are there patterns of activity that seem anomalous?
- How are they anomalous: Consistent with malfunction? Consistent with maliciousness?
- Can we identify any potentially bad actors?
- Can we determine if a bad actor successfully executed an attack?

Evidence: Joe has provided you with a packet capture (wlan.pcap) and permission to inspect it in any way you need to either solve his problem, catch InterOptic, or both. He also helpfully tells you that his own system's MAC address is 00:11:22:33:44:55, and reiterates that no one else should be using his WAP.

6.7.1 Inspecting the WAP

The most obvious place to begin analysis is Joe's WAP. Along the way we expect—or at least hope—to learn something about the stations with which it was communicating, and to be able to infer a whole lot from the anomalous traffic we're about to examine. Let's begin by identifying and inspecting the WLAN under investigation.

6.7.1.1 Inspecting Beacon Frames

Probably the most straightforward way to identify the WAPs in a packet capture is to simply filter on Beacon frames. Figure 6–12 demonstrates how Wireshark can be used with a display filter on the appropriate frame type (0) and subtype (8): “wlan.fc.type_subtype == 0x08.” Note also the “BSS Id” in the frame: 00:23:69:61:00:d0.

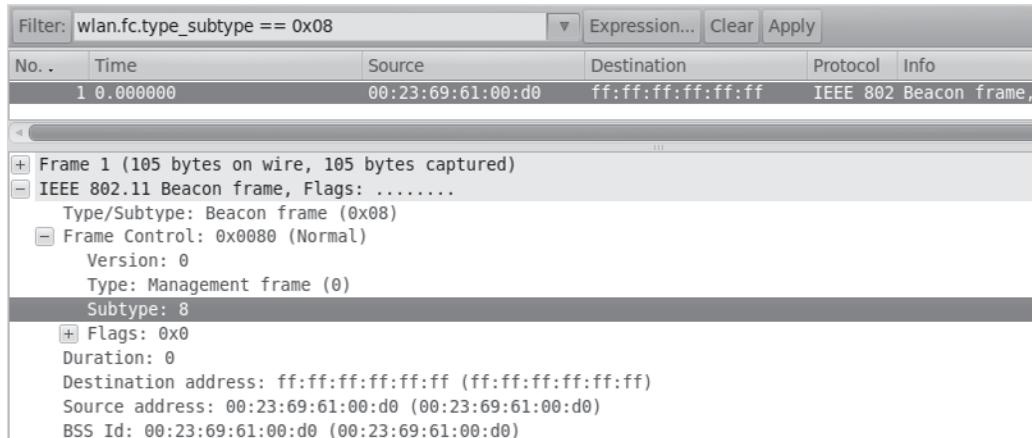


Figure 6–12. An 802.11 management frame shown in Wireshark. As you can see in the Packet Details pane, this frame is type 0, subtype 8: a Beacon frame.

Using tcpdump with the BPF language, we can easily find this Beacon frame too, so long as we mind our endianness:

```
$ tcpdump -nne -r wlan.pcap 'wlan[0] = 0x80' reading from file wlan.pcap,
link-type IEEE802_11 (802.11) 09:56:41.085810 BSSID:00:23:69:61:00:d0 DA:
ff:ff:ff:ff:ff:ff SA:00:23:69:61:00:d0 Beacon (MentOrNet) [1.0* 2.0* 5.5*
11.0* 18.0 24.0 36.0 54.0 Mbit] ESS CH: 2, PRIVACY
```

We see the same BSSID as before, and some other useful information (SSID, channel, etc.). But what if the WAP of interest was specifically configured *not* to send Beacon frames? That's not as big a problem for us as many people might think.

6.7.1.2 Filter on WAP-Announcing Management Frames

Let's use our tshark invocation from Section 6.3.3.1 to filter traffic and display only Beacon and Probe Response frames that have the ESS subfield set to 1 and the IBSS subfield set to 0. (Recall that, by specification, WAPs set these fields accordingly.) Even if a WAP is not broadcasting Beacon frames, it may still send Probe Responses to stations that initiate Probe Requests.

```
$ tshark -nn -r wlan.pcap -R '((wlan.fc.type_subtype == 0x08 || wlan.fc.
type_subtype == 0x05) && (wlan_mgt.fixed.capabilities.ess == 1) && (
wlan_mgt.fixed.capabilities.ibss == 0))'
1  0.000000 00:23:69:61:00:d0 -> ff:ff:ff:ff:ff:ff 802.11 105 Beacon frame
, SN=3583, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
265 20.409086 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
Response, SN=3801, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
270 20.597504 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
Response, SN=3804, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
335 23.318463 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
Response, SN=3837, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
412 26.317951 00:23:69:61:00:d0 -> 00:11:22:33:44:55 802.11 211 Probe
Response, SN=3873, FN=0, Flags=. ...., BI=100, SSID=MentOrNet
[...]
```

As you can see in Figure 6–13, you can use the same display filter in Wireshark to find Beacon or Probe Response frames that emanate from WAPs. To list the BSSIDs of the known WAPs in the packet capture, let’s use tshark along with some simple shell scripting. In the command below, we tell tshark to print only the BSSID field (“wlan.bssid”), and then we use the Linux tool “uniq” to get a count of the number of occurrences of each BSSID.

```
$ tshark -nn -r wlan.pcap -R '((wlan.fc.type_subtype == 0x08 || wlan.fc.type_subtype == 0x05) && (wlan_mgt.fixed.capabilities.ess == 1) && (wlan_mgt.fixed.capabilities.ibss == 0))' -T fields -e wlan.bssid | uniq -c
174 00:23:69:61:00:d0
```

As you can see, only one WAP sent Beacon or Probe Response frames in the packet capture—and it sent 174 of these frames. This WAP had the BSSID “00:23:69:61:00:d0.” We can confirm with Joe that that is indeed the BSSID printed on the label that Linksys put on the device, and that it is configured with the SSID “Ment0rNet.”

6.7.1.3 Take Inventory of Stations on the WLAN

Returning to the Beacon frame from earlier, let us inspect it further with Wireshark so that we can record a few more details about the WLAN that we may need later (and perhaps even create some display filters we can use later on with “tshark” to speed things up). Along the way we’ll answer a number of the questions posed in the challenge.

Figures 6–14, 6–15, and 6–16 highlight the BSSID, SSID, and channel, respectively. In these frames, we see that the WAP’s BSSID is “00:23:69:61:00:d0,” the SSID is “Ment0rNet,” and the WAP is operating on channel 2.

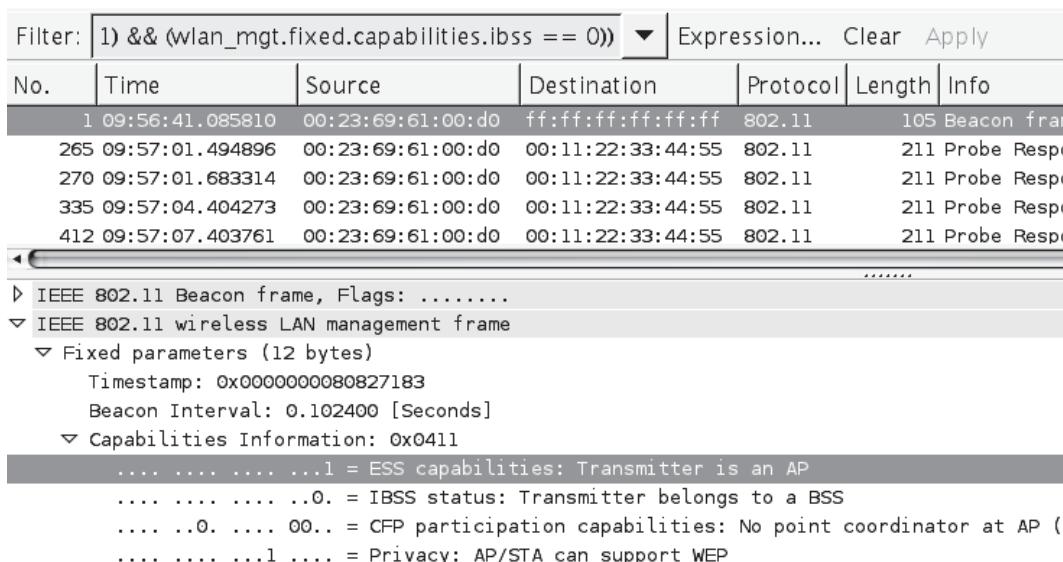


Figure 6–13. A screenshot of Wireshark used with a display filter that tests for frames that emanate from WAPs. The display filter used is “((wlan.fc.type_subtype == 0x08 || wlan.fc.type_subtype == 0x05) && (wlan_mgt.fixed.capabilities.ess == 1) && (wlan_mgt.fixed.capabilities.ibss == 0)).”

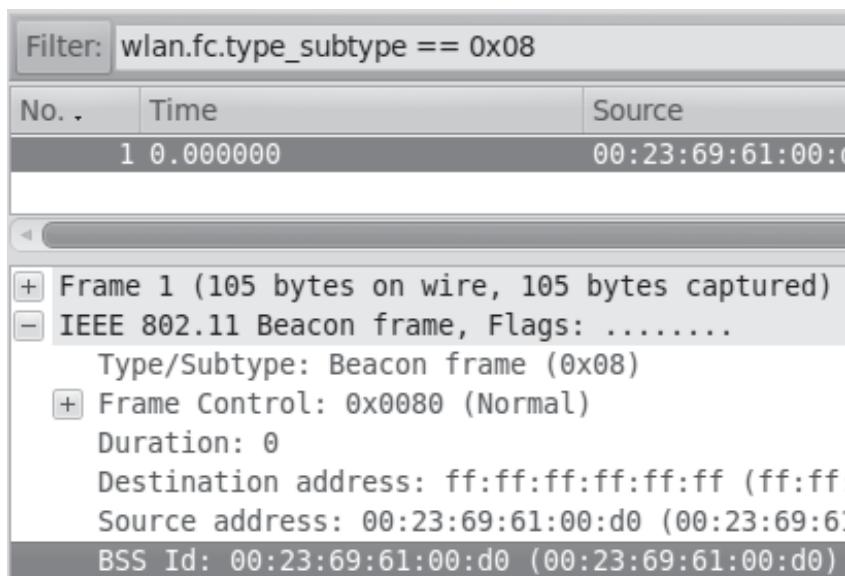


Figure 6–14. An 802.11 management frame with subtype 0x8 (a Beacon frame). The BSSID (00:23:69:61:00:d0) is highlighted.

