

4 FREE BOOKLETS

YOUR SOLUTIONS MEMBERSHIP



Securing **IM and P2P** **Applications** for the Enterprise

Are You Ready for the Network Equivalent of Guerrilla Warfare?

- Learn Why 40% of the Top Network Threats Are Instant Messaging or Peer-to-Peer Based
- Create Policies to Distinguish Legitimate from Malicious Traffic
- Prepare for the Threats Inherent in the Next Generation of VoIP Products, Including Skype and Google Talk

Paul L. Piccard

Brian Baskin

Craig Edwards

George Spillman

Marcus H. Sachs Technical Editor

**FOREWORD
BY KEVIN BEAVER**

Register for Free Membership to

s o l u t i o n s @ s y n g r e s s . c o m

Over the last few years, Syngress has published many best-selling and critically acclaimed books, including Tom Shinder's *Configuring ISA Server 2004*, Brian Caswell and Jay Beale's *Snort 2.1 Intrusion Detection*, and Angela Orebaugh and Gilbert Ramirez's *Ethereal Packet Sniffing*. One of the reasons for the success of these books has been our unique **solutions@syngress.com** program. Through this site, we've been able to provide readers a real time extension to the printed book.

As a registered owner of this book, you will qualify for free access to our members-only solutions@syngress.com program. Once you have registered, you will enjoy several benefits, including:

- Four downloadable e-booklets on topics related to the book. Each booklet is approximately 20-30 pages in Adobe PDF format. They have been selected by our editors from other best-selling Syngress books as providing topic coverage that is directly related to the coverage in this book.
- A comprehensive FAQ page that consolidates all of the key points of this book into an easy-to-search web page, providing you with the concise, easy-to-access data you need to perform your job.
- A "From the Author" Forum that allows the authors of this book to post timely updates and links to related sites, or additional topic coverage that may have been requested by readers.

Just visit us at **www.syngress.com/solutions** and follow the simple registration process. You will need to have this book with you when you register.

Thank you for giving us the opportunity to serve your needs. And be sure to let us know if there is anything else we can do to make your job easier.

SECURING
IM and P2P
Applications
for the Enterprise

Paul L. Piccard

Brian Baskin

Craig Edwards

George Spillman

Marcus H. Sachs Technical Editor

Syngress Publishing, Inc., the author(s), and any person or firm involved in the writing, editing, or production (collectively “Makers”) of this book (“the Work”) do not guarantee or warrant the results to be obtained from the Work.

There is no guarantee of any kind, expressed or implied, regarding the Work or its contents. The Work is sold AS IS and WITHOUT WARRANTY. You may have other legal rights, which vary from state to state.

In no event will Makers be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out from the Work or its contents. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

You should always use reasonable care, including backup and other appropriate precautions, when working with computers, networks, data, and files.

Syngress Media®, Syngress®, “Career Advancement Through Skill Enhancement®,” “Ask the Author UPDATE®,” and “Hack Proofing®,” are registered trademarks of Syngress Publishing, Inc. “Syngress: The Definition of a Serious Security Library”™, “Mission Critical™,” and “The Only Way to Stop a Hacker is to Think Like One™” are trademarks of Syngress Publishing, Inc. Brands and product names mentioned in this book are trademarks or service marks of their respective companies.

KEY SERIAL NUMBER

001	HJIRTCV764
002	PO9873D5FG
003	829KM8NJH2
004	HJ563LLM8C
005	CVPLQ6WQ23
006	VBP965T5T5
007	HJJJ863WD3E
008	2987GVTWMK
009	629MP5SDJT
010	IMWQ295T6T

PUBLISHED BY

Syngress Publishing, Inc.
800 Hingham Street
Rockland, MA 02370

Securing IM and P2P Applications for the Enterprise

Copyright © 2006 by Syngress Publishing, Inc. All rights reserved. Printed in Canada. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

Printed in Canada

1 2 3 4 5 6 7 8 9 0

ISBN: 1-59749-017-2

Publisher: Andrew Williams
Acquisitions Editor: Jaime Quigley
Technical Editor: Marcus H. Sachs
Cover Designer: Michael Kavish

Page Layout and Art: Patricia Lupien
Copy Editor: Amy Thomson
Indexer: Richard Carlson

Distributed by O'Reilly Media, Inc. in the United States and Canada.

For information on rights, translations, and bulk purchases, contact Matt Pedersen, Director of Worldwide Sales and Licensing, at Syngress Publishing; email matt@syngress.com or fax to 781-681-3585.



Acknowledgments

Syngress would like to acknowledge the following people for their kindness and support in making this book possible.

Syngress books are now distributed in the United States and Canada by O'Reilly Media, Inc. The enthusiasm and work ethic at O'Reilly are incredible, and we would like to thank everyone there for their time and efforts to bring Syngress books to market: Tim O'Reilly, Laura Baldwin, Mark Brokering, Mike Leonard, Donna Selenko, Bonnie Sheehan, Cindy Davis, Grant Kikkert, Opol Matsutaro, Steve Hazelwood, Mark Wilson, Rick Brown, Tim Hinton, Kyle Hart, Sara Winge, Peter Pardo, Leslie Crandell, Regina Aggio, Pascal Honscher, Preston Paull, Susan Thompson, Bruce Stewart, Laura Schmier, Sue Willing, Mark Jacobsen, Betsy Waliszewski, Dawn Mann, Kathryn Barrett, Karen Montgomery, John Chodacki, and Rob Bullington.

The incredibly hardworking team at Elsevier Science, including Jonathan Bunkell, Ian Seager, Duncan Enright, David Burton, Rosanna Ramacciotti, Robert Fairbrother, Miguel Sanchez, Klaus Beran, Emma Wyatt, Chris Hossack, Krista Leppiko, Marcel Koppes, Judy Chappell, Radek Janousek, and Chris Reinders for making certain that our vision remains worldwide in scope.

David Buckland, Marie Chieng, Lucy Chong, Leslie Lim, Audrey Gan, Pang Ai Hua, Joseph Chan, and Siti Zuraidah Ahmad of STP Distributors for the enthusiasm with which they receive our books.

David Scott, Tricia Wilden, Marilla Burgess, Annette Scott, Andrew Swaffer, Stephen O'Donoghue, Bec Lowe, Mark Langley, and Anyo Geddes of Woodslane for distributing our books throughout Australia, New Zealand, Papua New Guinea, Fiji, Tonga, Solomon Islands, and the Cook Islands.



Lead Author



Paul L. Piccard serves as Director of Threat Research for Webroot, where he focuses on research and development, and providing early identification, warning, and response services to Webroot customers. Prior to joining Webroot, Piccard was manager of Internet Security Systems' Global Threat Operations Center. This state of the art detection and analysis facility maintains a constant global view of Internet threats and is responsible for tracking and analyzing hackers, malicious Internet activity, and global Internet security threats on four continents.

His career includes management positions at VistaScape Security Systems, Lehman Brothers, and Coopers & Lybrand. Piccard was researcher and author of the quarterly Internet Risk Impact Summary (IRIS) report. He holds a Bachelor of Arts from Fordham University in New York.



Technical Editor



Marcus H. Sachs, P.E., is SRI International's Deputy Director of the Department of Homeland Security's Cyber Security Research and Development Center, a portfolio of several dozen cyber security R&D projects managed by DHS and supported by SRI. Marc also volunteers as the director of the SANS Internet Storm Center and is a cyberspace security researcher, writer, and instructor for the SANS Institute. After retiring from the US Army in 2001 following a 20-year career as a Corps of Engineers officer, Marc was appointed by President George W. Bush to serve on the staff of the National Security Council as part of the White House Office of Cyberspace Security from 2002 to 2003.

Brian has been instructing courses for six years, including presentations at the annual DoD Cyber Crime Conference. He is an avid amateur programmer in many languages, beginning when his father purchased QuickC for him when he was 11, and has geared much of his life around the implementations of technology. He has also been an avid Linux user since 1994, and enjoys a relaxing terminal screen whenever he can. He has worked in networking environment for over 10 years from small Novell networks to large, mission-critical, Windows-based networks

Brian lives in the Baltimore, MD area with his lovely wife and son. He is also the founder, and president, of the Lightning Owners of Maryland car club. Brian is a motor sports enthusiast and spends much of his time building and racing his vehicles. He attributes a great deal of his success to his parents, who relinquished their household 80286 PC to him at a young age, and allowed him the freedom to explore technology.



George Spillman is a Director for Acadine Informatics, president of the computer consulting group PixelBlip Digital Services, and one of the principals behind ToorCon, the highly respected computer security conference that draws in and educates some of the best hackers and security experts from around the globe. As such, he travels well in hacker circles and takes great pleasure in poking and prodding the deep dark underbelly of the Internet. George is a frequent guest on television news programs for his expertise and his ability to communicate complex computer security and identity theft issues to non-technical audiences. His consulting clients include representatives from both the Fortune 100 and the Fortune 100,000,000. In the past he has been lured away from consulting by large wheelbarrows of stock options to serve as Director of IT for an international pharmaceutical R&D company, and would most likely do that again if the wheelbarrow was included to sweeten the deal. George was a reviewer for the Syngress book, *Phishing Exposed*, (ISBN: 159749030X).

Marc has contributed to Syngress titles *IT Ethics Handbook*, *Cyber Adversary Characterization*, and *Zero-Day Exploits*.

Marc holds a Master of Science in Computer Science with a concentration in Information Security from James Madison University, a Master of Science in Science and Technology Commercialization from the University of Texas, and a Bachelor of Civil Engineering from the Georgia Institute of Technology. He is a graduate of the Army's Command and General Staff College, the Army Engineer School, the Army Signal School, and the Army's Airborne and Air Assault schools. Marc holds an advanced class amateur radio license, is a registered Professional Engineer in the Commonwealth of Virginia, and is a life member of the Signal Corps Regimental Association and the Armed Forces Communications and Electronics Association. A native of Tallahassee, Florida, he currently lives in Virginia with his wife and children.



Contributing Authors



Brian Baskin (MCP, CTT+) is a researcher and developer for Computer Sciences Corporation, on contract to the Defense Cyber Crime Center's (DC3) Computer Investigations Training Program (DCITP). Here, he researches, develops, and instructs computer forensic courses for members of the military and law enforcement. Brian currently specializes in Linux/Solaris intrusion investigations, as well as investigations of various network applications. He has designed and implemented networks to be used in scenarios, and has also exercised penetration testing procedures.

Contents

Foreword	xxiii
Part I Instant Messaging Applications.	1
Chapter 1 Introduction to Instant Messaging.	3
Introduction	4
Major Instant Messaging Services	6
Instant Messaging Popularity	7
Common Features	8
Third-Party Clients	10
Common Security Issues	11
Social Engineering and Identity Theft	12
File Transfers and Messages Spread Malicious Software	12
Worms and File TransferCircumvent Gateway	
Security Devices	13
IP Address of Workstation Revealed During Usage	14
Messages and Files are not Encrypted	15
Message Logging	15
SPIM and Offensive Material	15
Client Security	16
Summary	18
Solutions Fast Track	19
Frequently Asked Questions	22
Chapter 2 AOL Instant Messenger (AIM).	25
Introduction	26
AIM Architecture	26
AIM Protocol	30
AIM Features and Security Information	31
Instant Messaging	32

Encryption	32
Group Chat	33
Audio Chat	34
File Transfer	35
File Share	36
Malicious Code and Client Security	37
AIMDES	39
Oscarbot/Opanki	42
Velkbot	43
Client Security	44
Description:	45
Platforms Affected:	45
Remedy	45
Consequences:	45
References:	45
Summary	47
Solutions Fast Track	47
Frequently Asked Questions	49
Chapter 3 Yahoo! Messenger	51
Introduction	52
Yahoo! Messenger Architecture	52
Yahoo! Messenger Protocol	57
Features and Security Information	59
Instant Messaging	60
Encryption	61
Message Archiving	61
Conferences	62
Voice Chat	63
Yahoo! Chat Rooms	64
File Transfer	65
File Share	66
Web Camera Settings	66
Yahoo! Messenger Malicious Code and Client Security	68
Worm Examples	69
W32.Chod.B@mm	69
W32.Picrate.C@mm	81
Client Security	87

Summary	89
Solutions Fast Track	90
Frequently Asked Questions	92
Chapter 4 MSN Messenger	95
Introduction	96
MSN Messenger Architecture and Protocol	96
Features and Security Information	104
Instant Messaging	104
Encryption	106
Message Archiving	106
Whiteboard	107
Application Sharing	108
Remote Assistance	110
Voice Chat	111
File Transfer	112
Web Camera Settings	114
Malicious Code and Client Security	114
Malicious Code	114
Worm	120
W32.Kelvir.R	120
W32.Picrate.C@mm	122
Client Security	126
Vulnerability Description	126
Vulnerability Solution	127
Summary	128
Solutions Fast Track	128
Frequently Asked Questions	131
Chapter 5 ICQ.	133
Introduction and History of ICQ	134
ICQ Features	135
Instant Messaging	136
Encryption	137
Group Chat	137
Message Archiving	138
Voice Chat	139

File Transfer	140
Web Camera Settings	141
Malicious Code	141
Worm Examples	143
WORM_VAMPIRE.A	143
Identification and Termination	144
WORM_CHOD.B	147
Client Security	149
Multiple Vulnerabilities in Mirabilis ICQ Client	149
Vulnerability Description	150
Vulnerable Packages	151
Credits	151
Technical Description	152
Summary	155
Solutions Fast Track	156
Frequently Asked Questions	157

Chapter 6 Trillian, Google Talk, and Web-based Clients 159

Introduction	160
Trillian Features	160
Trillian Features	161
Trillian Malicious Code and Client Security	166
Google Talk	168
Google Talk Features	170
Instant Messaging	170
Encryption	171
Voice Chat	171
Web-based Clients	172
Web-based Client Features	172
Instant Messaging	172
Encryption	173
Circumventing Workstation Controls	173
Summary	174
Solutions Fast Track	175
Frequently Asked Questions	176

Chapter 7 Skype	179
Introduction	180
Skype Architecture	181
Features and Security Information	183
Instant Messaging	183
Encryption	184
Chat History	184
Skype Calls(Voice Chat)	185
Group Chat	186
File Transfer	188
Malicious Code	189
Client Security	190
A Word about Network Address Translation and Firewalls	192
Home Users	195
Small to Medium-Sized Businesses	195
Large Corporations	195
What You Need to Know About Configuring Your	
Network Devices	197
Home Users or Businesses Using a DSL/Cable	
Router And No Firewall	197
Small to Large Company Firewall Users	198
TCP and UDP Primer	198
NAT vs. a Firewall	199
Ports Required for Skype	200
Home Users or Businesses Using a DSL/Cable	
Router and No Firewall	200
Small to Large Company Firewall Users	200
Skype's Shared.xml file	201
Microsoft Windows Active Directory	202
Using Proxy Servers and Skype	205
Display Technical Call Information	207
Small to Large Companies	211
How to Block Skype in the Enterprise	211
Endnote	212
Summary	213
Solutions Fast Track	214
Frequently Asked Questions	215

Part II Peer-to-Peer Networks.	217
Chapter 8 Introduction to P2P	219
Introduction	220
Welcome to Peer-to-Peer Networking	221
Enter Napster	223
Gnutella and a Purer P2P Network	225
The Rise of the Ultrapeer	226
The Next Step: Swarming	227
eDonkey (Kademlia/OverNet)	227
BitTorrent	228
Other Networks	228
Concerns with Using P2P Networks	231
General Concerns	231
Infected or Malicious Files	231
Legal Concerns	233
Sony Corp v. Universal City Studios	233
A&M Records Inc. v. Napster Inc.	234
MGM Studios Inc. v. Grokster Ltd.	234
RIAA vs. The People	235
The Future of P2P Networks	236
Frequently Asked Questions	237
Chapter 9 Gnutella Architecture	239
Introduction	240
Gnutella Clients and Network	240
Gnutella	240
LimeWire	241
BearShare	242
Gnucleus	243
Morpheus	243
Gnutella Architecture	243
UltraPeers	245
Gnutella Protocol	246
Peer Connections	246
Descriptor Packets	247
Ping/Pong Descriptor Packets	248

Summary	313
Solutions Fast Track	314
Frequently Asked Questions	316
Chapter 12 FastTrack	319
Introduction	320
History of Clients and Networks	320
The FastTrack Network	320
Kazaa	321
History of Kazaa	323
Morpheus	325
Grokster	326
iMesh	327
Spyware Bundling and Alternative Clients	328
AltNet	328
Kazaa Lite Client	329
Kazaa Loaders	330
External Utilities	331
Kazaa Lite Resurrection Client	331
K-Lite Client	332
Network Architecture	332
Supernodes	334
Protocol Analysis	336
Connecting Clients	337
Performing a Search	339
Transferring Files	339
The X-KazaaTag	341
Features and Related Security Risks	343
Downloading and Copyright Violations	343
Malicious Software	343
Fake Files	344
Sharing	346
Legal Threats	346
Vulnerabilities	347
Bandwidth Issues and Mitigation Steps	347
Supernode Clients	348
Firewall Rules	348

Query Descriptor Packets	249
QueryHits Descriptor Packets	250
File Transfers	252
Features and Related Security Risks	254
Problems Created by P2P in the Enterprise	254
Infected Files: Trojans and Viruses	255
Misconfigured File Sharing	256
Copyright Infringement	257
File Transfers Reveal IP Address	257
Technical Countermeasures for Gnutella	257
Firewall Rules	259
IPTables String Match Module	260
Snort IDS Rules	262
Summary	263
Solutions Fast Track	263
Frequently Asked Questions	265
Chapter 10 eDonkey and eMule	267
Introduction	268
History of the eDonkey and eMule Clients and Networks	268
The eDonkey and eMule Networks	271
Features and Related Security Risks	275
Copyright Infringement	275
Malicious Software	275
Poisoned Files	277
Misconfigured Sharing	277
Vulnerabilities	278
Vulnerability Description	278
Vulnerability Solution	278
Vulnerability Provided and/or Discovered by	
PivX Bug Researcher	278
Vulnerability Description	279
Vulnerability Solution	279
Vulnerability Provided and/or Discovered By	279
Summary	280
Solutions Fast Track	281
Frequently Asked Questions	282

Chapter 11 BitTorrent	285
History of the Network	286
BitTorrent	287
BitTornado	288
Azureus	288
BitComet	289
Other Clients	290
ABC	290
μ Torrent	290
G3 Torrent	291
Shareaza	291
Network Architecture and Data Flow	291
Torrent Files	292
Trackers	292
Of Leechers and Seeders	294
Trackerless	295
Protocol Analysis	296
Bencoding	296
Torrent Files	297
Tracker Connections	299
Peer Connections	302
Peer States	304
Peer Wire Protocol Messages	305
Peer Requests	307
Peer Data Transmission	307
DHT Connections	307
Features and Related Security Risks	308
Copyright Infringement	308
Poison Peers	309
Automatic Sharing of Data	310
Bandwidth Issues and Mitigation Steps	310
Bandwidth Scheduling	311
Trackers	311
Sharing of Data	311
Snort IDS Rules	312

IPTables String Match Module	349
P2PWall	350
Snort IDS Rules	352
Frequently Asked Questions	356
Part III Internet Relay Chat Networks	359
Chapter 13 Internet Relay Chat—Major Players of IRC 361	
Introduction	362
History	362
IRC Jargon	363
Nick	364
Ident or Username	364
Channel Operator	364
Nick Delay and Time Stamps	365
Nick Delay	366
Time Stamps	367
IRC Server Software Packages	368
ircd 2.11.x	369
ircd-hybrid	369
bahamut	369
ircu (and Derivatives)	370
UnrealIRCd	370
Major Networks	371
Quakenet	371
Undernet, IRCnet, DALnet and EFnet	372
Rizon	372
GameSurge	372
Freenode	373
Summary	374
Solutions Fast Track	374
Frequently Asked Questions	376
Chapter 14 IRC Networks and Security	377
Introduction	378
IRC Networks	378
EFnet	379
DALnet	381

NickServ	381
ChanServ	382
Undernet	384
IRCnet	385
IRC Servers in Sum	385
File Transfer Protocols	386
IRC Botnets	388
Automated Shares/Fserve Bots	388
File-Sharing Botnets	390
Channel Protection Botnets	390
Channel Takeover Botnets	391
Channel Flooding Botnets	391
Spamming Botnets	392
DDoS Botnets	392
Proxy Botnets	392
Other Uses for IRC Bots	393
Summary	394
Solutions Fast Track	394
Frequently Asked Questions	396
Chapter 15 Global IRC Security	399
Introduction	400
DDoS Botnets Turned Bot-Armies	400
Methods of Botnet Control	401
Reprisals	404
The ipbote Botnet: A Real World Example	405
Information Leakage	407
Copyright Infringement	408
Other Forms of Infringement	408
Transfer of Malicious Files	411
How to Protect Against Malicious File Transfers	413
What to Do if a Malicious File Infects Your Network	414
Prevention of Malicious File Sends in the Client	414
DCC Exploits	414
Firewall/IDS Information	415
Port Scans	415
IDS	415

Summary	417
Solutions Fast Track	417
Frequently Asked Questions	419
Chapter 16 Common IRC Clients by OS	421
Introduction	422
Windows IRC Clients	422
mIRC	422
X-Chat	424
Opera IRC Client	425
ChatZilla	425
WinBot	425
Visual IRC (vIRC)	425
Trillian	425
UNIX IRC Clients	426
X-Chat	426
IRSSI	427
BitchX	427
KVIrc	428
sirc	428
ircII	428
Apple Macintosh IRC Clients	428
ChatNet	428
Snak	429
Homer	429
Ircle	429
MacIRC	429
Colloquy	430
Other IRC Clients	430
PJIRC	431
J-Pilot	431
CGI:IRC	431
SILC	431
Summary	432
Solutions Fast Track	433
Frequently Asked Questions	435
Index.	437

Foreword

I've been expressing my concerns about IM and P2P security to colleagues, students, and clients for nearly a decade. Initially, what I saw coming down the pike and communicated to others fell on deaf ears. I heard things like “yeah, yeah, this is all just novelty software for home users, hackers, and copyright violators” and “these technologies will never have a place in the enterprise.” But I knew this was going to be big. Not to mention a good opportunity for me as an information security consultant. So I stuck with it.

Over three years ago I gave a presentation on instant messaging security at several security conferences. The interesting thing about these sessions is that they were chock full of IT and security professionals eager to learn how to secure their corporate conversations. Later that same year, I served on a panel (which included a member of the RIAA of all people!) to talk about P2P use and concerns. Again, this session was full of people eager to see what it was all about, and how to keep it under wraps. People were starting to come around.

Even to this day, network managers will make you think that IM and P2P will never come to fruition in a business environment. However, year after year, studies show increasing usage of IM and P2P within business networks. I can certainly attest to seeing tons of IM and P2P traffic on networks that I'm assessing as well. The reality is these technologies are everywhere on corporate networks and they're not going away. People are only going to become more and more dependent on them—especially once their business value sinks in. Further fueling the fire, more and more vendors (especially Microsoft) are jumping aboard the IM and P2P bandwagon. This will only perpetuate their use.

As with any new technology, there are always going to be security issues to contend with. Security flaws and general misuse of IM and P2P can lead to innumerable losses of intellectual property, personal information, network bandwidth, and even employee productivity. But this is nothing new. We've all experienced the security pains associated with e-mail, Web-based applications, wireless networks, and so on—we just have to apply old solutions in a new context.

In all but the most stringently controlled networks, it's futile and counter-productive to ignore the presence of IM and P2P in your enterprise. I'm the first to admit that serious business value can come from these applications. However, as with anything of value, IM and P2P do have their risks. But this can be controlled, especially if it's approached from all the critical angles—not just from a technical perspective.

If you're going to be effective and successful in managing and securing IM and P2P long-term, it'll require some effort. You'll need to develop organizational standards and policies, ensure policies are being enforced with technical solutions where possible, and perform ongoing security testing to make sure no new risks have been introduced by these applications or the people using them. The best way to go about doing this is to have the involvement and support of upper management.

There has never been a better time for IT professionals to get that buy-in and get a grip on the security risks associated with IM and P2P. The most logical place to start is here—the best resource I've ever seen on IM and P2P security—to point you in the right direction.

—Kevin Beaver
*Founder and information security
consultant for Principle Logic, LLC*

Part I

Instant Messaging

Applications

Introduction to Instant Messaging

Solutions in this chapter:

- Major Instant Messaging Services
- Instant Messaging Popularity and Common Features
- Third-party Clients
- Common Instant Messaging Security Issues

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Instant messaging (IM) and peer-to-peer services are steadily increasing in popularity and are becoming a greater concern for security professionals and network administrators. Instant messaging usage has increased dramatically in recent years, and has become a mainstay in corporate environments, with or without the approval of networking and security groups. According to a study by the Radicati Group published in July 2004, instant messaging is used in 85% of corporate environments in North America. According to the report, it was forecast that there would be 362 million instant messaging users in corporate environments, with 768 million accounts, using the same public instant messaging services available to home users.

Peer-to-peer networks are also often accessed from work, where the higher available bandwidth makes downloading large files more efficient. Instant messaging and peer-to-peer networks function very differently in terms of architecture and how they impact the network they reside on. Recently, however, these programs are starting to blur the lines between each other and now ship with many of the same features, such as file sharing and communicating with other users. This presents a problem for network administrators and security professionals. Clients that access these services are often installed on workstations without permission or company consent, and are designed to work around many of the typical security measures, such as firewalls, that have been put in place to stop the activity of these services. Instant messaging and peer-to-peer applications open up a host of security issues in any environment where they are run, from the obvious risks of client-side vulnerabilities to the less obvious issues and risks associated with copyright infringement, information leakage, and unregulated communications and file sharing with users outside the corporate environment.

Instant messaging services are designed to send instant messages to another user. This form of communication can send messages in near real-time, resulting in conversations that are more like a telephone conversation where there is instant feedback. Instant messaging clients provide a wide variety of features that will be discussed in detail in Chapter 2. Most of these features are carried out with the assistance of a central server. When you sign into an instant messaging service, your username and password are sent to a central server for authentication. Once your username and password are verified and you have access to the service, almost all of your communication with others is sent through a central server before reaching its destination. Servers are responsible for looking up the IP (Internet Protocol) address of the intended recipient and delivering the instant message or other communication. There are times when a central server is not involved in instant messaging. This

direct connection with another client helps reduce the load of the server and also can speed up the delivery of information. For instance, most instant messaging services provide users with the ability to send files to one another. These services have little, if any, size limitations on file transfers. Rather than sending the file to the central server first, you establish a direct connection with the intended recipient and transfer the file directly.

Internet Relay Chat (IRC) is a near real-time chat system that was developed in the late 1980's. Like instant messaging, this system is based on a client/server model, but messaging is not limited to two users. There are many networks for IRC, including Dalnet, EFNet, IRCnet, Quakenet, and Undernet. IRC networks are located all over the world, and there are smaller, local networks available to connect to as well. IRC networks are not connected to each other, so a client can communicate with other clients only if it is signed into the same network. These networks are comprised of servers that are responsible for routing messages and hosting channels, which are similar to chat rooms. A user connects to a specific IRC channel hosted on one of the servers on an IRC network by using one of many available IRC clients. There are multiple users on this channel, all taking part in a conversation. One server can host multiple channels, but a user can only see the conversation on the particular channel that he or she has joined. IRC supports features beyond chatting with multiple users, including private messaging and direct file transfers.

Peer-to-peer networks are used mainly for connecting with other users in order to share files. These networks are a group of workstations that share the same client software and connect to each other, creating an ad hoc network. There are several different architectures used in peer-to-peer networking, some of which rely on a centralized server, while others treat all connected clients as equals. In a true peer-to-peer network, there is no centralized server and no sign in process to connect to these networks. In these networks, files are sent and received with a direct connection to another client. This creates some problems with usability, since there is no index of files that are available for download, and searching can take a long time. Most peer-to-peer networks use a semi-centralized architecture, where a workstation must know the IP address of another workstation (which may function as a *super node*) or server in order to connect. These servers or super nodes aid in locating files by indexing nearby files or passing a file search onto workstations closest to it.

This book will provide details on instant messaging, peer-to-peer, and IRC systems, including popular clients and networks, security vulnerabilities, and best practices for protecting a corporate network or individual workstations against security threats from these messaging and chat systems.

Major Instant Messaging Services

Most instant messaging services require some type of centralized infrastructure in order to operate. Functions such as routing messages and authentication have to be handled by servers. Because of the expense required in building and maintaining these infrastructures, the larger and more popular services are owned by several companies, including AOL, Yahoo!, and Microsoft. Due to the investment each company has made in building these services, there is little, if any, desire to share these resources with each other (or with third parties), which is why each service requires that you use a specific client for access. For instance, if you wish to use the MSN service, Microsoft, the owner of the service, requires the MSN Messenger client to connect. The following clients, and their corresponding services, if applicable, will be covered in this book:

- AIM
- Yahoo!
- MSN
- ICQ
- Trillian
- Web-based Clients
- Skype

IM Backgrounder

What is the Difference Between a Service, Network, and a Client?

A *service* is generally run by a major provider, including AOL, Yahoo!, or MSN. These services are made up of multiple servers that are responsible for authenticating users via usernames and passwords, acting as intermediaries between network clients, and passing messages and other functions through servers first and routing them to the appropriate user.

A *network* is a collection of servers that act in concert to provide an IM service to users who have been signed into the service and appropriately authenticated.

Continued

A *client* is the application that you use to connect to a service network, and it is what you interface with when typing messages and using other features of instant messaging services. The client is generally tied to a particular instant messaging network, allowing you to access your list of contacts on that particular network, and use some functions that may be unique to that service.

There are some clients that are not published by an owner of a service (AOL, Yahoo!, Microsoft, etc.) that are able to connect to multiple services at the same time. These clients rely on protocol information from reverse engineering or documentation to connect to these multiple services, and are generally free or open source software. One of these clients, Trillian, is able to connect to multiple services and provides a greater level of security than other clients. Trillian will be discussed in greater detail in Chapter 3.

Web-based clients are not used very often due to their limited feature sets. These clients generally lack most of the features of locally installed clients, and focus only on delivering instant messages. AOL and Microsoft produce versions of their instant messaging clients that can be used via the Web. These Web-based clients can circumvent workstation restrictions regarding software installation since they operate completely from the Web using Java. Additionally, these clients are able to bypass gateway security devices such as firewalls, circumventing security measures that may already be in place to prevent these services from being used in a particular environment.

Another client that offers some unique features is Skype. The beta for Skype was launched in August 2003 and is available for multiple platforms including Windows, MacOS X, Linux, and Pocket PC. By October 2004, Skype had seen rapid growth and over one million clients were connected to the service simultaneously. What fueled this rapid adoption was its focus on Internet telephony with free long distance and international communication with other Skype users. Skype originally provided Internet telephony services for free and has recently added new features including file transfers, instant messaging, and the ability to place calls to Public Switched Telephone Network (PTSN) numbers around the world, which are received on standard phones. Another unique feature of this client is that voice conversations and instant messages are encrypted between parties, reducing some security concerns on networks.

Instant Messaging Popularity

Research from comScore Media Metrix conducted in July 2004 determined the popularity of instant messaging services. This study measured which client was used for sending instant messages. Keeping in mind multiple instant messaging services can be used, the most popular clients are detailed in Table 1.1.

- *Whiteboard* features allow for a Microsoft Paint canvas to be launched and shared with another user for collaboration.
- *Mobile Messaging* provides the ability to send messages and alerts from the instant messaging client to a mobile phone.
- *Multi-Network* capabilities provide the ability to use one client to connect to several major instant messaging services.
- *Web Services Integration* is a feature of several instant messaging clients and provides access to some of the features that are available on websites. E-mail notification alerts, stock quotes, and other services that are usually associated with a particular website may be available through an instant messaging client as well.

IM Backgrounder

Are Instant Messaging Clients the Same as Peer-to-Peer Networks?

Although they both are starting to share many of the same features, instant messaging services are very different from true peer-to-peer networks. Instant messaging services require you to sign into a particular service and route your information through a server first before reaching its destination. Peer-to-peer networks are based on a system where most users and their workstations are considered equals. There are no centralized servers and therefore no usernames or other unique information is used to identify users. Additionally, these services are different in their architectures. Peer-to-peer networks are designed for efficiency when searching for and transferring large files across many workstations, while instant messaging services have features and functionality geared towards interpersonal communication between users.

Third-Party Clients

There are many other clients that may be used for connecting to instant messaging services. Since services are closed networks owned by providers, there is a chance that some functionality may be blocked. Additionally, services may change the protocol used for connection and communication in an effort to prevent third-party

Table 1.1 Instant Messaging Market Penetration by Client

Client	Market	
AOL	37%	America Online’s service for AOL members only
Yahoo!	33%	Users spent more time online than other messengers
AIM	31%	Favored by college students over other messengers
MSN	25%	Microsoft’s instant messaging client
ICQ	6%	Owned by AOL
Trillian	1%	Third-party multiprotocol client

This study also found that Yahoo! Messenger was the most popular service used at work, where users were signed into the service for an average of over 7 hours a day.

Common Features

Instant messaging clients have evolved to provide features other than instant messaging in order in order to entice users to spend more time signed into these services actively utilizing its features. There are some basic features that all messaging clients have, which may not aid in messaging, but add or enhance other methods of communication such as file transfers or video and audio chat. Table 1.2 summarizes the functionality available in each instant messaging client.

Table 1.2 Instant Messaging Client Features

	AIM	Yahoo!	MSN	ICQ	Trillian	Skype
Text Chat	X	X	X	X	X	X
Group Chat	X	X		X		X
Audio Chat	X	X	X	X	X	X
VoIP	X	X		X		X
Video Chat	X	X	X	X	X	
File Transfer	X	X	X	X	X	X
File Share	X					
App Sharing			X			
Encryption	X				X	X
Whiteboard			X			

Continued

Table 1.2 continued Instant Messaging Client Features

	AIM	Yahoo!	MSN	ICQ	Trillian	Skype
Mobile Messages	X	X	X	X		
Multi-Network					X	
Web services Integration	X	X	X	X	X	

- *Text Chat* allows for communication with other users by typing messages back and forth to each other. This is the foundation of instant messaging.
- *Group Chat* is similar to a chat room. Invitations can be sent to multiple users, and they join a chat where all parties can see what is typed during that session.
- *Audio Chat* is similar to a phone conversation, but it takes place through the instant messaging client. Both parties must have a microphone and speakers in order to participate.
- *VoIP* services within an instant messaging client are services that are provided by a separate VoIP provider and allow for phone calls to be made. These calls are initiated by the client and are received by a telephone. Skype includes this feature natively.
- *Video Chat* utilizes a webcam or other camera and provides the ability to share video feeds with another user. Some clients may vie for control of another user's webcam without permission.
- *File Transfer* provides the ability to send individual files from one workstation to another. This creates a direct connection between both workstations, bypassing the instant messaging architecture.
- *File Sharing* shares the file contents of an entire directory with other users. This can be a security issue if incorrect permissions are used or if sensitive files are stored in the same directory.
- *Application Sharing* allows two users to utilize the same program at the same time. The executable is hosted on one workstation and both users have full access to the program and its features.
- *Encryption* solves one of the basic security concerns of instant messaging. Messages cannot be intercepted and easily read if they are encrypted.

clients from connecting to the service, which would mean that a third-party client would experience an interruption in service. Some of these clients provide unique features or run on operating systems that are not supported by operators of instant messaging services. The following is a list of several of these clients:

- **IM2** (www.im2.com) IM2 began its beta in January 2004 and has steadily increased its available features. It is now able to encrypt communications from all supported protocols (AIM, Yahoo!, MSN, and ICQ).
- **Miranda Instant Messenger** (www.miranda-im.org) Miranda Instant Messenger is an open source (GPL) client that connects to all major instant messaging services. It supports a plugin architecture, making it highly extensible. Miranda runs on Microsoft Windows platforms.
- **Gaim** (<http://gaim.sourceforge.net>) Gaim is a multiprotocol client and is available in multiple platforms including Microsoft Windows and Linux. It is free software available under the GNU GPL.
- **Kopete** (<http://kopete.kde.org>) Kopete is a multiprotocol client, packaged with the KDE desktop environment for Linux.
- **Qnext** (www.qnext.com) Qnext is a Java-based multiprotocol instant messaging client with features that include streaming music from another client, photo sharing, and remote PC access.
- **Jabber** (www.jabber.org) Jabber is an XML-based system for instant messaging. There are multiple Jabber server implementations available for download, and there are multiple Jabber clients, which are multiprotocol.