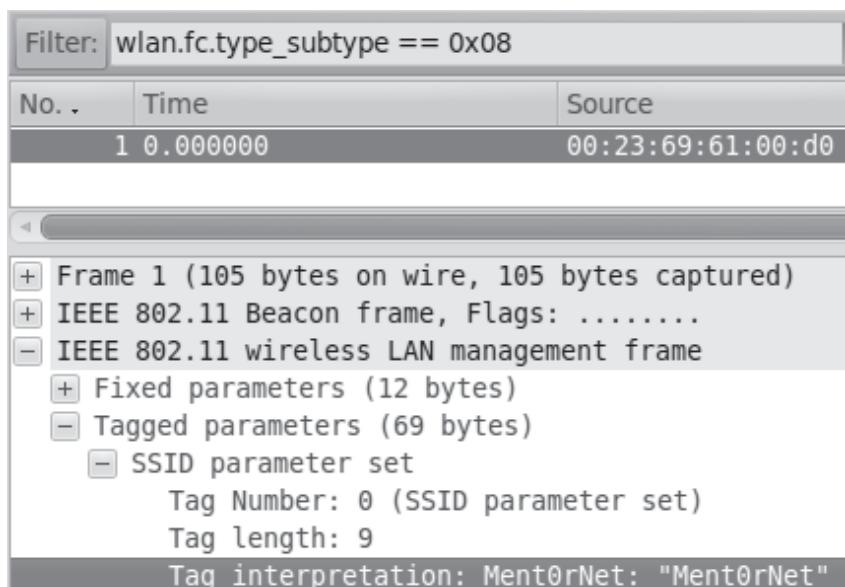


Figure 6–15. An 802.11 management frame with subtype 0x8 (a Beacon frame). The SSID parameter set is shown, with the tag interpretation ("Ment0rNet") highlighted.

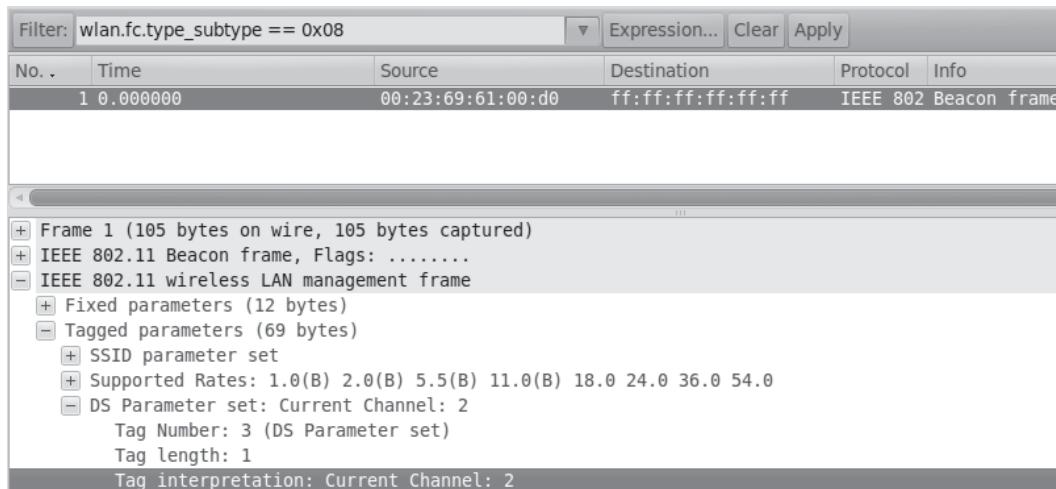


Figure 6–16. An 802.11 management frame with subtype 0x8 (a Beacon frame). The field containing “Current Channel” (channel 2) is highlighted.

6.7.1.4 WLAN Encryption

Is the WLAN using encryption? Yes, it does appear to be. To answer this question, first, we can ask Wireshark to show us only the data frames: “wlan.fc.type_subtype == 0x20.” Figure 6–17 shows an example of a data frame. Notice the Protected bit (highlighted) is set to 1.

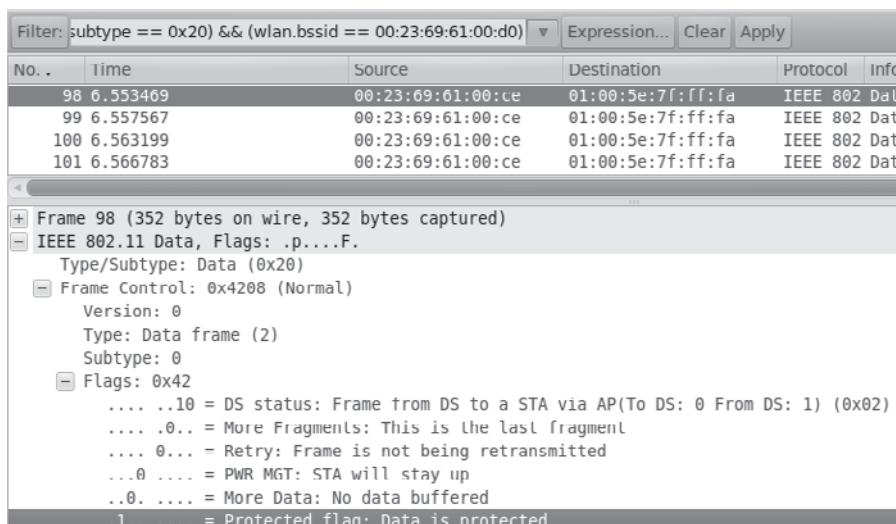


Figure 6–17. An 802.11 data frame shown in Wireshark. Notice that the “Protected” flag (highlighted) is set, indicating that the data is encrypted.

Using tcpdump and some BPF inspection, we can easily demonstrate that *all* the data frames in the capture are WEP protected. First, let's count the data frames. By filtering on the version (0), type (2) and subtype (0) in the first byte of the frame (the byte at "offset zero"), and sending the output to the "wc" command to count lines, we get 59,274 total data frames:

```
$ tcpdump -nne -r wlan.pcap 'wlan[0] = 0x08' | wc -l
reading from file wlan.pcap, link-type IEEE802_11 (802.11)
59274
```

Next, let's also filter on the "Protected" bit in the next byte (again, keeping in mind the endianness of the bits in transmission):

```
$ tcpdump -nne -r wlan.pcap 'wlan[0] = 0x08 and wlan[1] & 0x40 = 0x40' | wc -l
reading from file wlan.pcap, link-type IEEE802_11 (802.11)
59274
```

The resulting count of data frames is the same with or without a filter on a positive value for the "Protected" bit: 59,274. This means that all 59,274 data frames are also encrypted.

6.7.1.5 Associated Stations

Using Wireshark, we can easily construct a filter on the Association Response subtype of the management frames, and further filter on the 2-byte status code indicating successful association: "wlan.fc.type_subtype == 0x01 && wlan_mgt.fixed.status_code == 0x0000." Presumably the source of such frames should all be our known BSSID, and the various destinations would be the stations that successfully associated. Figure 6–18 illustrates this approach, as well as the large number of frames which show successful association.

With a little bit of tcpdump and BPF language, and some more command-line *fu*, we can aggregate data from these frames in a useful way. We begin by filtering on the version/type/subtype byte for the appropriate value for an Association Response (0x01, which translates to 0x10 in transmission order), and then locate the 2-byte field that indicates the status code (the 2 bytes starting at the 26th byte offset (remember, counting from 0)).

In the tcpdump output produced by the command below, the destination MAC address is the third field. We use the "awk" command to print only the third field, and then send these

Filter: wlan.fc.type_subtype == 0x01 && wlan_mgt.fixed.status_code == 0x0000		Expression...		Clear	Apply
No.	Time	Source	Destination	Protocol	Info
278	20.630783	00:23:69:61:00:d0	00:11:22:33:44:55	IEEE 802 Association Resp	
3884	130.691710	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3707	131.552957	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3723	132.576510	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3742	133.540671	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3759	134.542204	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3774	135.547326	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3775	135.548863	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3813	138.535551	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3838	139.541183	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3871	140.564223	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
3926	141.527871	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
4696	142.531967	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
4374	143.534527	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	
4415	143.643581	00:23:69:61:00:d0	00:11:22:33:44:55	IEEE 802 Association Resp	
4701	144.525311	00:23:69:61:00:d0	1c:4b:d6:69:cd:07	IEEE 802 Association Resp	

Figure 6–18. So very many successful association messages ...

values line-by-line to “sort,” whose output then allows the “uniq” program to aggregate and count them. By sending the remaining output through the last “sort” invocation, we can see them listed in order of frequency.

```
$ tcpdump -nne -r wlan.pcap 'wlan[0] = 0x10 and wlan[26:2] = 0x0000' | awk '{  
    print $3}' | sort | uniq -c | sort -nr  
reading from file wlan.pcap, link-type IEEE802_11 (802.11)  
68 DA:1c:4b:d6:69:cd:07  
 4 DA:00:11:22:33:44:55  
 1 DA:de:ad:be:ef:13:37
```

It seems that Joe successfully associated four times, an unknown station with the MAC address of “de:ad:be:ef:13:37” associated once, and another unknown station with the MAC address of “1c:4b:d6:69:cd:07” successfully associated 68 times. That seems a bit odd.

6.7.2 Quick-and-Dirty Statistics

Let’s gather some statistics on the traffic in this packet capture in order to better understand the actors and time frame.

6.7.2.1 Who Are the People in Your Neighborhood?

Let’s look again at the encrypted data frames. How many encrypted data frames are there? Who are they coming from and where are they going? Are there any that seem unusual?

Tshark is an excellent tool to help us look at statistics, as well as individual packets. Let’s begin by trying out a display filter that should provide a known result: the number of protected data frames in the capture. Since we know the BSSID of the WAP that is the focus of our investigation (00:23:69:61:00:d0), we will include the filter “wlan.bssid == 00:23:69:61:00:d0” in our tshark invocations in order to narrow the scope to our identified target of interest.

Note that in the display filter below, we specify the value of the type/subtype fields as they appear in the protocol specification (0x20), NOT the order in which they are transmitted (0x08). This is an important difference between the display filters we use with tshark/WireShark and the BPF filters we used previously with tcpdump.

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.  
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0)' | wc -l  
59274
```

That number corresponds with the results of our previous inspection. From there, we can begin to use tshark to extract individual fields from the WLAN protocol for aggregation and comparison. This invocation, the end of which should be familiar now, shows us an aggregated count of the number of encrypted data frames transmitted from each MAC address:

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.  
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0)' -T fields -e wlan.  
sa | sort | uniq -c | sort -nr  
42816 1c:4b:d6:69:cd:07  
14127 00:11:22:33:44:55  
1574 00:23:69:61:00:ce  
 757 de:ad:be:ef:13:37
```

In the example above, by extracting only the “sending address” field of the WLAN protocol (“wlan.sa”) from the data frames, and sorting and counting their occurrences, we can see that one of our unknown stations (1c:4b:d6:69:cd:07) sent roughly three times the number of data frames as Joe’s station during the same time period.

We also see a source of “00:23:69:61:00:ce,” which differs from the WAP’s BSSID by only the last octet. It’s interesting that this MAC address is very similar to that of the WAP’s BSSID. Recall that WAPs typically serve at least two functions: First, they provide access to wireless distribution services, and second, they act as stations on the network which provide services for WAP management, DHCP, logging, and other functionality. It is common to see different addresses used for different purposes. In this case, let’s hypothesize that the similarity between the MAC address “00:23:69:61:00:ce” and the WAP’s BSSID (00:23:69:61:00:d0) is not a coincidence. It is likely that “00:23:69:61:00:ce” is the WAP’s MAC address which it uses to participate as a logically distinct station in the wireless network. We will henceforth refer to this informally as the WAP’s “station” (STA) interface.⁵²

Last but not least, we also see a few data frames from the odd MAC address, “de:ad:be:ef:13:37.”

What if we were to look at the comparative count of frame *destination* addresses (“wlan.da”) in the same fashion?

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0)' -T fields -e wlan.
da|sort|uniq -c|sort -nr
42837 ff:ff:ff:ff:ff:ff
14816 00:23:69:61:00:ce
858 00:11:22:33:44:55
654 de:ad:be:ef:13:37
59 01:00:5e:7f:ff:fa
25 33:33:00:00:00:02
17 33:33:00:00:00:16
6 33:33:ff:33:44:55
2 33:33:ff:ef:13:37
```

Interesting: The results shown above indicate that approximately 42,837 encrypted data frames were sent to the broadcast MAC address (ff:ff:ff:ff:ff:ff). This is almost the same as the number of frames sent from station 1c:4b:d6:69:cd:07. Similarly, station 00:23:69:61:00:ce (likely the WAP’s STA interface) received roughly the same number of encrypted data frames as Joe’s station (00:11:22:33:44:55). Coincidence?

Let’s look at both source and destination:

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0)' -T fields -e wlan.
sa -e wlan.da|sort|uniq -c|sort -nr
42816 1c:4b:d6:69:cd:07 ff:ff:ff:ff:ff:ff
14076 00:11:22:33:44:55 00:23:69:61:00:ce
858 00:23:69:61:00:ce 00:11:22:33:44:55
740 de:ad:be:ef:13:37 00:23:69:61:00:ce
```

52. IEEE, “IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications” (June 12, 2007): 40–41, <http://standards.ieee.org/getieee802/download/802.11-2007.pdf> (accessed December 31, 2011).

```

654 00:23:69:61:00:ce de:ad:be:ef:13:37
59 00:23:69:61:00:ce 01:00:5e:7f:ff:fa
18 00:11:22:33:44:55 33:33:00:00:00:02
14 00:11:22:33:44:55 ff:ff:ff:ff:ff:ff
13 00:11:22:33:44:55 33:33:00:00:00:16
7 de:ad:be:ef:13:37 33:33:00:00:00:02
6 00:11:22:33:44:55 33:33:ff:33:44:55
4 de:ad:be:ef:13:37 ff:ff:ff:ff:ff:ff
4 de:ad:be:ef:13:37 33:33:00:00:00:16
3 00:23:69:61:00:ce ff:ff:ff:ff:ff:ff
2 de:ad:be:ef:13:37 33:33:ff:ef:13:37

```

Sure enough, that mystery station (1c:4b:d6:69:cd:07) sent all 42,816 of its data frames to the broadcast address. Most of the remaining frames were sent between Joe’s station and the WAP’s presumed STA interface (with the majority of those frames sent *from* Joe’s station). The next most common exchanges appear to be between the other odd station, “de:ad:be:ef:13:37,” and the WAP.

To put this in perspective, of 59,274 data frames:

- 72% were sent from an unknown station (1c:4b:d6:69:cd:07) to the broadcast address (ff:ff:ff:ff:ff:ff)
- 25% were transmitted between Joe’s station (00:11:22:33:44:55) and the WAP (00:23:69:61:00:ce)
- 2% were transmitted between an unknown station (de:ad:be:ef:13:37) and the WAP (00:23:69:61:00:ce)

There are relatively few reasons that data frames would be sent to the broadcast MAC address. Perhaps the most common case is when clients send out ARP requests. However, the volume of traffic sent by 1c:4b:d6:69:cd:07 compared with other stations seems abnormally high to be legitimate ARP requests.

Another possibility is that 1c:4b:d6:69:cd:07 was conducting some sort of attack on the wireless network. WEP-cracking attacks often involve the attacker sending out a large number of 802.11 data frames. Recall that an attacker’s ability to leverage the related key attack depends on the volume of unique IVs exposed. In order to capture lots of unique IVs quickly, the attacker needs to send out traffic that triggers other stations on the network to respond. An effective tactic is to replay ARP requests, because ARP requests elicit timely responses from other systems.

A malicious actor can capture 802.11 frames and replay them on the network, even without knowing the WEP key. How does an attacker know which 802.11 traffic to replay if the traffic is encrypted? In the classic ARP replay attack, the attacker listens for data frames sent to the broadcast MAC address—likely ARP requests—and just blindly replays them. In this way, the attacker has a good chance of generating traffic that will trigger a response from other stations on the WLAN. By replaying data packets to the broadcast Layer 2 address, an attacker can cause other stations to generate frames with unique IVs, which can then be captured and used to dramatically speed up the WEP-cracking attack.

6.7.2.2 Patterns and Time Frames

Are there any patterns or stations that seem unusual (perhaps considering the known vs. unknown stations by volume)? Let's use the Wireshark suite's "capinfos" tool to quickly view the duration of the packet capture and other high-level information.

```
$ capinfos wlan.pcap
File name: wlan.pcap
File type: Wireshark/tcpdump/... - libpcap
File encapsulation: IEEE 802.11 Wireless LAN
Number of packets: 133068
File size: 8685397 bytes
Data size: 6556285 bytes
Capture duration: 414 seconds
Start time: Fri Sep 17 09:56:41 2010
End time: Fri Sep 17 10:03:34 2010
Data byte rate: 15852.64 bytes/sec
Data bit rate: 126821.09 bits/sec
Average packet size: 49.27 bytes
Average packet rate: 321.75 packets/sec
```

Note that our forensic workstation's time zone was MDT (GMT-6) at the time of this analysis. The human-readable dates and times will be listed in MDT where applicable throughout the remainder of this case study. Please adjust for time zone accordingly when you do our own analysis.

For more granular timestamps, we can use tcpdump to print the first and last frame:

```
$ tcpdump -nnr wlan.pcap|head -1
reading from file wlan.pcap, link-type IEEE802_11 (802.11)
09:56:41.085810 Beacon (MentOrNet) [1.0* 2.0* 5.5* 11.0* 18.0 24.0 36.0 54.0
Mbit] ESS CH: 2, PRIVACY

$ tcpdump -nnr wlan.pcap|tail -1
reading from file wlan.pcap, link-type IEEE802_11 (802.11)
10:03:34.662764 Request-To-Send TA:00:02:6f:05:cb:11
```

From the output above, we see that the total duration of the packet capture is approximately 414 seconds, or 6.9 minutes. In that time frame, Joe's station (11:22:33:44:55:66) sent the following data frames to the destinations shown below:

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa ==
00:11:22:33:44:55)' -T fields -e wlan.da|sort|uniq -c|sort -nr
14076 00:23:69:61:00:ce
    18 33:33:00:00:00:02
    14 ff:ff:ff:ff:ff:ff
    13 33:33:00:00:00:16
     6 33:33:ff:33:44:55
```

In that same amount of time, the unknown station 1c:4b:d6:69:cd:07 sent 42,816 data frames to broadcast, and none to anyone else:

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.
protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b
:d6:69:cd:07)' -T fields -e wlan.da|sort|uniq -c|sort -nr
42816 ff:ff:ff:ff:ff:ff
```

As we see below, this broadcast traffic from 1c:4b:d6:69:cd:07 occurred over a span of less than 69 seconds, from 09:59:42 to 10:00:50:

```
$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07)' -T fields -e frame.time|awk '{print $4}'|head -1
09:59:42.220425000

$ tshark -r wlan.pcap -R '((wlan.fc.type_subtype == 0x20) && (wlan.fc.protected == 1)) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07)' -T fields -e frame.time|awk '{print $4}'|tail -1
10:00:50.972590000
```

That seems unusual, and worthy of further inquiry. Typically, being unable to explain such a large proportion of network activity would be a huge red flag. In this case, however, our sample size is so temporally short that it may only seem anomalous on a small scale. Perhaps some benign (if annoying) process on this device causes it to do the same thing every 10 minutes, and we've caught only one such burst.

Another interesting pattern to examine might be the distribution of communication partners with 00:23:69:61:00:ce (the WAP's STA interface). It would be useful to find out what Layer 3+ services this device provides (perhaps from Joe, or through our own inspection). Typically DHCP is provided, as would be ARP service for the associated devices.⁵³ In addition, most WAPs provide remote administrative access through the application layer—most commonly via HTTP (or better, HTTPS).

We've seen that there are data frames coming from the WAP's BSSID and STA MAC addresses. Let's drill down further and inspect the data frames *sent* by the WAP. We look first at the distribution of destinations by number of data frames coming from the WAP's BSSID (00:23:69:61:00:d0):

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 2) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0)' -T fields -e wlan.da |sort|uniq -c|sort -nr
44 00:11:22:33:44:55
2 1c:4b:d6:69:cd:07
1 de:ad:be:ef:13:37
```

Notice that Joe's station (00:11:22:33:44:55) received far more data frames from the WAP's BSSID interface than any other station on the network.

Let's also look at the data frames coming from the WAP's STA interface (00:23:69:61:00:ce) and the distribution of their destinations:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 2) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:ce)' -T fields -e wlan.da |sort|uniq -c|sort -nr
858 00:11:22:33:44:55
654 de:ad:be:ef:13:37
59 01:00:5e:7f:ff:fa
3 ff:ff:ff:ff:ff:ff
```

⁵³. Address resolution between MAC and IP addresses is a service that any Layer 2 protocol must provide. However, any protocol for doing so must be dependent upon the details of both Layer 2 and Layer 3 protocols to function properly, and so 802.11's ARP is necessarily different from Ethernet's. In the former case, it becomes necessary to put ARP traffic into data frames.

As we might reasonably expect, most of the data frames from 00:23:69:61:00:ce (the WAP’s STA interface) were sent to Joe’s station. After all, he’s likely logged onto the device’s web-based administrative interface in hopes of troubleshooting the situation. Given that Joe is the network administrator, if anyone is interacting directly with the WAP above Layer 2, it should be him. However, as we saw in the command output above, a significant percentage of frames from the 00:23:69:61:00:ce were sent to the unknown station de:ad:be:ef:13:37.⁵⁴ It’s certainly reasonable to wonder why de:ad:be:ef:13:37 appears to have had significant interaction directly with the WAP.