Common Security Issues

Since instant messaging clients have the same basic functionality, it only makes sense that they share many of the same security risks. Some of these risks are the result of the client itself, while others take advantage of social engineering to exchange sensitive information. Specific client security issues will be discussed in each section dedicated to a particular instant messaging client. The following sections detail the security issues found in almost all instant messaging clients. Most of these issues are due to social engineering or are problems inherent with the features of instant messaging clients. Many features of instant messaging clients can be used to replace common protocols that may already be blocked in a corporate setting. For instance, features such as file transfers allow users to evade network controls that restrict FTP (File Transfer Protocol) access or limit attachment sizes in e-mail messages. These

features many be highly configurable, operating on user-defined ports, making it harder to block specific features while allowing access to more benign features such as instant messaging, which may enhance productivity. Since each client uses specific ports for its features, and may be configurable, when discussing each client we will be providing counter measures to these and other security issues.

Social Engineering and Identity Theft

Social engineering is especially pervasive on instant messaging services. This is because of the idea of *buddy lists*, where users add contacts that they are familiar with to their lists. The assumption is that when someone contacts you, they have received your name from a friend, when in fact it could have been gained through a simple dictionary attack.

Identity theft can lead to several problems. By posing as an employee of an instant messaging service, a malicious user can convince someone to divulge information such as usernames, passwords, and credit card information. This information can be used to compromise other systems and services and can lead to theft. Additionally, this information can be used to impersonate the user on the instant messaging service. Once the malicious user has access to all of the legitimate user's online contacts, he or she can begin to contact them and ask for sensitive and confidential information. There is a good chance that this information will be obtained since the malicious user appears as an acquaintance to others.

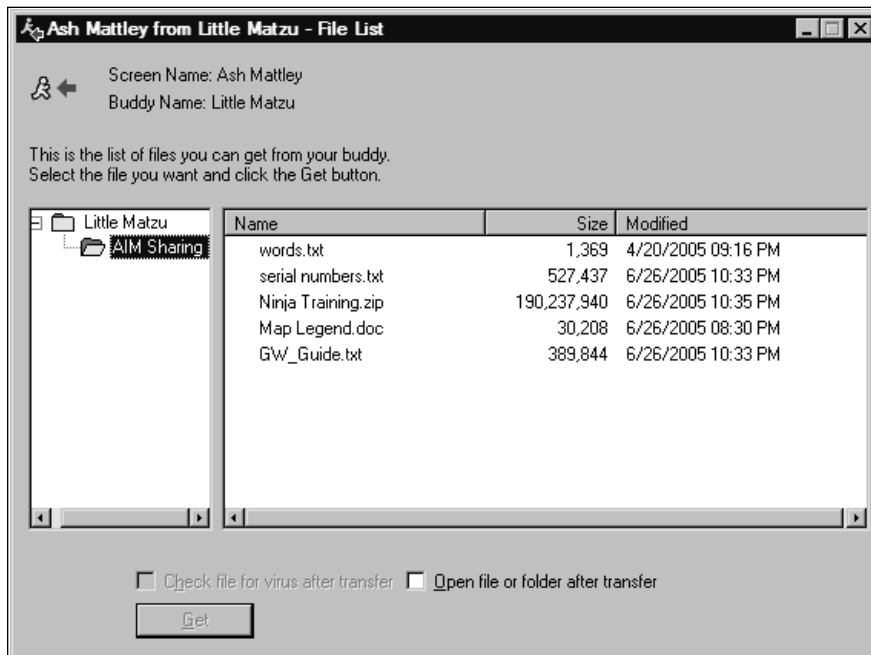
Another method of identity theft involves obtaining usernames or passwords through decryption on the local workstation or through a packet capture utility. Programs such as *dsniff* (located at www.monkey.org/~dugsong/dsniff), are able to decrypt passwords for AIM and ICQ over a network on the fly. Other utilities, such as *Cain and Able*, a popular utility to monitor network activity and decrypt passwords, can be found at www.oxid.it/cain.html.

File Transfers and Messages Spread Malicious Software

One of the most dangerous security risks for instant messaging clients is the ability to send Trojans and viruses to users with the file transfer feature. Sending files in this manner creates a direct connection between users, bypassing any gateway antivirus scanning that would normally protect a network from becoming infected. Once these pieces of malware infect a machine, they are able to spread to other machines, creating massive amounts of network traffic and overloading a network. Depending on how a client is set up, it is possible for files to be transferred without the host's

knowledge. This may allow sensitive information to be transferred from a workstation without permission. Figure 1.1 shows a dialog box that a user would see in AIM when requesting files from a machine set up to share a directory. It is possible to allow all users access to this directory, and the end user who hosts these files does not receive any notification that a file transfer has been established. Additionally, no logs are provided for this feature, providing no forensic data to determine whether or not files were transferred.

Figure 1.1 Hostile Request for a File Transfer

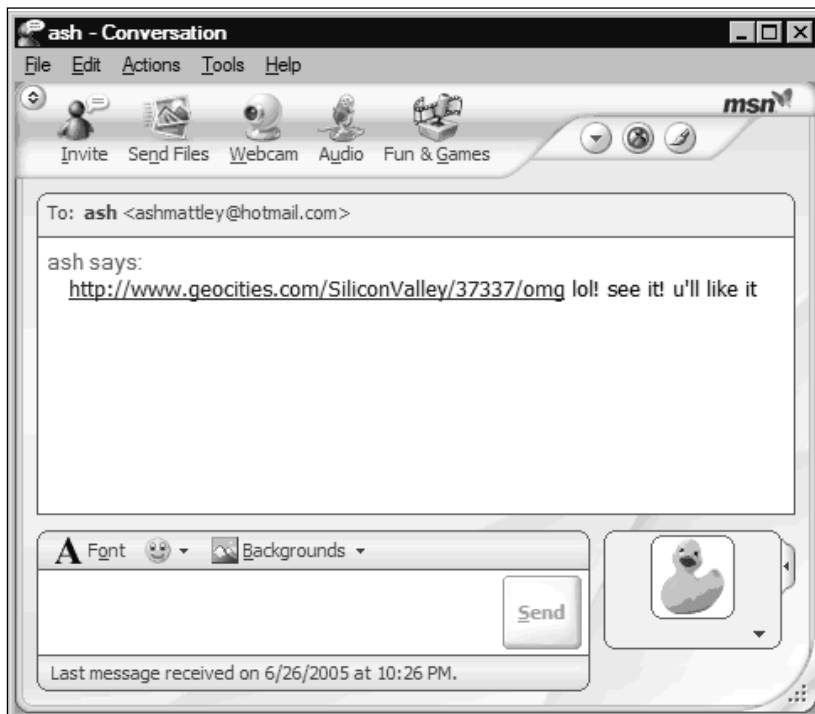


Worms and File Transfer Circumvent Gateway Security Devices

Worms are capable of spreading over instant messaging services, and generally appear as a URL (Uniform Resource Locator). Since these messages come from what appears to be someone on a buddy list, it is more likely that these URLs will be accessed. Once these URLs are clicked, the worm will infect the machine and spread to everyone on the buddy list. Some worms and viruses that spread via instant messenger send an infected file to users and are able to avoid being detected by gateway antivirus devices. These malicious files are written for a specific instant mes-

saging client, and several of them will be discussed with each particular client. Figure 1.2 shows an MSN dialog box and a message that may be from a worm. This worm, like most worms that spread via instant messaging, sends a message to all online contacts. The message contains a URL, which points to a location where a malicious file is available for download.

Figure 1.2 MSN Worm



IP Address of Workstation Revealed During Usage

Some features, including file transfers, reveal the IP address of the workstation being used. This is generally not revealed during instant messaging or other activities, but it becomes necessary when a direct connection is needed. Figure 1.3 shows a dialog box that is presented to a user in ICQ when a file transfer is initiated. After an IP address has been revealed, a malicious user can concentrate on the machine in order to gain access into a network.

Figure 1.3 IP Addresses Revealed Through Client Usage

Messages and Files are not Encrypted

Another major flaw with instant messaging is the lack of encryption for sending instant messages. All clients covered in this book (with the exception of AIM, Trillian, and Skype) do not encrypt information. This information is routed over the Internet through centralized servers to its destination. Any information, including file transfers, can be intercepted by anyone using packet capture software. An example of software capable of monitoring AIM messages, EtherBoss Monitor, is located at www.efeotech.com/aim-sniffer/index.htm. If files are not encrypted, it is recommended that they not be sent via instant messenger. Additionally, sensitive information should never be discussed over instant messaging software unless the conversation is encrypted.

Message Logging

Yahoo! Messenger, MSN Messenger, ICQ, Trillian, and Skype all provide the capability to log online conversations with other users. This information is stored in a text file on the local workstation. A malicious user who has access to this workstation can retrieve this file and have access to all information that was exchanged during an online conversation.

SPIM and Offensive Material

SPIM, or instant messaging SPAM, is not necessarily a security problem, but may cause Human Resources problems depending on the nature of the marketing materials. SPIM is carried out by automated bots that harvest instant messaging names and send marketing messages to users. Currently, these messages do not consume very much network resources, but often contain links to pornographic material. These messages are often more intrusive than SPAM e-mail, since instant messaging

clients alert users when a new instant message arrives. Users in a corporate environment often believe it is the responsibility of the company to protect and prevent objectionable material from being viewed, making this an issue that has to be prevented. Individual users can prevent unwanted SPIM messages by changing the settings in their instant messaging client to ignore messages from unknown users.

Client Security

Worms and other malware target specific clients and services. Since instant messaging services are incompatible, an instant messaging worm generally affects only one client at a time. Instant messaging malware functions somewhat differently than those that affect e-mail. Generally, gateway security devices can block infected e-mails from entering a network, protecting it from infection. However, instant messaging traffic is not checked by many gateway security products since clients can add HTTP (Hypertext Transfer Protocol) headers to instant messaging traffic to avoid firewalls with protocol analysis. Additionally, instant messaging clients can be configured for multiple ports, can utilize proxies, and can automatically configure themselves if a firewall is detected. Luckily, instant messaging worms require user interaction in order to propagate. Usually in the form of a URL, instant messaging threats require an unknowing user to click on a link or download a file. Once the malware is executed, it may not only cause damage to the user's machine, but may also send copies of itself to users on the contact list.

Backdoors and keyboard loggers are especially dangerous on instant messaging clients, since traffic generated from these pieces of malware appear as legitimate instant messaging activity. In this case, there is no need for a system monitor to open a new port for communication, and can instead rely on the instant messaging client to send information back.

An example of this type of activity is the AIM-Canbot Trojan, which was discovered on March 27, 2003. This Trojan, after infecting a workstation, had the ability to download and execute files from malicious users. Once run on a workstation, the Trojan created a bot, which was responsible for automating much of the activity. First, a new AIM username was created in order to connect to the AIM service, registry keys were created in order to allow it to run on system startup. After the AIM account was created and the system changes were made, the Trojan would connect to a specific chat session and notify malicious users that the compromised machine was online with the message "**aimb0t reporting for duty...**" The malicious users, with the help of the online bot, had the ability to gather workstation information including hostname and IP address, alter AIM's sound settings, and instruct the bot

to download and execute files. Since this traffic appeared on standard AIM ports, it was hard to recognize whether or not this traffic was legitimate.

Users are often fooled by these messages since they appear to come from known sources, increasing the likelihood that these worms will continue spreading. In many cases, the messages seem legitimate and instruct the user to look at pictures on a website or download a file. In order to properly protect against instant messaging malware, desktop antivirus protection is strongly recommended.

F-Secure released a report in March 2005 that stated that instant messaging worms were growing at a rate of 50% per month due to the ability to spread worms faster than e-mail. Additionally, F-Secure noted that a worm released to instant messaging clients was capable of spreading to all machines running instant messaging software in less than 15 seconds. Based on this efficient mechanism for delivering malware, instant messaging is becoming a more likely vector for distributing malicious code.

Summary

Instant messaging, IRC, and peer-to-peer networks are all similar in that they allow users to exchange information and files in an efficient but unmanaged way. The impact for a business revolves around several issues, including security of the client, the unregulated flow of sensitive information and files, and copyright infringement. Many of these issues are exacerbated due to social engineering. Whether posing as an employee of an instant messaging service or using one of several tools, a malicious user can obtain a username and password from a user of an instant messaging service. This username and password, in turn, can be used by malicious parties to pose as someone trusted, making it easier to obtain sensitive information including usernames, passwords, and credit card information.

Instant messaging is becoming more popular, and according to the Radicati Group's study, the amount of instant messages sent to other users will grow from 11.4 billion today to 45.8 billion in 2008. This increased usage in the work environment proves that instant messaging is becoming a tool that can be used for efficient communication, but proper security safeguards must be followed. Not only is security becoming an issue, but compliance with regulations such as Sarbanes-Oxley require tracking and logging of instant messaging conversations in the workplace. New regulations treat instant messaging with the same standards as e-mail, whether or not it is authorized by the company. Companies are not only required to protect sensitive information and prevent it from disclosure to the wrong parties, but are now also required to control access of instant messaging and archive conversation activity for periodic review. Certain countermeasures, including instant messaging-specific gateway security devices, have the ability to provide this functionality as well as provide a more secure environment. These devices will be discussed briefly later in this book.

Client security in terms of vulnerabilities are not a primary issue, since publishers regularly update their clients as necessary to protect end users against security exploits. Several exploits have been released in 2005 that required changes to the client software. A security advisory, published in February 2005, detailed a vulnerability in Microsoft's MSN Messenger versions 5.0, 6.1, and 6.2 that exploited a vulnerability in PNG image processing. This vulnerability made it possible for a malicious user to remotely execute code on a vulnerable machine. Microsoft's response was to update its client software, thus fixing the vulnerability and denying access to its instant messaging service unless users upgraded to the non-vulnerable versions of the software.

Instant messaging services, having grown in popularity, have also attracted the attention of malicious users, including virus writers, who look at this medium as a

new vector for spreading advertising as well as malicious code such as Trojans and worms. These forms of attacks have increased rapidly as instant messaging usage has increased. Part of this increase is the efficiency that this code spreads. It is much faster to infect instant messaging clients due to the speed at which messages travel and the ability of these messages to appear as if they came from a trusted source.

Links to Sites

- **www.aim.com** America Online's AOL Instant Messenger (AIM) client.
- **http://messenger.yahoo.com** Yahoo! messenger download site.
- **http://messenger.msn.com** Download site for MSN Messenger from Microsoft.
- **www.icq.com** America Online's ICQ Messenger.
- **www.trillian.cc** The third party client Trillian has the ability to connect to all major instant messaging services, and provides unique security features.
- **www.skype.com** Skype provides Internet telephony, instant messaging, and chat features.

Solutions Fast Track

Major Instant Messaging Services

- ☑ Many people have started using instant messaging clients for features other than text messaging. These programs can be used to transmit video from a webcam or other camera connected to your computer. This can be useful for communicating with people in faraway places. Another often-used feature is audio chat, which allows you to communicate with your friends by attaching a microphone and can replace much of your phone conversations.
- ☑ File transfers are an efficient way to send information to others, and do not have size restrictions like e-mail attachments. For people who are not technically savvy, sending or receiving a large file via FTP can be confusing. Instant messaging clients can be an easier way to send these files. Additionally, this feature can circumvent restrictions on transferring large files.

Instant Messaging Popularity and Common Features

- ✓ *Text Chat* allows for communication with other users by typing messages back and forth to each other. This is the foundation of instant messaging.
- ✓ *Group Chat* is similar to a chat room. Invitations can be sent to multiple users, and they join a chat where all parties can see what is typed during that session.
- ✓ *Audio Chat* is similar to a phone conversation, but it takes place through the instant messaging client. Both parties must have a microphone and speakers in order to participate.
- ✓ *VoIP* services within an instant messaging client are services that are provided by a separate VoIP provider and allow for phone calls to be made. These calls are initiated by the client and are received by a telephone. Skype includes this feature natively.
- ✓ *Video Chat* utilizes a webcam or other camera and provides the ability to share video feeds with another user. Some clients may vie for control of another user's webcam without permission.
- ✓ *File Transfer* provides the ability to send individual files from one workstation to another. This creates a direct connection between both workstations, bypassing the instant messaging architecture.
- ✓ *File Sharing* shares the file contents of an entire directory with other users. This can be a security issue if incorrect permissions are used or if sensitive files are stored in the same directory.
- ✓ *Application Sharing* allows two users to utilize the same program at the same time. The executable is hosted on one workstation and both users have full access to the program and its features.
- ✓ *Encryption* solves one of the basic security concerns of instant messaging. Messages cannot be intercepted and easily read if they are encrypted.
- ✓ *Whiteboard* features allow for a Microsoft Paint canvas to be launched and shared with another user for collaboration.
- ✓ *Mobile Messaging* provides the ability to send messages and alerts from the instant messaging client to a mobile phone.
- ✓ *Multi-Network* capabilities provide the ability to use one client to connect to several major instant messaging services.

- ☑ *Web Services Integration* is a feature of several instant messaging clients and provides access to some of the features that are available on websites. E-mail notification alerts, stock quotes, and other services that are usually associated with a particular website may be available through an instant messaging client as well.

Third-party Clients

- ☑ Although not officially approved, third-party clients may be worth looking into. They can provide unique features that are not available with other clients. Some of these clients are open source, which can provide the ability to rebuild the client to suit your own unique needs.
- ☑ Older clients generally are not allowed on networks since there are protocol changes that require new clients. Additionally, these older clients may contain security vulnerabilities that have been fixed in newer versions of the clients. It is always a good idea to use the newest client available. Use beta software at your own risk, since it may be unstable and contain bugs.
- ☑ Each client utilizes different ports for authentication and communication. There is no one *instant messaging port* that can be blocked to restrict access to these clients. We will be exploring specific client and their port details later on in the book.

Common Instant Messaging Security Issues

- ☑ Social engineering is especially pervasive on instant messaging services.
- ☑ One method of identity theft involves obtaining usernames or passwords through decryption on the local workstation or through a packet capture utility. Programs such as dsniff (located at www.monkey.org/~dugsong/dsniff), are able to decrypt passwords for AIM and ICQ over a network on the fly.
- ☑ Sending trojans and viruses is possible through a direct connection between users, bypassing any gateway antivirus scanning that would normally protect a network from becoming infected. Once these pieces of malware infect a machine, they are able to spread to other machines, creating massive amounts of network traffic and overloading a network.

- ☑ Some features, including file transfers, reveal the IP address of the workstation being used. After an IP address has been revealed, a malicious user can concentrate on the machine in order to gain access into a network.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Which instant messaging client should be used?

A: There are several issues that have to be explored when deciding on an instant messaging client. Security issues are present in all clients, and social engineering and data leakage are not exclusive to any particular client. There really is no “secure client.” Some instant messaging clients, notably Trillian and Skype, provide encryption for instant messaging, which are very desirable in order to prevent users from obtaining sensitive information. If you are locking down workstations to prevent users from installing software, it makes sense to standardize on one client for the entire environment. All of the major clients are responsive to patching software based on security issues, so much of the decision can be based on features.

Q: What platforms do these clients run on?

A: Instant messaging clients can run on a variety of platforms, including desktop operating systems such as Windows, MacOS, and Linux, as well as mobile devices and personal digital assistants (PDAs). Most clients also include a way to launch via a Web interface using Java, so even if an unsupported platform is used, there is a way to connect to an instant messaging network, although there may be some limited functionality. Many cellular phones are now shipped with instant messaging clients built into them. Each phone may have one or more clients to choose from, so the phone itself may determine which client is used. Some clients are also available for download for PDAs.

Q: Why are these companies giving away software?

A: The major instant messaging providers give their software away for several reasons, but they benefit by displaying advertising in small banners within the client. Advertising is a fairly new aspect to these instant messaging clients, and most of the clients did not have any advertising when they were first released. With so many subscribers on these systems, and so much interaction with these programs, it made sense to convert this into an opportunity to profit based on the service that was being provided. Generally, these ads are not intrusive and do not detract from the user's experience.

Q: Why do instant messaging networks prevent interoperability between clients?

A: Instant messaging began with the architecture of a client logging into a network of servers. This authentication allowed people to locate others who were signed in, since a unique name was assigned, allowing others to identify an individual. As new clients appeared by various companies, it was not in the provider's interest to allow another company access to their users. With millions of users at stake, there was no incentive for the network's company to provide marketing and other opportunities to allow other companies to profit off of their work in building out a network. Also, if multi-network clients were able to connect to a network, they had the opportunity to steal users away to another network, effectively erasing the work spent building a user base and canceling any possibilities for profiting off of that community. Interoperability and security concerns have been cited by these companies as reasons for restricting access to third-party clients.

AOL Instant Messenger (AIM)

Solutions in this chapter:

- AIM Architecture
- AIM Protocol
- AIM Features and Security Information
- Malicious Code and Client Security

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

America Online (AOL) has been offering two nearly identical clients for instant messaging, Buddy List and AIM. What separates these clients is that the Buddy List software is exclusive to AOL subscribers and is integrated into AOL's proprietary access software. This Buddy list software was the first instant messaging client produced by AOL in the mid 1990's and was available to subscribers of AOL only, providing a way for members of the AOL service to communicate with each other. In 1996, Mirabilis, an Israeli-based company, released an instant messaging client that was freely available and compatible with most workstations and Internet Service Providers (ISPs). This software, ICQ, was met with immediate success, and after only 6 months of availability had registered 850,000 users. AOL, noticing the rapid increase in popularity of ICQ and the increased interest in instant messaging, released their own client to compete with this freely available software. AIM was released for multiple platforms as a competitor to ICQ and was released to be available to all Internet users, regardless of the ISP they were using. In 1998, AOL acquired Mirabilis, and ran the ICQ service separately from AIM. In late 2002, AOL began testing interoperability of AIM and ICQ, enabling users of both services to communicate with each other and share contact lists between services.

AIM Architecture

AIM's architecture is like most instant messaging services in that it employs a client-server model for authentication to the service and for communication with other clients. Importantly, the default behavior of AIM is to send messages to an intermediate server, which routes them to the correct clients. Messages are not sent directly to another contact unless a direct connection with a particular client is requested and accepted by the recipient.

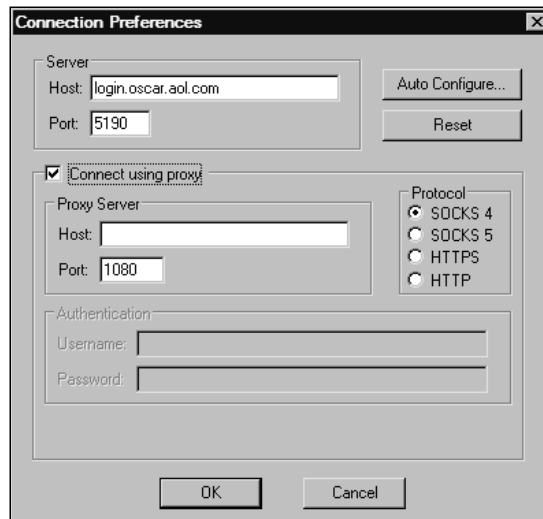
Communication on the AIM service takes place with the assistance of one of two protocols, OSCAR (which is not an acronym) or TOC (which is not an acronym either). AOL has never released specifications on the OSCAR protocol, so most of the information that is currently known about this protocol has come from users observing its behavior through packet capturing utilities and reverse engineering it. This information is now well known and has been published on various websites. Since this protocol has been reverse engineered, it is possible for third-party developers to produce clients that can access the AIM network using an implementation of the OSCAR protocol. The process for reverse engineering the AIM protocol is not very difficult since the client does not encrypt its information, allowing the packets to be analyzed fairly easily.

AOL is very protective of the OSCAR protocol, and will make changes to the way OSCAR communicates and authenticates with the AIM servers in order to prevent third-party clients from successfully connecting to the network. AOL has made several changes to prevent these clients from connecting, and this begins a cat-and-mouse game in which the developers of these clients need to reverse engineer the protocol again and make changes to their client in order to maintain connectivity. A second protocol AOL released in order to connect to the AIM service, the TOC protocol, was released in 1999 under the GNU public license. This protocol is a subset of the OSCAR protocol and was released to be used primarily by third parties interested in developing instant messaging clients on operating systems and platforms that were not supported by AOL. This protocol is not very robust, and is missing many of the advanced features expected of instant messaging clients, such as the ability to transfer files and display buddy icons. Although some third party clients use this protocol, developers looking for full featured communication options with the AIM service need to rely on a reverse engineered version of OSCAR. AOL currently uses this protocol for its Web-based client, AIM Express, which is available at <http://toc.oscar.aol.com/>. This client will be discussed later in this book along with Microsoft's Web-based client.

The process for gaining access to the AIM service begins when a user enters his or her username and password and signs into the service. This information is part of the first FLAP (which is not an acronym) packets sent to the AIM service. FLAP is a proprietary protocol that AIM uses for communication and will be discussed later in this section. The login information is weakly encrypted using Exclusive-OR (XOR) encryption to protect this information, and as previously discussed, utilities such as *dsniff* can be used to decrypt these passwords while they are being transmitted over a network. More information on XOR encryption can be found at <http://cprogramming.com/tutorial/xor.html>. Figure 2.1 shows the main screen for signing into the AIM service, while figure 2.xxx shows *dsniff* revealing AIM passwords.

Figure 2.1 Signing Into the AIM Service

Communications between the client and server for this login process are handled over port 5190 by default. This port number can be easily changed via the **Connection Preferences** options on AIM. Options for changing AIM sign-in settings are not limited to the port number, but are also available for configuring proxy servers. This allows AIM to bypass a network's protections and allow AIM to communicate with its login servers even when it is prohibited and steps are taken to prevent it. There are many proxy servers freely available for use, so more advanced users may be able to bypass restrictions on instant messaging in a particular environment. Figure 2.2 shows the **Connection Preferences** screen used for changing connection settings.

Figure 2.2 AIM Connection Preferences

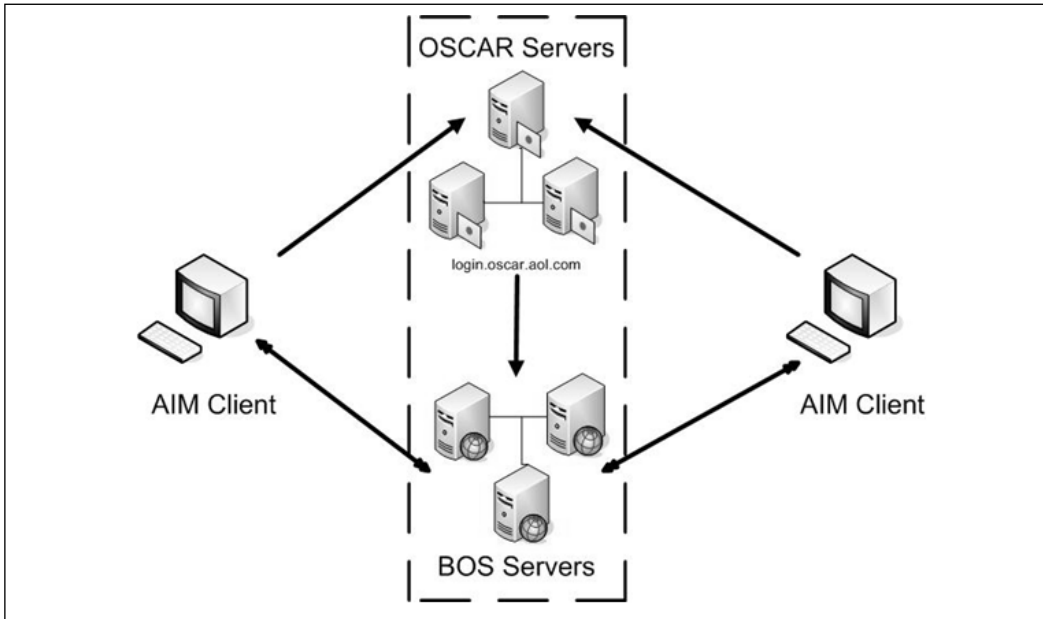
The following proxies are supported by AIM:

- SOCKS 4
- SOCKS 5
- Hypertext Transfer Protocol (HTTP)
- HTTP over Secure Sockets Layer (HTTPS)

Assuming the user is able to communicate either directly or through a proxy, the login information is sent to an OSCAR server, which is located at login.oscar.aol.com. There are many OSCAR servers available to accept client connections, which are necessary considering the amount of users simultaneously signing into and using the AIM service. These servers verify that the login information is correct and pass the client a cookie and an Internet protocol (IP) address for a Basic OSCAR Service (BOS) server as well as the user's Buddy List. BOS server access is necessary to any user of the AIM service. One of the major features of these servers is to provide lookup information used for routing messages. By default, a client does not have any knowledge of the IP address of the recipient of messages and uses the BOS servers for looking up the correct client to receive the message. There are some AIM services that require a client to request or reveal an IP address, usually for services that require a direct connection to another client such as file transfers. For these transactions, the BOS servers negotiate on behalf of the clients and reveal the IP addresses of each client in order to begin transmitting data. These cookies for BOS server access are temporary, and become

invalid as soon as a user logs off the AIM service. The only way a user can reconnect to a BOS server is by signing into the AIM service again and receiving another cookie. Figure 2.3 shows the architecture of the AIM network.

Figure 2.3 AIM Architecture



AIM Protocol

All communication between clients and servers uses the FLAP protocol. This is a low-level protocol and sits above the Transmission Control Protocol/Internet Protocol (TCP/IP). This protocol must be followed exactly, and any third-party client that does not implement this protocol correctly will be disconnected. FLAP headers make up the first 6 bytes of all AIM messages and use a feature called *channels*, which allows several types of information to be transmitted at once. The header consists of the following information:

Offset	Function
0	Command Start
1	Frame Type (0x01 - 0x05)
2	Sequence Number
4	Data Length

- The **Start** command identifies the beginning of the FLAP packet.
- **Frame Type** denotes which channel is used for the information that is being transmitted. The available channels are as follows:

0x01	New Connection/Login
0x02	SNAC Data
0x03	Error
0x04	Signoff
0x05	Keep Alive

- The first channel is used to denote the beginning of a new connection.
- Channel two contains the SNAC data, which is where most communication occurs, such as messaging.
- The remaining channels communicate the status of the client and server communication, where errors or disconnections are reported.
- **Sequence Number** denotes information that is used to ensure that packets are received in the correct order. Although this feature is redundant (since TCP contains sequence numbering itself), it is still necessary to implement in third-party clients. Sequence numbers are picked randomly, and for each command sent, the sequence numbers will increment upwards, until reaching 0xFFFF (at which point it wraps to 0x0000). Sequences for client to server and server to client communication are not related, but follow the same rules for incrementing. If this sequence is out of order, the client will be disconnected.
- **Data Length** information is populated with the amount of data being sent for the particular series of packets.

More details on the FLAP specification and SNAC data can be found at <http://iserverd1.khstu.ru/oscar/basic.html>.

AIM Features and Security Information

AIM has many features that are useful for different aspects of communication. However, many of these features can lead to a leakage of sensitive information and opportunities for a malicious user to obtain confidential information through social engineering. Additionally, the client itself has been vulnerable to a number of security exploits, and recent worms and other malicious code can be delivered through links displayed in the messaging window.

Instant Messaging

Instant messaging is the most basic function of instant messaging clients, with information being exchanged between two users. Since these users are not face-to-face, there are several security issues that can arise from using this feature. First, a malicious user can attempt to persuade another user through social engineering to disclose user accounts, passwords, or other sensitive information. Another security issue arises due to the unencrypted nature of this communication. Since this information moves between users in plain text, a malicious user can install packet capturing software and record the entire conversation, gaining a good amount of information without resorting to social engineering. Figure 2.4 shows the AIM messaging window used to communicate with other users. Since AIM has the ability to use any port for signing into the service and using instant messaging, it is nearly impossible to block this function by blocking ports on a gateway security device. Blocking login.oscar.aol.com on all ports will not only prevent users from instant messaging, but will also prevent users from signing into the service. This, however, does not stop users from connecting to a proxy server to access the AIM service.

Figure 2.4 AIM Messaging Window

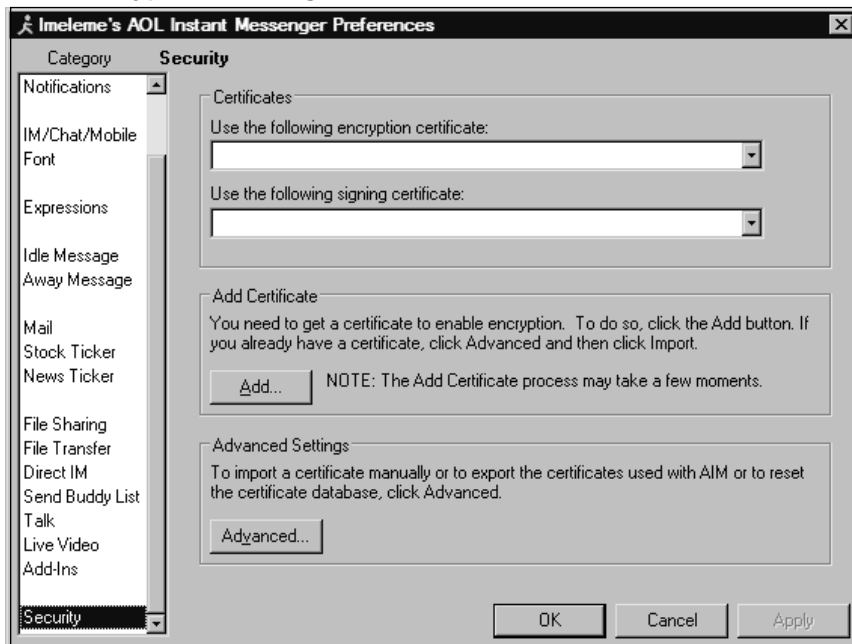


Encryption

Encryption is necessary to prevent packet monitoring utilities from gaining access to all information that users of instant messaging services exchange. AIM is the only instant messaging client (besides third-party applications) that provides the ability to encrypt communications, preventing malicious users from spying on the little matzu

conversations. AIM uses a different encryption method than Trillian, which is discussed later in this book. AIM relies on certificates for encrypting instant messaging traffic. There is some administration involved in issuing and revoking certificates for users. Although certificates are an important feature that should be used to prevent data leakage, there are better solutions that are easier and more cost-efficient. Figure 2.5 details the security settings for AIM, with options for using certificates for encryption.