Why not look at the distribution of destination MAC addresses for data frames originating from de:ad:be:ef:13:37? That might give us some context:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 2) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == de:ad:be:ef:13:37)' -T fields -e wlan.da | sort | uniq -c | sort -nr
740 00:23:69:61:00:ce
    7 33:33:00:00:00:02
    4 ff:ff:ff:ff:ff:ff
    4 33:33:00:00:00:16
    2 33:33:ff:ef:13:37
    2 00:23:69:61:00:d0
```

It’s clear from the above output that within the time frame of this packet capture, most of the data frames sent from de:ad:be:ef:13:37 were directed to one other station—the WAP’s STA interface. Given that we don’t yet know who or what de:ad:be:ef:13:37 is, other than that it appears to have successfully associated with our WAP, it might be a good idea to place those frames in context. In particular, how does the traffic from de:ad:be:ef:13:37 fit into our timeline? Let’s use tshark to obtain the start and end times of traffic originating from de:ad:be:ef:13:37.

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 2) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == de:ad:be:ef:13:37) && (wlan.da == 00:23:69:61:00:ce)' -T fields -e frame.time | awk '{print $4}' | head -1
10:02:14.181505000

$ tshark -r wlan.pcap -R '(wlan.fc.type == 2) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == de:ad:be:ef:13:37) && (wlan.da == 00:23:69:61:00:ce)' -T fields -e frame.time | awk '{print $4}' | tail -1
10:03:32.836868000
```

It appears that traffic occurred toward the end of the packet capture.

6.7.2.3 The Timeline So Far ...

Putting together the information we’ve gathered so far, we can start to build a timeline, as shown below (times are rounded to the nearest second):

- **09:56:41**—Packet capture begins
- **09:59:42 to 10:00:51**—1c:4b:d6:69:cd:07 broadcasted large volumes of data frames

54. Obviously the MAC address itself is odd—almost like someone configured it to be easily identified in a lab environment (much like Joe’s station’s MAC). Our attacker, Eric, has been kinder in this regard than most actual perpetrators.

- **10:02:14 to 10:03:33**—de:ad:be:ef:13:37 sent small volume of data frames to the WAP’s STA interface (00:23:69:61:00:ce)
- **10:03:35**—Packet capture ends

6.7.3 A Closer Look at the Management Frames

Given the stations of interest we identified, let’s look at management frames specific to the BSSID of interest. To do this we can use Wireshark’s display filter “(wlan.fc.type == 0) && (wlan.bssid == 00:23:69:61:00:d0).”

6.7.3.1 Frequency by Source and Destination

Let’s start by obtaining a breakdown of the number of management frames by sender, to see if anything stands out. We can use tshark to show us this information:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 0) && (wlan.bssid == 00:23:69:61:00:d0)' -T fields -e wlan.sa|sort|uniq -c|sort -nr
14858 00:23:69:61:00:d0
  146  1c:4b:d6:69:cd:07
  100  00:11:22:33:44:55
    6  de:ad:be:ef:13:37
```

It is reasonable to expect the WAP to send out more management frames than the stations, but two orders of magnitude more seems unusual, especially when there are only a few stations.

Let’s examine the WAP’s outbound management frames (from the BSSID interface) and sort them by destination MAC address:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 0) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0)' -T fields -e wlan.da|sort|uniq -c|sort -nr
12217  1c:4b:d6:69:cd:07
  2455  ff:ff:ff:ff:ff:ff
  126  00:11:22:33:44:55
    60  de:ad:be:ef:13:37
```

The overwhelming majority of management frames sent by the WAP’s BSSID interface were destined to one of the unknown stations—two orders of magnitude more than were sent to Joe’s station in the same time period. For that matter, the WAP’s BSSID interface sent more than 20 times as many management frames to the broadcast address as it did to Joe’s station.

Let’s view statistics regarding the subtypes of this traffic. In the output below, the first column is the number of matching frames, the second column is the management frame subtype, and the third column is the destination MAC address.

```
$ tshark -r wlan.pcap -R '(wlan.fc.type == 0) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0)' -T fields -e wlan.fc.subtype -e wlan.da|sort|uniq -c|sort -nr
12076  10  1c:4b:d6:69:cd:07
  2454  12  ff:ff:ff:ff:ff:ff
  118   5  00:11:22:33:44:55
```

```

73  11  1c:4b:d6:69:cd:07
68  1   1c:4b:d6:69:cd:07
55  5   de:ad:be:ef:13:37
  4 11  00:11:22:33:44:55
  4 1   00:11:22:33:44:55
  2 11  de:ad:be:ef:13:37
  1 8   ff:ff:ff:ff:ff:ff
  1 3   de:ad:be:ef:13:37
  1 1   de:ad:be:ef:13:37
  1 12  de:ad:be:ef:13:37

```

Based on our analysis, the overwhelming majority of management frames from the WAP's BSSID interface were sent to the unknown station 1c:4b:d6:69:cd:07, and were subtype 10 (0x0a): Disassociation. In other words, Joe's WAP spent most of its frames trying to tell this station to "get lost."

Its next-highest count of frames were broadcasting subtype 12 (0x0c): Deauthentication! In other words, it looks like Joe's WAP *also* spent a bunch of time telling *everyone* to get lost! No wonder Joe was having problems.

Let's look at the timing of these particular management frames. We'll start by testing out a filter that should match all of the subtype 10 traffic within the WLAN of interest that was sent from the WAP's BSSID interface to the unknown station 1c:4b:d6:69:cd:07. We should get 12,076 frames—the same as the number above:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0a) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == 1c:4b:d6:69:cd:07)' -T fields -e frame.time|wc -l
12076
```

That worked, so now let's use the same filter with the "head" and "tail" programs in order to find the time boundaries:

```
$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0a) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == 1c:4b:d6:69:cd:07)' -T fields -e frame.time|awk '{print $4}'|head -1
09:59:42.221489000

$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0a) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == 1c:4b:d6:69:cd:07)' -T fields -e frame.time|awk '{print $4}'|tail -1
10:00:47.611120000
```

It appears that Joe's WAP told 1c:4b:d6:69:cd:07 to Disassociate 12,076 times in the 65 seconds that passed from 09:59:42 to 10:00:47. That seems unusual.

While we're at it, let's look at the timing of the subtype 12 frames that the WAP was broadcasting (that second-most common type of management frame seen):

```
$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0c) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == ff:ff:ff:ff:ff:ff)' -T fields -e frame.time|wc -l
2454

$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0c) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == ff:ff:ff:ff:ff:ff)' -T fields -e frame.time|awk '{print $4}'|head -1
09:59:03.241923000
```

```
$ tshark -r wlan.pcap -R '(wlan.fc.type_subtype == 0x0c) && (wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 00:23:69:61:00:d0) && (wlan.da == ff:ff:ff:ff:ff:ff)' -T fields -e frame.time|awk '{print $4}'|tail -1
10:00:57.672520000
```

Joe's WAP broadcast 2,454 Deauthentication messages during roughly the same time period (09:59:03 to 10:00:57). That also seems unusual.

6.7.3.2 Partial Explanation

Recall that the 802.11 specification does not include a mechanism for verifying the authenticity of a sender. As a result, management frames can be spoofed. This leaves WLANs vulnerable to trivial denial-of-service attacks. Attackers can broadcast Disassociation or Deauthentication frames in order to cause network-wide outages or knock specific stations off the network. It is entirely possible that some of the Disassociation and Deauthentication messages shown above were actually sent by an attacker masquerading as Joe's WAP, and not the WAP itself.

Could the traffic we've seen have caused problems for Joe? Of course. Joe's station likely interpreted the Deauthentication messages as valid communications from the WAP, which would cause his station to attempt a new authentication/association negotiation. *It is also possible that another station in physical range could have spoofed some or all of these messages.*

Let's investigate further.

6.7.4 A Possible Bad Actor

Based on what we've seen so far, let us hypothesize that 1c:4b:d6:69:cd:07 is a suspicious actor (and for the moment our most obvious one, based on voluminous aberrant behavior).

Let's catalog the activities of interest. We should make sure we record *all* the activities we see coming from this station and update our timeline.

The command below produces a count of frames sent by 1c:4b:d6:69:cd:07 sorted and counted by type and subtype. In the output below, the first column is the number of matching frames, the second column is the 802.11 frame type, and the third column is the frame subtype.

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07)' -T fields -e wlan.fc.type -e wlan.fc.subtype|sort -n|uniq -c|sort -nr
42816 2 0
    77 0 11
    69 0 0
```

As you can see, there were 42,816 data frames from 1c:4b:d6:69:cd:07, 77 Authentication Requests, and 69 Association Requests. Let's sort by type/subtype to help us build our timeline further:

- The following command produces the timestamp for the first Association Request (type 0 subtype 0) seen from 1c:4b:d6:69:cd:07:

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07) && (wlan.fc.type_subtype == 0x00)' -T fields -e frame.time|head -1
Sep 17, 2010 09:58:51.776452000
```

- The following command produces the timestamp for the last Association Request (type 0 subtype 0) seen from 1c:4b:d6:69:cd:07:

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07) && (wlan.fc.type_subtype == 0x00)' -T fields -e frame.time|tail -1
Sep 17, 2010 10:00:47.612104000
```

- The following command produces the timestamp for the first Authentication Request (type 0 subtype 11) seen from 1c:4b:d6:69:cd:07:

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07) && (wlan.fc.type_subtype == 0x0b)' -T fields -e frame.time|head -1
Sep 17, 2010 09:58:51.773386000
```

- The following command produces the timestamp for the last Authentication Request (type 0 subtype 11) seen from 1c:4b:d6:69:cd:07:

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa == 1c:4b:d6:69:cd:07) && (wlan.fc.type_subtype == 0x0b)' -T fields -e frame.time|tail -1
Sep 17, 2010 10:00:47.606473000
```

6.7.5 Timeline

Based on the investigation so far, we can now cobble together a timeline of the events of interest to us on September 17, 2010 (times are rounded to the nearest second):

- **09:56:41**—Packet capture begins
- **09:58:52**—Station “1c:4b:d6:69:cd:07” begins sending both Authentication and Association Requests, essentially simultaneously, at a rate exceeding one per second
- **09:59:03**—The WAP appears to begin broadcasting a larger flood of Deauthentication messages
- **09:59:42**—The station “1c:4b:d6:69:cd:07” begins broadcasting an even larger flood of data frames
- **09:59:42 to 10:00:47**—The WAP appears to send 12,076 Disassociation messages to 1c:4b:d6:69:cd:07
- **10:00:47**—The station “1c:4b:d6:69:cd:07” stops sending Authentication and Association Requests
- **10:00:51**—The station “1c:4b:d6:69:cd:07” stops sending broadcasted data frames

- **10:00:58**—The WAP’s apparent Deauthentication broadcasts stop
- **10:02:14**—Station “de:ad:be:ef:13:37” sends first data frame to the WAP’s STA interface
- **10:03:33**—Station “de:ad:be:ef:13:37” sends last captured data frame
- **10:03:35**—Packet capture ends

6.7.6 Theory of the Case

Based on Joe’s categorical assertion, the unknown station with MAC address “1c:4b:d6:69:cd:07” is an interloper at best. Taking into consideration its activities relating to Joe’s WAP, it may well be malicious.

Previously in this chapter we discussed how an attacker might seek to force the generation of unique IVs in order to brute-force a WEP key. The evidence suggests that in this case, the station 1c:4b:d6:69:cd:07 may have been an attacker attempting to gain access to the WLAN. In this section, we further explore this theory of the case.

6.7.6.1 Stimulus and Response

Based on the evidence we’ve seen so far, we hypothesize that the abnormal volume of broadcasted data frames sent by 1c:4b:d6:69:cd:07 are ARP requests which have been replayed in order to generate additional unique IVs to facilitate a WEP-cracking attack. If this is the case, and the attack was successful, then the number of unique IVs generated by stations on the WLAN would have significantly increased during the time period that the attacker sent the broadcasted data frames.

Recall that the packet capture began at 09:56:41.085810 and ended at 10:03:34.662764 on September 17, 2010. The station 1c:4b:d6:69:cd:07 began sending broadcasted data frames 09:59:42.220425 and ending at 10:00:50.972590 on the same day.

Let’s check to see how many unique IVs were generated by other stations on the WLAN during the time period that 1c:4b:d6:69:cd:07 was generating large amounts of data frames, and compare this result to other time periods.

- Before the flood of data frames began (09:56:41.085810 to 09:59:42.220425), 694 unique IVs were generated in 181.134615 seconds, as shown below. This is an average of **3.831405** unique IVs per second.

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa != 1c:4b:d6:69:cd:07) && (frame.time < "Sep 17, 2010 09:59:42.220425000")' -T fields -e wlan.wep.iv | sort -u | wc -l
694
```

- During the flood of broadcasted data frames (09:59:42.220425 through 10:00:50.972590) 13,657 unique IVs were generated in 68.752165 seconds, as shown below. This is an average of **198.641017** unique IVs per second.

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa != 1c:4b:d6:69:cd:07) && (frame.time <= "Sep 17, 2010 10:00:50.972590000") && (frame.time >= "Sep 17, 2010 09:59:42.220425000")' -T fields -e wlan.wep.iv | sort -u | wc -l
13657
```

- After the flood of broadcasted data frames ended (10:00:50.972590 through 10:03:34.662764) 1,240 unique IVs were generated in 163.690174 seconds, as shown below. This is an average of **7.575287** unique IVs per second.

```
$ tshark -r wlan.pcap -R '(wlan.bssid == 00:23:69:61:00:d0) && (wlan.sa != 1c:4b:d6:69:cd:07) && (frame.time > "Sep 17, 2010 10:00:50.972590000")' -T fields -e wlan.wep.iv|sort -u|wc -l
```

1240

The frequency of unique IVs generated during the flood of data frames was over 26 times greater than the frequency of unique IVs generated beforehand—and over 51 times greater than the frequency of unique ICs generated afterwards. In other words, it is entirely plausible that the flood of data frames sent from 1c:4b:d6:69:cd:07 could be indicative of a successful WEP-cracking attack.

6.7.6.2 WEP Cracking Attack

Given what we know, a reasonable theory is that a station with the MAC address of 1c:4b:d6:69:cd:07 waged an attack on Joe's WAP and its associated stations beginning at 09:58:51 and lasting for a few minutes at most. If successful, it may have yielded sufficient key material for a “related key” attack on the WEP key.

We can test this theory, and will, in our answers below.

As a corollary, we could further theorize that the station with the MAC address of de:ad:be:ef:13:37 was able to authenticate and associate as a result of the previous activity. (Exactly *how* 1c:4b:d6:69:cd:07 and de:ad:be:ef:13:37 are related is a matter of interest, but *that* they are related is *strongly implied* by the timeline.) At a minimum, we can attempt to inspect de:ad:be:ef:13:37's traffic to see if it is malicious.

6.7.7 Response to Challenge Questions

Now let's answer the questions posed at the beginning of this case:

- **What are the BSSID and SSID of the WAP of interest?**

We were able to identify and verify these values from the traffic capture provided to us:

- BSSID: 00:23:69:61:00:d0
- SSID: Ment0rNet

- **Is the WAP of interest using encryption?**

Yes, every single data frame inspected was encrypted.

- **What stations are interacting with the WAP and/or other stations on the WLAN?** There were three stations that we could easily detect based on an analysis of management frame activity:

- 00:11:22:33:44:55
- 1c:4b:d6:69:cd:07
- de:ad:be:ef:13:37

- **Are there patterns of activity that seem anomalous?** There were several, including a flood of data frames coming from an unknown station that coincided with a spike in unique IVs being sent out from the WAP. Though it didn't last for very long, the unusual activity was followed by yet *another* unknown station which appeared to associate successfully.
- **How are they anomalous: Consistent with malfunction? Consistent with maliciousness?** The anomalous behavior is consistent with a WEP-cracking attack. One station emitted a disproportionate number of data frames destined for the Layer 2 broadcast address. These data frames apparently triggered a response from the WAP (and other stations as well), which resulted in the WAP generating a large number of frames with unique IVs. The proportion of data frames broadcasted one station was unusually high. However, it is a common symptom of a WEP-cracking attack because in order to crack the WEP key, an attacker needs to collect a large number of data frames with unique IVs.
- **Can we identify any potentially bad actors?** Station 1c:4b:d6:69:cd:07 emitted a large number of data frames sent to the Layer 2 broadcast address, which caused the WAP to respond with a large number of frames with unique IVs. Based on this, it is likely that 1c:4b:d6:69:cd:07 was an attacker attempting to launch a WEP-cracking attack. Shortly after the apparent WEP-cracking attack ended, the unknown station de:ad:be:ef:13:37 successfully associated with the WAP. It is possible that this system's activities are associated with the attack.
- **Can we determine if a bad actor successfully executed an attack?** Let's investigate whether the WEP-cracking attack could have been successful. One way to figure that out would be to simply use aircrack-ng to try to recover the WEP key from the packet capture. If enough key material was revealed during that packet capture, we should be able to crack the key. Let's find out:

```
$ aircrack-ng -b 00:23:69:61:00:d0 wlan.pcap
          Aircrack-ng 1.0

[00:00:02] Tested 938 keys (got 26805 IVs)

KB      depth    byte(vote)
 0      3/  4    D0(33536) 1F(33024) 27(33024) BC(33024) 2F(31744) 7B
        (31744)
 1      0/  1    E5(38656) 82(33024) 0C(32256) 3C(32000) EB(31744)
        42(31488)
 2      0/  6    9E(34048) 27(33792) 7A(32768) E9(32512) 8B(31744) 0E
        (31744)
 3      0/  4    B9(35328) D4(35072) 2E(34048) B9(33024) 00(32768)
        06(32512)
 4      8/ 10    6D(31488) 10(31232) B9(31232) 7A(30976) 95(30976) A5
        (30976)

          KEY FOUND! [ D0:E5:9E:B9:04 ]
Decrypted correctly: 100%
```

We were able to successfully recover the WEP key from the captured packets, which tells us that yes, it is certainly possible that the attacker’s WEP-cracking attack was successful. If we recovered the WEP key from these packets, then the attacker could have, too.

Recall that shortly after the apparent attack ended, the unknown station “de:ad:be:ef:13:37” successfully authenticated to the network, and there was additional WEP-encrypted communication between “de:ad:be:ef:13:37” and Joe’s WAP. It is entirely possible that after recovering the WEP key with station 1c:4b:d6:69:cd:07, the attacker used the key to authenticate with station “de:ad:be:ef:13:37.” Now that we have recovered the WEP key, we can decrypt the Layer 2 traffic and investigate further using higher-layer protocol analysis techniques.

6.7.8 Next Steps

At this point, we should probably verify with Joe that the 40-bit key we were able to expose (D0:E5:9E:B9:04) is the WEP key that he had set for his WAP. Then again a more certain confirmation can be had if the recovered WEP key decrypts the packets.

- **Decrypting the data frames.** Assuming all due permission and authority to proceed, the obvious next step would be to “decapsulate” the WEP-encrypted frames. The “airdecap-ng” tool performs this function nicely. The command below produces a file named “wlan-dec.pcap” with the decrypted WEP frames.

```
$ airdecap-ng -l -b 00:23:69:61:00:d0 -w D0:E5:9E:B9:04 wlan.pcap
Total number of packets read          133068
Total number of WEP data packets      56692
Total number of WPA data packets      0
Number of plaintext data packets     0
Number of decrypted WEP packets      56692
Number of corrupted WEP packets      0
Number of decrypted WPA packets      0
```

- **Inspecting the decrypted content.** It seems prudent to proceed with the assumption that some unknown station had forcibly gained access to Joe’s WLAN. Every indication seems to support this interpretation of events—including our ability to replicate an attacker’s likely successes so far. If we could crack the WEP key and decrypt the network traffic, anyone else with access to the RF waves could have done the same.

At this point we can return to our usual tools and techniques for network traffic and communications analysis—but not leaving aside what we’ve already learned and documented in our timeline. In addition, if we were to look further, we would discover that immediately after the traffic spike caused by 1c:4b:d6:69:cd:07, and after all of the Deauthentication frames were broadcast (perhaps spoofed), the station “de:ad:be:ef:13:37” appeared, associated, and authenticated for the first time. Once it authenticated, it immediately began talking to the WAP on port 80/TCP. According to Joe, this station is not an authorized network user, and so this is clearly suspicious traffic.

Reconstructing some of the last few decrypted streams reveals that “de:ad:be:ef:13:37” authenticated to the WAP’s administrative web interface.

```

Follow TCP Stream
Stream Content
POST /Security.tri HTTP/1.1
Host: 192.168.1.1
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15) Gecko/20091028
Firefox/3.0.15
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://192.168.1.1/WSecurity.htm
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: application/x-www-form-urlencoded
Content-Length: 78

SecurityMode=3&CipherType=1&PassPhrase=hahp0wnedJ00&GkuInterval=3600&layout=en<
<head>
<meta http-equiv="expires" content="0">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="pragma" content="no-cache">

<meta http-equiv=Content-Type content="text/html; charset=iso-8859-1">
<SCRIPT language="javascript" type="text/javascript" src="share.js"></SCRIPT>

```

Figure 6–19. A screenshot of Wireshark showing a reconstructed stream between “de:ad:be:ef:13:37” (192.168.1.109) and the WAP’s STA interface “00:23:69:61:00:ce” (192.168.1.1). Notice the HTTP “Authorization” header, which indicates that HTTP Basic Access Authentication was used.