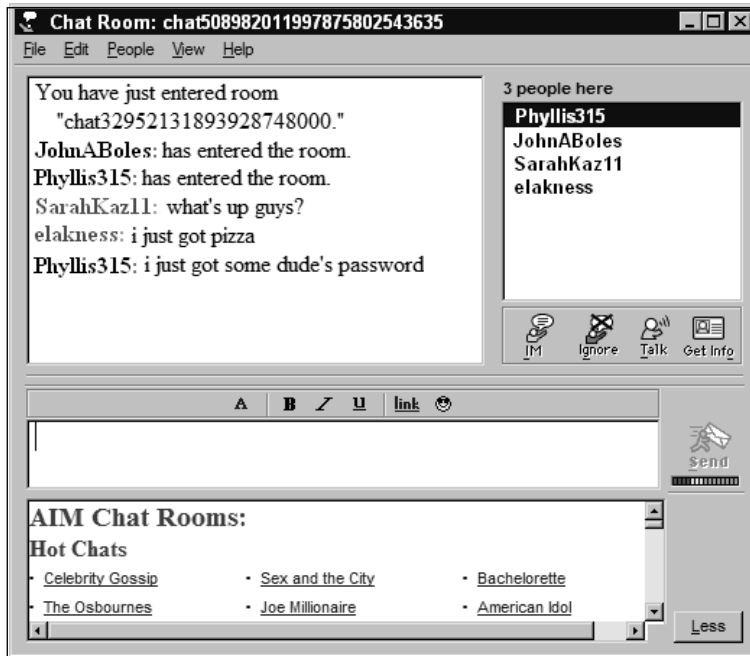
Figure 2.5 Encryption Settings in AIM



Group Chat

Group chat initiates a chat room where multiple users can communicate with each other at the same time. The identical security issues that affect instant messaging also apply to group chat, which communicates over several ports, including port 80, which cannot be blocked without restricting all Web access. This makes it necessary to block access to the entire AIM service in order to prevent group chat sessions. Figure 2.6 shows a chat room window. An invitation must be accepted before a user is connected to a chat room.

Figure 2.6 Private Chat Room



Audio Chat

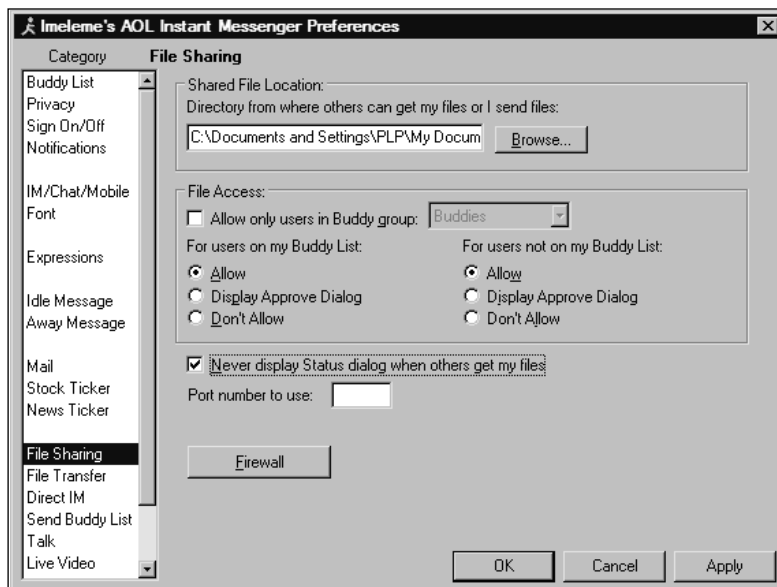
Audio chat provides the capability for two users to communicate via voice as long as their machines are equipped with microphones and speakers. This functionality has similar security risks as instant messaging. By bypassing telephone systems and e-mail systems, users may communicate sensitive information to others. Like many features of instant messaging clients, it is difficult to authenticate the user you are communicating with, and could possibly be entering into exchanges with unknown parties or users posing as acquaintances. Figure 2.7 shows a voice chat session initiating. A user initiating the voice chat sends a request to begin Audio Chat. The recipient must accept this invitation for Audio Chat to begin. This feature can not be activated without the knowledge of both parties.

Figure 2.7 Audio Chat

File Transfer

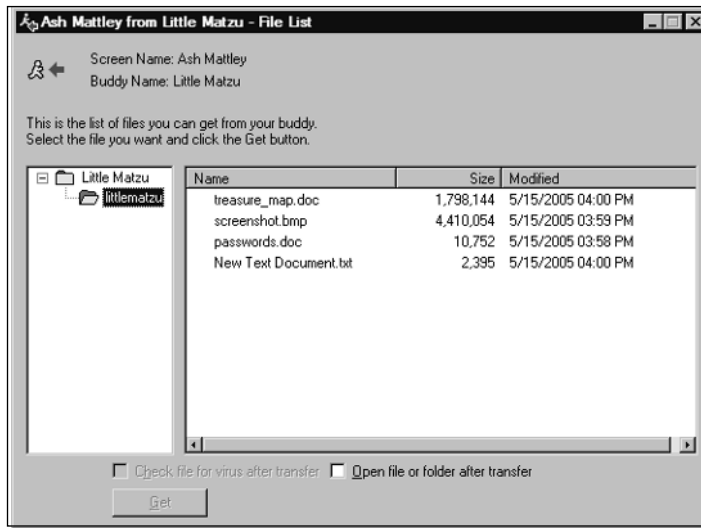
The file transfer feature in AIM allows users to send files or directories to remote users. This allows unregulated file transfers outside of an organization, and has the ability to bypass file and size restrictions of e-mail systems. Additionally, security measures that prevent File Transfer Protocol (FTP) or other file transfers via various protocols are circumvented by using this feature within AIM. Files and directories sent through AIM do not have a file size restriction, allowing for large files, including sensitive documents and copyrighted material such as MP3s and movies to be transferred between users, whether or not other security measures are in place. This feature can be configured to automatically transfer files without notification, and can be configured within AIM to use any port, making this feature available to all users who are able to log into AIM. Figure 2.8 details the file transfer options that can be set within AIM.

Figure 2.8 File Transfer Options



File Share

The file-sharing feature of AIM is similar to the file transfer feature, but it is configured to share an entire directory on a workstation. Multiple options are available, allowing for notification and status of transfers and permissions to enable contact list members access while blocking unknown users. It is important to note that this feature can be configured to allow anyone on the AIM service to have access to the files on a workstation, increasing the security risk with regards to data leakage. This feature can be misconfigured to share the wrong directory on a workstation, making the security implications even worse. This feature can be configured to use any port, making this service available to all users that log into the AIM service. Figure 2.9 is an example of an AIM user browsing for files on a remote workstation.

Figure 2.9 Browsing Files on a Remote Workstation

Malicious Code and Client Security

This section deals with malicious code, such as worms and viruses, as well as security vulnerabilities found in the AIM client. There has been a rapid increase in worms and other malicious code released to take advantage of instant messaging clients. IMlogic, which provides several products to filter, block, and manage instant messaging traffic, maintains a comprehensive list of security threats for instant messaging clients and IRC networks. According to the IMlogic Threat Center (http://imlogic.com/im_threat_center/index.asp), there were 42 security threats related to malicious code that was identified, from January 1, 2004 to July 1, 2004. In 2005, there was an alarming increase in security threats, with 674 recorded from January 1, 2005 to May 1, 2005. This increase is due to the rising adoption of instant messaging clients, which is seen by malicious users as a new vector of attack that can be exploited for the spreading of malicious code.

Of these security threats, AIM had 41 in 2005, of which seven were considered medium severity while the remainder was categorized as low severity. One of the more serious threats to AIM began spreading in January 2005 and is known as WORM_WOOTBOT.GX. This is a worm that when installed connects to an IRC channel and listens for commands from a malicious user. A malicious user can take control of the infected machine and issue commands through the IRC channel.

Most threats for instant messaging clients are worms that distribute Uniform Resource Locators (URLs) to other clients, and require interaction from the user in

order to continue spreading. The other medium severity threat for AIM in 2005, W32/Worm.Aimdes.A, is a worm that spreads by sending information to users that are listed in the instant messenger's contact list. This worm sends a copy of itself to everyone on this list, along with a message to trick users into opening the file. Since these messages appear to come from a trusted source, it is more likely to be opened, which in turn allows the worm to continue propagating. These worms can also contain backdoors and connect to IRC channels for communication. Although the distribution of these worms and the necessity for user interaction makes it seem like a trivial matter, these worms can provide malicious users with large amounts of sensitive data and even remote control of a remote machine.

The Table 2.1 lists all the security threats affecting AIM from January 1, 2005 to May 1, 2005:

Table 2.1 AIM Security Threats

Name	Severity	Date
Troj/LdPinch-JD	Low	January 23, 2005
WORM_WOOTBOT.GX	Low	January 31, 2005
W32/Worm.Aimdes.A	Med	February 10, 2005
Troj/LdPinch-AV	Low	April 19, 2005
Trojan.Aiminfo	Low	April 21, 2005
W32.Velkbot.A	Low	April 23, 2005
W32.Gabloliz.A	Low	April 25, 2005
Gabby-A	Med	April 26, 2005
W32.Allim.A	Low	April 26, 2005
W32.Allim!gen	Low	April 27, 2005
W32.Allim.B	Low	April 27, 2005
Backdoor.Doyorg	Med	May 1, 2005
W32/Oscarbot	Low	May 1, 2005
WORM_OPANKI.F	Low	May 6, 2005
Troj/BudInk-A	Low	May 10, 2005
W32/Oscabot-B	Low	May 10, 2005
W32.lmspread.Worm	Low	May 10, 2005
W32/Opanki-C	Low	May 12, 2005
W32/Opanki-A	Low	May 12, 2005
WORM_OPANKI.G	Low	May 12, 2005

Continued

Table 2.1 continued AIM Security Threats

Name	Severity	Date
W32/Oscabot-C	Low	May 15, 2005
W32/Oscabot-D	Low	May 16, 2005
W32/Oscabot-E	Med	May 16, 2005
W32.Opanki	Low	May 18, 2005
W32/Opanki-I	Low	May 18, 2005
W32/Oscabot-F	Low	May 18, 2005
W32/Oscabot-G	Low	May 19, 2005
W32/Oscabot-H	Low	May 21, 2005
WORM/FUNNY .MOVIE.AOL	Med	May 24, 2005
W32/Rbot-ADN	Low	May 27, 2005
W32.Pinkton.A	Low	May 31, 2005
W32/Oscabot-I	Low	May 31, 2005
Gpic.aol	Med	June 1, 2005
W32/Rbot-AEO	Low	June 3, 2005
OPANKI.APIF.HEY	Med	June 15, 2005
W32/Sdbot-ZJ	Low	June 15, 2005
W32/Oscabot-J	Low	June 20, 2005
W32/Opanki-E	Low	June 21, 2005
Trojan.Aimsend	Low	June 22, 2005
Oscarbot.AY	Low	June 28, 2005
Oscarbot.AY	Low	June 29, 2005

The sections to follow are prime examples of worms executable through AIM.

AIMDES

The AIMDES worm was discovered on February 10, 2005 and spreads via AIM and e-mail. Symantec's Research Center (<http://sarc.com>) describes it as the following:

When W32.Aimdes.A@mm is executed, it:

1. Creates the following files:
C:\Windows\Msvbdl.pif
C:\Program Files\Sony\VAIO Action Setup\MsVBdll32.exe
%Windir%\msVBdll.exe
%UserProfile%\Start Menu\Programs\Startup\msVBdll.exe
2. Adds the value “MsVBdll” = “%Windir%\MsVBdll.pif” to the HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run and HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run registry keys so the worm is executed every time Windows starts.
3. In an attempt to disable notification of firewall, antivirus, and update status through the Windows Security Center, W32.Aimdes.A@mm adds the following registry entries to the HKEY_CURRENT_USER\Software\Microsoft\security center and HKEY_LOCAL_MACHINE\Software\Microsoft\security center registry keys:
“FirewallDisableNotify” = “1”
“UpdatesDisableNotify” = “1”
“AntiVirusDisableNotify” = “1”
4. Adds the following registry entries to the HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System registry key to disable access to the Windows Task Manager and registry editing tools:
“DisableTaskMgr” = “1”
“DisableRegistryTools” = “1”
5. Adds the registry entry “NoAutoUpdate” = “1” to the HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Windows Update\AU to disable Windows Update.
6. Deletes the HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\CurrentVersion\Run\ “Windows” = “Auto Update.exe” entry if present.

7. Displays one of the following dialog boxes:

Title: "Blow Me"

Text: "Hello Windows has suffered from a serious error, it may never recover unless you perform oral sec on the cd drive"

or

Title: "Disgusting"

Text: "You are viewing this message because someone in the house is homosexual"

8. Sends keystrokes to open AOL Instant Messenger windows causing the following message to be sent:

Message:

"Hey whats up!! look what I did to my hair...lol!!"

Attachment:

C:\Windows\picture.pif.

The worm does not appear to create this file.

9. May attempt to download a file from the domain geocities.com/vip_asshole and save it as C:\Fix_SP2.zip.
10. May send itself by e-mail to all addresses in the Outlook Address book. The **From** address is not spoofed and the e-mail has the following attributes:

Subject: Service Pack 2 BUG!!

Message body:

Dear user I have been informed that there was a BUG in Windows Service Pack 2 which was fixed I recommend you to download this Patch version which will fix the bug and keep your system safe.

You will find the Patch file in the attachment, feel free to send it to anyone.

I'll be in touch with you as soon as another bug is found.

Regards,

A.H

11. It attaches the file C:\Fix_SP2.zip.

NOTE

After the e-mails are sent, they are deleted from Sent Items folder and the file C:\Fix_SP2.zip is deleted.

12. Terminates the svchost.exe and lsass.exe processes.
13. Puts the PC to sleep and attempts to copy itself to A:\homework.exe.
14. If it cannot access the A: drive, the following error is displayed and the worm terminates:
“Run-time error ‘71’: Disk not ready”
15. Terminates if the disk in the A: drive ceases to be available.

NOTE

In testing, while the A: drive was available, the worm went into an infinite loop of putting the PC to sleep and attempting to copy itself to the A: drive.

Oscarbot/Opanki

Oscarbot, also known as Opanki, was discovered on May 1, 2005. According to McAfee’s AVERT group (<http://vil.nai.com/vil/default.asp>), the worm has the characteristics described below.

This threat spreads via a hyperlink that is received via AOL Instant Messenger. Recipients may receive a message such as the following:

hey check out *this*
hehe :) i found *this* funny movie

When a user clicks on the hyperlinks, he/she will be prompted to save/run an executable file (such as pictures@gallery.com). If users choose to download and/or run this file, it will contact a remote IRC server, log on to a specified channel and wait for further instructions. One of these instructions can result in the bot program sending the aforementioned hyperlink to all recipients on the infected users buddy

list. Technically not a worm, this threat requires a bot commander to initiate the *spimming* (IM spam) routine.

This threat copies itself to the WINDOWS (%WinDir%) directory as svchost.exe (note that a valid svchost.exe file exists in the WINDOWS SYSTEM directory). The shell is hooked via the registry to ensure the threat is run at system startup. The shell entry is HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon “Shell” = Explorer.exe C:\WINDOWS\svchost.exe.

The bot will attempt to connect to a remote IRC server, such as d205.yi.org or ftpd.there3d.com.

Velkbot

Velkbot was discovered on April 28, 2005 and affects AIM, Yahoo! Messenger, and MSN Messenger. Trend Micro (www.trendmicro.com/vinfo/virusencyclo/) details the following information on this backdoor Trojan:

Upon execution, this backdoor drops a copy of itself in the Windows system folder as MSKEV.EXE.

It then adds the following registry entries to ensure its automatic execution at every Windows startup:

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run Windows Kev Messenger = “mskev.exe”

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices Windows Kev Messenger = “mskev.exe”

It also disables the Windows Task Manager and Registry Tools by adding the following registry entries:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System DisableTaskMgr = dword:75000031

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System DisableRegistryTools = dword:75000031

Below are the other registry entries added by the backdoor:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa Windows kev Messenger = “mskev.exe”

HKEY_LOCAL_MACHINE\Software\Microsoft\Ole Windows key
Messenger = “mskev.exe”

Backdoor Capabilities

Once Velkbot is installed on a system, it connects to a certain IRC server and joins a specific channel using a random nickname. It monitors and responds to the private messages, which usually come from a malicious user employing specific keyword triggers.

It is capable of the following operations:

- Downloading file(s)
- Remotely executing file(s)
- Updating itself

Client Security

To defend against these types of worms, it is necessary to restrict access to the AIM service. Since these worms are spread as instant messages, there is no way to block a specific port to restrict that functionality. Block login.oscar.aim.com to restrict access to the entire AIM service and all functionality. This will not, however, stop more advanced users from setting up or connecting to a proxy server in order to access the AIM service.

The AIM client is in a constant state of development, with frequent releases occurring many times throughout the year. In 2004 alone, AIM had nine releases, which included several beta clients. Some of these releases were due to security issues found in the client that required updates to patch vulnerabilities. Internet Security Systems' X-Force Database is the largest security database, cataloging over 13,000 threats and vulnerabilities (available at <http://xforce.iss.net/xforce/search.php>). According to this source, AIM has been affected by 10 denial of service (DoS) attacks and 11 buffer overflows, which may allow a remote attacker to execute arbitrary code on an affected workstation. There is a software vulnerability in version 5.9.3797 of the client, which has not been remedied at the time of this writing in July 2005. This vulnerability, AOL AIM ateimg32.dll denial of service, was reported on June 7, 2005 and is described by the ISS X-Force as the following:

Description:

America Online AOL Instant Messenger (AIM) is a program that Internet users can use to chat and exchange files and images. AIM version 5.9.3797 when running on Microsoft Windows is vulnerable to a denial of service attack in the GIF parser in the `ateimg32.dll`. A remote attacker could supply a specially-crafted .gif argument file in Trillian and then message another user that uses AIM to cause AIM to crash.

Platforms Affected:

- AOL/Time Warner: AOL Instant Messenger 5.9.3797
- Microsoft Corporation: Windows 98
- Microsoft Corporation: Windows 98 Second Edition
- Microsoft Corporation: Windows Me
- Microsoft Corporation: Windows XP
- Microsoft Corporation: Windows 2000 Any version
- Microsoft Corporation: Windows 2003 Any version

Remedy

No remedy available as of June 2005.

Consequences:

Denial of Service

References:

- AOL Instant Messenger website, Download AIM for Windows at http://www.aim.com/get_aim/win/latest_win.adp?aolp=.
- Full Disclosure Mailing List, Tue Jun 07 2005 - 02:38:21 CDT , AOL AIM Instant Messenger Buddy Icon `ateimg32.dll` DoS at <http://archives.neohapsis.com/archives/fulldisclosure/2005-06/0061.html>.

This highlights the need to centrally manage instant messaging software if it is allowed on a large network. Vulnerabilities and exploits may be present in the software, and may need to be updated rapidly as security issues are disclosed and new versions of the client become available. AOL hosts a website dedicated to providing information on AIM security issues, located at http://aim.com/help_faq/security/faq.adp?aolp=.

The best prevention against vulnerable software is to remove it from a workstation if it is not critical. If this is not an option, patches or new executables may be provided by the software publisher that will eliminate the risk of running the software. This is one of the reasons the AIM client is updated frequently. Not only are there new features that are added, but AOL will also release these clients to fix security vulnerabilities that are reported throughout the year.

Summary

There are multiple types of security issues affecting AIM, and they fall into the categories of data leakage, social engineering, malicious code, and client vulnerabilities. Data leakage involves the distribution of sensitive information to others outside of an organization. Unlike e-mail, instant messaging software does not have the capability to track the transfer of files. Confidential files can be shared on a workstation, allowing any user of the AIM service to access and transfer any of these files.

Additionally, these file transfers do not have a size limit similar to e-mail attachments at most organizations. Large files such as CD images, movies, and MP3s may be transferred in and out of a network without the knowledge of an administrator.

Social engineering is another security issue that faces users of instant messaging software. Since you cannot reliably identify who is at the other end of a conversation, you can only go by their user name registered on the system. Names that are slightly different from an acquaintance or appear to be from a certain company may be able to obtain password, credit card, or other information. In order to prevent this from happening, it is recommended that no confidential information be divulged over an instant messaging session, especially if the session is not encrypted. It is also recommended that you do not save your password on the AIM client. This can prevent unauthorized use of your client, so a malicious user who has access to your workstation may not sign into the AIM network and masquerade as you.

Perhaps the greatest danger of instant messaging clients is the unregulated spread of malicious code. Instant messaging clients are able to bypass firewalls, URL filtering devices, or other safeguards to protect and secure a network. Generally, worms and other malicious code are spread through instant messaging clients via URLs contained in messages. Since they appear to come from a trusted source, there is a higher probability that a user will click on the link, infecting their system and further spreading this code to other users.

Solutions Fast Track

AIM Architecture

- ☑ AIM uses a client-server model for its architecture, and by default all messages and communications are sent through an intermediate server then routed to the intended user.

- ☑ The default port AIM uses to connect to the server is 5190, which can be changed to any available port.
- ☑ AIM can be connected via several different proxies, which can evade security controls put in place to prevent the client from connecting. AIM supports SOCKS 4, SOCKS 5, HTTP, and HTTPS proxies.
- ☑ There are a number of free AIM proxies available throughout the Internet, which will allow any user with this knowledge to bypass restrictions against instant messaging by utilizing these services.

AIM Protocol

- ☑ The FLAP protocol sits above TCP/IP and contains information regarding communication as well as the message itself.
- ☑ The FLAP protocol also transmits multiple commands to the client, including the message data, error messages, and whether or not the client should disconnect from the service.
- ☑ Sequence numbers are used in FLAP to ensure that information is received in the correct order. If the sequence of data is not correct, the client will be disconnected.

Features and Security Information

- ☑ Many AIM features are useful for communicating rapidly. These communications can bypass current systems that may track usage. This lack of control can lead to data leakage, where sensitive information is disseminated outside of an organization. Additionally, this may lead to a situation where there are no records that can be used for forensics.
- ☑ File transfers can be used to bypass size constraints on email systems as well as bypass FTP restrictions. This may result in the transmission of sensitive information, specifically large files that may contain confidential data.
- ☑ Text messages between clients are not encrypted by default. This may allow a malicious user, with a packet capturing utility, to recreate a conversation.

Malicious Code and Client Security

- ☑ Malicious code such as worms are generally distributed through instant messaging services via URLs. Some worms make these messages appear as though these URLs are coming from a known user, increasing the chance that these URLs will be visited, thereby infecting a workstation.
- ☑ Some worms have the ability to receive remote commands to control the infected workstation. This gives a malicious user complete control over the workstation, including access to all information stored on the workstation.
- ☑ Vulnerabilities within the AIM client are fixed via a new client, which needs to be installed. Users may not be aware of the availability of a new, more secure version. There is a risk that these users have software installed which contains one or several security vulnerabilities.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: How can I tell if my version of AIM has security issues?

A: AOL times many releases of AIM to remedy security issues in the client. You should always ensure you have the latest version of AIM to protect your workstation against vulnerabilities. The latest versions of AIM include a feature that automatically check for updates to the client and download any that are available.

Q: What is the best way to prevent worms from spreading?

A: The easiest way to stop a worm is not to click on the URL containing the malware. If a URL looks suspicious, it should not be clicked on, especially if it is coming from an unknown user. Additionally, any file with a .pif, .exe, or other extension that does not lead to a Web page should always be avoided. Remember that websites may take advantage of system vulnerabilities and install software without knowledge or consent. Another way to prevent

worms or other malware from spreading is to change the Privacy settings on AIM to restrict messages unless they are from known users on your contact list or specific users that are known. This will prevent unwanted messages from unknown users, which is usually related to SPIM (SPAM on instant messaging) and worms.

Q: Is there a way to restrict some features of AIM while allowing messaging?

A: Most services of AIM operate on a large range of ports, making it difficult to restrict specific ports and functionality. For example, direct connection for file transfers uses ports 1024–5000, making it difficult to maintain controls for all services offered through the AIM client.

Q: Is there a good way to block AIM on a network?

A: There are several options for blocking AIM traffic. There are dedicated gateway devices such as IMlogic's which allow for control over instant messaging services on a network. Firewalls that provide packet filtering or stateful inspection have the ability to specifically block AIM traffic. Additionally, an administrator can block traffic to `aim.login.oscar.com` on all ports. This will prevent clients from signing into the service, even if the user has changed his or her port number from the default. This will not, however, prevent users from configuring the client to connect to the service via a proxy.

Q: Can AIM share files securely?

A: AIM has many controls that will allow you to share files in a more secure manner. You can restrict access to certain folders, but you need to make sure that no sensitive information is available in these areas. Another way to restrict access is to configure the access to this shared folder. AIM provides controls to specify which groups of contacts may have access to the files in this folder, including everyone on your contact list or contacts that are listed in a specific group, which can be defined through the contact list options. Additional settings are recommended that configure the client to display a dialog box to approve of a user accessing data in your shared folders.

Yahoo! Messenger

Solutions in this chapter:

- Yahoo! Messenger Architecture
- Yahoo! Messenger Protocol
- Features and Security Information
- Malicious Code and Client Security

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Soon after ICQ and AIM (AOL Instant Messenger) gained popularity and amassed millions of users, Yahoo! built its own proprietary instant messaging service. The client that Yahoo! released, Yahoo! Pager, had the ability to connect to the AIM network. AOL, both with its AIM and ICQ services, had locked up the majority of instant messaging users. Since AIM and ICQ were operating on closed networks, and since they had signed up so many users, the only way someone could talk to an online contact was by either using one of AOL's clients or by using a client that was compatible with these services that allowed for communication with multiple networks. Yahoo! realized that the quickest way to build their own user base for instant messaging was to build clients with the ability to connect with the majority of online instant messaging users, while at the same time connecting these users to their own networks.

Interoperability did not last long, as AOL was quick to change its protocol in order to restrict access to its network, and after multiple fixes to the client and multiple changes to AOL's protocol, Yahoo! gave up on building multi-protocol clients, and focused on building its own service and increasing distribution of their own clients.

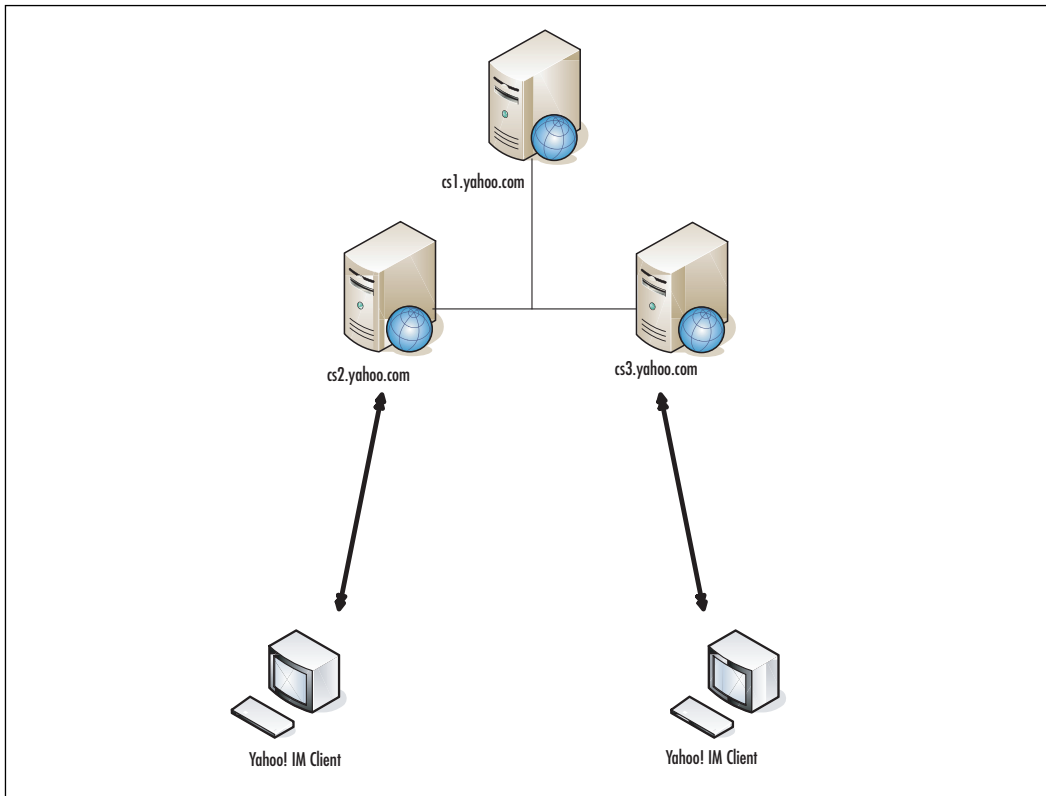
Yahoo! has also been successful in attracting users, and has been growing steadily since its early days when it was known as Yahoo! Pager. Its current version, Yahoo! Messenger, compares favorably with features of other instant messaging clients, and its ability to utilize many features of the Yahoo! Internet portal makes it a logical choice for people who are frequent users of other Yahoo! services. Figure 3.1 shows Yahoo! Messenger's main window. Notice the tabs on the bottom of the window, showing the various features of the portal available for use within the client. Many personalization features of the customized My Yahoo! portal can be accessed directly from the messenger window.

Yahoo! Messenger Architecture

The architecture of Yahoo! Messenger is based on a client server model, with each client connecting to a main server for authentication purposes. Yahoo! Messenger uses port 5050 by default to communicate with its servers, which are cs1.yahoo.com, cs2.yahoo.com, and cs3.yahoo.com. If this port is not available, Yahoo! Messenger will attempt to communicate via port 80, which is the standard port for Web communication. If this port is not available, the software will attempt to communicate to the centralized servers by using any available port. This makes Yahoo! Messenger extremely difficult to block on a network due to its ability to use any available port to authenticate and begin communication. Additionally, Yahoo! Messenger will add

an HTTP (Hypertext Transfer Protocol) header to the packets, allowing communication to evade firewalls that employ protocol analysis.

Figure 3.1 Yahoo! Messenger Architecture



Yahoo! Messenger uses the YMSG protocol for communication, which changes very rapidly. Yahoo! has been very vigilant in restricting access to its network, attempting to block all third-party clients from connecting to its service. The desire to block these clients has provided Yahoo! with the incentive to continuously refine its protocol. Another benefit for Yahoo! is that since its protocol is rapidly evolving, it is not as well documented and understood as other protocols for competing instant messaging services.

The sign-in process for Yahoo! Messenger is initiated by the user, who enters his or her Yahoo! ID and password. This information is encrypted and sent to the Yahoo servers for authentication. Yahoo uses a challenge-response method for verifying sign-in information. Both the sign-in ID and password are encrypted during this process, and if this is accepted by the server, the user is signed into the service and a

cookie is sent to the workstation. This cookie allows the user to access other functions of Yahoo such as signing into the e-mail service, checking stock prices, or other customized portal functions. The service provides the client with a list of available contacts and what their status is (offline, busy, etcetera), and delivers messages that were sent while the client was not signed into the service.

Figure 3.2 Signing Into Yahoo! Messenger



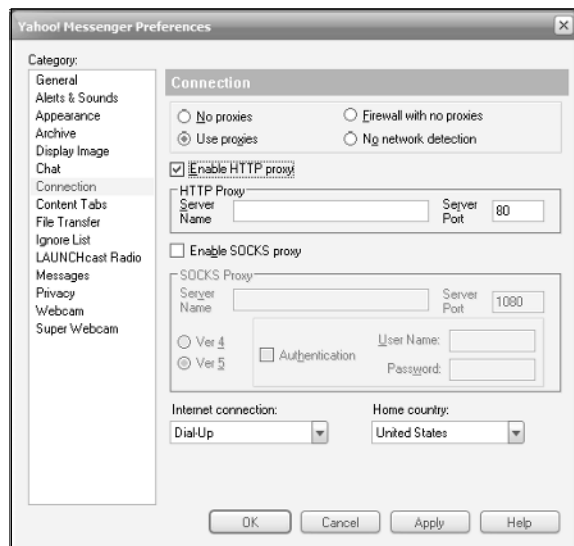
Communication between client and server takes place over port 5050, but this can be changed by the client if access is restricted. There are several settings a user can change to provide access and evade network controls such as firewalls. Yahoo! Messenger supports the use of proxies, which can be used to evade network controls such as firewalls. By configuring a proxy, a user is able to tunnel their traffic through another service, bypassing the default network for particular services. Yahoo supports the following proxies:

- SOCKS 4
- SOCKS 5
- HTTP

Table 3.1 Yahoo! Messenger Connection Details

Connection Type	Features
No proxies	This is the default setting for Yahoo! Messenger, and is used when there is no need for a proxy, if there is no firewall present, or if there are no restrictions.
Firewall with no proxies	This setting forces Yahoo! Messenger to connect using only HTTP. Several features of the client, including voice chat, are not enabled when using this setting.
Use proxies	When this is selected, the user has the option to configure one of the supported proxies to enable access to the service, including HTTP, SOCKS 4, or SOCKS 5 with authentication.
No network detection	This option ignores proxy settings and does not detect any settings on the available network.

These proxies can be configured to work on multiple ports, so the user can configure the service as necessary to gain access to the Yahoo! instant messaging network. Yahoo! Messenger also contains other connection options, which help users work around network restrictions in order to connect to the service. Figure 3.3 details the options available:

Figure 3.3 Yahoo! Messenger Connection Options

Once a user is connected to the service, he or she is able to communicate with other users by using a server as an intermediary. Just as in AIM, all messages are routed through a central server in order to locate the user and send the message to the intended recipient. Other servers may be utilized by the Yahoo! Messenger service to assist in handling specific requests. The following is a list of servers and ports for the Yahoo! Messenger service:

- **Webcam Connection:**

Protocol: TCP

Server: webcam.yahoo.com

Port: 5100

- **File Transfer Connection:**

Protocol: HTTP

Server: filetransfer.msg.yahoo.com

Port: 80

- **File Sharing Connection:**

Protocol: HTTP

Port: 80

- **Voice Chat Connection:**

Protocol: UDP or TCP

Servers:

v1.vc.scd.yahoo.com

v2.vc.scd.yahoo.com

v3.vc.scd.yahoo.com

v4.vc.scd.yahoo.com

v5.vc.scd.yahoo.com

v6.vc.scd.yahoo.com

v7.vc.scd.yahoo.com

v8.vc.scd.yahoo.com

v9.vc.scd.yahoo.com

v10.vc.scd.yahoo.com

v11.vc.scd.yahoo.com

v13.vc.sc5.yahoo.com

vc1.vip.scd.yahoo.com

Ports: 5000-5010

Yahoo! Messenger Protocol

Yahoo! Messenger uses the YMSG protocol, which is in a constant state of development. This protocol runs over TCP (Transmission Control Protocol) and HTTP. There is very little information available about this protocol, with a good source being a free library available for interfacing with Yahoo! Messenger at <http://libyahoo2.sourceforge.net/>.

YMSG packets are made up of the following information:

- **Message Start** The first four bytes are the name of the protocol, YMSG. This is always the start of a packet used for communicating with Yahoo Messenger.
- **Protocol Version** The four bytes after the message start provide the protocol version number. Some earlier versions of YMSG are not allowed on the Yahoo Messenger service. This identification keeps older clients off the service and forces users to upgrade.
- **Data Length** Denotes the size, in bytes, of the data section of the packet. This value is two bytes.
- **Service Identification** Identifies which service is being responded to or requested from the Yahoo Messenger service. See the list of services below for more detail.
- **Request Status** Denotes whether a request or response to a service has succeeded or failed. This also sends information regarding the status of the client (busy, available, typing a message).
- **Session ID** Set by the server and kept at the same value until the user logs off. This is a pseudorandom number that identifies the user.
- **Data** The contents of the message, or other data sent to another user or to the server for communications.

According to documentation on the libyahoo website, the following 45 services codes are used by Yahoo! Messenger to identify services that are being requested or responded to:

YAHOO_SERVICE_LOGON
YAHOO_SERVICE_LOGOFF
YAHOO_SERVICE_ISAWAY
YAHOO_SERVICE_ISBACK
YAHOO_SERVICE_IDLE
YAHOO_SERVICE_MESSAGE
YAHOO_SERVICE_IDACT
YAHOO_SERVICE_IDDEACT
YAHOO_SERVICE_MAILSTAT
YAHOO_SERVICE_USERSTAT
YAHOO_SERVICE_NEWMAIL
YAHOO_SERVICE_CHATINVITE
YAHOO_SERVICE_CALENDAR
YAHOO_SERVICE_NEWPERSONALMAIL
YAHOO_SERVICE_NEWCONTACT
YAHOO_SERVICE_ADDIDENT
YAHOO_SERVICE_ADDIGNORE
YAHOO_SERVICE_PING
YAHOO_SERVICE_GROUPRENAME
YAHOO_SERVICE_SYSMESSAGE
YAHOO_SERVICE_PASSTHROUGH2
YAHOO_SERVICE_CONFINVITE
YAHOO_SERVICE_CONFLOGON
YAHOO_SERVICE_CONFDECLINE
YAHOO_SERVICE_CONFLOGOFF
YAHOO_SERVICE_CONFADDINVITE

YAHOO_SERVICE_CONFMSG
YAHOO_SERVICE_CHATLOGON
YAHOO_SERVICE_CHATLOGOFF
YAHOO_SERVICE_CHATMSG
YAHOO_SERVICE_GAMELOGON
YAHOO_SERVICE_GAMELOGOFF
YAHOO_SERVICE_GAMEMSG
YAHOO_SERVICE_FILETRANSFER
YAHOO_SERVICE_VOICECHAT
YAHOO_SERVICE_NOTIFY
YAHOO_SERVICE_P2PFILEXFER
YAHOO_SERVICE_PEERTOPEER
YAHOO_SERVICE_AUTHRESP
YAHOO_SERVICE_LIST
YAHOO_SERVICE_AUTH
YAHOO_SERVICE_ADDBUDDY
YAHOO_SERVICE_REMBUDDY
YAHOO_SERVICE_IGNORECONTACT
YAHOO_SERVICE_REJECTCONTACT

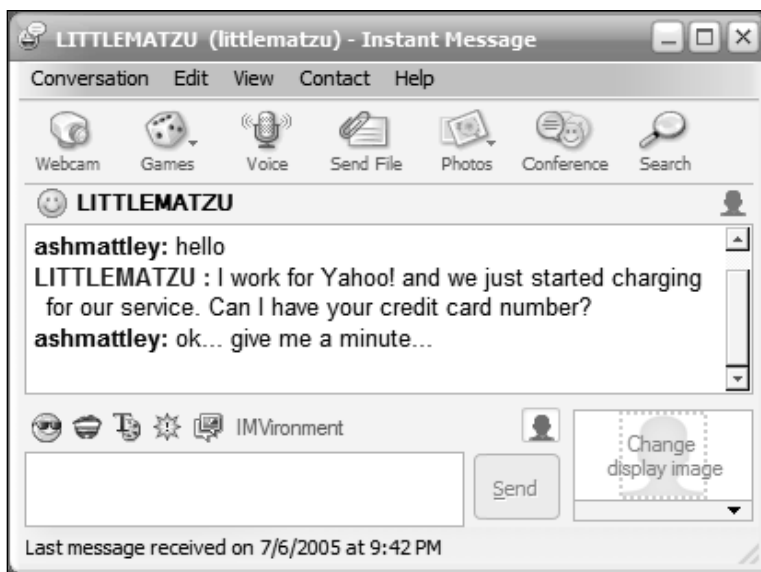
Features and Security Information

Yahoo! Messenger is a very capable communication tool, and has some very unique features due to its ability to integrate with the Yahoo! portal. Many features of the messenger client are subsets of the data found on <http://yahoo.com> and its personalized site, <http://my.yahoo.com>. By having a Yahoo! ID, a user can easily pull down personalized information into tabs within the messenger, as well as alert contacts throughout the Yahoo! site when he or she is online. However, some features such as message logging and file transfers may allow a malicious user access to confidential or sensitive information, as well as compromise a system based on any security vulnerability contained within Yahoo! messenger.

Instant Messaging

Conversations, Yahoo! Messenger's term for the instant messaging function of their service, allow users to send text messages to each other in almost real time. This feature is one of the main uses of instant messaging software, and can present some problems for users since there is no way to really authenticate who you are communicating with. By using social engineering, a malicious user has the ability to obtain information about any user. By posing as an employee of Yahoo! or by using an ID that appears to be the same as an acquaintance, a user is more likely to give out sensitive or personal information, including credit card data, addresses, or other sensitive information. Additionally, since this data between users is not encrypted, it is possible that this information can be obtained by a malicious user by utilizing a packet capturing utility. The entire conversation can be recorded through this type of utility without the knowledge of any end user. Figure 3.4 shows Yahoo! Messenger's conversation window used to communicate with other users. Yahoo! has the ability to switch to any available port for signing into the service, making it fairly difficult to block access to this function. The ability to use a proxy server also prevents an administrator from successfully blocking the service.

Figure 3.4 Yahoo! Messenger Conversation Window

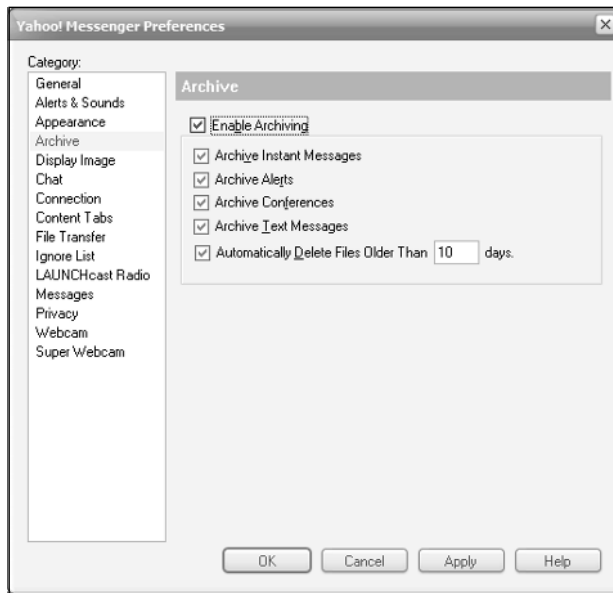


Encryption

Yahoo! Messenger does not provide any method for encrypting messages between users. This feature is necessary to prevent packet monitoring utilities from gaining access to all information that is exchanged through the use of instant messaging services.

Message Archiving

Message archiving is a very useful feature for users of instant messaging software. This feature can record any conversation automatically and save it to a file on your hard drive. Yahoo! Messenger provides this feature and enables to save all conversations, including text messages to cellular phones, group conferences, alerts, and standard messages. There is a setting to delete the archive after a definable amount of days, but this information is still stored as a local data file on your hard drive. Although this file is encrypted, it can be used by malicious users to gain access to sensitive information and conversations as long as they have access to your data. One program, known as Super Yahoo Messenger Archive Decoder (available at <http://piravi.com/>), allows any user to gain access to all logged messages on a local workstation. This software reads all messages for every Yahoo! ID that ever used that particular workstation, and does not require a password. Due to utilities like this that run locally, and the ability for a malicious user to compromise a workstation and gain access to the file, it is recommended that this feature be disabled. Figure 3.5 shows the settings that can be defined for this feature.

Figure 3.5 Message Logging in Yahoo! Messenger

Conferences

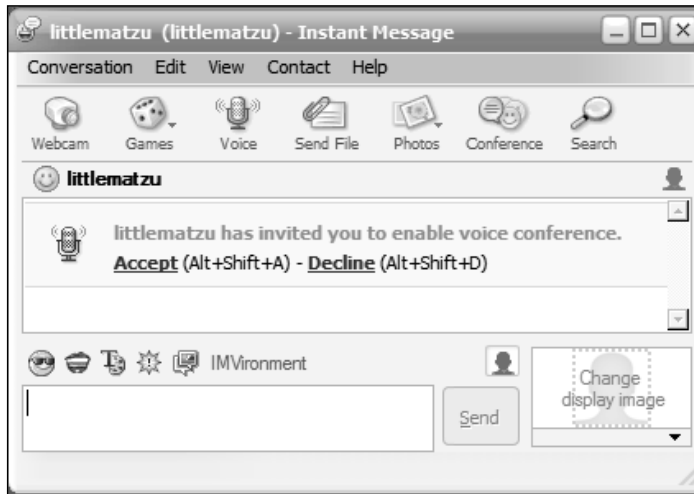
A conference is basically a chat room that is spawned on demand and includes the users that are invited to participate. Multiple users are able to join this conference, and are able to communicate at the same time. Users are not automatically entered into a conference, but must accept an invitation before joining one. Yahoo! also features the ability for all participants to use voice chat to communicate with each other at the same time. The same issues related to instant messaging also affect conferences, in that social engineering can be employed to gain access to a user's sensitive and confidential data. Figure 3.6 shows a conference window.

Figure 3.6 Yahoo! Messenger Conference

Voice Chat

Voice chat allows two users to communicate via voice as long as their machines are equipped with microphones and speakers. This functionality has similar security risks as instant messaging. By bypassing telephone systems and e-mail systems, users may communicate sensitive information to others. Like many features of instant messaging clients, it is difficult to authenticate the user you are communicating with, and could possibly be entering into exchanges with unknown parties or users posing as acquaintances. Figure 3.7 shows a voice chat session initiating.

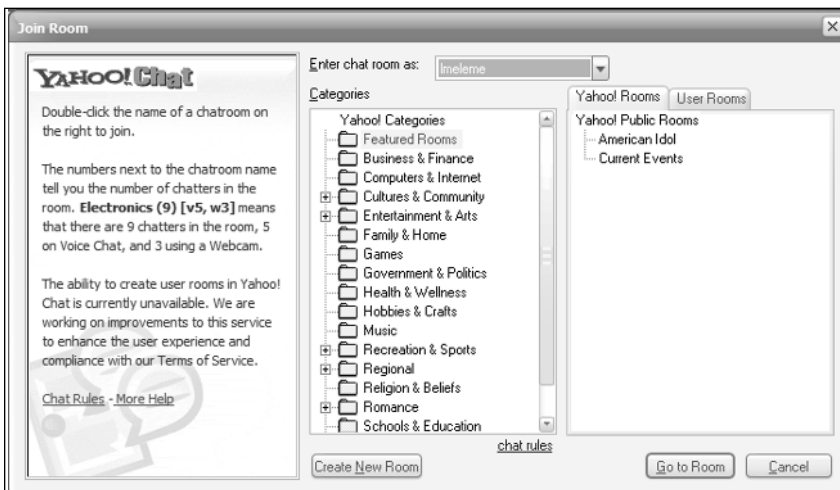
Figure 3.7 Yahoo! Messenger Voice Chat



Yahoo! Chat Rooms

Yahoo! Messenger integrates a chat client with its program, providing users with the ability to join rooms with other who are using either the dedicated Yahoo! chat client or the Messenger client. This feature makes it possible for a user to have a conversation with over 20 unknown users. This should be considered a security violation, as it is possible to release confidential information and leak data to a large audience of known and unknown users on the Internet.

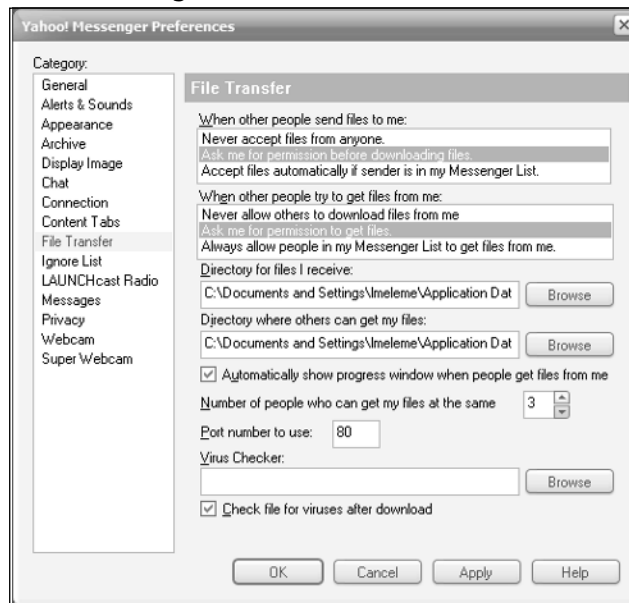
Figure 3.8 Yahoo! Chat Room



File Transfer

Yahoo! provides the capability for users to exchange files through the instant messaging service. This activity occurs over port 80 by default, and can be configured to operate over any available port, making it difficult to detect and block. This feature provides a way for users to exchange files in an unregulated fashion, often bypassing restrictions placed on file exchanges that prevent large files from being e-mailed or that prevent users from utilizing FTP (File Transfer Protocol) servers. There are no file restrictions on the feature, allowing users to exchange large files such as confidential documentation and copyrighted material such as music and movies even though security measures may be in place to prevent file sharing and P2P services. This feature can be configured to provide different levels of notification to users, including always accept from users on the contact list to never accept any file transfers. This provides some level of protection, but assumes that the person initiating a file transfer is actually the person logged in. Since there is no way of really authenticating this user, it can be a security problem even though he or she is listed as a contact. Figure 3.9 highlights the File Transfer preferences for Yahoo! Messenger.

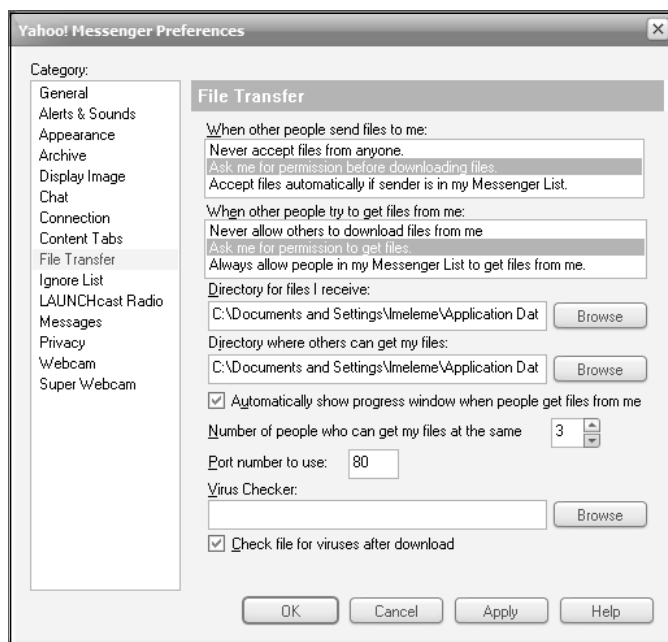
Figure 3.9 Yahoo! Messenger File Transfer Preferences



File Share

Yahoo! Messenger's file sharing feature is configured as part of the File transfer option. By default, the client sets up a directory for files to be downloaded by another client at C:\Documents and Settings\ppiccard\Application Data\Yahoo! Messenger*user-name*\shared. This directory can be changed, and restrictions can be placed on who has access to these files. Settings determine whether all users have access, no users have access, or only known contacts have access to the files. See figure 3.10 for configuration options. Options also include how many users can concurrently download files and what port number to use, which has a default setting of 80.

Figure 3.10 Yahoo! Configuration Options

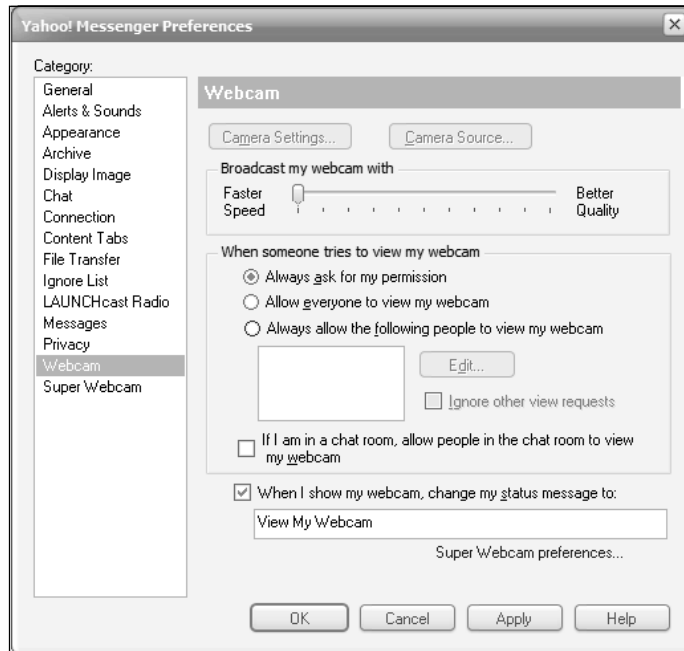


Web Camera Settings

Yahoo! Messenger provides extensive support for Web cameras, which allow users to share images of themselves and their surroundings with others. Like many of the security issues with instant messaging software, this feature provides an unregulated communications device, and allows for users to release proprietary or confidential information to other users without a network administrator's knowledge. This feature can also be configured to allow all users access to the Web camera, so anyone

who is able to send a message to a client may also be able to view and control the Web camera as well. Yahoo! Messenger uses ports 5000–5010 for this service, and it is recommended that these ports be disabled to restrict access to this feature. Figure 3.11 shows the options available for setting up a Web camera within the application. Super Webcam mode provides users with higher resolution images relying on broadband settings for both users in order to establish the enhanced service.

Figure 3.11 Web Camera Settings in Yahoo! Messenger



Yahoo! Messenger

Malicious Code and Client Security

Yahoo! Messenger has not been the target of many malicious code attacks compared to other instant messaging clients such as AIM and MSN Messenger. According to the IMLogic Threat Center, Yahoo! Messenger had sixteen security threats between January 1, 2005 and June 1, 2005, of which only one was medium severity. In 2004, there were only four security threats against Yahoo! Messenger, which is a large increase and shows that there is an increased interest among malicious users of instant messaging as a vector of attack. Instant messaging clients are an efficient medium for distribution of information due to its ability to send information rapidly and the likelihood that a recipient of an instant message will assume that a message is coming from a trusted source and click on a URL (Uniform Resource Locator) or accept any file transfer. Users should configure Yahoo! Messenger to ignore all users who are not on their contact list. This will prevent many of these worms from spreading, especially those that come from unknown sources. Figure 3.12 shows the **Ignore List** options available in the preferences.

Figure 3.12 Ignore List Preferences

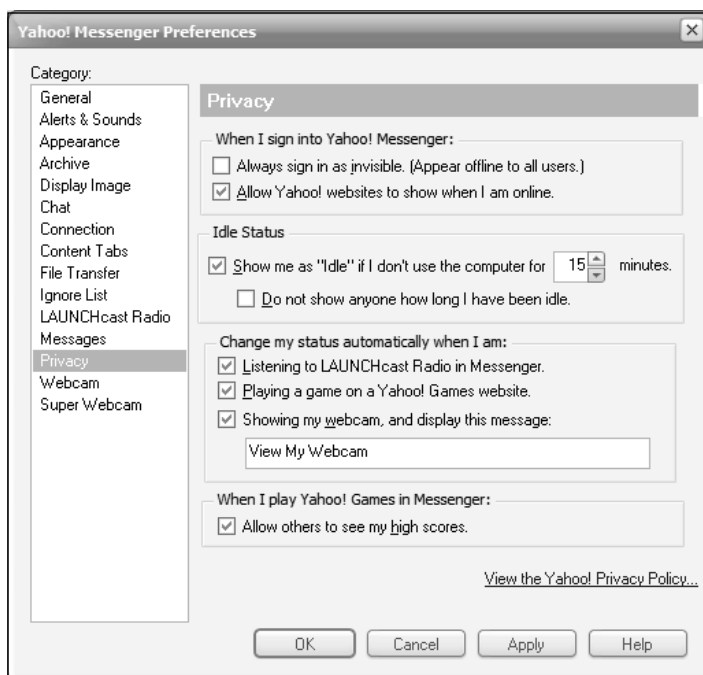


Table 3.2 lists the security threats for 2005 listed by the IMLogic Threat Center:

Table 3.2 Yahoo! Security Threats

Name	Severity	Date
WORM_WOOTBOT.GX	Low	January 31, 2005
W32/Ahker-B	Low	February 3, 2005
Yahoo! Messenger File Transfer Spoofing	Low	February 18, 2005
Yahoo! Messenger Privilege Escalation	Low	February 18, 2005
Yahoo! Phishing Attack	Low	March 24, 2005
Trillian Multiple Plug-ins Buffer Overflow Vulnerability	Low	March 25, 2005
W32.Chod.B@mm	Low	April 2, 2005
W32.Picrate.A@mm	Low	April 14, 2005
W32.Picrate.B@mm	Low	April 17, 2005
W32.Velkbot.A	Low	April 23, 2005
Yahoo-Log	Low	May 18, 2005
Trojan.Dazheb	Low	May 20, 2005
Trojan.Dazheb	Low	May 20, 2005
W32.Picrate.C@mm	Low	May 23, 2005
W32.Picrate.C@mm	Low	May 23, 2005
PHISH.Yahoo.STAR	Med	May 24, 2005

Worm Examples

W32.Chod.B@mm

W32.Chod.B@mm was discovered on April 2, 2005 and affects both Yahoo! Messenger and MSN messenger. Symantec's Research Center (<http://sarc.com>) describes it as the following:

When W32.Chod.B@mm is executed, it performs the following actions:

1. Drops the following files:
 - %System%\cpu.dll
 - %System%\[random folder name]\csrss.dat

- %System%\[random folder name]\csrss.exe
- %System%\[random folder name]\csrss.ini
- %Application Data%\Microsoft\CD_Burning\Autorun.exe

2. Displays the following message:

Run-time Error

Run-time error #7: Out of memory

3. May drop the C:\Documents and Settings\All Users\Start Menu\Programs\Startup\csrss.Ink shortcut file.
4. Adds the value “**Csrss**” = “%System%\[random folder name]\csrss.exe” to the following registry subkeys so that the worm runs every time Windows starts:
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
5. Adds the “**run**” = “%System%\[random folder name]\csrss.exe” value to the HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows so that the worm runs every time Windows starts.
6. Adds the “Installed” = “1” value to the following registry subkeys as an infection marker:
- HKEY_CLASSES_ROOT\Chode
 - HKEY_CURRENT_USER\Software\Chode
7. Deletes the following registry subkeys to prevent the programs that are associated with those keys from running when Windows starts:
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\CAISafe
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ccProxy
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ccPwdSvc

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ccSetMgr
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ISSVC
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\MCAgentExe
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\navapsvc
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OutpostFirewall
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\PcCtlCom
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SAVScan
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SBService
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SmcService
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SPBBCSvc
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\vsmon
8. Adds the following values to the HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced registry subkey so that the dropped file stays hidden:
- “Hidden” = “2”
 - “SuperHidden” = “0”
 - “ShowSuperHidden” = “0”
9. Deletes the following values from the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run to prevent several legitimate programs from running when Windows starts:

- TmPfw
 - tmpoxy
 - Tmntsrv
 - net stop
 - sc config
 - start
 - CleanUp
 - MCUpdateExe
 - VirusScan Online
 - VSOCheckTask
 - ccApp
 - Symantec NetDriver Monitor
 - Outpost Firewall
 - gcasServ
 - pccguide.exe
 - KAVPersonal50
 - Zone Labs Client
 - services
 - microsoft antispware
 - hijackthis
10. Adds the following values to the HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System registry key to disable registry tools.
 - “DisableRegistryTools” = “1”
 - “NoAdminPage” = “1”
 11. Adds the following values to the HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows registry subkey:
 - “Load” = “%System%\[random folder name]\csrss.exe”
 - “run” = “%System%\[random folder name]\csrss.exe”

12. Collects e-mail addresses from files with the following extensions:

- .adb
- .asp
- .cgi
- .ctt
- .dbx
- .dhtm
- .doc
- .eml
- .htm
- .html
- .msg
- .oft
- .php
- .pl
- .rtf
- .sht
- .shtm
- .sql
- .tbb
- .txt
- .uin
- .vbs
- .wab
- .xml

The worm avoids e-mail addresses with a domain name containing the following strings:

- .gov
- .mil

- abuse
- antivirus
- avp
- bitdefender
- f-pro
- f-secure
- fbi
- kaspersky
- mcafee
- messagelabs
- microsoft
- norton
- spam
- symantec

13. Uses its own SMTP (Simple Mail Transfer Protocol) engine to send e-mail messages to any addresses found.

The e-mail may have the following characteristics:

From: Spoofed.

- security@microsoft.com
- security@trendmicro.com
- securityresponse@symantec.com

Subject:

- Warning - you have been infected!
- Your computer may have been infected

Message body:

- Your message was undeliverable due to the following reason(s):
- Your message could not be delivered because the destination server was unreachable within the allowed queue period. The amount of time a message is queued before it is returned depends on local configuration parameters. Most likely there is a network problem that prevented

delivery, but it is also possible that the computer is turned off, or does not have a mail system running right now.

- Your original message has been attached.

Attachment:

- netsky_removal.exe
- removal_tool.exe
- message.pif
- message.scr

14. Attempts to send itself through MSN Messenger. The worm monitors for any change in the status of MSN Messenger contacts. The worm then sends commands to MSN Messenger to send a copy of the worm to the contacts whose status has changed. The message has the following characteristics:

Message Body:

- check out what I just found on some stupid website
- dude check this out, it's awesome! :D
- haha you have to see this, I almost couldn't believe it! :O
- holy shit you have to see this... :|
- I just found this on a CD... you won't believe it! :|
- LOL! look at this, I can't explain it in words..
- naked lesbian twister
- omg check this out, it's just wrong :O
- ROFL!! you have to see this... wtf...
- you have to see this, it freaked me out :S
- you have to see this, it's amazing!

with a copy of itself as any of the following files:

- check this out
- gross
- my sister's webcam
- mypic

- paris hilton
- picture
- rofl
- us together
- wtf

The file has one of the following extensions:

- .pif
- .scr

15. Blocks access to the following security-related websites by appending text to the Hosts file:

- avp.com
- ca.com
- customer.symantec.com
- dispatch.mcafee.com
- download.mcafee.com
- f-secure.com
- fastclick.net
- ftp.f-secure.com
- ftp.sophos.com
- grisoft.com
- housecall.trendmicro.com
- kaspersky.com
- liveupdate.symantec.com
- mast.mcafee.com
- mcafee.com
- merijn.org
- my-etrust.com
- nai.com

- networkassociates.com
- pandasoftware.com
- phpbb.com
- rads.mcafee.com
- secure.nai.com
- securityresponse.symantec.com
- service1.symantec.com
- sophos.com
- spywareinfo.com
- support.microsoft.com
- symantec.com
- trendmicro.com
- update.symantec.com
- updates.symantec.com
- us.mcafee.com
- vil.nai.com
- viruslist.com
- www.avp.com
- www.awaps.net
- www.ca.com
- www.f-secure.com
- www.fastclick.net
- www.grisoft.com
- www.kaspersky.com
- www.mcafee.com
- www.merijn.org
- www.microsoft.com
- www.my-etrust.com

- www.nai.com
- www.networkassociates.com
- www.pandasoftware.com
- www.phpbb.com
- www.sophos.com
- www.spywareinfo.com
- www.symantec.com
- www.trendmicro.com
- www.viruslist.com
- www.zonelabs.com
- www3.ca.com
- zonelabs.com

16. Ends the following processes, some of which may be security-related:

- bbeagle.exe
- ccapp.exe
- ccevtmgr.exe
- ccproxy.exe
- ccsetmgr.exe
- d3dupdate.exe
- enterprise.exe
- gcasdtserv.exe
- gcasserv.exe
- hijackthis.exe
- i11r54n4.exe
- irun4.exe
- isafe.exe
- issvc.exe
- kav.exe

- kavsvc.exe
- mcagent.exe
- mcdash.exe
- mcinfo.exe
- mcmnhdlr.exe
- mcshield.exe
- mcvsescn.exe
- mcvstsn.exe
- mcvsrte.exe
- mcvsshld.exe
- mpfagent.exe
- mpfservice.exe
- mpftray.exe
- msblast.exe
- msconfig.exe
- mscvb32.exe
- mskagent.exe
- mwincfg32.exe
- navapsvc.exe
- navapw32.exe
- navw32.exe
- npfmntor.exe
- outpost.exe
- pandaavengine.exe
- pccguide.exe
- pcclient.exe
- pcctlcom.exe
- penis32.exe

- regedit.exe
 - smc.exe
 - sndsrvc.exe
 - spbbcsvc.exe
 - symclsvc.exe
 - sysinfo.exe
 - sysmonxp.exe
 - teekids.exe
 - tmntsrv.exe
 - tmpfw.exe
 - tmproxy.exe
 - usrprmt.exe
 - vsmon.exe
 - wincfg32.exe
 - winsys.exe
 - winupd.exe
 - zapro.exe
 - zlclient.exe
17. Opens a back door and allows a remote attacker to have unauthorized access to the compromised computer by connecting to an IRC server.
18. Listens for commands from the attacker to perform some of the following actions:
- Download and execute files
 - Install or uninstall IRCD
 - Perform ping, TCP, or UDP (User Datagram Protocol) denial of service attacks
 - Send e-mail
 - Shut down and reboot the computer
 - Spread itself by e-mail
 - Spread itself through MSN messenger

19. Attempts to steal passwords for the following applications:
 - AOL Instant Messenger (older versions)
 - AOL Instant Messenger/Netscape 7
 - GAIM
 - ICQ Lite 4.x/2003
 - Miranda
 - MSN Messenger
 - Trillian
 - Windows Messenger (on Windows XP)
 - Yahoo Messenger (Versions 5.x and 6.x)
20. Uses one of the following tools to steal passwords from the above applications:
 - Intelligent TCPIP.SYS patcher
 - MessenPass
 - Protected Storage PassView Subhead Level 3

W32.Picrate.C@mm

W32.Picrate.C@mm was discovered on May 23, 2005, with messenger. Symantec's Research Center (<http://sarc.com>) describing it as:

When W32.Picrate.C@mm is executed, it performs the following actions:

1. Opens the following URL in the default browser:
[[http://solair.eunet.yu/\[REMOVED\]/coyote.jpg](http://solair.eunet.yu/[REMOVED]/coyote.jpg)]
2. Drops the following files, to disable certain programs:
 - %System%\netstat.com
 - %System%\ping.com
 - %System%\tracert.com
 - %System%\tasklist.com
 - %System%\taskkill.com
 - %System%\regedit.com
 - %System%\cmd.com

3. Drops the following files:
 - %System%\zzx.tmp
 - %System%\File.zip
 - %System%\bszip.dll
 - %System%\ANSMTP.DLL
4. Creates the following registry subkeys when it installs ANSMTP.DLL:
 - HKEY_CLASSES_ROOT\ANSMTP.OBJ.1
 - HKEY_CLASSES_ROOT\ANSMTP.OBJ
 - HKEY_CLASSES_ROOT\ANSMTP.MassSender.1
 - HKEY_CLASSES_ROOT\ANSMTP.MassSender
 - HKEY_CLASSES_ROOT\CLSID\{253664FB-EDFC-4AC6-BD69-B322F466AEED}
 - HKEY_CLASSES_ROOT\CLSID\{887A577B-406B-48FF-80CB-70752BFCD7B4}
 - HKEY_CLASSES_ROOT\Typelib\{DE6317F7-6EF0-45C2-88D1-8E09415817F1}
 - HKEY_CLASSES_ROOT\Interface\{68B8DCDB-EFA4-420A-BB8A-71B9892A2063}
 - HKEY_CLASSES_ROOT\Interface\{1E98666F-6260-42C9-B846-32B20fDEFEB7B}
 - HKEY_CLASSES_ROOT\Interface\{A5F6C90C-ABE4-4C57-A421-8C5A202AA9F8}
 - HKEY_CLASSES_ROOT\Interface\{B13281CF-8778-4C98-AE23-ABBA4637A33D}
5. Gathers e-mail addresses from the Yahoo! Messenger address book.
6. Sends itself as an attachment to the e-mail addresses that it has gathered. The worm checks the country code of the current user and sends an e-mail in English, German, French, Italian, Portuguese, Spanish, or Dutch. The e-mail has one of the following four subject and message body combinations for each language:

English:

Subject: Attachment Returned

Message Body: This file was rejected by the recipient

Subject: You suck!

Message Body: I have enclosed why you suck and you're not going to like it

Subject: My new details

Message Body: Hi ive changed email address if you would like to keep in contact i have enclosed my new details

Subject: Party Invite!!

Message Body: You have been invited to my party please download the details and tell me if you will be able to make it ,Thanks!

German:

Subject: Zubehoer Ging

Message Body: Diese Akte wurde von der Empfaenger zurueck gewiesen

Subject: Sie saugen!

Message Body: Ich habe umgeben, warum Sie saugen und Ihr Gehen nicht zu wie ihm :@

Subject: Meine neuen Details

Message Body: Hallo aenderte ive email address, wenn Sie zu moechten Unterhalt im Kontakt habe ich meine neuen Details umgeben

Subject: Beteiligtes Laden!!

Message Body: Sie sind zu meinem Beteiligten eingeladen worden
ddownloaden Sie bitte die Details und erklären Sie mir wenn Sie in
der LageSIND, es zu bilden, Danke!

NOTE

The “a” umlaut is represented as “ae”, the “u” umlaut as “ue”, and the
“o” umlaut as “oe.”

French:

Subject: L'Attachment Est retourne

Message Body: Ce dossier a ete rejete par le destinataire

Subject: Voud sucez!

Message Body: J'ai enferme pourquoi vous sucez et votre ne pas aller a
comme lui :@

Subject: Mes nouveaux details

Message Body: Bonjour l'ive a change le email address si vous voudriez a
subsistence en contact j'ai joint mes nouveaux details

Subject: La Partie Invitent!!

Message Body: Vous avez ete invites a ma partie telechargez svp les details
et me dites si vous pourrez la fair, merci!

Italian:

Subject: Il Collegamento Ha rinviato

Message Body: Questa lima e stata rifiutata dal destinatario

Subject: Succhiate!

Message Body: Ho accluso perche succhiate e vostro non andare come ad esso :@

Subject: I miei nuovi particolari

Message Body: Hi il ive ha cambiato il email address se gradiste a conservazione in contatto ho accluso i miei nuovi particolari

Subject: Il partito Invita!!

Message Body: Siete stati invitati al mio partito prego trasferite i particolari dal sistema centrale verso i satelliti e mi dite se potrete farlo, ringraziamenti!

Portuguese:

Subject: O Acess

Message Body: Esta lime foi rejeitada pelo receptor

Subject: Voce suga!

Message Body: Eu inclui porque voce suga e seu nao lhe ir como :@

Subject: Meus detalhes novos

Message Body: Hi o ive mudou o email address se voce gostasse a sustento no contato eu inclui meus detalhes novos

Subject: O Partido Convida!!

Message Body: Voce foi convidado a meu partido download por favor os detalhes e diz-me se voce pudesse o fazer, agradecimentos!

Spanish:

Subject: El Accesorio Volvi

Message Body: Este archivo fue rechazado por el recipiente

Subject: Usted aspira!

Message Body: He incluido porque usted aspira y el su no ir como a el :@

Subject: Mis nuevos detalles

Message Body: Hi el ive cambio email address si usted quisiera a mantener contacto he incluido mis nuevos detalles

Subject: El partido Invita!!

Message Body: Le han invitado a mi partido descarga por favor los detalles y me dice si usted puede hacerlo, gracias!

Dutch:

Subject: Het aanhechtsel Keerde Terug

Message Body: Dit bestand werd door de ontvanger afgekeurd

Subject: U zuigt!

Message Body: Ik heb waarom u zuigt bijgevoegd en uw gaand alsof het niet :@

Subject: Mijn nieuwe details

Message Body: Hi veranderde ive e-mail aanspreekt of u van naar zou houden Hou in contact ik heb bijgevoegd mijn nieuwe details bij

Subject: De partij Uitnodig!!

Message Body: U bent naar mijn partij alstublieft download de details uitgenodigd worden en vertel mij indien u hem zult kunnen maken, Bedankt!

Attachment: File.zip

NOTE

File.zip contains one of the following files:

- File.pif
- Corrupt.pif
- details.pif
- Party.pif
- File.scr
- Corrupt.scr
- details.scr

7. Drops and executes one of the following files, which is a copy of a W32.Randex variant:

- %System%\p2pnetwork.exe
- %System%\winlogins.exe
- %System%\wini.exe
- %System%\winis.exe
- %System%\muamgr.exe

Client Security

To properly secure yourself against malicious code such as worms and viruses spread through instant messaging services, it is advised that you restrict access to these clients. Since worms and other malicious code are spread as a URL embedded in a message, you would have to restrict access the Yahoo! Messenger entirely to avoid these security issues. Yahoo! Messenger is very adept at evading network controls, and is able to use any port and avoid detection by adding HTTP headers to its communication. Since a specific port cannot be blocked, you must block access to the authentication servers located at cs1.yahoo.com, cs2.yahoo.com, and cs3.yahoo.com.