Figure 6–19 shows a reconstructed stream between “de:ad:be:ef:13:37” (192.168.1.109) and the WAP’s STA interface “00:23:69:61:00:ce” (192.168.1.1). Notice the HTTP “Authorization” header:

```
Authorization: Basic YWRtaW46YWRtaW4=
```

This indicates that the client “de:ad:be:ef:13:37” attempted to authenticate to the WAP’s administrative web interface using HTTP Basic Access Authentication (the credentials are Base64-encoded in transit, not encrypted). The shell command below decodes the Base64-encoded credentials, as shown:

```
$ echo "YWRtaW46YWRtaW4=" | base64 -d
admin:admin
```

It appears from Figure 6–19 that the WAP was configured with a weak or default password of “admin,” and that we have a “volunteer” system administrator who is about to help us fix something. On further inspection it becomes clear that something adverse to Joe was done.

Exactly what, and how, is left as an exercise for the reader.

Chapter 7

Network Intrusion Detection and Analysis

“IDS is dead.”
—Gartner, 2003

It may seem, based on the title of this chapter, that we’re somewhat behind the times. After all, Gartner famously pronounced intrusion detection dead many years ago,¹ asserting in 2003 that intrusion detection systems (IDSs) would be obsolete by 2005 and that everyone would be better off putting their money into preventative technologies (i.e., firewalls). Subsequently, most vendors followed suit, rebranding all of their detection solutions as “intrusion prevention systems (IPSs).” This wasn’t all that difficult to do, as many already included automated remediative actions as configurable options. It wasn’t a magical new technology so much as a marketing strategy for selling a “new” product to existing consumers to replace the “old” in a way that generated revenues that mere feature upgrades couldn’t.

Meanwhile, on the ground, we’ve been facing the same issues as always, regardless of the acronym. In a very dynamic and fluid environment (say, the Internet) it is sometimes difficult to detect adverse and malicious events, particularly when the authors of such events seek not to have them noticed.

In some cases we can construct sophisticated pattern- or behavioral-matching algorithms that are exceedingly unlikely to miss the bad actor, rarely producing “false positives” (alerting when nothing bad actually happened) or “false negatives” (failing to alert when something bad *really did* happen)—the worst kind of failure. In such cases automated remediation is a really good idea. However, automated remediation in response to false positives typically results in a self-inflicted denial-of-service condition. The result is typically that the IPS gets “tuned down” a bit, until it is simply less capable of alerting on events that *might* be of interest.

As investigators, we have to be prepared to evaluate events that *might* be of interest, and so we often prefer to have the *most* data available, even that which may not be deemed automatically actionable by an IPS. As a result, we must sift events of interest from the noise.

1. Allison Haines, “Gartner Information Security Hype Cycle Declares Intrusion Detection Systems a Market Failure: Money Slated for Intrusion Detection Should Be Invested in Firewalls,” June 11, 2003, http://www.gartner.com/press_releases/pr11june2003c.html.

TMA (Too Many Acronyms)

Over the years, the IDS/IPS product space has developed two separate niches: network intrusion detection/prevention systems (NIDS/NIPS) and host instruction detection/prevention systems (HIDS/HIPS). The difference is as follows:

- NIDS/NIPS monitor network traffic and alert on suspicious network events;
- HIDS/HIPS monitor system events and alert on suspicious system activities.

In this book, we focus on forensic analysis of NIDS/NIPS.

7.1 Why Investigate NIDS/NIPS?

In an environment that has been appropriately instrumented to detect potentially adverse events via network monitoring, it's very likely that a NIDS/NIPS alert is the starting point of our investigation. After all, that's what such deployments are designed to do: tell us when there's something going on that we ought to look into. Unfortunately NIDS/NIPS can't always reconstruct a sequence of events and explain them to us, at least not easily. When present, though, they can be as important an event recorder as the "little black box" aviation authorities search for and analyze after every crash, for similar reasons. NIDS/NIPS are often involved in network forensics investigations for the following reasons:

- NIDS/NIPS alerts/logs may include details regarding illicit connections—or even attempts—that are not recorded anywhere else. NIDS/NIPS tend to inspect traffic in ways that even application-layer firewalls may not, and so can provide a richer source of information.
- NIDS/NIPS can be configured to alert on, or at least log traffic that firewalls deem perfectly acceptable. For example, a firewall may not record a port-80 connection to the web server, but the NIDS/NIPS may note if it contains a malicious URI.
- An investigator could potentially modify a NIDS/NIPS configuration to begin detecting events it wasn't previously configured to record. These devices are usually well positioned as inspection points for network traffic.
- Rarely, the NIDS/NIPS itself might be suspected of compromise.

7.2 Typical NIDS/NIPS Functionality

NIDS/NIPS are specialized sniffers, with the added capability of evaluating captured traffic to determine whether it is malicious or legitimate. There are many different commercial NIDS/NIPS systems on the market. They all behave differently but provide many of the same features. NIDS/NIPS typically include:

- **Rules** Descriptions of how to compare a packet or stream with known malicious traffic.
- **Alerts** Lists of suspicious packets/streams.
- **Packet captures** Certain NIDS/NIPS can be configured to capture suspicious packets and save them for later analysis. They are not always configured to do this by default, due to storage constraints.

7.2.1 Sniffing

At a minimum, a network-based intrusion detection system must have access to network traffic to collect and inspect. This can be done passively, either by connecting it to a mirroring port on a switch or through the use of an inline tap. This can also be done by putting the NIDS/NIPS itself inline between two devices in a choke point position—typically between a set of firewalls or routers.

Once the traffic has been collected, several approaches to its inspection can be leveraged, all of which take some amount of time. If the device is sniffing traffic passively—meaning that the traffic is not perturbed or detained but merely copied on its way by—this delay is of little to no consequence. The delay caused by the inspection is likely to be minuscule compared to the time lag between an alert being sent and a human responder beginning an investigation.

However, if the device is inline in the traffic's path (typical for a NIPS), and is actively filtering the traffic as it inspects it, the delay can cause noticeable latency. Consequently, many NIPS are not configured to perform as much analysis as a passive NIDS because the latency introduced by the analysis might interfere with business operations.

This is important for the network investigator to understand. While it might be advantageous to *begin* sniffing and analyzing traffic during the course of an investigation, particularly in ways that weren't being done previously, modifying an existing NIPS might not be feasible without disrupting the business. In such situations it might be much better to temporarily install a separate passive NIDS.

7.2.2 Higher-Layer Protocol Awareness

In the arms race between attackers and network defenders, attackers began to break malicious traffic into multiple pieces, or encode traffic in sneaky ways, in order to evade detection. In response, NIDS and NIPS developers incorporated higher-layer protocol analysis into their systems. Protocol reassembly and normalization of content based on protocol awareness are two ways in which NIDS and NIPS have evolved to protect against attacks. These techniques are costly in terms of processing power, but effective.

7.2.2.1 Protocol Reassembly

In order for either a NIDS or a NIPS to accurately detect a wide range of malicious traffic, these devices must be programmed to properly parse a wide range of protocols. For example, many years ago a common technique that attackers would use to evade detection by a NIDS was to intentionally fragment the attack traffic. The assumption was that only the end recipient of the traffic—presumably the target—would reassemble the fragments. Consequently we learned that we had to build our NIDS to perform fragment reassembly in order to be

able to see and evaluate what the target would receive and process. Likewise it is common for the telltale signs of malicious activity to span multiple segments in a TCP stream. In order for a NIDS/NIPS to accurately inspect such traffic, it must be able to understand TCP’s statefulness and to perform stream reassembly prior to doing content inspection.

It is this actual content inspection that lends the NIDS/NIPS its power. Most simple firewalls can easily be programmed to allow only port 80 traffic to a web server, but if the “legitimate” port 80 connection contains a malicious request, the firewall has no way of knowing. Because the NIDS/NIPS understands the HTTP protocol, it can look inside the request and make sure that it is formatted properly, and alert if it contains something it shouldn’t.

7.2.2.2 Normalization

For certain types of traffic, most notably HTTP, the protocol allows for many different ways of encoding data. For example, the following GET requests are functionally identical, and would be handled by a web server in the same manner (hopefully not allowed!):²

```
GET ../../../../../../etc/passwd HTTP/1.1
GET %2f%2f%2f%2fetc%2fpasswd HTTP/1.1
GET %2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2f%65%74%63%2f%70%61%73%73%77%64
HTTP/1.1
```

This is an example of a “directory traversal” attack, whereby if the webserver is buggy, it can be convinced to traverse up and out of its document hierarchy, and then display the system /etc/passwd file. A NIDS/NIPS that understands the GET method of the HTTP protocol should have no problem identifying such an attempt. However, given that there are a nearly infinite number of ways to encode any particular malicious string, we have two options: either write a “signature” to detect *each and every one* of the various ways such a string can be represented, or first “normalize” each string to its “canonical” form, and only then compare them against the list of “known bad” things.

7.2.3 Alerting on Suspicious Bits

The different ways that NIDS/NIPS are typically built to perform detection are described below. All NIDS/NIPS can be configured to communicate alerts via various means. The most common of these include:

- Sending email alerts
- Logging events to a syslog server
- Sending SNMP traps
- Logging events directly to a queriable database

Additionally, a NIDS/NIPS sensor may store alert and event data locally, although as with other network devices, storage space is likely to be fairly limited, and so these logs will likely be kept for only short periods of time.

2. Tim Berners-Lee, “RFC 3986—Uniform Resource Identifier (URI): Generic Syntax,” IETF, January 2005, <http://rfc-editor.org/rfc/rfc3986.txt>.

7.2.3.1 Fidelity

The manner in which the alerts are recorded and delivered can matter greatly when considering the value of the data/evidence produced. Alerts logged via syslog or SNMP traps are normally fairly “lo-fidelity” in that very little detail about the event is available. Database logging (usually as part of some sort of proprietary event aggregation system) can often produce much more detail about how and why the event was detected and determined to be of interest.

At the other end of the spectrum, some NIDS/NIPS can produce very “hi-fidelity” event logs, capturing and storing the packets in their entirety (usually in libpcap format). This can allow an analyst or investigator the ability to see the actual traffic that caused the alerts, and perhaps best understand why it was deemed “bad” by the NIDS/NIPS.

7.3 Modes of Detection

NIDS/NIPS typically work in three basic ways: using signature-based analysis, protocol awareness, and/or behavioral analysis techniques.

7.3.1 Signature-Based Analysis

Signature-based analysis is the oldest and most common strategy. This method compares headers, contents of packets, and streams of packets against databases of known, malicious byte sequences in order to identify suspicious traffic. Individual packet analysis does not always work because malicious activity may be spread across groups of packets. Hence, signature-based NIDS/NIPS have evolved to inspect related traffic that spans multiple fragments of packets, or multiple segments in a stream.

7.3.2 Protocol Awareness

Normal traffic typically conforms to RFC specifications. Malicious traffic often does not. Attackers often intentionally bend protocols in order to evade detection. Just as the arms race exists between attacker and defender, it also exists between attacker and detector. Protocol-aware NIDS/NIPS reassemble fragments (Layer 3), reassemble streams (Layer 4), and even reconstruct entire protocols (Layer 7) because they have to understand how the endpoint of the communication will interpret the data. It’s possible for protocols to be broken in a nonmalicious way by a misconfigured or malfunctioning device. However, protocol abuse is often a symptom of malicious intent. Either way, it is worth detecting.

7.3.3 Behavioral Analysis

Behavioral analysis is a more recent development in NIDS. The idea behind this approach is to spend time building a “baseline” of normal network behavior against which to compare future behavior. The system “learns” what is normal, in order to later detect aberrations. Alerts may be triggered, for example, if the volume of data per transaction going out of your web server is two standard deviations greater than the expected mean. Though there is no indication of protocol violations, or that the content of the web transactions matches any known bad behavior, this could nonetheless be a very important event to be made aware of.

7.4 Types of NIDS/NIPSS

NIDS/NIPS can be sold as a commercial product or built in-house. Commercial NIDS/NIPS are usually made from special hardware designed to improve performance. “Roll-your-own” NIDS are often built from open-source software such as Snort, which gives users the option to download free signature updates, or purchase a subscription that provides more timely access to updates.

7.4.1 Commercial

Regardless of what Gartner may have predicted in 2003, the commercial market for IDS is very much alive and well, with many of the top IT vendors having made significant acquisitions to cover their positions relative to competitors. One of the early leaders in detective technologies was Marcus Ranum’s “Network Flight Recorder” (NFR), which was bought by Check Point (after a failed attempt to purchase Marty Roesch’s “Snort”) and which presumably has been adopted as the core technology behind Check Point’s IPS-1 product. Notably, NFR was one of the few NIDS that offered an open language for specifying packet and traffic signatures, allowing NFR Security’s customers to build their own custom rules (though the open-source Snort was evolving a simpler and more widely adopted language around the same time).

Other long-time players in this market have evolved their product lines as well. Cisco’s “Secure IDS” has morphed into “Cisco IPS” and Enterasys’ Dragon is now “Enterasys IPS.” IBM purchased Internet Security Systems (ISS), a market leader in both IDS and network vulnerability scanning technologies. ISS’s RealSecure sensor has morphed into the “IBM Security NIPS,” which is now offered as a component of IBM’s Tivoli suite of enterprise management tools. Meanwhile, TippingPoint’s popular product was subsumed by Hewlett-Packard.

Sourcefire continues to stand alone as the commercial vendor/licenser of Snort, which in its various for-profit and open-source forms undoubtedly leads the market as the single most widely deployed NIDS/NIPS technology on the planet. Consequently, we’ll look at its architecture in depth below.

As of the time of this writing, some common commercial products network forensic investigators are likely to encounter include (listed alphabetically):

- Check Point IPS-1³
- Cisco IPS⁴
- Corero Network Security⁵
- Enterasys IPS⁶

3. “Check Point IPS-1—intrusion detection & prevention system (IDS/IPS),” 2011, <http://www.checkpoint.com/products/ips-1/index.html>.

4. “Cisco Intrusion Prevention System—Products & Services—Cisco Systems,” 2011, <http://www.cisco.com/en/US/products/sw/secursw/ps2113/index.html>.

5. “Intrusion Prevention System (IPS)—Stop DoS, DDoS, SYN Flood, and Cyber Attacks,” 2011, http://www.corero.com/content/products/intrusion_detection/attack_mitigator.jsp.

6. “Intrusion Prevention System, Secure Networks—Enterasys,” 2011, <http://www.enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx>.

- HP TippingPoint IPS⁷
- IBM Security NIPS⁸
- Sourcefire 3D System⁹

7.4.2 Roll-Your-Own

It is clear why Snort is the most deployed NIDS/NIPS technology: it is both free for use under the GNU Public License (GPL), and at the same time supported by commercially funded research and development efforts. Consequently you may encounter Snort used “unofficially” even within corporate and government environments that have opted to purchase another commercial vendor’s offering as an “official” solution.

As with any other IT product niche, commercial support, development, and ease of integration can provide compelling reasons to write checks. However, as an investigator, you should be aware that you may find custom, “roll-your-own” solutions sitting side-by-side with a commercial deployment. These devices are often more flexible in their use (being “unofficial” anyhow), and can more easily be refactored or repurposed during the course of an investigation—and yet the evidence they collect can be no less valid and useful.

For a long time, Snort was the only real open-source NIDS/NIPS capable of industrial-scale use, though it appears that with some National Science Foundation funding and an increasingly energetic following, the Bro system is gaining both in competitiveness and adoption.

Again, here are the most common noncommercial systems you should expect to encounter, with references to more information about each. Furthermore, these are systems that you can and should be prepared to work with during an investigation. Pretty much everything you need to know about how to do so is available in online manuals, and both systems have robust communities surrounding them:

- Snort¹⁰
- Bro¹¹

Though both are open-source and free for use, they are really not equivalent projects in either scope or goal. Due to its commercial ownership and support, Snort is much more targeted toward enterprise, production deployment. Meanwhile Bro’s development team states overtly that it “has been developed primarily as a research platform for intrusion detection and traffic analysis.” These are very complementary goals, and by all appearances, are very complementary projects where either research or raw investigation is the goal.

7. “HP Networking products: switches, wireless, routers, TippingPoint security, network management, uc&c,” 2011, http://h10163.www1.hp.com/products_ips.html.

8. “IBM—IBM Security Network Intrusion Prevention System—Software,” 2011, <http://www-01.ibm.com/software/tivoli/products/security-network-intrusion-prevention/>.

9. “Next-Generation Intrusion Prevention System (NGIPS) | Sourcefire Cybersecurity,” 2011, <http://www.sourcefire.com/security-technologies/cyber-security-products/3d-system>.

10. “Snort,” 2011, <http://www.snort.org/>.

11. “The Bro Network Security Monitor,” 2011, <http://bro-ids.org/>.

7.5 NIDS/NIPS Evidence Acquisition

Given the wide variation in hardware and software used to build NIDS/NIPS, the precise evidence that forensic investigators can gather, and the methods for gathering it, can vary considerably. In this section, we review the general categories of evidence that can be gathered from NIDS/NIPS and discuss the value to investigators and common interfaces for devices.

7.5.1 Types of Evidence

Depending on the vendor, alerts, rules, and packet captures may be accessed via local devices' file systems, web management interfaces, client software, email, etc. In general, the evidence that we may be able to gather from NIDS/NIPS includes:

- Configuration
- Alert data
- Packet header and/or flow record information
- Packet payloads
- Activities correlated across multiple sensors

7.5.1.1 Configuration

The configuration of the NIDS/NIPS is fundamentally important for any analysis of alerts from the device. For example, suppose your investigative environment has a number of IDS sensors deployed throughout the environment. A very critical alert resulted in escalation and investigation. The alert in question was only seen in one critical segment. Is that because the events that triggered it only occurred in that segment, or because sensors in other segments weren't configured to look for those events? How would your strategy and the outcome differ depending on that answer? Without evaluating the relative configurations of the sensors in the various segments, you have no way of knowing what the possible scope of events may be.

If a NIDS/NIPS has not been configured to alert upon a particular event, then the absence of alerts for that event is meaningless. Similarly, it is not possible to determine the meaning of an alert with certainty, or evaluate whether it is a false positive or negative, unless you can see precisely what rule triggered the alert. What you do or do not see coming from an IDS system is wholly dependent upon where the device is placed and *what it has been configured to alert upon*. For this reason, we recommend that you gather device configuration as part of NIDS/NIPS evidence acquisition.

In addition to gathering the boot configuration of the device (usually found in persistent storage on the device), it's important to gather the current running configuration whenever possible. Depending on the NIDS/NIPS software, an administrator may be able to modify the configuration while the system is running, without writing these changes to disk. In these instances, the running configuration may be quite volatile, and unless it is manually written to persistent storage, it will be lost when the device is powered down.

7.5.1.2 Alert Data

The purpose of a NIDS is to evaluate network traffic based on a set of rules or learned patterns, identify traffic of interest, and produce alerts that humans can act upon. NIDS/NIPS produce alerts in various formats, ranging from text files to emails to messages in pretty graphical user interfaces.

Different NIDS/NIPS systems categorize, label, and prioritize alert data differently. Even when alerts are aggregated from multiple sources, it can be a real challenge to discover that the manner in which a NIDS/NIPS reports an event on one segment of a network is not at all the same way an older model on a different segment reports it. There are Security Information and Event Management (SIEM) systems that seek to “normalize” event data from disparate sources, but this is often a problem that ends up being solved by humans, hunkered over lots of events, trying to puzzle them out.

7.5.1.3 Packet Header and/or Flow Record Information

When a packet or flow triggers a NIDS/NIPS alert, the device may log corresponding packet header and/or flow record data. This can help you identify the origin, destination, and patterns of activity. It can also help you correlate events from multiple sources.

7.5.1.4 Content Data

In some instances, the NIDS/NIPS may be configured to capture full packet contents from packets that trigger alerts, and often a certain number of subsequent packets. Packet contents can be invaluable to the investigator, allowing you to carve out the malware that triggered an alert or view the actual obfuscated Javascript that an attacker attempted to serve a client.

Regardless of how hard it can be to sort through volumes of data to figure out what's going on, it is usually *infinitely* more difficult to figure out the nature of events given *too little* information about them. Full content captures are limited by the amount of disk space available on the NIDS/NIPS, in addition to processing power.

7.5.1.5 Activities Correlated Across Multiple Sensors

Topologically and temporally correlated event data is exceptionally useful. Unfortunately, it is not always available. Relatively few organizations are set up to properly aggregate security event data, and for good reason: The products that can correlate NIDS/NIPS data effectively are often prohibitively expensive, and require even more expensive staff and skills to monitor and maintain. This is challenging for small- and mid-sized operations. Even large-scale enterprises don't always go all in with such technologies.