Keep in mind that these server addresses can change at any time, and it may be necessary to update firewall rules regularly to continue blocking this service. Blocking these addresses will not, however, prevent advanced users from using a proxy to connect to the Yahoo! Messenger service, and creating strict rules will result in blocking all Yahoo! traffic, including access to their Web content and search engine.

Yahoo! frequently updates its clients, and adds new features on a regular basis, often updating its protocol as well. Client updates are not only the result of an upgrade, but also are based on security vulnerabilities within the product that need to be addressed. According to Secunia (<http://secunia.com/>), a security services company, Yahoo! Messenger has been susceptible to seven vulnerabilities since 2003. Some of the vulnerabilities have led to increased privileges for a malicious user, including the vulnerability detailed on February 18, 2005:

Secunia Research has discovered a vulnerability in Yahoo! Messenger, which can be exploited by malicious, local users to gain escalated privileges.

The vulnerability is caused due to a combination of weak default directory permissions and the Audio Setup Wizard (asw.dll) invoking the ping.exe utility insecurely during the connection testing phase. This can be exploited to execute arbitrary code with the privileges of another user by placing a malicious ping.exe file in the application's Messenger directory.

Successful exploitation requires that a user runs the Audio Setup Wizard and that the application has been installed in a non-default location (not as a subdirectory to the Program Files directory).

The vulnerability has been confirmed in version 6.0.0.1750 for Windows. Other versions may also be affected. The solution is to update to version 6.0.0.1921 or later (<http://messenger.yahoo.com/>).

Four of these advisories are related to system access, where a malicious user may execute arbitrary code on the affected system after exploiting the vulnerability. Specific information regarding Yahoo! Messenger security is provided by Yahoo! and is located at <http://messenger.yahoo.com/security/>.

These vulnerabilities are often remedied by Yahoo! releasing a new version of their client. If a user leaves a vulnerable copy of software on his or her workstation, this can affect not only that workstation, but the network in general. If instant messaging software is approved for use on a large network, it is recommended that the software be centrally managed in order to distribute updates and protect against malicious users and vulnerable code. In order to prevent attacks against vulnerable software, it is recommended that it is removed or disabled if it is not critical. If this is not an option, clients should be updated regularly with new executables to provide clients that are free from software vulnerabilities and pose less risk to the network.

Summary

Yahoo! Messenger provides a great amount of tools to enable users to communicate through text, voice, and video. Many of the features of Yahoo! Messenger are also avenues for security issues, such as data leakage, social engineering, malicious code, and client vulnerabilities. Data leakage allows the flow of sensitive and confidential information into and out of an organization or individual's workstation. The usual safeguards against file transfers are bypassed through instant messaging. These transfers can be done without knowledge or consent from an affected user and can involve the transfer of confidential files or copyrighted files such as MP3s, movies, and licensed software. Because instant messaging clients do not impose size restrictions on file transfers, there is an easy workaround for users who have restrictions on e-mail attachments and FTP transfers. These transfers can be configured to appear as standard Web traffic on port 80, making it difficult to monitor and record.

Social engineering plays a role in instant messaging security since it is nearly impossible to identify a user at the other end of a conversation. If a contact's account has been compromised, they are able to masquerade as that user and ask for sensitive information. Malicious users may also pose as employees of an instant messaging service, and ask for credit card information, your username and password, or other information. To prevent this type of situation, no confidential information should be discussed over an instant messaging service. In order to prevent another user from posing as you, it is recommended that you do not save your Yahoo! ID and password locally, which can prevent a malicious user who has access to your workstation from posing as you.

Malicious code has the ability to spread quickly through instant messaging networks, circumventing protection that has been put in place to prevent worms and viruses from spreading. Instant messaging clients can bypass security devices such as firewalls, URL filtering, and gateway security devices, promoting the spread of malicious code. Many of these worms spread via URLs displayed within a message from a contact. By clicking on this type of link, a user will infect his or her workstation. Users should never click on a URL embedded within an instant message, especially from an unknown contact.

Solutions Fast Track

Yahoo! Messenger Architecture

- ☑ Yahoo! Messenger is based on client server architecture, and has several login servers as well as a pool of servers that handle functions such as Webcam usage and file transfers.
- ☑ The default port for Yahoo! Messenger is 5050, but the client will try to communicate over any available port until a connection is established.
- ☑ Yahoo! Messenger will add an HTTP header to the information it sends, allowing it to pass through firewalls that use protocol analysis.
- ☑ Yahoo! Messenger supports several proxies, including SOCK4, SOCKS5, and HTTP, giving advanced users the opportunity to bypass controls and connect to the service.

YMSG Protocol

- ☑ The YMSG protocol is in constant development, and changes based on new features added to the client and to prevent third-party clients from connecting to the service.
- ☑ Upon connection and authentication, YMSG send the client a contact list for messaging and ignoring, as well as cookies to enable access to other services of Yahoo!

Features and Security Information

- ☑ Message archiving is a capability that allows the client to record all conversations on the instant messaging service. This information is saved in a file locally, and encrypted for a greater amount of security. However, utilities exist which are able to decrypt that information for all users who have logged in from a particular workstation.
- ☑ There is no encryption on communications between clients. Messages are routed over the Internet through a central server and may be vulnerable to a malicious user employing a packet capturing utility. With this utility, the entire conversation can be recorded.

- ☑ File transfers are an efficient way to send large files to other users outside of a particular network or organization. With no restrictions on file types and sizes, a user can send confidential or sensitive information as well as exchange files that infringe on copyright laws.

Malicious Code and Client Security

- ☑ Malicious code such as worms are generally distributed through instant messaging services via URLs. Some worms make these messages appear as though these URLs are coming from a known user, increasing the chance that these URLs will be visited, thereby infecting a workstation.
- ☑ Some worms have the ability to receive remote commands to control the infected workstation. This gives a malicious user complete control over the workstation, including access to all information stored on the workstation.
- ☑ Yahoo! Messenger checks for updates to its program when it authenticates to the service. This allows for users to be notified when there is an updated, and possibly more secure client, available.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Is Yahoo! Messenger a separate service than Yahoo! or do I need to use the same sign-in?

A: The Yahoo! ID you would use to access and customize information on their websites is the same as your sign-in. This provides you with the ability to access much of your customized content including e-mail and other services through tabs within the messenger. Another feature of this client that uses some customization is its Launchcast radio service, which provides music and customized radio stations to users. This integration with their websites is one reason for the popularity of Yahoo! and makes it a very good choice for users who frequent Yahoo! websites.

Q: What is the best way to prevent worms from spreading?

A: The easiest way to stop a worm is not to click on the URL. If a URL looks suspicious, it should not be clicked, especially if it is coming from an unknown user. Additionally, any file with a .pif, .exe, or other extension that does not lead to a Web page should always be avoided. Remember that websites may take advantage of system vulnerabilities and install software without knowledge or consent. Another way to prevent worms or other malware from spreading is to change the Ignore List setting on Yahoo! Messenger to restrict communication with other users unless they are from known users. This will prevent unwanted messages from unknown users, which is usually related to SPIM (SPAM on instant messaging) and worms.

Q: Is there a good way to block Yahoo! Messenger on a network?

A: Without using a gateway device such as IMlogic's, the only way to block Yahoo! Messenger is to restrict access on all ports to the authentication servers for Yahoo! Messenger. This will prevent clients from signing into the service, even if the user has changed his or her port number from the default. This will not, however, prevent users from configuring the client to connect to the service via a proxy.

Q: How secure is Yahoo! Messenger's file transfer abilities?

A: Yahoo! Messenger has some options for configuring which users have access to files, what type of notification, if any, is available, and what directories house these files. By default, the client is set to ask for permission before accepting files or sending files another user has requested. You can change the setting for shared files to restrict all users from downloading files from your workstation, and make sure that there are no confidential or sensitive files in this directory.

MSN Messenger

Solutions in this chapter:

- MSN Messenger Architecture and Protocol
- Features and Security Information
- Malicious Code and Client Security

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Microsoft's MSN Messenger, much like Yahoo! Messenger, started out as a multi-protocol client with the capability to connect to AOL's AIM (AOL Instant Messenger) service. As a new instant messaging service, MSN did not have a large user base and used interoperability with AIM as a way to co-opt that user community while building out its own service. Microsoft released Windows Messenger in 1999, and included the ability to connect to AIM. Just as with Yahoo! Messenger, AIM quickly changed its protocol to prevent access from this client.

AOL changed its protocol, which was answered with Microsoft changing its client to regain connectivity to the service. This is one of the reasons Microsoft released 21 versions of its instant messaging client in 1999 (see <http://meetingby-wire.com/OldVersions.htm> for more details). Microsoft finally abandoned its goal of interoperability at the end of 1999 with version 2.0 and concentrated on building out its own service.

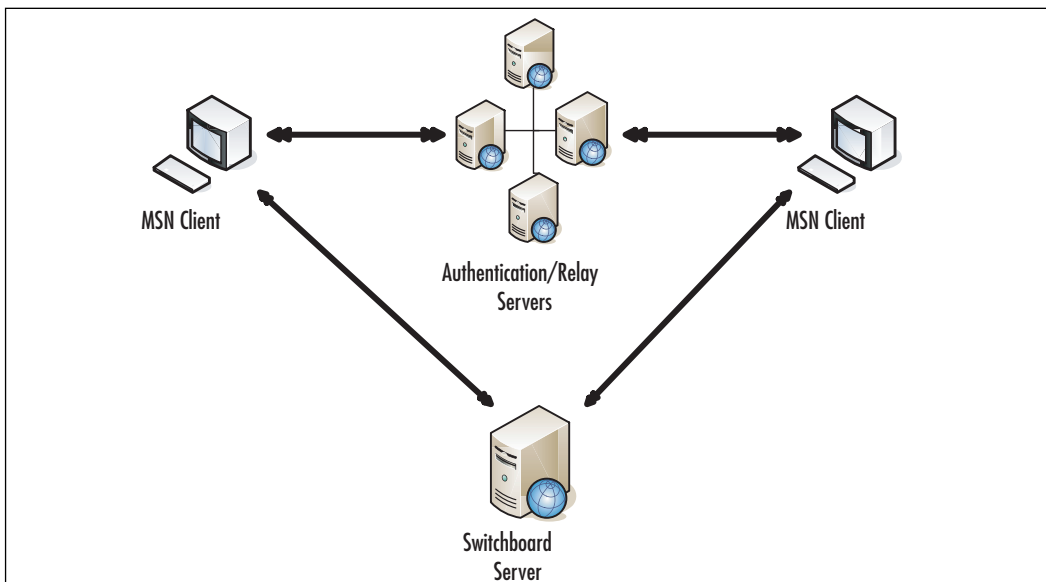
This client for Microsoft's service has had several names including Windows Messenger (the original name and also the version shipped with the Windows operating system), .net Messenger, and MSN Messenger, the current version available for download. The latest client has changed considerably from its early versions, tying many of its features to the MSN portal, including the Hotmail e-mail service and other MSN properties and services. Today's client bears little resemblance to the previous versions, and contains new features such as Webcam communication, audio chat, photo and calendar sharing, and includes the ability to provide or request a remote connection with another workstation. Many of MSN Messenger's features tie back to MSN Web properties, and include a new blogging service called MSN Spaces.

MSN Messenger Architecture and Protocol

The architecture of MSN Messenger is very complicated compared to other instant messaging services such as AIM and Yahoo! since it relies on five different types of servers to handle the communication and operation of its service. The MSN Messenger servers and their functions are featured in Table 4.1 and Figure 4.1.

Table 4.1 Servers and Functions

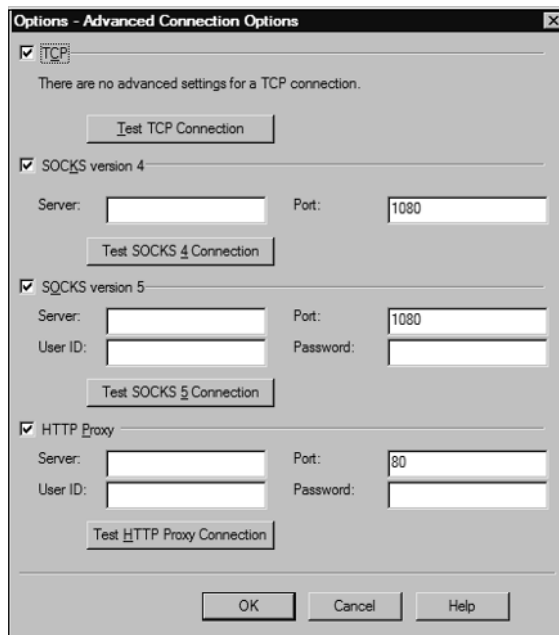
Server	Function
Dispatch Server	Track locations for the notification servers and communicate the IP (Internet Protocol) address of these servers to clients.
Notification Server	One of the main servers of the MSN Messenger service. Clients must maintain connection to the notification server at all times. Provides information on client locations for routing purposes and provides connections to the switchboard servers. Provides presence information notifying other users if whether or not a particular user is connected.
.NET Passport Login Server	Part of the authentication process, provides the client attempting to sign in with a ticket to complete the authentication process with a notification server.
Nexus Server	Another part of the authentication process, provides a client with a URL (Uniform Resource Locator) of the passport server and information necessary to authenticate.
Switchboard Server	Provides messaging and file transfer functionality between two clients.

Figure 4.1 MSN Messenger Architecture

MSN Messenger uses port 1863 to communicate with servers, and this setting is not configurable via the client. MSN, like all other messengers covered so far, has the capability to use a proxy to facilitate communications with its servers and clients if the communication is not available. Like all other instant messengers, this feature allows advanced users to bypass security controls that may be present on a network. MSN Messenger is able to utilize the following proxies (see also Figure 4.2):

- SOCKS4
- SOCKS5
- HTTP (Hypertext Transfer Protocol)

Figure 4.2 Proxy Settings



MSN Messenger uses the Mobile Status Notification Protocol (MSNP) for communication, which has changed several times due to the addition of new features to the client. Some changes to the protocol have also been made to prevent third-party clients such as Trillian and GAIM from connecting to the service. Microsoft released a draft of version 2 of the protocol to the Internet Engineering Task Force (IETF) in 1999. The document is no longer available from IETF's website, but can still be

found at http://hypothetic.org/docs/msn/ietf_draft.txt while Microsoft's announcement of the release of its protocol can be found at;

<http://microsoft.com/presspass/press/1999/Aug99/Protocolpr.msp>.

Support for version 2 of MSNP was discontinued in October 2003, and at that time users were forced to upgrade to clients capable of communicating using MSNP version 8 or later. Microsoft has not publicly released any information on protocols after version 2, so much of the information known about MSNP comes from protocol analysis and reverse engineering. MSN Messenger uses the following ports for communicating with the service (Table 4.2):

Table 4.2 MSN Messenger Ports and Services

Port	Service
1863	Sign-in
6891-6900	Direct Connect, File Transfer
1503	Application Sharing
6901	VOIP
5004-65535	Video/Webcam
1838	Game Audio
5004-65535	Audio Chat

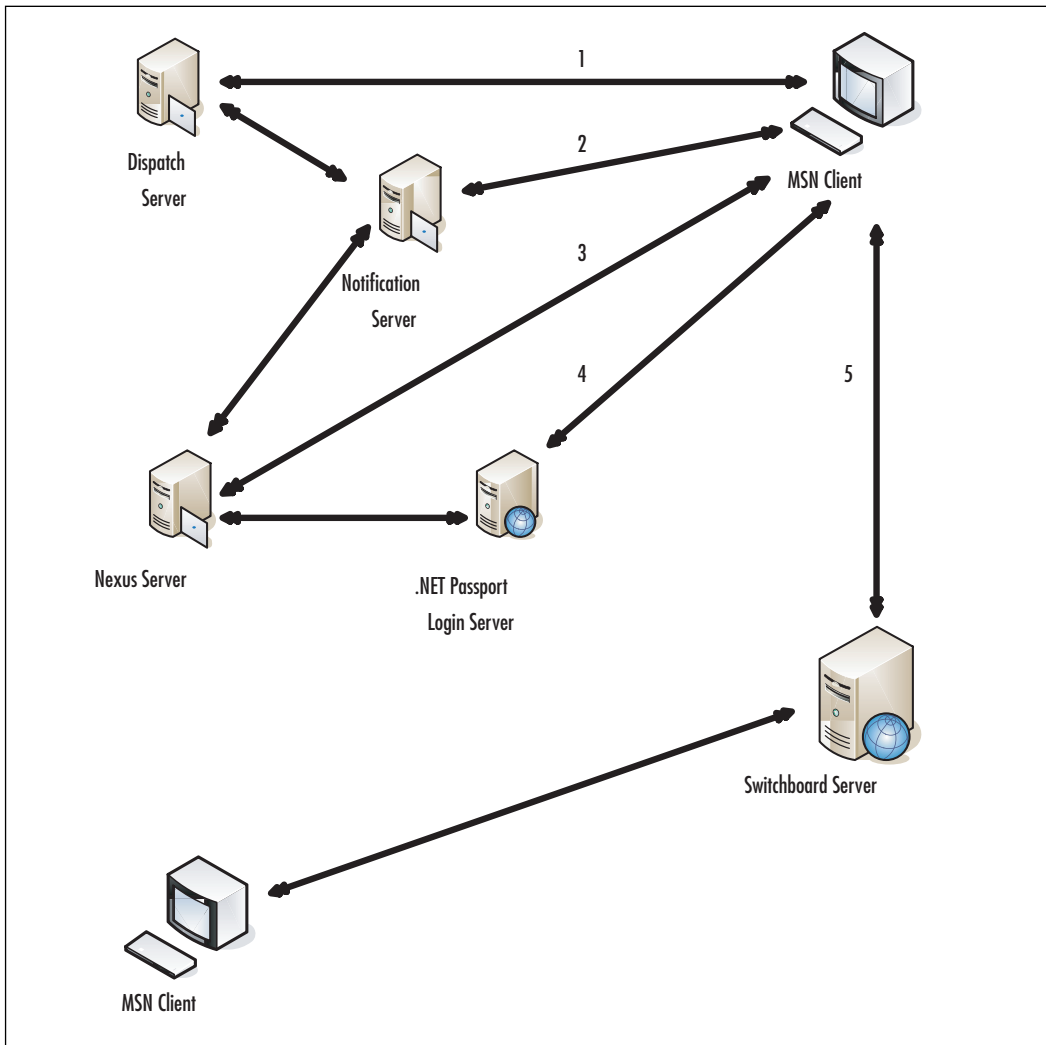
MSN Messenger, like many other Microsoft and MSN websites and services, utilizes the .NET Passport to sign into the MSN Messenger service. An MSN Messenger sign-in session is based on a challenge-response mechanism to authenticate user credentials. This process begins when a client connects to a dispatch server. This server, located at messenger.hotmail.com, is responsible for communicating the IP address and port information of a notification server to the client. Importantly, the location of a notification server is maintained in the client, and is used until it is no longer available. If a notification server is offline, the client will connect to the dispatch server to receive information on another notification server that is available. This process allows the MSN Messenger service to take notification servers offline, or in the case of system failure, clients can be redirected to another server.

Figure 4.3 Signing Into MSN Messenger

Once the client establishes a connection with a notification server, the client and server communicate protocol version information, client version information, and verify Microsoft Passport identification. The client uses the notification server for many services, and must remain connected during the entire instant messaging session. If the connection between the client and the notification server is not available, the client will be disconnected from the service.

The notification server begins the authentication process after verifying that the Passport is in the proper format and issues the client the challenge string. The client contacts the Nexus server to receive a Passport server URL in order to send the Passport and password to complete the authentication process. The communication with the Passport server is conducted over the HTTPS (Hypertext Transfer Protocol over Secure Sockets Layer) protocol, ensuring that the sign-in information is encrypted. The client sends the challenge string, Passport username, and password to the Passport URL. If the credentials for signing in are confirmed, the Passport server issues a *ticket*, which is passed back to the notification server to complete the authentication procedure. Figure 4.4 details the entire authentication procedure for MSN Messenger.

Figure 4.4 MSN Authentication Process



Once signed into the service, the client communicates to the notification server and transmits the details of its presence, which is communicated in a three-letter code. This is used to communicate your presence to others using the service, and denotes your availability to send and receive messages. The notification server handles these settings, and requires that a connection be maintained with the client. If the connection is severed, the client will appear offline, and be unable to send and receive messages. There are several options are set through the client, and currently include the items shown in Table 4.3.

Table 4.3 Code and Status

Code	Status
NLN	Online
BSY	Busy
BRB	Be Right Back
AWY	Away
PHN	On The Phone
LUN	Out To Lunch
FLN	Appear Offline

When logging into the service, the client sends the status to the notification server with the following syntax:

```
FLN ashmattley@hotmail.com
```

When manually setting the status in the client, the information is sent to the notification server using the change command (CHG) followed by a transaction ID, which identifies the set of commands the individual communication belongs to. This transaction ID is later used to identify the status of other users on your contact list. Setting your status involves the client using the following syntax:

```
CHG 5 BRB
```

Once the presence is set, the notification server sends information to the client regarding the Passport account that was used to sign in. Hotmail account information (the number of unread e-mails) is sent to the client, along with other communication regarding the Passport. After this information is received, the client is ready to accept information regarding the contact lists. There are four different lists that are sent to the client, communicated with a two-letter code (Table 4.4).

Table 4.4 Code/List Type and User Information

Code and List Type	User Information
FL – Forward List	Users who were added to your contact list
RL – Reverse List	Users who added you to their contact list
AL – Allow List	Users who are able to see your status
BL – Block List	Users who are blocked from seeing your status

The MSN Messenger client can use one of two commands to retrieve this information. The SYN command is almost always used and synchronizes all lists while the

list command (LST) requests a specific list one at a time. The following is the syntax for the LST command between the client and the server:

Client request:

LST 8 FL

Server response:

LST 8 FL 11 1 3 ashmattley@hotmail.com Ash 0

LST 8 FL 11 2 3 lmeleme@hotmail.com Lee 0

LST 8 FL 11 3 3 lmatzu@hotmail.com LittleMatzu 0

The client request has three parameters. The first parameter, the LST command, denotes that a list is being requested from the server. The second part of the command is the transaction ID, which identifies the communication as belonging to a specific set of requests and responses. The next portion, FL, denotes the type of list being requested. The server response has more parameters, and the first part is the LST command, which is a response that a list is being sent from the server. The second parameter is the transaction ID, which associates the communication as belonging to the list request. The next portion, FL, denotes to the client the type of list being received. The fourth parameter, the number 11, tracks the version number of the list, representing how many times the list was modified. The fifth and sixth parameters are the amount of contacts that are contained in the contact list. In this example, there are three contacts, each with their specific number and total numbers. The next portion is the Passport ID of the user followed by the nickname they have configured within the client. This can change based on when a user reconfigures their client, and is updated through the notification server. The last parameter is the group ID, which corresponds to the group that you have placed them in when configuring your contact list.

Messaging using MSN Messenger is also more complicated than other services. Rather than relaying messages over one server like AIM or Yahoo!, MSN relies on several connections to different servers to carry out communication, as seen in Figure 4.5. For example, when beginning a messaging session, Bert initiates a session by typing a message a particular contact, Ernie. The client for Bert communicates with the notification server, informing the server that a messaging session is requested. The notification server forwards Bert to a switchboard server. The notification server then contacts Ernie's client and notifies it that a messaging session is beginning, and is connected to the same switchboard server as Bert. Once both users connect to the same switchboard sever, the messaging session commences. More information on the MSN protocol can be found at <http://hypothetic.org/docs/msn/>.

Features and Security Information

MSN Messenger has standard features that are common in other instant messengers, with some functionality and features that exist as links to MSN websites, making it somewhat of a promotional tool to drive traffic to Microsoft's websites. For instance, the messenger contains a radio feature which is merely a link to MSN's website for Internet radio stations. There are several unique features in MSN Messenger, which are whiteboard, remote assistance, and application sharing. These features provide functionality that is not present in other instant messengers, while providing the ability to remote control a machine or share applications or data in a way other instant messaging clients do not provide.

Instant Messaging

MSN Messenger provides instant messaging functions in order to allow users to send text messages to each other in near real time. As with all instant messaging clients, the danger of using the instant messaging feature is the inability to authenticate with whom you are communicating. Without being able to recognize a voice or other feature of the user at the other end, there is a possibility that this person may be a malicious user. Additionally, less experienced users may not notice slight differences in usernames. These issues are useful for malicious users, who are capable of using this to

their advantage. By posing as an acquaintance or an employee of MSN, a malicious user can obtain sensitive information, including banking, credit card, or other personal information, as well as confidential information and files from a work environment. Since this information is not encrypted, a malicious user may use a packet capturing utility to spy on users and gain access to their messaging session. Specialized utilities like MSN Track Monitor, available at <http://tallsoft.com/msntrack.htm>, have the ability to capture an entire MSN Messenger conversation that is being conducted on the same network. This utility, and others like it, are designed to be used by less advanced users, providing an easy to use environment for features that would normally be hard to use with a standard packet capturing utility like Ethereal (<http://ethereal.com>). Figure 4.5 shows MSN Messenger's conversation window used to communicate with other users. By blocking port 1863 and login.passport.com, a network administrator can limit the availability of MSN Messenger to users in the environment. These settings can prevent users from logging into the service, although the ability to use a proxy server prevents an administrator from successfully blocking the service in all cases.

Figure 4.5 MSN Messenger Conversation Window



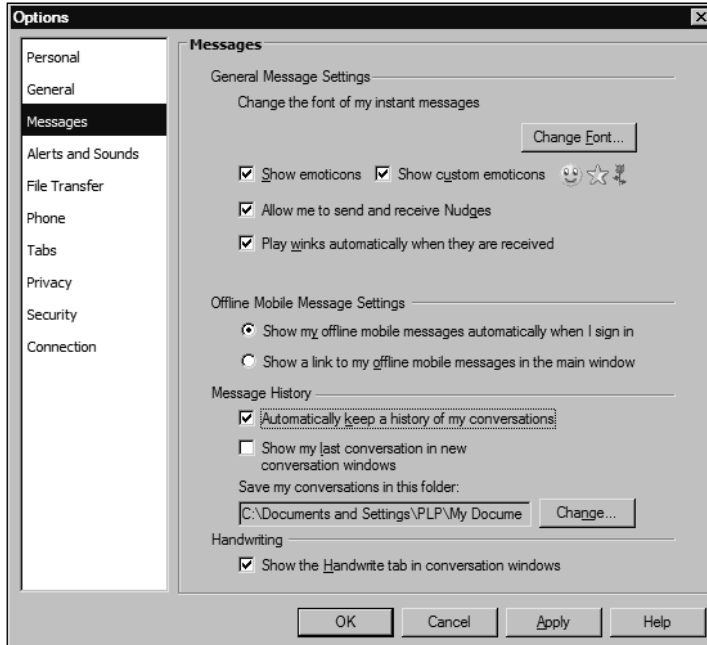
Encryption

MSN Messenger does not provide any method for encrypting messages between users. This feature is necessary to prevent packet monitoring utilities from gaining access to all information that is exchanged through the use of instant messaging services.

Message Archiving

MSN Messenger provides users with the ability to store a history of all conversations as a local file on your workstation. This feature is disabled by default, but can be enabled to record conversations, with a user specified location. This information is stored as an XML document, and is not encrypted. Figure 4.6 shows an example of a message log. Any user, either known or malicious, who has access to your workstation, can view this information and retrieve any personal or sensitive information that was discussed while communicating with another user. This feature should remain disabled to protect any information that is discussed during a messaging session. If this feature is required the log should be deleted on a regular basis. Figure 4.6 shows the settings that can be defined for this feature.

Figure 4.6 Message Logging in MSN Messenger

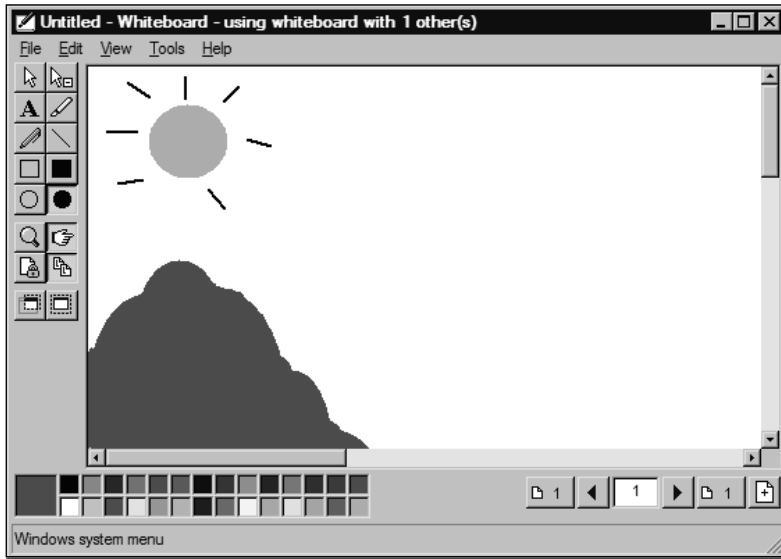


Whiteboard

The whiteboard feature of MSN Messenger allows users to share a common canvas in Microsoft's Paint accessory that ships standard on Windows operating systems. By using the application-sharing feature of MSN Messenger, two users are able to establish a connection and share the Paint application. Figure 4.7 shows an example of this feature being used. This feature is not necessarily a security issue, since it is very inefficient at displaying and communicating data between users. Other features such as text, voice, and video messaging are more versatile and efficient at communicating messages. This feature is initiated by a user and is communicated via a switchboard server. The recipient must reply to this invitation, with either an accept or deny message. Figure 4.8 is an example of this dialog box for accepting a shared whiteboard session to begin. If accepting, this message will contain an IP address for the workstation in order to begin the connection. Once the acceptance is received by the client initiating the message, another message is sent to reveal its IP address, and the connection starts. Communication is now directly negotiated between the two clients without the use of the switchboard server and uses TCP (Transmission Control Protocol) port 1503.

Figure 4.7 Whiteboard Approval Dialog Box in MSN Messenger



Figure 4.8 Shared Whiteboard Session

Application Sharing

Application sharing is based on Microsoft's NetMeeting technology. For more information about NetMeeting, visit <http://microsoft.com/windows/NetMeeting/default.ASP>. This feature provides the ability for users of MSN Messenger to share application remotely, and even taking control of other applications on a remote workstation. The process for negotiating a connection is similar to the process used in the whiteboard feature, with communications occurring over TCP port 1503.

Figure 4.9 shows the toolbar that appears after a connection is established between two clients. Sharing is not automatic, and a user must interact with the software and explicitly accept the invitation for application sharing in order for a session to begin.

Figure 4.10 shows an example of the dialog box a user is presented with at the beginning of an application sharing session. This feature can be considered a security issue since it allows users to share out information, or even allow users outside an organization to control programs on a restricted network. Proprietary programs and confidential and sensitive information may be at the disposal of an unauthorized or malicious user if this feature is available. Once application sharing begins, a user decides which application to share. Figure 4.11 shows an example of this dialog box, which is presented to both users during the initiation of the session. Each user can decide which programs to share, regardless of who initiated the session. Users may also block programs from being shared.

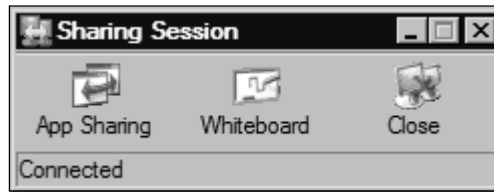
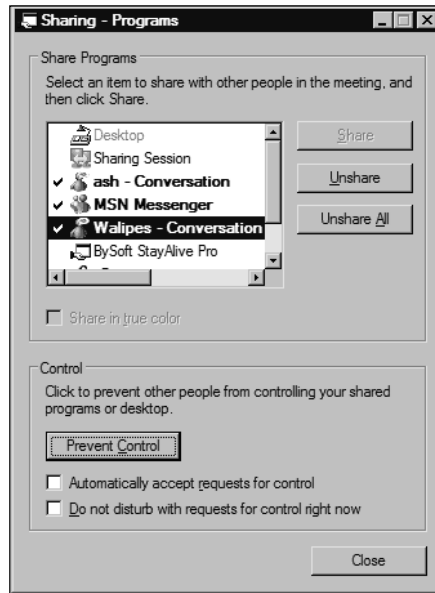
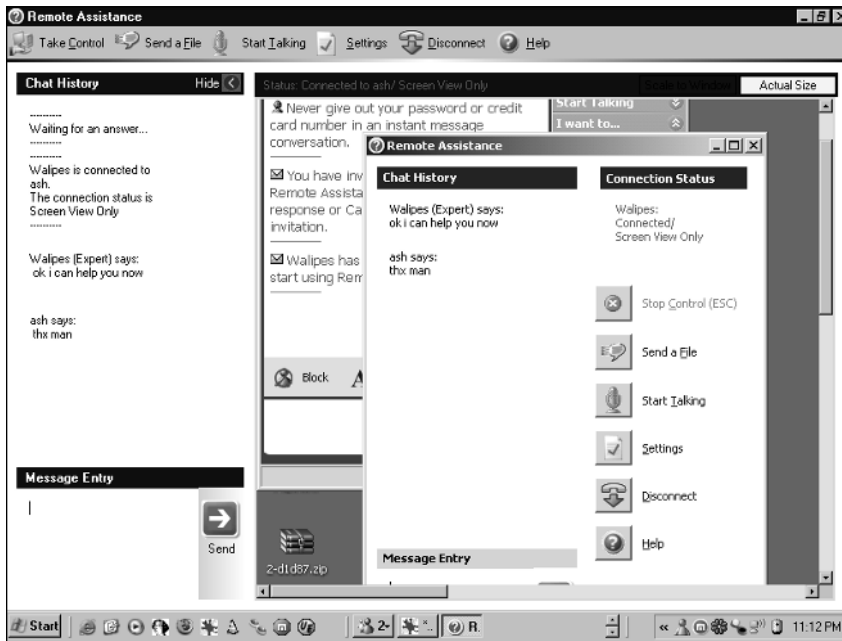
Figure 4.9 Application Sharing Toolbar**Figure 4.10** Accepting an Application Sharing Request

Figure 4.11 Selecting Programs to Share in MSN Messenger

Remote Assistance

The remote assistance function of MSN Messenger launches the remote assistance feature built into Windows XP Home and Professional editions. The feature is built on the Remote Desktop Protocol (RDP). For more information on RDP, visit Microsoft's support page at <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q186607>. The invitation mechanism to begin the Remote Assistance session is similar to all other client-to-client exchanges, detailed in the "Whiteboard" section of this chapter. Figure 4.12 shows an example of remote assistance. This feature uses TCP port 3389 for negotiating the connection between two workstations and sharing the desktop. This feature provides users with the ability to give another user complete control of their workstation as though the remote user was actually sitting in front of the workstation controlling it. All functions and programs are available to the remote user, including network access and all files on the remote workstation. By using this feature, any unauthorized or malicious user may be given access to sensitive files and information, and may use this functionality to install malicious code such as a backdoor or Trojan, compromising the security of the machine and allowing for future unauthorized access. Microsoft has provided documentation about remote access at http://microsoft.com/resources/documentation/Windows/XP/all/reskit/en-us/Default.asp?url=/resources/documentation/Windows/XP/all/reskit/en-us/prmb_tol_drft.asp.

Figure 4.12 Remote Assistance



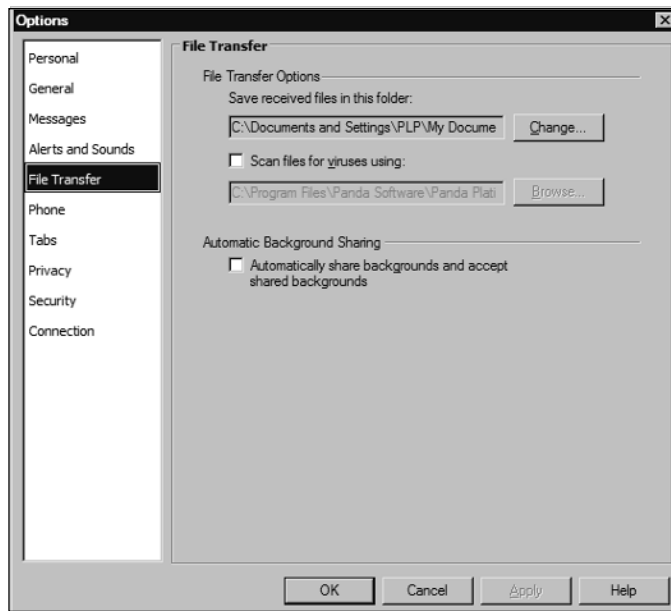
Voice Chat

Voice chat allows two users to communicate via voice chat. Once a small setup is run to ensure that a microphone is present on the workstation, a user can establish a conversation with another user. This functionality has similar security risks as instant messaging. By bypassing telephone systems and e-mail systems, users may communicate sensitive information to others. Like many features of instant messaging clients, it is difficult to authenticate the user you are communicating with, and you could possibly be entering into exchanges with unknown parties or users posing as acquaintances. Figure 4.13 shows a voice chat session initiating. Voice chat utilizes a wide range of ports, TCP 5004–65535.

Figure 4.13 MSN Messenger Voice Chat

File Transfer

MSN Messenger offers the functionality to send and receive files between users directly. This feature is not as robust or configurable as in other clients such as Yahoo! Messenger, but still provides a useful feature. The file transfer has to be accepted by the receiving user before it begins, and the process is similar to the one used to accept whiteboard and application sharing requests. Once a user initiates a file transfer on his or her workstation, a message is sent through the switchboard server to the intended user. This invitation is sent along with the name of file being sent and the size of the file. The recipient must reply to this invitation, and can either accept or deny the file transfer request. If the transfer is accepted, the message will contain the IP address of the recipient's workstation in order to begin the connection. The acceptance message is received by the client initiating the transfer and sends another file transfer message to reveal the IP address of the workstation sending the file, allowing the workstations to identify each other and begin a direct connection to start the file transfer. This activity occurs over a range of ports, TCP 6891-6900. Figure 4.14 shows the settings MSN Messenger provides for file transfers, while Figure 4.15 shows the request for a file transfer. This feature can be used to circumvent network controls that restrict the exchanging of file outside of an organization. A user can utilize this feature to avoid the FTP (File Transfer Protocol) and e-mail size restrictions set by an administrator and send and receive sensitive, confidential files without detection.

Figure 4.14 MSN Messenger File Transfer Preferences**Figure 4.15** MSN File Transfer Acceptance Dialog Box

Web Camera Settings

MSN Messenger provides features that enable the sharing of Web cameras for viewing or for a bi-directional conference. In order to use this feature, the same process for sending files and sharing applications apply, utilizing messages sent through the switchboard server to accept or deny requests for video conversations. Video and Web camera information communicates over a wide range of ports, TCP 5004–65535. This functionality may be used by a malicious user, however it is unlikely due to the need to accept a request and the more efficient communication methods that can be used with MSN Messenger.

Malicious Code and Client Security

This section of the book deals with several types of security issues concerning MSN Messenger. The Malicious Code section details security issues which involve worms, Trojans, or other malicious programs which spread via MSN's instant messaging service. The Client Security section of this chapter involves vulnerabilities that MSN Messenger has been exposed to.

Malicious Code

MSN Messenger has been targeted by more malicious users than any other instant messenger. This is due in part to its large distribution. Microsoft ships a version of MSN Messenger with Windows XP, making the client a large and attractive target for malicious users, who can reach a large, guaranteed user base for their attack. Additionally, since the client is developed by Microsoft, this increases the likelihood that it will be targeted by malicious users, many of whom simply do not like the company. According to the IMLogic Threat Center, MSN Messenger had 127 security threats between January 1, 2005 and June 1, 2005, of which 3 were rated as high severity and 8 were rated as medium severity security risks. In 2004, there were only 17 security threats against MSN Messenger, which is the largest increase over any other instant messaging client. Since instant messaging clients have the ability to spread information rapidly, they are an increasing target for malicious users as a way to distribute their code efficiently. In order to combat many of these worms that affect MSN Messenger, users should configure MSN Messenger's privacy options to allow messages only from users who are on their contact list. This will prevent messages from unknown users who may be distributing malicious URLs and code. Figure 4.16 shows the allow and block lists available in privacy options.

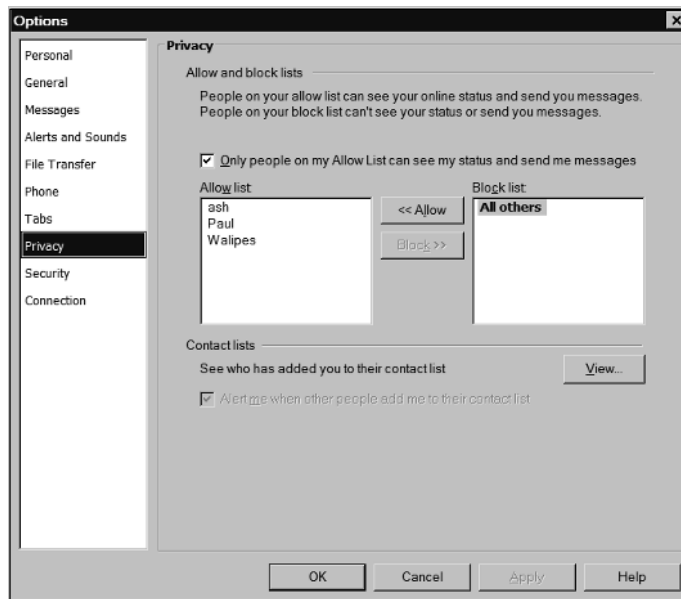
Figure 4.16 Allow and Block Lists

Table 4.5 lists the security threats for MSN Messenger reported in 2005 by the IMLogic Threat Center:

Table 4.5 MSN Messenger Threats 2005

Name	Severity	Date
Microsoft PNG vulnerability	High	February 8, 2005
MSN Messenger GIF Image Processing Vulnerability	High	April 12, 2005
W32/Kelvir-Re	High	April 14, 2005
W32/Bropia-A	Med	January 20, 2005
W32.Bropia.J	Med	February 2, 2005
W32/Kelvir-A	Med	February 25, 2005
W32/Kelvir-B	Med	March 6, 2005
W32/Kelvir-D	Med	March 7, 2005
W32/Sumom-A	Med	March 7, 2005

Continued

Table 4.5 continued MSN Messenger Threats 2005

Name	Severity	Date
W32.Kelvir.P	Med	April 11, 2005
Weebs.World	Med	June 22, 2005
WM97/Lebani-A	Low	January 1, 2005
Troj/LdPinch-JD	Low	January 23, 2005
Troj/Singu-P	Low	January 24, 2005
W32.Unfunner.A	Low	January 28, 2005
W32/Mugly.i@MM	Low	January 30, 2005
WORM_WOOTBOT.GX	Low	January 31, 2005
W32/Ahker-B	Low	February 3, 2005
Worm.Bropia.O	Low	February 15, 2005
W32.Serflog.C	Low	February 15, 2005
Worm_Stang.A	Low	February 23, 2005
W32/Bropia-S	Low	February 27, 2005
Bropia.F	Low	March 2, 2005
W32/Sumom-B	Low	March 10, 2005
W32.Kelvir.H	Low	March 14, 2005
W32.Kelvir.G	Low	March 14, 2005
W32.Chod@mm	Low	March 14, 2005
W32/Sumom-C	Low	March 17, 2005
W32/Chode-A	Low	March 18, 2005
W32.Kelvir.I	Low	March 18, 2005
Trillian Multiple Plug-ins Buffer Overflow Vulnerability	Low	March 25, 2005
Troj/Agent-CT	Low	March 29, 2005
W32/Kelvir-F	Low	March 29, 2005
W32.Kelvir.K	Low	April 1, 2005
W32.Chod.B@mm	Low	April 2, 2005
W32.Kelvir.L	Low	April 3, 2005
WORM_CHOD.B	Low	April 4, 2005

Continued

Table 4.5 continued MSN Messenger Threats 2005

Name	Severity	Date
W32.Kelvir.M	Low	April 4, 2005
W32/Kelvir-G	Low	April 4, 2005
W32/Bropia-L	Low	April 7, 2005
W32.Kelvir.O	Low	April 7, 2005
W32/Kelvir-H	Low	April 7, 2005
W32.Kelvir.Q	Low	April 12, 2005
Bloodhound. Exploit.33	Low	April 12, 2005
W32.Kelvir.Q	Low	April 12, 2005
W32.Kelvir.R	Low	April 12, 2005
Bloodhound. Exploit.33	Low	April 12, 2005
W32.Kelvir.V	Low	April 14, 2005
W32.Kelvir.S	Low	April 14, 2005
W32.Kelvir.U	Low	April 14, 2005
W32.Picrate.A @mm	Low	April 14, 2005
W32/Kelvir-I	Low	April 14, 2005
W32.Kelvir.W	Low	April 15, 2005
W32/Kelvir-Q	Low	April 15, 2005
W32/Kelvir-J	Low	April 15, 2005
W32.Kelvir.X	Low	April 16, 2005
W32.Kelvir.Y	Low	April 17, 2005
W32.Picrate.B @mm	Low	April 17, 2005
W32.Kelvir.AA	Low	April 18, 2005
W32.Kelvir.AB	Low	April 18, 2005
W32.Kelvir.AC	Low	April 19, 2005
Troj/Kelvir-P	Low	April 19, 2005
Troj/Kelvir-M	Low	April 19, 2005
W32.Kelvir.AF	Low	April 20, 2005
W32/Chode-B	Low	April 20, 2005

Continued

Table 4.5 continued MSN Messenger Threats 2005

Name	Severity	Date
W32/Kelvir-L	Low	April 20, 2005
Troj/Kelvir-O	Low	April 20, 2005
WORM_KELVIR.Z	Low	April 21, 2005
W32.Kelvir.AJ	Low	April 22, 2005
W32.Kelvir.AI	Low	April 22, 2005
W32.Kelvir.AH	Low	April 22, 2005
W32.Velkbot.A	Low	April 23, 2005
W32.Kelvir.AL	Low	April 23, 2005
W32.Kelvir.AN	Low	April 24, 2005
W32/Kelvir-S	Low	April 25, 2005
W32.Kelvir.AO	Low	April 25, 2005
W32.Kelvir.AP	Low	April 26, 2005
W32/Kelvir-T	Low	April 26, 2005
W32.Kelvir.AP	Low	April 27, 2005
W32.Kelvir.AW	Low	April 28, 2005
W32/Bropia. worm.aj	Low	April 28, 2005
W32.Kelvir.AX	Low	April 28, 2005
Kelvir-AZ	Low	April 29, 2005
W32.Kelvir.BA	Low	May 1, 2005
W32.Kelvir.BD	Low	May 2, 2005
Troj/Kelvir-V	Low	May 3, 2005
W32.Kelvir.BF	Low	May 4, 2005
WORM_KELVIR.AQ	Low	May 4, 2005
W32/Kelvir-W	Low	May 10, 2005
Kelvir.AS	Low	May 11, 2005
W32/Kelvir-Gen	Low	May 12, 2005
Win32.Bropia.AQ	Low	May 12, 2005
W32/Kelvir-U	Low	May 18, 2005
W32/Kelvir-Y	Low	May 18, 2005
W32.Kelvir.CG	Low	May 19, 2005

Continued

Table 4.5 continued MSN Messenger Threats 2005

Name	Severity	Date
WORM_KELVIR.W	Low	May 19, 2005
W32/Kelvir. worm.bh	Low	May 21, 2005
WORM_BROPIA.W	Low	May 21, 2005
W32/Kelvir-Z	Low	May 25, 2005
W32/Bropia-V	Low	May 26, 2005
WORM_KELVIR.BE	Low	May 26, 2005
W32/Kelvir-AA	Low	May 26, 2005
WORM_KELVIR.AF	Low	June 1, 2005
Troj/Kelvir-AC	Low	June 5, 2005
W32/Chode-C	Low	June 6, 2005
W32/Kelvir-AD	Low	June 7, 2005
W32/Kelvir-AE	Low	June 9, 2005
W32.Kelvir.DA	Low	June 13, 2005
W32.Kelvir.DE	Low	June 14, 2005
W32.Kelvir.DD	Low	June 14, 2005
W32/Kelvir-AF	Low	June 14, 2005
Worm/Exchange-x3	Low	June 17, 2005
WORM_KELVIR.BK	Low	June 22, 2005
W32/Kelvir-AP	Low	June 23, 2005
W32/Kelvir-AG	Low	June 24, 2005
W32/Kelvir-AH	Low	June 24, 2005
W32.Kelvir.DQ	Low	June 26, 2005
W32/Kelvir-CB	Low	June 27, 2005
W32/Kelvir-AI	Low	June 27, 2005
W32/Kelvir-AJ	Low	June 27, 2005
W32.Kelvir.DR	Low	June 27, 2005
Troj/Multidr-DP	Low	June 27, 2005
W32.Kelvir.DT	Low	June 28, 2005
W32.Kelvir.DU	Low	June 29, 2005
W32/Kelvir-AK	Low	June 29, 2005

Continued

Table 4.5 continued MSN Messenger Threats 2005

Name	Severity	Date
Troj/Singu-R	Low	June 30, 2005
W32/Kelvir-BR	Low	June 30, 2005

Worm

The following sections demonstrate examples of potentially malicious MSN Messenger Worms.

W32.Kelvir.R

W32.Kelvir.R was discovered on April 12, 2005, and Symantec's Research Center (<http://sarc.com>) describes it as the following:

When W32.Kelvir.R is executed, it performs the following actions:

1. Sends the following message with a link to all the MSN Messenger contacts on the compromised computer:
 - picture of you!
 - [www.\[domain removed\].com/pics/pictures.php?email=](http://www.[domain removed].com/pics/pictures.php?email=)

NOTE

To activate the worm, a recipient must click on the link, download the file (unknown@hotmail.com), and then execute the file. The downloaded file unknown@hotmail.com is a copy of the worm. It's a self-extracting RAR file.

2. Drops the following files after the file unknown@hotmail.com self-extracts the following files:
 - **%ProgramFiles%\KEVIN\rBot.exe** a variant of W32.Spybot.Worm.

- **%ProgramFiles%\KEVIN\kevin.exe** a worm component that sends the abovementioned message to all the MSN Messenger contacts on the compromised computer.
3. Adds the value “C%%Program Files%KEVIN” = “%ProgramFiles%\KEVIN” to the HKEY_CURRENT_USER\Software\WinRAR SFX registry subkey.
 4. Copies itself as %System%\svchost32.exe. It sets the file attributes to hidden, read only, and system.
 5. Adds the value “Universal USB Service” = “svchost32.exe” to the following registry subkeys so that the W32.Spybot.Worm variant runs every time Windows starts:
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
 - HKEY_CURRENT_USER\Software\Microsoft\OLE
 6. Adds the value “load” = “C:\Program Files\KEVIN\Kevin.exe” to the HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows registry subkey.
 7. Spreads the W32.Spybot.Worm variant by exploiting the following vulnerabilities:
 - The Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability (described in Microsoft Security Bulletin MS03-026).
 - The Microsoft Windows Local Security Authority Service Remote Buffer Overflow (as described in Microsoft Security Bulletin MS04-011).
 - The vulnerabilities in the Microsoft SQL Server 2000 or MSDE 2000 audit (as described in Microsoft Security Bulletin MS02-061) using UDP port 1434.
 8. Attempts to enumerate users in order to copy itself to network shares.
 9. Performs any of the following actions:
 - Opens a back door on the compromised computer, allowing a remote attacker to have unauthorized access.

- Steals CD activation keys for many games.
- Attempts to terminate processes and services, some of which may be security related.
- Installs a keylogger.
- Uses the compromised computer as a traffic relay or proxy.

W32.Picrate.C@mm

W32.Kelvir.U is one of many variants of the Kelvir worm. This particular variant was posted on Symantec's SARC website on April 14, 2005. SARC describes it as the following:

When W32.Kelvir.U is executed, it performs the following actions:

1. Creates the following files:
 - C:\Service.exe
 - %Windir%\hosts.exe
2. Adds the value "Windows Host" = "%Windir%\hosts.exe" to the following registry subkeys so that it is executed every time Windows starts:
 - HKEY_CURRENT_USER\Software\Microsoft\CurrentVersion\Run
 - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\CurrentVersion\Run

Attempts to end the following services:

Ahnlab Task Scheduler	AVP control center service
altiris client service	AVP.EXE
ANTIVIR	AVP32
ATRACK	AVSync Manager
avast! antivirus	AVSYNMGR
avast! iavs4 control service	Background Intelligent Transfer Service
AVCONSOL	BlackICE
AVG6 Service	carbon copy access edition
AVG7 Alert Manager Server	CFINET
AVG7 Update Service	

CFINET32	IOMON98
config loader	iroff
Detector de OfficeScanNT	Kaspersky
directupdate engine	Kaspersky Antivirus
dllhost	Kaspersky Anti-Virus
dns	kaspersky auto protect service
eTrust Antivirus Job Server	Kaspersky Client
etrust antivirus job server	kav
eTrust Antivirus Realtime Server	KAV Monitor Service
etrust antivirus realtime server	kerio personal firewall
eTrust Antivirus RPC Server	Kingsoft AntiVirus Service
etrust antivirus rpc server	LOCKDOWN2000
Eventask	LUALL
FireBall	LUCOMSERVER
FireBaum	MastDLL
fix-it task manager	MCAFEE
F-PROT95	McAfee Agent
FP-WIN	mcafee framework service
fxsvc	McAfee.com McShield
gear security	McAfee.com VirusScan Online
IAMAPP	Realtime Engine
ICMON	mcshield
intel file transfer	MonSvcNT
intel pds	msclol2
Internet Connection Firewall (ICF) / Internet Connection Sharing (ICS)	msclol8
internet prOtocol	msinit
InternetFirewallProc	MsInt
	MsIntScan
	NAV Alert

NAV Auto-Protect	RemoteAgent
NAVAPW32	remotely possible/32
NAVW32	rising process communication center
NISSERV	Rising Process Communication Center
NISUM	rising realtime monitor service
NMAIN	Rising Realtime Monitor Service
noipducservice	rundll
NORTON	SAFEWEB
Norton Internet Security Proxy Service	savroam
Norton Internet Security service	ScriptBlocking Service
Norton Unerase Protection	scvhost
ntiVirus Corporate Edition	secur2
NVC95	Security Center
nvscv	services32 service: msinit
officescant listener	servu
OfficeScanNT Monitor	Serv-U
officescant realtime scan	serv-u-ftp
outpost firewall service	smss
P2P Networking	snake sockproxy service
Panda Antivirus	Sophos Anti-Virus
pcanywhere host service	Sophos Anti-Virus Network
PC-cillin Personal Firewall	Sygate Personal Firewall
PCCIOMON	Sygate Personal Firewall Pro
PCCMAIN	SyGateService
PCCWIN98	symantec antivirus
POP3TRAP	symantec central quarantine
psaxesvc	symantec quarantine agent
Quick Heal Online Protection	

symantec quarantine scanner	vnc server
syslock	VNC server
System Event Notification	VSHWIN32
systemsecuritydll	VSSTAT
task manager	WEBSCANX
Trend Micro Proxy Service	WEBTRAP
Trend NT Realtime Service	win32sl
V3MonNT	Windows Firewall
V3MonSvc	Windows Internet Connection Sharing(ICS)
ViRobot Expert Monitoring	ZoneAlarm
ViRobot Lite Monitoring	
ViRobot Professional Monitoring	

4. Downloads a copy of W32.Spybot.NLI from the IP address 65.75.134.170. The worm saves the file as C:\Service.exe and executes it.
5. Sends one of the following instant messages to all the MSN messenger contacts on the compromised computer:
 - What a loser, who does something like this
 - Getting f*cked is never the same, see this!
 - This face, it looks like a alien
 - People say this is real, u might wanna check this out
 - Who does something like this..
 - Bleh :| What a filthy sh*t is this, dude check it out.
6. The instant message will contain a URL pointing to one of the following domains:
 - ubb.cc
 - dd.vg
 - 00mbitde.info

7. Downloads a copy of W32.Kelvir.U onto the compromised computer if the recipient visits the URL.

Client Security

In order to minimize your exposure to worms, viruses, and other malicious code, it is recommended that you block instant messaging services altogether. MSN Messenger can be blocked for most users by restricting access to port 1863 and login.passport.com, which will prevent the client from authenticating to the service. However, advanced users may be able to configure a proxy server to evade network controls that restrict usage of MSN Messenger.

MSN updates its client when new features are added to the service or when security issues are reported and have to be patched. According to Secunia's database, (<http://secunia.com/>) MSN Messenger has been susceptible to 7 vulnerabilities since 2003. Some of the vulnerabilities have led to increased privileges for a malicious user and the ability to remotely execute arbitrary code, such as the vulnerability detailed on April 12, 2005:

Vulnerability Description

Hongzhen Zhou has reported a vulnerability in MSN Messenger, which can be exploited by malicious people to compromise a user's system.

The vulnerability is caused due to an error within the processing of GIF images and can be exploited by sending a specially crafted emoticon or display picture to a user.

Successful exploitation allows execution of arbitrary code with the privileges of the user running MSN Messenger, but requires that it is possible to send messages to the user (only possible for people in a user's contact list by default).

The vulnerability affects versions 6.2 and 7.0 beta.

Vulnerability Solution

Apply patch. MSN Messenger 6.2: www.microsoft.com/download/details.aspx?id=5556-D4D0-42D6-9F05-1FF3C799BB10

Vulnerabilities are remedied by Microsoft releasing a new version of their client. Unless there is a specific policy related to the installation of instant messaging clients, users need to be vigilant and always keep aware of security issues and what steps need to be taken to lessen the exposure. Rather than rely on users who may not be aware of these issues, it is recommended that the software be centrally managed in order to distribute updates and protect against malicious users and vulnerable code. In order to prevent attacks against vulnerable software, it is recommended that it is removed or disabled if it is not critical.

Summary

MSN Messenger contains features that are similar to other instant messaging clients, including communication through text, voice, and video. These features that provide communication cannot be regulated and logged in ways that other systems, such as e-mail, can. These unregulated communications may result in the distribution of confidential information to parties that should not be privy to such information. Social engineering may play a part in data leakage, by convincing users to part with sensitive information or files. By posing as a known contact, a malicious user may be able to obtain information easily. It is recommended that sign-in and passwords are not stored on the client, which will reduce the risk of someone using your workstation to masquerade as you to obtain information from your contacts. Another way to safeguard information would be to limit sensitive and personal information from being discussed with others online. You should never give out credit card or other personal information to anyone online. Without the network controls that are available to other systems, files and executables can be transferred with almost no safety, creating avenues of attack for malicious users. These transfers can also involve the movement of copyrighted materials, such as MP3s, movies, and licensed software. MSN, like other instant messaging clients, does not have a restriction on the size of file transfers.

Malicious code has the ability to spread quickly through instant messaging networks, circumventing protection that has been put in place to prevent worms and viruses from spreading. Instant messaging clients can bypass security devices such as firewalls, URL filtering, and gateway security devices, promoting the spread of malicious code. Many of these worms spread via URLs displayed within a message from a contact. By clicking on such a link, a user will infect his or her workstation. Users should never click on a URL embedded within an instant message, especially from an unknown contact.

Solutions Fast Track

MSN Messenger Architecture

- ☑ MSN Messenger utilizes many different servers to carry out its functions. For signing into the service, it uses five different types of servers in order to authenticate username and password information.
- ☑ The switchboard server is responsible to communication between two clients, while the notification server sends a particular client's status and

presence information to anyone on your contact list to determine whether or not you are online.

- ☑ When utilizing features that require a direct connection, MSN Messenger sends an accept/refuse confirmation and negotiates the IP addresses necessary for connection. This process is handled through the switchboard server.
- ☑ Advanced users may be able to set up a proxy server to bypass controls restricting access to instant messaging services. MSN Messenger included the ability to connect to SOCK4, SOCK5, and HTTP proxies.
- ☑ The MSNP protocol version 2 was released to the public as an IETF draft in 1999. No other information has been made public about the protocol, but documentation exists on the Internet regarding details of the protocol.
- ☑ MSN Messenger utilizes the .NET Passport for signing into the service. This can be used for multiple websites in the MSN network. The credentials are encrypted to avoid packet-capturing software from gaining access to this information.

Features and Security Information

- ☑ MSN Messenger can archive all conversations that take place using the client. The file that contains the message history is stored locally on the workstation, and can be configured to be in any location. This file is not encrypted, and can be accessed and read by anyone who has access to your workstation.
- ☑ The client has the ability to launch utilities to share applications or give control of the workstation to a remote user. This functionality can be dangerous because of the amount of information that can be obtained from a user who may be unknown. By sharing applications and remote controlling the desktop, the remote user has access to all information available to the workstation, including local files and information contained on network shares.
- ☑ File transfers can be used to send large files to other users outside of a particular network or organization. Since there are no restrictions on file sizes sent in this manner, a user can send confidential or sensitive information as well as exchange files that infringe on copyright laws.

Unlike e-mail servers, there is no centralized way to log this information so it is difficult to evaluate what information was sent outside an organization.

Malicious Code and Client Security

- ☑ Malicious code such as worms are generally distributed through instant messaging services via URLs. Some worms make these messages appear as though these URLs are coming from a known user, increasing the chance that these URLs will be visited, thereby infecting a workstation.
- ☑ Some worms have the ability to receive remote commands to control the infected workstation. This gives a malicious user complete control over the workstation, including access to all information stored on the workstation.
- ☑ MSN Messenger is the largest target of all instant messaging clients. Worms and other malicious code have been created to take advantage of its wide distribution.
- ☑ There have been multiple high priority vulnerabilities that affect MSN Messenger. One vulnerability, an error in the processing of GIF files, allows a remote user to execute arbitrary code on the affected workstation.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: How does MSN Messenger differ from AIM and Yahoo?

A: MSN Messenger shares a similar feature set with other instant messaging clients, such as text messaging, file transfers, voice chat, and Web camera support. These basic features are available to all the major instant messaging clients. There are, however, some features which are unique to MSN Messenger, such as white boarding and remote assistance. These features provide users of the MSN Messenger service with the ability to collaborate on a whiteboard or to send a request for another user to take control of their workstation. Besides these features, MSN is also able to provide integration with MSN Web properties and provide data from the network of Web sites. Hotmail notifications, CNBC news, and personalized notifications are available from this client.

Q: Why is MSN more complicated than other messaging clients?

A: Although MSN’s service architecture is more complicated than other instant messaging services, users of the client should notice no difference in its ease of use when compared with other instant messaging clients. The choice and complexity of an instant messaging client is very subjective, and each user may feel more comfortable with a different client. Additionally, the choice of a client will depend on what service your clients are using. If all of your contacts are using MSN, the only way to communicate with these people is by using the MSN client as well. Another option would be to install a multi-protocol client, which has the ability to connect to multiple services. Multi-protocol clients will be covered in Chapter 6.

Q: What is the best way to secure MSN from worms, viruses, and other malicious code?

A: The best way to protect yourself from malicious code such as worms and viruses is to refrain from using an instant messaging client altogether. This is not always feasible, and instant messaging is a very useful communication tool and can be used in ways to minimize exposure to malicious code. Since most worms are spread as links within text messages, it is recommended that users do not click on any URLs sent within messages unless absolutely sure of the source. Another way to prevent the spread of malicious code is to restrict who you communicate with. There are settings in all instant messaging clients to limit communication with others unless they have previously been added to your contact list. By restricting your communications, you can prevent an unknown user from sending worms or other malicious code to your workstation.

Q: Why is MSN the most targeted messaging client by malicious users?

A: MSN's instant messaging service is targeted by the writers of malicious code due to its large distribution and because it is developed by Microsoft. Microsoft Windows ships with a version of MSN Messenger by default, which is one of the reasons for this service's popularity. However, since it has been installed on millions of workstations by default, it has become a very large target for virus writers. Rather than concentrate on clients with minimal distribution, these malicious users can target MSN and create a large amount of damage for the same amount of work. Another reason for MSN being targeted over other instant messaging clients is because malicious users, in many cases, prefer to target Microsoft products. There are many malicious users who do not like Microsoft and target their products to bring bad press and harm to the company. By creating malicious code, these users hope to portray Microsoft's products as insecure and drive users to other competing products.

ICQ

Solutions in this chapter:

- Introduction and History of ICQ
- ICQ Features
- Malicious Code
- Client Security

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction and History of ICQ

ICQ can be considered the first modern, freely available instant messaging client for Microsoft Windows-based workstations. Although there were several clients for UNIX systems (such as talk), and IRC, which allowed for communication with others, ICQ was the first popular client produced for Microsoft Windows. ICQ was the work of an Israeli-based company, Mirabilis, which was formed in the middle of 1996. Its sole product was met with immediate success after its launch in November 1996, with an impressive 850,000 subscribers for the service within six months. Part of the product's popularity was due to the software providing a way for users to send invitations to their friends to join them in online messaging. This viral distribution of its software was also successful due to its ability to work on current computer systems, and only required a connection to the Internet to begin chatting. Any PC user, with any ISP (Internet Service Provider), was able to install ICQ and begin sending messages to their friends. Being the only program of its kind for Windows-based workstations was helpful as well in boosting the software's popularity. All this activity was noticed by larger companies, including AOL, who were interested in the massive and rapidly growing userbase that ICQ generated.

AOL already had multiple chat systems available to members for a few years, and even introduced a system similar to ICQ for its members. This was not as successful as ICQ since it was available exclusively to AOL subscribers, and did not allow AOL to increase its membership and advertising potential. In 1997, AOL released AOL Instant Messenger (AIM), which was freely available to non-AOL subscribers. This allowed the AIM service to grow rapidly, and became a strong competitor to ICQ. However, ICQ was still the most dominant instant messaging service, and in 1998, AOL acquired Mirabilis and its ICQ technology for \$287 million. AOL has operated both instant messaging systems as separate services up until today.

Since ICQ was developed as a completely separate service from AOL, it relied on a different protocol and used different servers for its functionality. Beginning in 1999, coinciding with the release of the ICQ2000 client, the ICQ protocol was changed to mirror much of AIM's proprietary OSCAR protocol. The ICQ Protocol v7 was released with this client, and continues to evolve in step with the OSCAR protocol. In October 2002, AOL released AIM beta 5.1, which gave AIM users the ability to communicate with ICQ users. This interoperability allowed for users on both services to communicate with each other, without having to sign up for both services or load both clients. In fact, version 5.9 of AIM, released in September 2004, allows ICQ users to log into the ICQ service with the AIM client. When logging into the services, the major difference between AIM and ICQ is the way screen names are handled. AIM uses names, while ICQ assigns a unique number (ICQ#) to

each user who signs up for the service. Users of ICQ can, in turn, create a screen name to make it easier for users to look them up in the ICQ directory. These screen names are non-unique and can be changed, however the ICQ# must remain the same to identify each user. Detailed information on the ICQ protocol can be found at www.micq.org/ICQ-OSCAR-Protocol-v7-v8-v9/index.html.

In December 2002, AOL was granted patents for much of the core functionality surrounding instant messaging. This patent was filed by Mirabilis in 1997, and could have been another reason for AOL wishing to acquire its rival in the instant messaging space. US Patent #6449344 grants rights to instant messaging functionality such as tracking of user's presence on a service, the ability to create lists of users and track their online status, and locating users online for instant messaging purposes.

ICQ Features

For connection and communication, ICQ utilizes the same ports as does AIM, which are detailed in Chapter 2. For connecting to the ICQ service, the client uses the address login.icq.com, which resolves to the same servers that AIM does, and communicates via TCP (Transmission Control Protocol) port 5190 by default. Figure 5.1 shows the Connection Settings screen for ICQ.

Figure 5.1 ICQ Connection Settings



Instant Messaging

Both ICQ and AIM have very similar features, now that both programs are owned and distributed by the same company, AOL-Time Warner. There are minimal differences between the two instant messaging clients such as offline message delivery and message logging. Offline message delivery allows a user to send messages to an offline contact. When the recipient signs into the ICQ service, he or she receives the messages that were sent while they were offline. Another feature unique to the ICQ client is the ability to send Short Message Service (SMS) messages to mobile phones directly through the client. When the message window is open, a user can specify whether to send the message via client, SMS, or both. Figure 5.2 shows an example of a messaging session in ICQ. Instant messaging provides malicious users with a tool for social engineering, and can be used to entice a user to divulge sensitive or proprietary information. Malicious users may pose as an acquaintance or as an employee of an instant messaging service. In either case, since it is nearly impossible to verify the identity of the user, it is recommended that personal, confidential, or other sensitive information not be discussed via instant messaging. The likelihood of a user masquerading as an acquaintance is made more likely due to programs such as Messenger Key, which enables a user to recover passwords from an ICQ client. Figure 5.3 shows Messenger Key recovering a password.