An increasingly common practice is for enterprises to deploy NIDS/NIPS sensor instrumentation, but then to direct all of its output to a third-party monitoring service. As an investigator, you'll need to determine very quickly if this is being done, as you'll almost certainly want to work with the outsourced service provider to gain access to a comprehensive view of correlated events. After all, this is what they're being paid to provide, and it sure beats repeating their work. Even if you are working with an outsourced provider, be aware that NIDS/NIPS data can be either inaccurate or incomplete, or both. Corroboration is important no matter where your evidence comes from.

7.5.2 NIDS/NIPS Interfaces

These days, NIDS/NIPS are almost uniformly designed to be information aggregation devices, so they are typically deployed with a central analysis console. This is certainly convenient for the forensic investigator; it's far better to collect evidence from one central source than from multiple disparate devices, perhaps in far-flung locations.

A standalone NIDS/NIPS sensor may be used without aggregation, as is common with point-deployments of Snort and other low-cost systems. In this situation, you may find a great deal of data without a convenient central analysis tool. The biggest challenge will be to manually aggregate and correlate the data.

Depending on the particular NIDS/NIPS, evidence can be gathered via a graphical user interface (GUI), command-line interface (CLI), or a remote log server. Please see Chapter 3 for a detailed discussion of active evidence acquisition.

7.5.2.1 GUI Interfaces

Nearly all modern NIDS/NIPS systems can be accessed, inspected, and configured via some sort of graphical user interface, whether it's a web application that can be accessed by a web client, or a proprietary interface that requires a special client. The degree of inspection and configuration supported by such interfaces varies, of course, but some vendors strive to expose every last bit of functionality this way, from signature configuration to alerts generated. Sometimes you can use the GUI to download the actual packet traces.

7.5.2.2 CLI Interfaces

It may be possible to log onto the NIDS/NIPS via SSH or direct console connection and use the command-line interface to view configuration, logs, and alerts. Some devices have historically provided constrained access to a subset of functionality via the command-line, while others expose entire application programming interfaces (APIs) for nearly infinitely extensible customizations.

As an example, the NIPS sensors that Sourcefire sells commercially run the same version of Snort that can be downloaded for free, though they provide an array of configuration and management utilities in the commercial devices that make enterprise deployments much simpler and more manageable. Many of these utilities leverage a library of custom Perl objects that can be used by anyone with the appropriate access credentials and the skills to employ them (check licensing terms and support contracts before doing so).

7.5.2.3 Off-System Logging

These days, both commercial offerings and open-source systems alike are capable of sending at least *some* of the critical evidence to an aggregation repository. Many NIDS/NIPS are also configured to send alerts via email to system administrators. You may need to inspect the NIDS/NIPS configuration in order to figure out where the system is sending logs (passive analysis of the traffic from the device may also do the trick). The local device configuration will also provide you with an understanding of the level of detail and fidelity of the remote logging.

7.6 Comprehensive Packet Logging

We've discussed various ways that packet activity can be logged: routers and firewalls can log the packet event with source and destination information; NIDS/NIPSs can record the total packet content from metadata (header info) down to the payload contents.

However, where full content packet logging is occurring on NIDS/NIPS, almost uniformly the only packets being captured and made available for future inspection are the ones that trigger actual alerts. NIDS/NIPS generally are not configured (or configurable) to capture the packets that preceded an incident, nor the ones that followed. That can pose a serious problem for the investigator: often very little can be gleaned from a single packet, as too much of the context is unavailable.

One way of countering this problem is to intentionally construct a persistent packet sniffing device. This is a sniffer configured to record every one and zero that traverses the link it monitors. Its job is not to alert on any specific condition, but rather to ensure that it has recorded all of the data available, along with critically accurate timestamp information.

The benefits of such a device are instantly obvious. Whereas NIDS/NIPS rarely capture much more than a single packet per event (the one that tripped the trigger), that single packet can subsequently be located in a comprehensive packet log, and its context examined. The stream that contained it, if it was part of a stateful session, can be isolated and reconstructed. If it was part of a stimulus, the response may be located and investigated. If it was part of a response, perhaps the stimulus can be determined.

All of the previous caveats with respect to packet captures still apply, however:

- It can require a lot of CPU.
- It can require a lot of disk.
- It can also pose a huge security risk!

7.6.0.4 Evidence that May Be Available

The possession of full-content packet captures, that contain packets which correlate with NIDS/NIPS alerts, is pretty much the dream scenario. There is virtually nothing left wanting—so long as time skew and time zone can be dealt with. As to the former, many tools can be programmatically configured to adjust for skews—either on input, or on display, or both—and to aggregate based on an absolute time (such as GMT-0). We discuss the latter issue in Chapter 8. To summarize the bounty:

- Packet headers
 - Suitable for traffic analysis
 - Similar to pen registers, trap/trace
- Packet payloads
 - Suitable for full transaction reconstruction
 - Similar to full wiretap

When such packet captures are available, they can be used in conjunction with other event data (such as NIDS logs) in order to reconstruct entire events. If full payloads are stored, entire sessions can be reconstructed, and files recovered intact. Perfect-fidelity copies of evidence can be acquired in this way, every bit as successfully as from the hard drive to which they were written, or from which they were transferred.

Since packet capturing is usually performed with tcpdump or another libpcap-based tool retrieval and analysis of the logs is usually performed with libpcap-based tool as well. The Berkeley Packet Filter language becomes critical for such analysis, as finding the critical traffic can pose an overwhelming task. See Chapter 3 for details.

7.7 Snort

As discussed, the open-source Snort product is arguably the single most widely distributed NIDS in the world. It is sometimes used alongside commercial products. In this section we focus on Snort in particular depth because of its large market share.

Snort is a libpcap-based utility, and therefore shares all of the functionality that tcpdump and others possess. However, rather than being limited to Layer 2 through Layer 4 analysis, Snort has the capability to do much deeper packet inspection. It can analyze arbitrary network traffic using two basic techniques: protocol analysis and signature analysis.

Snort's rule language is open source, so we can easily discuss how rules are written to find individual packets. Anyone who has used tcpdump on a busy network interface is quickly overwhelmed by the fact that there is so much traffic it's difficult to find what you're looking for. Snort is a tool that illustrates our ability to find the needle in the haystack.

As our friend Rob Lee says, “If you know *what* you're looking for and *where* to look, nothing is hidden.”

An overview of Snort:

- Most widely used NIDS
- Open-source code
- Open rule language
- Extremely versatile
- Actively improving, partly due to commercial support

7.7.1 Basic Architecture

Snort's basic architecture is as follows:

- Snort pulls packets in using libpcap, from whichever interface is specified, either at the command-line or in the configuration file.
- Snort passes all packets through preprocessors for reassembly and protocol analysis. These preprocessors roughly mirror the OSI layers:
 - At Layer 3, it reassembles fragments
 - At Layer 4, it reassembles streams

- At Layer 5, it reassembles circuits/sessions
 - At Layer 7, it reassembles transactions
- If at any of those layers, Snort detects anomalies, it can alert. The degree to which Snort does protocol analysis and reconstruction is configurable.
 - After all of the preprocessing has been completed, the analyzed information is handed off to the Snort rule engine. This engine can then use any and all of the protocol information, as well as the payload contents (parsed or raw) to detect malicious traffic.
 - The output engine is then invoked to determine how any resulting alerts will be communicated to the end-user. This can range from Snort's native text-based alerting to syslog and SNMP alerts.

Any packet that triggers an alert, either through preprocessors or signatures, is by default captured in libpcap format, and can be used by analysts to further investigate the cause of the alert. Snort can be configured to capture subsequent related traffic.

7.7.2 Configuration

By default on Linux systems, the important Snort files and directories are:

- **/etc/snort/snort.conf** This file is where global values for Snort are declared, including the internal/external network definitions, how the preprocessors will be configured, how the output processors will be configured for logging, and the inclusion of major chunks of rules.
- **/etc/snort/rules/** This directory contains the rules files themselves. Within each file, rules can be individually disabled or enabled.
- **/var/log/snort/** This directory contains the native alerts file, where Snort records text-based alerts, and the libpcap files with corresponding packet captures.

The locations of these Snort files/directories are fully configurable.

7.7.3 Snort Rule Language

Snort rules are defined by text-based language, one line per rule. These rules are contained in numerous files, which by default contain thousands of available rules distributed by Sourcefire's Vulnerability Research Team (VRT). Each rule may be enabled or disabled by default, based on what the VRT deems is useful in a typical environment. All Snort rules must contain a Snort ID (SID), which identifies that particular rule. SIDs lower than 1 million are reserved for the Sourcefire VRT; SIDs between 1 and 2 million are reserved for a freelance, open-source community called "Emerging Threats."¹² Local rules can be built, but should always be numbered greater than or equal to 2 million. When creating customized local rules, best practice is to copy an existing rule to the local rules file, disable the commercial

12. "Emerging Threats," 2011, <http://www.emergin threats.net/>.

rule, and enable the modified local version. This way if the original rule is updated by the VRT, the update won't clobber your local changes.

7.7.3.1 Rule Header

The Snort rule header consists of seven fields, space delimited, as follows:

- Action: This field describes what the Snort sensor should do when a packet is discovered to match the rule. Typical actions are alert, log, pass, and drop.
- Protocol: This field delineates the protocol the packet must be employing to match the rule. Possible values are icmp, tcp, udp, or the general case, ip.
- Source IP/network and port: These fields describe what the source of a packet must be to qualify as a match.
- Directionality operator: This field allows the rule author to specify whether a match can occur in one direction only (from source to destination) or bidirectionally. The allowed values are “->” and “<>”.
- Destination IP/network and port: These fields describe what the destination of a packet must be to qualify as a match.

Here are three examples of Snort rule headers, which we examine in turn.

- `alert tcp any any -> 192.168.2.1 80 (...)`

For any TCP segment that is from any source at all and that is destined specifically to port 80 on host 192.168.2.1, compare it with the rule body. If a match is found, both log the packet and issue an alert.

- `log udp 192.168.1.1 53 -> !192.168.1.0/24 any (...)`

For any UDP datagram that originates from port 53 on host 192.168.1.1 and that is destined for anywhere except the 192.168.1.0/24 network, compare it with the rule body. If a match is found, log the packet, but do not alert on the event.

- `drop ip $EXTERNAL_NET any <> $HTTP_SERVERS $HTTP_PORTS (...)`

Here we notice variable names instead of sources and destinations. This is the most common practice when writing rules, as keeping specific network ranges defined in a single place (typically the snort.conf file) allows us to change the network topology or the location of the sensor without having to edit all the rules. In this case we assume the rule we're looking at is used in an in-line IPS, because the action is “DROP.” This rule will evaluate any IP packet against the rule body if it is coming from an external source and destined for the predefined HTTP servers. If the rule body also matches the packet, the packet is dropped.