Figure 5.2 ICQ Instant Messaging Session

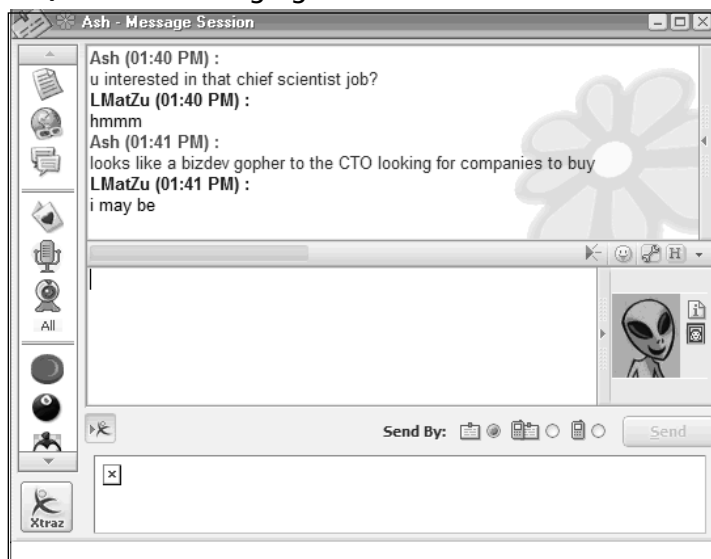
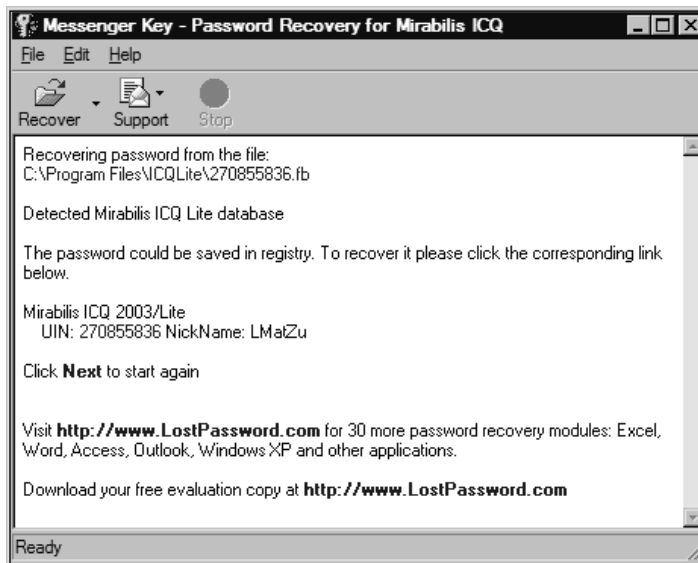


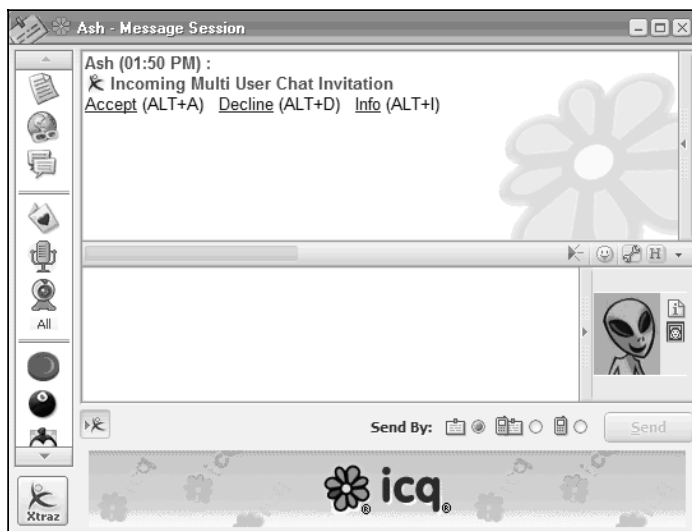
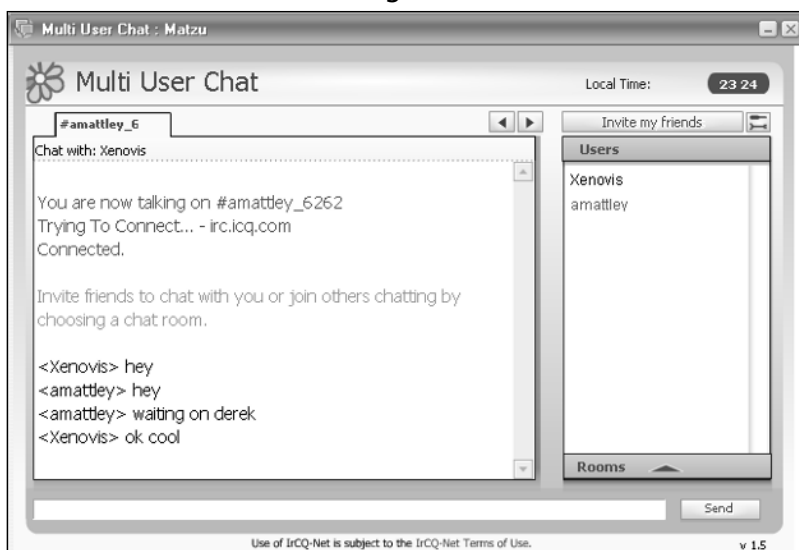
Figure 5.3 Messenger Key Recovering ICQ Passwords

Encryption

ICQ, like most other instant messengers, does not provide any encryption for messages between users. Encryption is necessary to prevent packet-capturing utilities from gaining access to all conversations that are exchanged through the use of instant messaging services. Utilizing packet-capturing utilities, such as Cain and Abel (www.oxid.it/cain.html), malicious users have the ability to record and play back conversations between each other via instant messaging software. Conversations with others over instant messaging services may be intercepted without your knowledge. It is recommended that users do not discuss sensitive or confidential information over instant messaging since it may be intercepted and viewed by others.

Group Chat

Group chat, known as Multi User Chat in ICQ, initiates a private room in which multiple users can communicate with each other at the same time. Similar security issues that affect instant messaging also apply to group chat, and users must accept an invitation before joining a session. Figure 5.4 shows an invitation for a Multi User Chat session, while Figure 5.5 shows a session in progress.

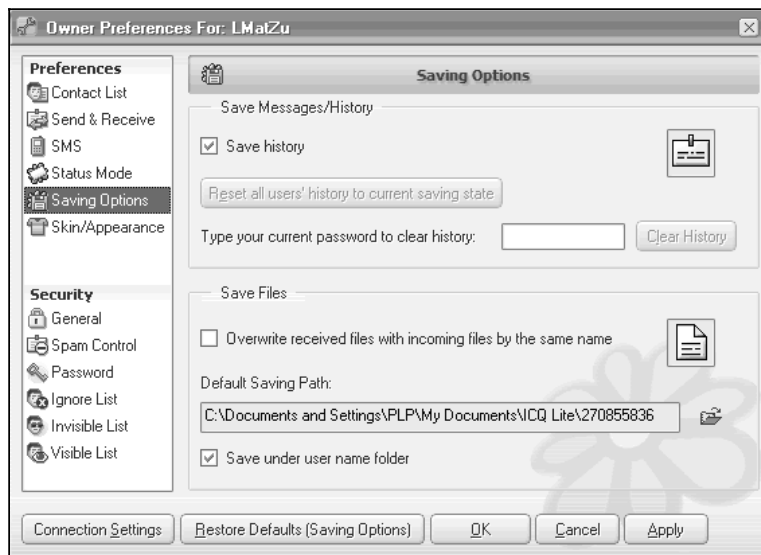
Figure 5.4 Multi User Chat Invitation**Figure 5.5** Multi Chat Session In Progress

Message Archiving

ICQ provides an option to allow users to record all conversation locally and export these messages to a text file on their local workstations. This feature is disabled by default, but can be enabled to record conversations for one or all users that you

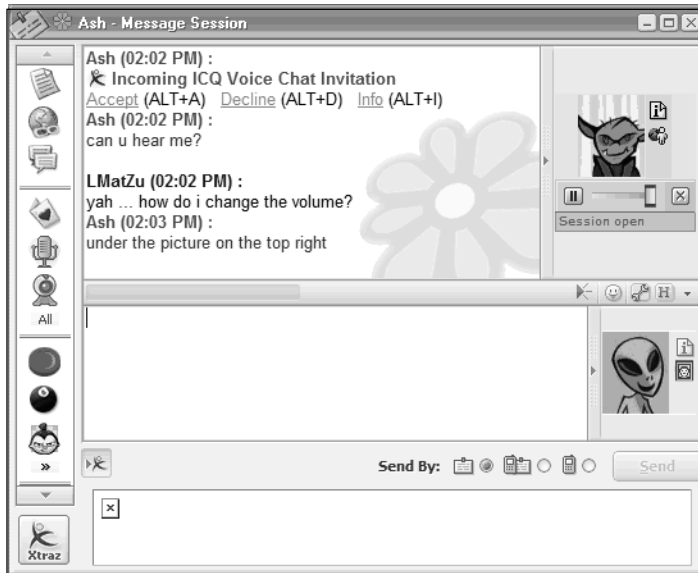
communicate with. ICQ provides options for managing the conversation record, including deleting entries and exporting to a file anywhere on your workstation. This file is written in plain text, and is not encrypted. This may pose a security risk if confidential or sensitive information is discussed and later logged, allowing any user with access to your workstation to read the contents of any conversation or copy this file for later retrieval. If this feature is enabled, users should manage these files on a regular basis and delete them after a period of time. Figure 5.6 details options related to the message history.

Figure 5.6 ICQ History Options



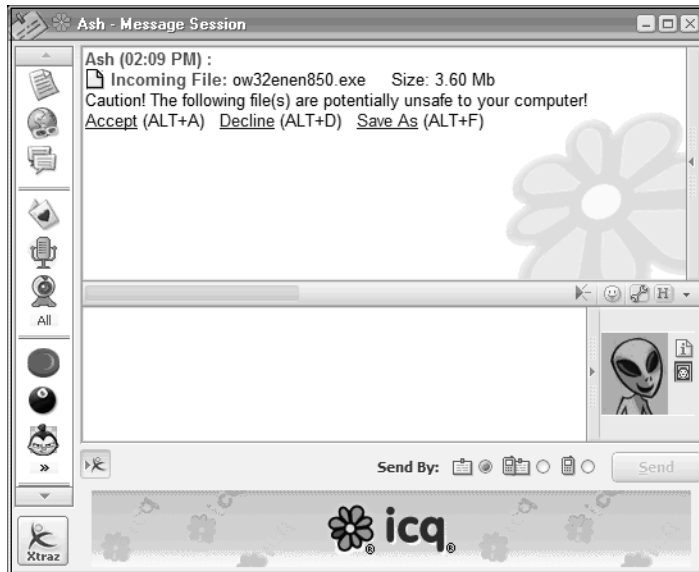
Voice Chat

Provided that two users have a microphone and speakers, Voice Chat establishes voice communication. This feature provides similar security risks as instant messaging, bypassing phone, e-mail, and other possibly regulated communication systems, allowing sensitive information to be communicated to people outside an organization without their knowledge. Users could mistakenly distribute confidential information to unauthorized parties with this feature. Figure 5.7 is an example of a voice chat session using ICQ.

Figure 5.7 ICQ Voice Chat

File Transfer

The file transfer feature of ICQ offers users the ability to exchange files of any size with one another directly, bypassing restrictions that may be in place in an organization. The recipient must accept the file transfer in order for the exchange to begin. The client warns users that engaging in a file transfer creates a direct connection between two users, and results in the ability to see the IP (Internet Protocol) addresses of both workstations involved in the transfer since there is a connection between the two. Files are saved in a specified location in the preferences screen for Saving Options. Figure 5.8 shows the dialog box used for accepting file transfers over ICQ. A user can utilize this feature to avoid the FTP (File Transfer Protocol) and e-mail size restrictions set by an administrator and send and receive sensitive, confidential files without detection.

Figure 5.8 ICQ File transfer Dialog Box

Web Camera Settings

ICQ includes a feature to enable the sharing of video over a webcam. This feature can be used by malicious users, but it is unlikely due to the way ICQ allows webcam sessions to begin. Just as in file transfers, a user sends a request to start a video session, and the recipient must accept the invitation to begin the session. There are more efficient methods for sending information, such as file transfers and instant messages, but this feature may provide the ability for a malicious user to see restricted areas that would otherwise be off limits.

Malicious Code

ICQ is not often the target of users writing malicious code. This is due to its slipping popularity, where authors of worms and viruses would rather spend their time writing code for programs with a larger audience that will result in increased damage. Of all the mainstream instant messaging clients, ICQ is the least used program, with only 6% of the market for instant messaging clients according to Media Metrix (Chapter 1 has the complete list of clients and usage details). This is a fairly unattractive target for writers of malicious code, especially when compared to larger targets. According to the IMLogic Threat Center, the ICQ client had only 9 security threats between January 1, 2005 and June 1, 2005. All of these security threats were

rated low. This is a sharp contrast to MSN Messenger, which had 127 security threats during the same period of time. In 2004, there were only 5 security threats against the ICQ client, which shows that although there is a growing threat, it is not as large as those affecting more popular clients such as MSN Messenger and AIM. Most worms that affect ICQ (as well as other instant messaging clients) spread via URL (Uniform Resource Locator) links embedded in messages. To prevent infection from these worms, it is recommended that you change the security settings of ICQ to restrict messages from unknown users. ICQ provides features in its spam control options to only accept messages from users on your contact list. This can prevent many worms from affecting your workstation and spreading, but you should also be wary of clicking on URLs embedded in messages. Figure 5.9 shows the options available to ICQ users that restrict messaging from unknown users.

Figure 5.9 Spam Control in ICQ

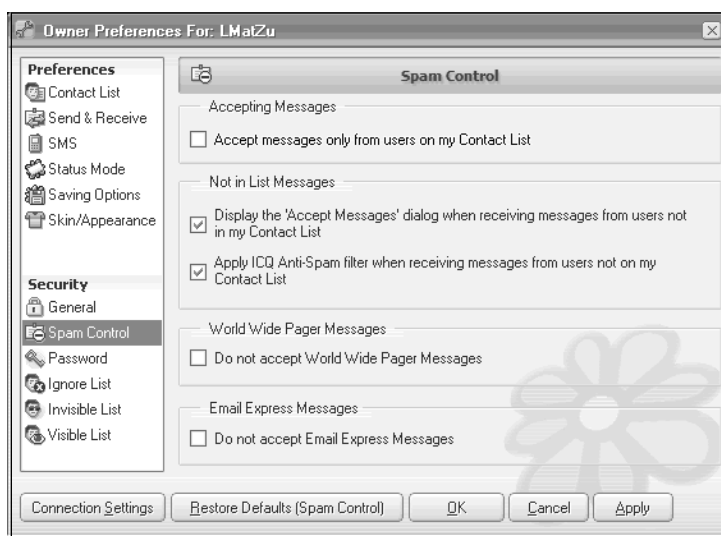


Table 5.1 lists the security threats for the ICQ client reported in 2005 by the IMLogic Threat Center:

Table 5.1 IMLogic's 2005 ICQ Client Security Threats

Name	Severity	Date
Troj/LdPinch-JD	Low	January 23, 2005
WORM_VAMPIRE.A	Low	January 25, 2005
W32/MyDoom-AN	Low	January 27, 2005
Troj/LdPinch-AR	Low	March 17, 2005
Troj/Agent-CT	Low	March 29, 2005
W32.Chod.B@mm	Low	April 2, 2005
Troj/LdPinch-AV	Low	April 19, 2005
Troj/LanFilt-I	Low	April 26, 2005
Troj/LdPinch-BA	Low	May 30, 2005

Worm Examples

The following are two examples of recent malicious code that affect ICQ.

WORM_VAMPIRE.A

WORM_VAMPIRE.A was discovered on January 25, 2005, and is described by Trend Micro (<http://trend.com>) as the following:

- Upon execution, this worm drops SYSMMSG32.EXE in the Windows system folder.
- It creates a registry entry to ensure its automatic execution at every system startup.
- It locates the address book for the ICQ instant messenger client and tries to send itself to all the contacts that it finds. Every 1–20 minutes, it hides the file transfer window while it sends itself to other ICQ users as THIS ROX.ZIP.EXE, which ICQ displays as THIS ROX.ZIP on a target recipient's window.

The worm's activities include the following:

- Hamper peer-to-peer file transfer
- Display unwanted messages
- Redirect file transfer
- Threaten user's privacy

WORM_VAMPIRE.A Installation and Autostart Technique

Upon execution, the WORM_VAMPIRE.A worm drops SYSMSG32.EXE in the Windows system folder.

It creates the following registry entry to ensure its automatic execution at every system startup:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
CurrentVersion\Run
SYSMSG = "%System%\SYSMSG32.EXE"
```

NOTE

%System% is the Windows system folder, which is usually C:\Windows\System on Windows 95, 98, and ME, C:\WINNT\System32 on Windows NT and 2000, and C:\Windows\System32 on Windows XP.

Propagation

WORM_VAMPIRE.A locates the address book for the ICQ instant messenger client and tries to send itself to all the contacts that it finds. Every 1–20 minutes, it hides the file transfer window while it sends itself to other ICQ users as THIS ROX.ZIP.EXE, which ICQ displays as THIS ROX.ZIP on a target recipient's window.

Identification and Termination

To remove this malware, first identify the malware program.

1. Scan your system with your Trend Micro antivirus product.
2. NOTE all files detected as WORM_VAMPIRE.A.

Trend Micro customers need to download the latest pattern file (www.trend-micro.com/download/pattern.asp) before scanning their systems. Other users can use Housecall, Trend Micro's online virus scanner (<http://housecall.antivirus.com>).

To terminate WORM_VAMPIRE.A, you will need the name(s) of the file(s) detected earlier.

1. Open Windows Task Manager.
2. On Windows 95, 98, and ME systems, press **Ctrl + Alt + Delete**. On Windows NT, 2000, and XP, press Ctrl + Shift + Esc, then click the **Processes** tab.
3. In the list of running programs, locate the malware file(s) detected earlier.
4. Select one of the detected files, then click **End Task** or **End Process**, depending on the version of Windows on your system.
5. Repeat steps 3 and 4 for all detected malware files in the list of running processes.
6. To check if the malware process has been terminated, close Task Manager, and then open it again.
7. Close Task Manager.

NOTE

On systems running Windows 95, 98, and ME, Windows Task Manager may *not* show certain processes. You can use a third-party process viewer such as Process Explorer (www.sysinternals.com/ntw2k/freeware/procexp.shtml) to terminate the malware process. Otherwise, continue with the next procedure, noting additional instructions.

Removing Autostart Entries for WORM_VAMPIRE.A from the Registry

Removing autostart entries from the registry prevents the malware from executing at startup.

1. To open the Registry Editor, click **Start** | **Run**, type **REGEDIT**, then press **Enter**.
2. In the left panel, double-click **HKEY_LOCAL_MACHINE** | **Software** | **Microsoft** | **Windows** | **CurrentVersion** | **Run**.
3. In the right panel, locate and delete the entry **SYSMMSG = “%System%\SYSMMSG332.EXE”**.

NOTE

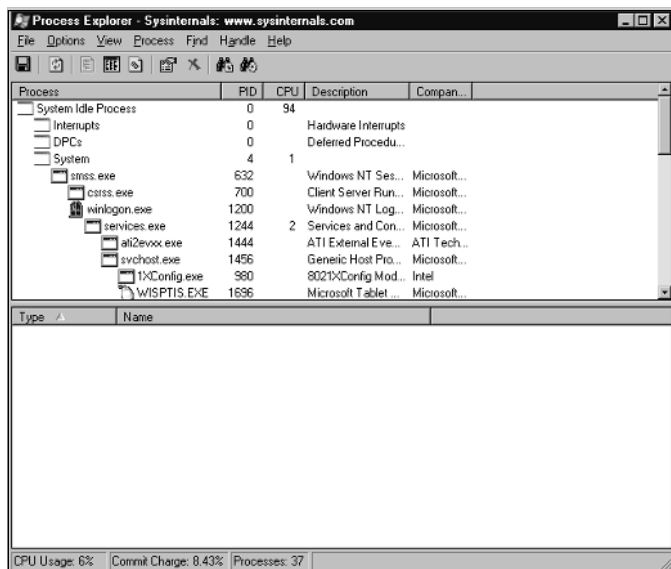
%System% is the Windows system folder, which is usually C:\Windows\System on Windows 95, 98, and ME, C:\WINNT\System32 on Windows NT and 2000, and C:\Windows\System32 on Windows XP.

4. Close the Registry Editor.

NOTE

Process Explorer is a utility published by SysInternals, and is available for free at www.sysinternals.com/Utilities/ProcessExplorer.html. This utility is a task monitor that graphically details process information. A small sample of information that can be obtained from this utility is process identifiers (PIDs), size and location of running processes in memory, dependencies and child processes. Figure 5.10 shows an example of Process Explorer examining threads.

Figure 5.10 Process Explorer's Main Interface



WORM_CHOD.B

WORM_CHOD.B was discovered on April 1, 2005, and is described by Trend Micro (<http://trend.com>) as the following:

- This worm may arrive via e-mail or the instant messaging application MSN Messenger.
- To propagate via e-mail, it searches for target e-mail recipients from files with certain file name extensions.
- It then sends copies of itself as an attachment e-mail addresses that it finds.

The e-mail message it sends may have any of the following details:

From: (any of the following)

- securityresponse@symantec.com
- security@microsoft.com
- security@trendmicro.com

Subject: (any of the following)

- Your computer may have been infected
- Warning – you have been infected!

Message Body:

- Your message was undeliverable due to the following reason(s):
- Your message could not be delivered because the destination server was unreachable within the allowed queue period. The amount of time a message is queued before it is returned depends on local configuration parameters. Most likely there is a network problem that prevented delivery, but it is also possible that the computer is turned off, or does not have a mail system running right now.
- Your original message has been attached.

Attachment: (any of the following)

- netsky_removal.exe
- removal_tool.exe
- message.pif
- message.scr

This worm also propagates via the instant messaging application MSN Messenger. It sends copies of itself to all available MSN Messenger contacts as any of the following files:

- check this out
- gross
- my sister's webcam
- mypic
- naked lesbian twister
- paris hilton
- picture
- rofl
- us together
- wtf

The file can have any of the following extension names:

- .pif
- .scr

It sends the copy of itself along with any of the following messages:

- check out what I just found on some stupid website
- dude check this out, it's awesome! :D
- haha you have to see this, I almost couldn't believe it! :O
- holy sht you have to see this... :|
- I just found this on a CD... you won't believe it! :|
- LOL! look at this, I can't explain it in words...
- omg check this out, it's just wrong :O
- ROFL!! you have to see this... wtf...
- you have to see this, it freaked me out :S
- you have to see this, it's amazing!

It modifies the HOSTS file to prevent access to antivirus-related websites. It also disables the firewall on affected systems, and is capable of disabling several other services.

This worm also has backdoor capabilities. It connects to a remote IRC server and joins a specific IRC channel, where it listens for commands coming from a remote malicious user. It executes these commands locally on an affected system, proving the remote user virtual control over the system. This worm contains an embedded password-recovery tool, which is capable of stealing passwords from various instant messaging applications.

Client Security

By blocking access to login.icq.com and port 5190, exposure to malicious code and security issues related to ICQ can be minimized. Although advanced users will be able to change port settings or configure proxies to continue using the ICQ client, less advanced users will be discouraged from using the ICQ service.

The ICQ client has not been the subject of many public vulnerabilities, and none have been published in 2005 up to the time of this book's publishing. The lack of security issues may be due to the lack of distribution that ICQ has, making more popular instant messaging clients the target of malicious users who wish to find vulnerabilities in more widespread software programs in order to increase the damage from worms and other malicious code written to exploit these security holes.

Multiple Vulnerabilities in Mirabilis ICQ Client

According to Secunia's database, (<http://secunia.com/>) the ICQ client has been susceptible to 7 vulnerabilities in 2003, and only 1 in 2004. While most vulnerabilities found during this timeframe were not severe, several of these vulnerabilities may provide a remote attacker full access to a vulnerable workstation. Five vulnerabilities were released on the same day on May 6, 2003 by Core Security Technologies.

The Core Security Technologies Advisory is as follows:

Date Published: 2003-05-05

Last Update: 2003-05-02

Advisory ID: CORE-2003-0303

Bugtraq IDs: 7461, 7462, 7463, 7464, 7465, 7466

CVE Names: CAN-2003-0235, CAN-2003-0236, CAN-2003-0237, CAN-2003-0238, CAN-2003-0239

CERT: VU#936164, VU#792988, VU#829860, VU#367156, VU#967316, VU#680788

Title: Multiple Vulnerabilities in Mirabilis ICQ Pro 2003a client

Class: Remote Code Execution, Denial of Service, Boundary Error Condition (Buffer Overflow), Design/implementation error

Remotely Exploitable: Yes

Locally Exploitable: Yes

Advisory URL:

www.coresecurity.com/common/showdoc.php?idx=315&idxsection=10

Vendors contacted: Mirabilis. We sent notifications mails to the following addresses: security@icq.com, secure@icq.com, webmaster@icq.com, support@icq.com, several times during March and April (2003-03-11, 2003-03-24, 2003-04-11) and never received an answer from Mirabilis.

Release Mode: USER RELEASE

Vulnerability Description

Mirabilis ICQ client is a popular program that enables users to communicate through instant messaging, chat, sending e-mails, SMS and wireless pager messages, as well as transferring files and URLs. The ICQ client offers other client services, which are detailed at www.icq.com/products/whatisicq.html. Six security vulnerabilities were found that could lead to various forms of exploitation ranging from denying users the ability to use ICQ services to execution of arbitrary commands on vulnerable systems.

The following vulnerabilities were found:

- **[BID 7461, CAN-2003-0235, VU#936164]** POP3 Client Format String in UIDL Field: ICQ provides an integrated POP3 (Post Office Protocol v3) client vulnerable to a format string his vulnerability can be successfully exploited by an attacker able to impersonate the POP3 server.
- **[BID 7462, CAN-2003-0236, VU#792988]** “Subject” signed overflow in POP3 Client: ICQ provides an integrated POP3 client vulnerable to a 16-bit sign in the **Subject** field of e-mail headers. An attacker may be able to execute arbitrary commands by sending a malformed e-mail header to a vulnerable client.

- **[BID 7463, CAN-2003-0236, VU#829860]** “Date” signed overflow in POP3 Client: ICQ provides an integrated POP3 client vulnerable to a 16bit sign overflow in the **Date** field of e-mail headers. An attacker may be able to execute arbitrary commands by sending a malformed e-mail header to a vulnerable client.
- **[BID 7464, CAN-2003-0237, VU#367156]** ICQ Features on Demand spoofing attack: ICQ provides a semi-automated functionality for upgrading client services (such as ICQ Phone, ICQ Web Search, etcetera) called *ICQ Features on Demand* vulnerable to a spoofing attack due to hard-coded information and lack of authentication signatures. By taking advantage of this vulnerability, an attacker will be able to install malicious software that could lead to execution of arbitrary commands as well as other important security breaches.
- **[BID 7465, CAN-2003-0238, VU#967316]** Message advertisements denial of service attack: ICQ displays advertisements inside a message window (called *Message Session*) by using a proprietary HTML (Hypertext Markup Language) parsing/rendering librarye to malformed tags input. By impersonating the static ADS server, an attacker may send malformed HTML code to the ADS rendering window freezing the ICQ interface and using 100% CPU.
- **[BID 7466, CAN-2003-0239, VU#680788]** Input validation error in ICQ’s GIF parsing/rendering library: ICQ implements its own image parsing/rendering library (found in ‘icqateimg32.dll’) vulnerable to an input validation error, causing a denial of service. The problem is triggered while parsing GIF89a headers.

Vulnerable Packages

Mirabilis ICQ Pro 2003a and previous versions.

Credits

These vulnerabilities were found by Lucas Lavarello, Daniel Benmergui, Norberto Kueffner and Fernando Russ from Core Security Technologies during Bugweek 2003 (March 3-7, 2003).

Technical Description

The following listings will better-define the insecure characteristics and potential consequence of each vulnerability.

[BID 7461, CAN-2003-0235, VU#936164]

POP3 Client Format String in UIDL field

ICQ's integrated POP3 client is a COM object found inside POP3.dll.

The client is vulnerable to a format string attack in the UIDL (unique ID listing) command server response string (the unique-id of a message):

"The unique-id of a message is an arbitrary server-determined string, consisting of one to 70 characters in the range 0x21 to 0x7E, which uniquely identifies a message within a maildrop and which persists across sessions," as described in RFC 1939 (www.ietf.org/rfc/rfc1939.txt).

By the insertion of format strings as part of a UIDL response message, the POP3 client can be forced to execute arbitrary commands.

[BID 7462, CAN-2003-0236, VU#792988]

"Subject" signed overflow in POP3 Client

ICQ's integrated POP3 client is a COM object found inside POP3.dll.

The client is vulnerable to a sign overflow attack in the "Subject" field of e-mail headers.

The length of the "Subject" field is stored in a 16 bit (short) signed integer, allowing an attacker to send a malicious e-mail along with a long "Subject" field of around 33k octets overflowing the sign of the variable and causing a negative value.

This attack results in the client throwing a self unhandled exception, crashing the client.

[BID 7463, CAN-2003-0236, VU#829860]

"Date" signed overflow in POP3 Client

ICQ's integrated POP3 client is a COM object found inside POP3.dll.

The client is vulnerable to a sign overflow attack in the “Date” field of e-mail headers.

The length of the “Date” field is stored in a 16 bit (short) signed integer, allowing an attacker to send a malicious e-mail along with a long “Date” field of around 32k octets overflowing the sign of the variable and causing a negative value.

This attack results in the client throwing a handled exception, instantly closing the client.

[BID 7464, CAN-2003-0237, VU#367156]

ICQ Features on Demand spoofing attack

The URL from where the requested ‘Features on Demand’ are downloaded is hard-coded inside a file called “Packages.ini” found inside the subdirectory “\DataFiles” in ICQ’s default installation path. The value named “DataURL” which belongs to the section “[General]” holds a static address from where the client will download user requested packages.

An attack is possible due to the lack of authentication methods applied to new downloaded packages. An attacker will be able to impersonate the ‘package repository service’ by spoofing the hard-coded address, being able to install malicious software that could lead to the execution of arbitrary commands as well as other important security breaches.

[BID 7465, CAN-2003-0238, VU#967316]

Message advertisements denial of service attack

The URL from where the HTML ads are downloaded has the following format: “http://web.icq.com/client/ate/ad-handler/ad_468/0,,[RANDOM],00.htm”

Being [RANDOM] a signed 16 bit random number. Note that the “,” characters don’t get encoded in their respective US-ASCII escape encoding.

The HTTP request follows certain rules:

- It is an HTTP/1.0 request
- The request has a “Refer:” to itself

- The “User-Agent:” is “Mozilla/4.08 [en] (WinNT; U; Nav)”
- The “Accept:” header must be “/”

The HTML parsing/rendering library is vulnerable to erroneous attributes specified in the <table> tag. By specifying a “width” attribute of value “-1”, the library will use 100% CPU, freezing the ICQ interface.

The attack is possible due to the lack of authentication methods applied to requests. An attacker will be able to impersonate the “ADS server” by spoofing the semi hard-coded address, being able to deny to users the usage of ICQ services.

[BID 7466, CAN-2003-0239, VU#680788]

Input validation error in ICQ’s GIF parsing/rendering library

While parsing GIF89a header files, ICQ’s GIF parsing/rendering library expects either an existing GCT (Global Color Table) or an LCT (Local Color Table) after an “Image Descriptor”. When none of these color tables exist, the library will malfunction leading to a denial of service.

The GIF89a file format has a section called Logical Screen Descriptor (from GIF89a specification, which can be found at <ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats/gif89a.doc>).

Summary

ICQ Messenger was once a very popular client, considered a pioneer in providing instant messaging services to the majority of users with Microsoft Windows operating systems. However, since AOL purchased its maker, Mirabilis, in 1998, ICQ has been slow to add new features and has developed an interface that differs considerably from standard Windows applications. Many original users of ICQ transitioned to other instant messaging services, leaving ICQ with a diminished user base as other services have grown. ICQ covers the basics of instant messaging such as text, voice, and video communication. ICQ also provides services via its website at <http://icq.com> such as chat rooms, message boards, and free Web-based e-mail services. The client and service do not provide central administration for the regulation and logging of information being exchanged, which may result in the distribution of confidential information to parties that should not be viewing this information. Users may be convinced to divulge sensitive, confidential, or personal data via social engineering, which will result in data leakage. Malicious users may pose as a known contact in order to obtain information easily. By forcing the client to ask for a sign-in and password for each session, the risk of someone using your workstation to masquerade as you can be greatly reduced. Another way to safeguard information is to limit sensitive and personal information from being discussed with others online. Credit card or banking information, social security numbers, or other personal information should never be discussed via instant messaging. Since there are limited controls compared to e-mail servers and firewalls, files, and executables can be transferred with almost no safety or scanning, creating avenues of attack for malicious users. These transfers can also involve the movement of copyrighted materials, such as MP3s, movies, and licensed software. There are no restrictions on the size or type of file transfers, allowing for restricted files to be transferred in spite of restrictions that block these file types.

Instant messaging services have become an increasingly popular vector for spreading malicious code since it can spread rapidly through instant messaging services. Files and URLs are often transferred through instant messaging clients without passing through security devices such as firewalls, URL filtering, and gateway security devices, promoting the spread of malicious code through an unregulated medium. Since worms are generally spread by unknown users through URL links in instant messages, many of these types of malicious code can be avoided by blocking unknown users from contacting you. This can decrease the risk of spreading a worm or other malicious code to your workstation and to other users.

Solutions Fast Track

Introduction and History of ICQ

- ☑ ICQ was founded in 1996 by an Israeli-based company, Mirabilis. In six months, it reached over 850,000 subscribers for the ICQ instant messaging service,
- ☑ The ICQ service can be considered the first modern instant messaging service for Windows-based workstations. AOL, Yahoo!, and Microsoft followed ICQ's lead, and released clients that could communicate using a PC and Internet connection to communicate with other users on the same network.
- ☑ AOL purchased Mirabilis in 1998, and has slowly been integrating parts of the ICQ service into AIM. In October 2002, AOL released an AIM client with the ability to communicate with users on the ICQ network.
- ☑ ICQ has moved to a protocol that is almost identical to the OSCAR protocol, originally used by AIM.

ICQ Features

- ☑ ICQ is very similar to AIM in terms of functionality, but provides unique features such as sending SMS messages to mobile phones and the logging of messaging history.
- ☑ ICQ uses the same ports for communication to its service that AIM uses, and uses many of the same servers for its functionality.

Malicious Code

- ☑ ICQ is not often targeted by malicious users, resulting in very few worms and public vulnerabilities.
- ☑ There were only 9 security threats between January 1, 2005 and June 1, 2005 according to IMLogic's Threat Center. These were all rated as low severity.

- ☑ Most worms are spread through instant messages and require the user to click on a URL and execute code manually. In order to prevent worms from affecting workstations, users should not allow messages to be received unless they are from known users.

Client Security

- ☑ The ICQ client has not been subject to many public security vulnerabilities. There have been no vulnerabilities published in 2005 at the time of this book's printing, and only 1 vulnerability published in 2004.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the "Ask the Author" form.

Q: What are the major differences between ICQ and AIM?

A: Since AOL bought ICQ, there has been a steady integration of the two services. ICQ originally had separate servers for sign-in and messaging. AOL has slowly changed the protocol of ICQ to match AIM, and has changed the architecture and servers of the ICQ service to be almost identical to AIM. The major differences between the two services lie in how users are identified on the services. For AIM, users choose a unique username. These names are alphanumeric, and are usually based on people's names, hobbies, or locations. ICQ, on the other hand, assigns users a unique number. These numbers are used to identify users of the service and route messages to contacts. Users also choose non-unique nicknames to identify themselves, and these nicknames can be published in the ICQ directory. This allows users of the service to have a friendly name to identify themselves with other users. The ICQ client itself has very few features that AIM does not have. Both clients are very basic, but ICQ provides the ability to log all messages sent between contacts, keeping a history for later reference. ICQ also gives its users the option to send messages via SMS to mobile phones, which AIM does not provide.

Q: Can I encrypt my messages using ICQ?

A: ICQ does not provide a way to encrypt your messaging session. Most instant messaging clients do not provide this feature. AIM provides the ability to use certificates to encrypt communications with other users, while Skype provides encryption by default for all communications. Third-party clients provide encryption, but you must be sure that the contact that you are communicating with has the same client and configuration in order for encryption to be activated. Trillian (<http://trillian.cc>) and IM2 (<http://im2.com>) provide built-in encryption, while Gaim (<http://gaim.sourceforge.net>) and Miranda (<http://miranda-im.com>) are two open source instant messaging clients that have plug-ins available to encrypt messaging.

Q: Is ICQ the safest instant messaging service to use?

A: There are really no safe instant messaging services. Although ICQ has not had many security vulnerabilities made public, it is unknown how susceptible this client really is. Its limited popularity makes it a smaller target for writers of worms, Trojans, and other malicious software, however the possibility always exists that malicious code can be written to take advantage of the network. All instant messaging services are not secure in that the very nature of instant messaging allows for users to communicate with each other without and controls. Sensitive or confidential information can be exchanged, bypassing any restrictions that would normally be in place. Instant messengers can exchange files and information as well as video and voice communication, all without restrictions and regulations. Additionally, only Skype provides complete encryption of this data, which means that users of any other instant messaging service may have their communications intercepted by a malicious users with a packet capturing utility.

Q: How can I prevent exposure to security issues related to ICQ?

A: ICQ is built to work within an environment that has a firewall, meaning that it can be configured with user-defined ports and connect to different proxy servers to gain access to the ICQ service. This makes it hard to restrict access to the ICQ service. Access to login.icq.com should be restricted from all ports, which will prevent users from logging into the service without using a proxy. A firewall with stateful packet inspection or packet filtering would be able to determine if there is ICQ network traffic, and can specifically block this communication.

Trillian, Google Talk, and Web-based Clients

Solutions in this chapter:

- Trillian Features
 - Google Talk
 - Web-based Clients
-
- ☑ Summary
 - ☑ Solutions Fast Track
 - ☑ Frequently Asked Questions

Introduction

There are many third-party instant messaging clients that have the capability to connect to multiple services such as AIM (AOL Instant Messenger), Yahoo! and MSN without needing to install the service's dedicated clients. Trillian, available at <http://trillian.cc>, is one of the most popular of these clients for Windows-based workstations, and is full featured and includes the feature of free, built-in encryption for messaging sessions. As of August 2005, Trillian has been downloaded 23,982,735 times according to CNet's Download.com (<http://download.com>).

Google Talk is a fairly basic instant messaging client. At the time of this writing, Google Talk's features are minimal compared to competitors, with none of the advanced tools that are now standard on other competing clients such as file transfers, video conferencing, or integration with portal features. Google Talk handles only two functions, text messaging and voice over IP (VoIP). This instant messaging client is in beta (as of September 2005) and its features and architecture may change before it reaches its finalized version. This instant messaging client is interesting because of Google's huge popularity and the fact that a GMail account is required to access Google Talk. The popularity of GMail may translate into millions of users for Google's instant messaging client.

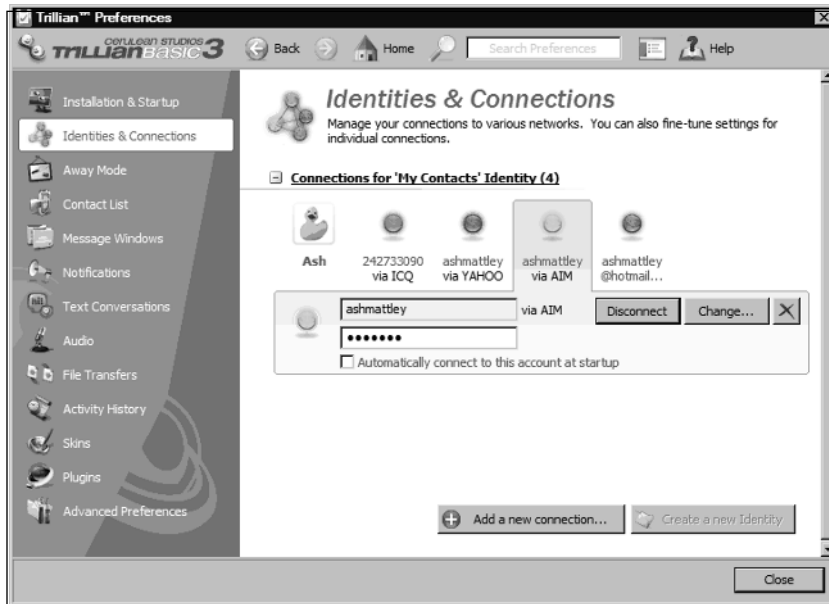
Web-based clients are not very popular due to their limited feature set. These clients are often used only when a user is unable to install a full-featured client on a system, possibly due to restrictions on a workstation. These clients are usually capable of very limited functionality, such as sending text messages. Web-based clients can present a unique security issue where restrictions on software installation can be bypassed, circumventing security and corporate policies that prevent instant messaging clients from being installed.

Trillian Features

Trillian was first released by Cerulean Studios on July 1, 2000 with version .50 and supported only IRC (Internet Relay Chat) communications. Trillian was refined and by version .74, which was released on September 10, 2002, was able to connect to all major instant messaging services. This release also brought a paid version of the client, Trillian Pro, which added support for plug-ins, which provided users with the ability to add features and functionality to the client, including the ability to connect to other, lesser known instant messaging services. Since Cerulean Studios does not control the services that their client connects to, there are times when instant messaging services change their protocols to block third-party clients such as Trillian. This has happened over the years, with both the AIM and Yahoo! services, and

Trillian has been very aggressive in responding with an updated client to regain connectivity to any service. Figure 6.1 shows the settings used to make connections with the different instant messaging services. Trillian is currently shipping version 3.1 of their client software, available in both a free version and a paid version, still known as Trillian Pro.

Figure 6.1 Trillian Connection Properties



Trillian Features

This section will focus on what sets Trillian apart from other instant messaging clients. The Pro version provides more functionality since it can be expanded through plug-ins, but the major feature that sets it apart today is the ability of users to encrypt messaging sessions among each other. Table 6.1 details the differences between the Basic and Pro versions of Trillian. The availability of encryption remedies one of the security issues present in instant messaging software. Since it is possible for a malicious user to install a packet capturing utility or other software to read messages and spy on others using instant messaging software, encrypting this data is the only real way to protect any sensitive data that is being discussed. Encryption provides a process by which the conversation that is being held can only be read by the intended recipient. Anyone using a packet capturing utility would see data that looked like random characters and would be unable to decipher the message. Unlike AIM, Trillian's encryption is built

into the product and is easily activated. This feature, known as SecureIM, provides encryption capabilities over AIM and ICQ with other Trillian users. Figure 6.2 shows the settings used to enable a SecureIM connection with the Trillian client.

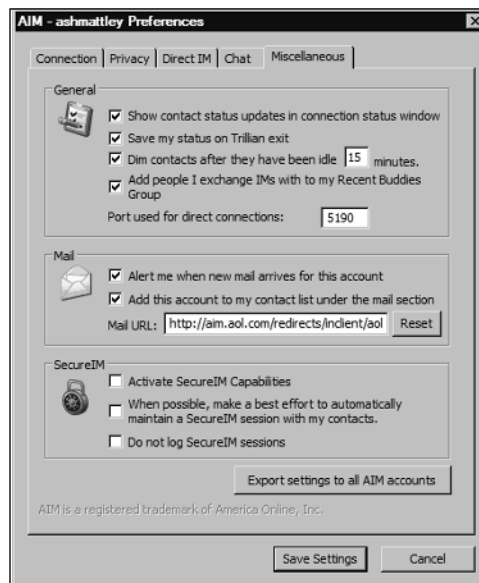
Table 6.1 Trillian Basic and Pro Features Compared

	Trillian Basic	Trillian Pro
Messaging Features		
Text Messaging	X	X
Encryption (SecureIM)	X	X
Conferencing	X	X
Audio Chat	X	X
File Transfers	X	X
Video Features		
Video Chat		X
Network Support		
IRC	X	X
Jabber®		X
Novell® GroupWise® Messenger		X
Rendezvous		X
History Viewer		
Timeline View		X
Summary View		X
Bookmarks		X
History Search		X
Assets		X

Continued

Table 6.1 continued Trillian Basic and Pro Features Compared

	Trillian Basic	Trillian Pro
Plug-ins		
Plug-in Support		X
Weather		X
HTML Minibrowser (views AIM Profiles)		X
News Syndication (RSS Feeds)		X
Mail Notification (POP3)		X

Figure 6.2 Trillian's SecureIM Settings

An encrypted session begins when a user with a Trillian client with SecureIM enabled sends a message to another user with the same configuration. Remember that this feature is only available on the AIM and ICQ protocols. This feature, shown in Figure 6.3, can be enabled by anyone using Trillian or Trillian Pro, but is activated only when both parties in a conversation are using Trillian and are communicating via the OSCAR protocol. Once the SecureIM session has been established, a random key is generated by both workstations. These keys are exchanged using the Diffie-Hellman-based key exchange protocol, which is described in detail at <http://rsasecurity.com/rsalabs/node.asp?id=2248>. Once this connection is established

and the keys have been exchanged, the connection is encrypted using 128-bit Blowfish encryption. More information on the Blowfish protocol can be found at <http://schneier.com/blowfish.html>. The resulting sessions provide a strong level of security. The ability to encrypt conversations should be a useful feature for users and businesses concerned about data leakage, where malicious users can spy on discussions with packet capturing utilities. Users and system administrators who are concerned about security and wish to obfuscate their communications should investigate this instant messaging client due to its ability to encrypt messaging sessions without installing extra utilities. Figure 6.3 shows an instant messaging session with SecureIM enabled, while Figure 6.4 shows a session with this feature disabled.

Figure 6.3 Conversation With SecureIM Enabled

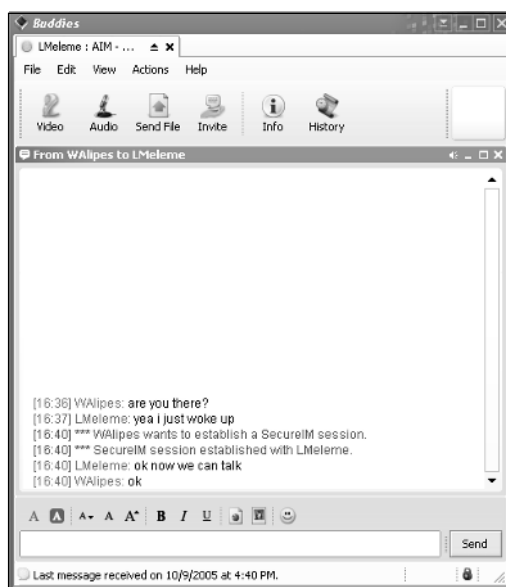
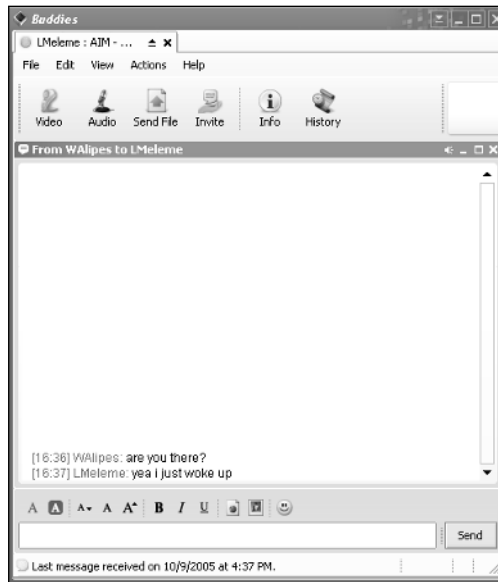


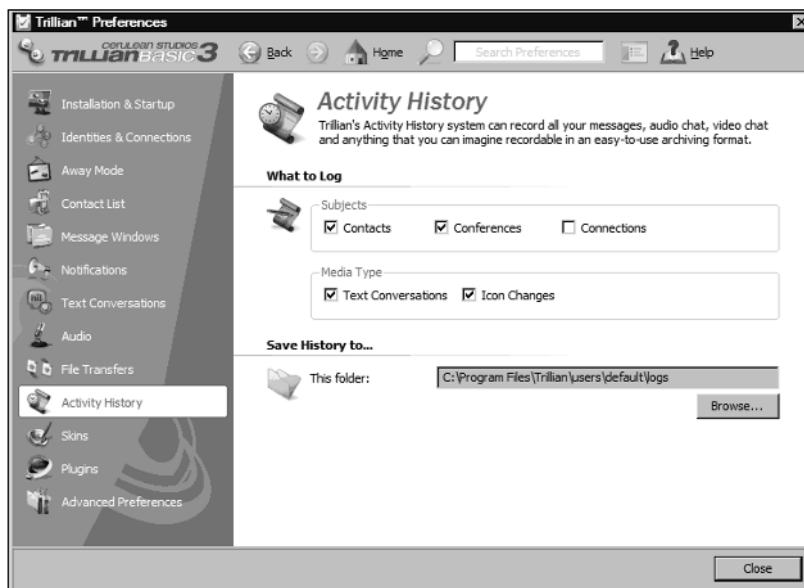
Figure 6.4 Conversation With SecureIM Disabled

Although not standard on other instant messaging clients, there are other options for encrypting data for instant messaging. AimEncrypt provides a free certificate for use with the AIM client, and is available at <http://aimencrypt.com/>. Miranda, available at <http://miranda-im.com/>, has a plug-in architecture, and an encryption plug-in is available at <http://miranda-im.org/download/details.php?action=viewfile&id=411>. GAIM, available at <http://gaim.sourceforge.net/> is also a popular instant messaging client with a plug-in architecture. This open source client also has an encryption plug-in available at <http://gaim-encryption.sourceforge.net/>. IM2, located at <http://im2.com>, provides a client with built-in encryption across all major instant messaging services. This client is currently free, but does not have the popularity that Trillian enjoys. Remember that both the originating user and the recipient must have the same client and plug-in in order to initiate an encrypted messaging session.

Trillian also includes a feature that gives it the ability to log all activity history (shown in Figure 6.5). All of Trillian's communication activity using this product can be logged, including text messages, conferences, and the date and time of these messages. The information this feature records is a security risk due to the fact that this information is logged to a plaintext file, which can be read by anyone with access to your workstation, including a malicious user if your workstation has been compromised or is susceptible to a vulnerability. If sensitive data was discussed during a conversation via Trillian, this information will be contained in these log files. It is

recommended that the message logging feature be disabled. Additionally, it is recommended that you specifically disable the logging of SecureIM conversations. Neglecting to disable this feature may provide a false sense of security since even though a conversation is encrypted and secured via SecureIM, the contents of that conversation can be retrieved by obtaining and viewing this text file.

Figure 6.5 Activity History



Trillian Malicious Code and Client Security

Trillian is a popular third-party application for instant messaging, but does not have the large user base that official instant messaging clients have. Trillian is not often the target of users writing malicious code due to its limited distribution. However, even though the client is not a particularly attractive target for malicious code writers, it still connects to several instant messaging services, which are carriers for worms and viruses. Therefore, by signing into various services, users will open themselves up to some of the same malicious code as if they were using the service's official client. To prevent infection from worms and other malicious code, Trillian users should restrict communications with unknown users. Each instant messaging service that is used within Trillian can be configured to accept messages from users on your contact list only. Figure 6.6 shows an example of these options in Trillian (using the AIM service), and each instant messaging service will have slightly different settings. By setting up these controls, you can prevent many worms from infecting a workstation

and spreading, but you should also restrict accessing URLs (Uniform Resource Locators) embedded in messages, which is how worms are often spread through instant messaging services. Trillian also includes an option to have all transferred files checked for viruses. Figure 6.7 shows the option available to users of Trillian that allows them to specify an antivirus client to scan files after transferring.

Figure 6.6 Trillian Options for Restricting Messages to Known Contacts

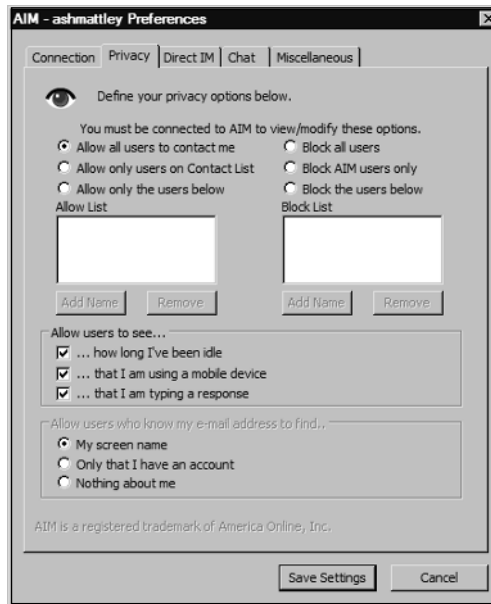
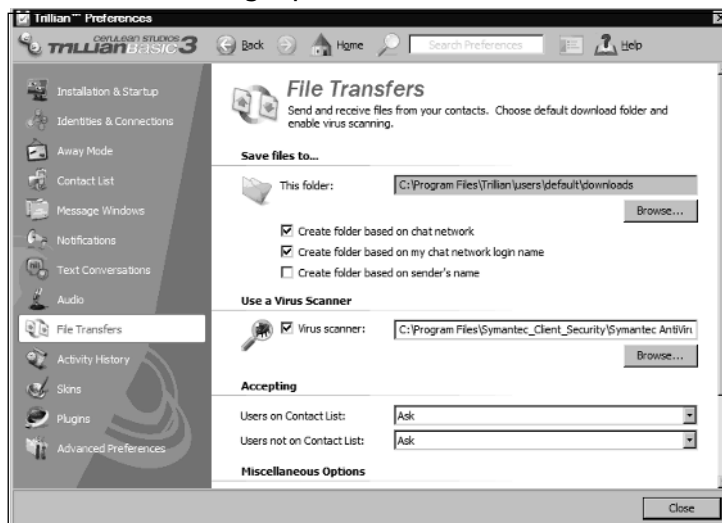


Figure 6.7 Antivirus Scanning Options



According to Secunia (<http://secunia.com>), Trillian has not been affected by many public vulnerabilities, with only 7 vulnerabilities affecting the product from September 2002 to August 2005. The lack of security issues may be due to the lack of distribution of Trillian, making more popular instant messaging clients the target of malicious users who wish to find vulnerabilities in more widespread software programs in order to increase the damage from worms and other malicious code written to exploit these security holes. While most vulnerabilities found during this timeframe were not severe, two of these vulnerabilities may allow a remote attacker to execute arbitrary code on the affected system. Table 6.2 lists Trillian vulnerabilities from Secunia's database.

Table 6.2 Trillian Vulnerabilities

Vulnerability	Date
Trillian Exposure of User Credentials	8/1/2005
Trillian Multiple Plug-ins Buffer Overflow Vulnerabilities	3/25/2005
Trillian Basic PNG Image Buffer Overflow Vulnerability	3/8/2005
Trillian MSN Module Buffer Overflow Vulnerability	9/8/2004
Trillian Protocol Handling Buffer Overflow Vulnerabilities	2/25/2004
Trillian buffer overflows	9/19/2002
Trillions Identd is susceptible to a buffer overflow	9/17/2002

Google Talk

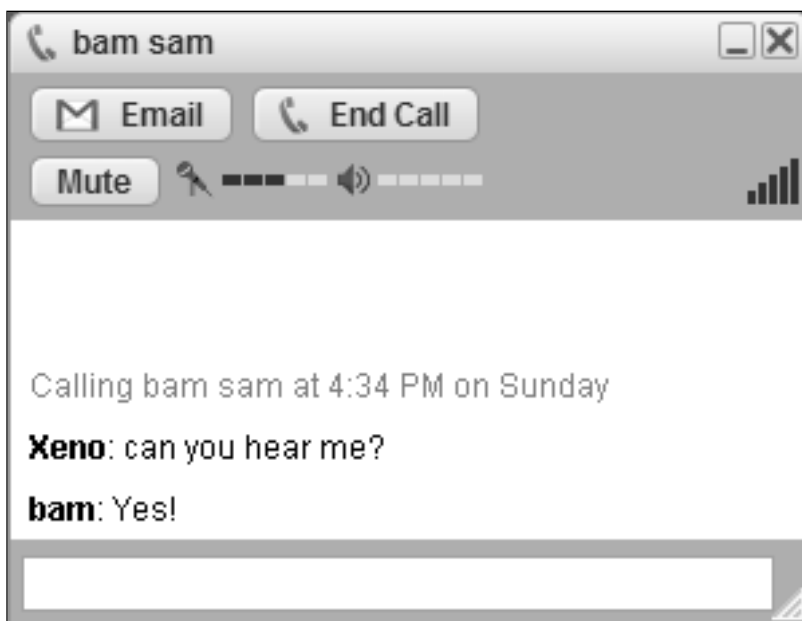
Google Talk is the newest service for instant messaging at the time of this book's publication. Google Talk was released as a beta on August 24, 2005 and is available at <http://talk.google.com>. At the time of this writing, Google's client is not very feature-rich, and contains only the functionality to communicate with other users via text or VoIP. What sets this service apart from the others mentioned in this book is the open nature in which Google has built their service. Rather than developing its own protocol, Google implemented the Jabber protocol, which is free and open. This protocol, also known as Extensible Messaging and Presence Protocol (XMPP),

is XML-based and is used by a wide variety of clients, which are listed on the Jabber Foundation's website at <http://jabber.org>. More information on the XMPP protocol is available at www.xmpp.org/. Since the protocol is freely available, there have been many implementations of servers for setting up a Jabber instant messaging system, as well as many clients that can communicate with these systems. One of the features of XMPP is the ability to communicate with multiple Jabber servers, creating an open community of instant messaging users. Google Talk is currently using its own servers and has not enabled server-to-server communication. This restricts Google Talk users from communicating with other users signed up on various Jabber servers. Google has stated that it will open the servers and allow this feature at a later date. Google does, however, encourage alternative instant messaging clients in order to communicate with Google Talk users. Google Talk's website provides examples and instruction for using other clients such as GAIM and Miranda to connect to and communicate with users of Google Talk. Instructions for connecting other clients to Google are available at <http://google.com/talk/otherclients.html>.

Connecting to Google Talk requires a GMail account, which is used as the user-name for the service. GMail is Google's free Web-based e-mail service, and accounts are given by invitation only from another GMail user or through SMS messages through a mobile phone. Google Talk connects to several servers on startup, including the following:

- mail.google.com, port 80
- talk.google.com, port 5222
- www.google.com, port 443

VoIP services are handled by customized XMPP, but Google has announced that it will be transitioning over to the Session Initiation Protocol, or SIP. This is another open protocol whose main purpose is to communicate multimedia content such as voice with other online users. An example of a VoIP session Google Talk is illustrated in Figure 6.8.

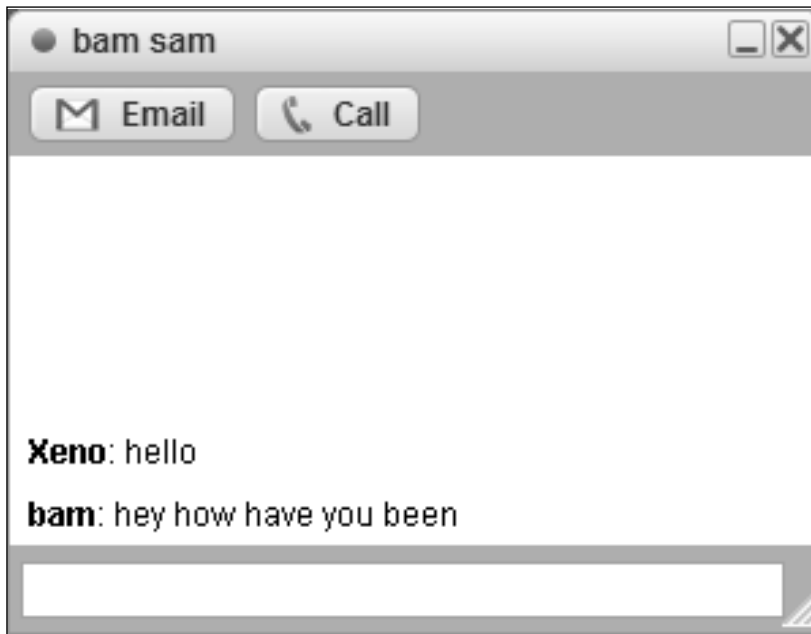
Figure 6.8 VoIP Session With Google Talk

Google Talk Features

Since Google Talk has a limited feature set, it does not have many areas for security concerns. Its feature set consists of two ways of communication, text messaging and voice calls. There are none of the other features that many users have come to expect from instant messaging clients, such as file transfers or group chat. This lack of features may be seen as beneficial since there are very few avenues for data to be distributed outside an environment.

Instant Messaging

Like all instant messaging software, Google Talk provides users with the ability to send text messages to each other in near real time. Figure 6.9 is an example of a messaging session in Google Talk. Instant messaging provides malicious users with a tool for social engineering, and can be used to entice a user to divulge sensitive or proprietary information. Malicious users may pose as an acquaintance, or as an employee of an instant messaging service. In either case, since it is nearly impossible to verify the identity of the user, it is recommended that personal, confidential or other sensitive information not be discussed via instant messaging.

Figure 6.9 Google Talk Messaging Session

Encryption

Google Talk does not provide any encryption for messages between users.

Encryption is necessary to prevent packet capturing utilities from gaining access to all conversations that are exchanged through the use of instant messaging services.

Utilizing packet-capturing utilities, such as Cain and Abel (located at www.oxid.it/cain.html), malicious users have the ability to record and play back conversations between each other via instant messaging software. Conversations with others over instant messaging services may be intercepted without your knowledge.

It is recommended that users do not discuss sensitive or confidential information over instant messaging since it may be intercepted and viewed by others.

Voice Chat

One of Google Talk's major features is its built-in voice chat feature. Many other instant messaging services have been implementing and enhancing their voice offerings, and it is one of the central (and few) features of Google Talk. As long as two users have a microphone and speakers, voice chat establishes voice communication. This feature provides similar security risks as instant messaging, bypassing phone, e-mail, and other possibly regulated communication systems, allowing sensitive infor-

mation to be communicated to people outside an organization without knowledge. Users could mistakenly distribute confidential information to unauthorized parties with this feature.

Web-based Clients

Web-based clients are offered by several of the large instant messaging services as a way to launch a limited version of their software and communicate with others without installing client software on a workstation. These Web clients offer very little in terms of functionality, often just text messaging and possibly e-mail alerts. With this limited functionality, these clients are often used only when a user is prohibited from installing applications on a workstation. In Enterprise environments, workstations are often locked down so the user is unable to install programs. By locking down workstations, administrators can restrict activities such as games and instant messaging from occurring. Web clients provide the means for a user to circumvent these controls and connect to instant messaging services. Table 6.3 lists Web-based clients and the URLs used for launching them.

Table 6.3 Web-based Clients

Name	URL
AIM Express	http://aimexpress.aim.com/
Yahoo! Web Pager	http://jpager.yahoo.com/jpager/messenger.html
MSN Web Messenger	http://webmessenger.msn.com/
ICQ2Go	http://go.icq.com/

Web-based Client Features

Web-based clients are essentially Web pages that provide connectivity to instant messaging services. These clients have limited functionality and minimal features available to users. These are simple utilities for text messaging only. AIM Express gives users the ability to establish a private chat room to communicate with multiple contacts.

Instant Messaging

Instant messaging is the only communication feature of Web-based instant messaging clients. Users have the ability to exchange data with each other in near real time via instant messages or conference rooms. This feature also provides malicious users with

a tool for social engineering, and can be used to entice a user to divulge sensitive or proprietary information. Malicious users may pose as an acquaintance, or as an employee of an instant re messaging service. In either case, since it is nearly impossible to verify the identity of the user, it is recommended that personal, confidential or other sensitive information not be discussed via instant messaging.

Encryption

Web-based clients do not provide any encryption for messages between users. Encryption is necessary to prevent packet capturing utilities from gaining access to all conversations that are exchanged through the use of instant messaging services. Conversations with others over instant messaging services may be intercepted without your knowledge. It is recommended that users do not discuss sensitive or confidential information over instant messaging since it may be intercepted and viewed by others.

Circumventing Workstation Controls

The features of these Web-based clients are so basic that they are used only when necessary. Their most important feature is their capability to be used on workstations that are locked down to prevent the installation of software. This is a security risk, since these clients have the ability to circumvent corporate policy against instant messaging.

Summary

There are several multi-protocol clients that enable users to connect with multiple instant messaging services. Chapter 1 contains a list of many of these clients. There are plenty of options for users looking to communicate with multiple contacts across the major instant messaging services. Many of these solutions are free and provide a plug-in architecture allowing a community of users to develop new functionality for these clients. Trillian stands out when compared to these other instant messaging clients due to its built-in encryption. Throughout this book, a security concern for instant messaging clients is that messages can be intercepted by users with packet capturing utilities. This creates a scenario where a rogue system administrator or malicious user can effectively spy on conversations. Trillian includes encryption for AIM and ICQ sessions, which ensures that only the intended recipient can actually read this information. Although social engineering may still be at play, it is important to ensure that no one between you and your contact are able to view any communication. This feature can prevent data leakage or other security issues as well.

Google Talk is a bare-bones instant messaging client that provides text and VoIP messaging. Google chose to employ open standards in building its service, utilizing the free XMPP protocol. This protocol, used by Jabber, is popular for many small organizations looking to build internal messaging platforms. It has not been popular when compared to other instant messaging services, mostly due to the lack of corporate sponsorship or single large server capable of handling millions of users. With Google throwing its support behind this protocol, it is very possible that there will be interest and a rise in popularity for this protocol and Jabber-compatible clients.

Web-based clients do not pose much of a security issue with regard to client vulnerabilities. However, these clients may be considered an even greater security risk than standard instant messaging clients due to their ability to bypass corporate policies that restrict the installation and usage of programs including instant messaging clients. These Web-based clients have limited functionality when compared to the standard clients that are offered by instant messaging services. The main issue with these clients is their ability to circumvent workstation policies that restrict instant messaging clients. Since they run within a Web browser, these clients can be employed by users to bypass restrictions against software installations.

Solutions Fast Track

Trillian

- ☑ Trillian ships a free version of its client software that communicates with the major instant messaging services and IRC. The paid version provides users with the ability to extend the client's functionality by using a plug-in architecture.
- ☑ Trillian provides encrypted messaging sessions, making it impossible for a user with packet capturing software to decipher the messages. The encryption feature, SecureIM, works on AIM and ICQ protocols only.

Google Talk

- ☑ Google Talk limits its feature set to text messages and VoIP. Google provides instructions for using other clients that are able to communicate using the Jabber protocol.
- ☑ Google has built much of its system on open standards, including its protocol, XMPP, which is used in many freely available servers that support Jabber and is compatible with many clients.
- ☑ VoIP is currently handled by a proprietary XMPP implementation, but Google has stated it will transition to SIP, which is an open standard for multimedia communications.

Web-based Clients

- ☑ Web-based clients are usually not the first choice of users due to the limited features available in these clients. Generally, these clients are only capable of text messaging.
- ☑ Web-based clients may pose a security issue due to their ability to bypass restrictions on locked down workstations. It is recommended that the URLs to these services be blocked if you are trying to block instant messaging usage within an organization.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Since Trillian can encrypt messages, is it the most secure instant messaging client?

A: Trillian is no more secure than other instant messaging clients. Instant messaging security involves many different aspects, including malicious code, security vulnerabilities, social engineering, and data leakage. If two Trillian users communicate with each other via OSCAR protocol (AIM or ICQ), they have the option to encrypt their messaging session. This eliminates one of the security issues involved in instant messaging, namely data leakage. However, the other issues instant messaging presents still exist. Users of Trillian may still be able to swap copyrighted files, including files with large sizes such as movies. Additionally, worms or other malicious code can still infect a machine and spread to others. Vulnerabilities have been found in Trillian’s software in the past, and some of these vulnerabilities have enabled malicious users to take control of a workstation or execute arbitrary code. It is important to note that Cerullian Studios, the publisher of Trillian, is very small compared to other software publishers. Since they have limited resources, it is possible that they would be unable to fix a software vulnerability or patch their software as quickly as a larger publisher such as AOL or Microsoft, with more resources to ensure the security of their software,

Q: Are there any security issues with using Google Talk?

A: As of September 2005, the time of this writing, Google Talk has no known vulnerabilities. This may be due to the very short time this client has been available to the public (less than a month). Google Talk does not have very many features, which in turn makes it more secure since there are less avenues for data leakage and social engineering. Google Talk does not provide users with the ability to trade files, which prevents a large portion of data leakage and copyright violations from occurring.

Q: What are the benefits of using a Web-based client?

A: One benefit of using a Web-based client is simply that there is no software to be installed. Users can sign into these services regardless of whether or not there are restrictions on their workstations that would prevent them from installing software locally. Users in an Enterprise environment with locked down workstations can go to the URL for the Web-based client and still log into the service, bypassing any restrictions. People who use public workstations, such as Internet cafes, workstations in libraries, or other areas may decide to use these clients as a way to sign into instant messaging services without installing software or leaving traces of their user accounts on a public workstation.

Skype

Solutions in this chapter:

- Skype Architecture
- Features and Security Information
- Malicious Code
- Client Security
- A Word about Network Address Translation (NAT) and Firewalls
- What You Need to Know about Configuring Your Network Devices
- Ports Required for Skype
- Using Proxy Servers and Skype
- How to Block Skype in the Enterprise

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Skype (available at <http://skype.com>) is a multi-purpose client that provides voice communication features as well as standard instant messaging features such as text messages and file transfers. The company highlights the voice communication features, and it is one of the most popular applications for making and receiving Internet-based calls. Skype's architecture resembles many peer-to-peer (P2P) services. This should come as no surprise, since its founders, Niklas Zennström and Janus Friis, were also the creators of Kazaa, one of the most popular P2P services. Rather than transmit all data for voice communication through a central server, Skype has the ability to use the workstations signed into the system to transfer data to and from callers. This allows the service to handle data efficiently while keeping costs minimal for scaling the service. The Skype program saw its first beta release on August 29, 2003. By October 2004, Skype had over one million users online simultaneously, and on May 18, 2005 had three million users online at once. Obviously, Skype has become incredibly popular in a very short amount of time. Of course, much of this popularity has to do with the fact that Skype provides free voice communications. Rather than pay fees to telephone companies for calls made to others, Skype provides a way to communicate with others, no matter where they are located in the world, for free. Additionally, the sound quality of these calls is very good, due to both efficient compression algorithms and the peer-to-peer nature of the network, which provides ample bandwidth for carrying the large amounts of data for voice communication. Skype also provides services to make and receive calls to standard phone lines, for which it charges a fee. The client is available for many different platforms and operating systems, including Microsoft Windows, Linux, Mac OSX, and PocketPC.

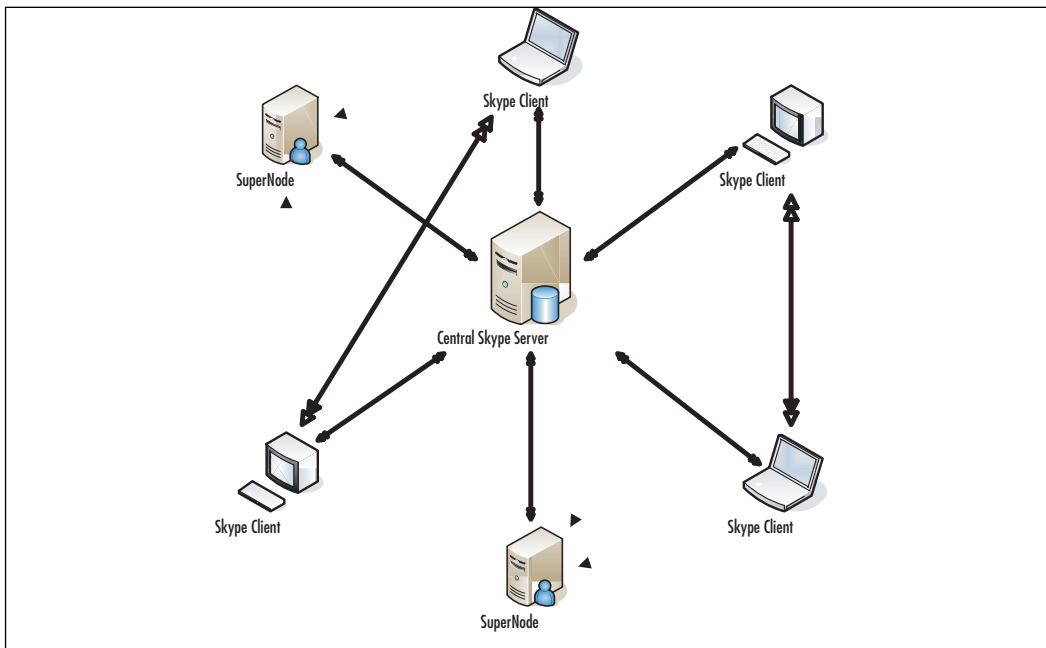
Skype's business model is based on providing services that users pay for. Skype charges for credits used for calling others and receiving calls from standard telephone lines. Local phone companies own and operate the public switched telephone network (PSTN), and charge for access to connect to these switches to make and receive calls. Skype sells two services, SkypeOut and SkypeIn. When you buy credits for these services, Skype provides you with the ability to dial or receive calls from anyone in the world using a standard phone. SkypeOut allows you to dial any phone number in the world, while SkypeIn provides you with a telephone number. This telephone number can be called from anywhere using a standard phone and allows Skype users to accept incoming calls to this assigned phone number. These services provide Skype users the ability to communicate with anyone in the world, with either another Skype client or anyone with a phone.

Skype Architecture

Skype uses architecture similar to Kazaa or other P2P networks (Figure 7.1). It is not a very strict P2P network since it employs a centralized server which helps the system sign up new users as well as authenticates existing users with user ID and password information. There are three main types of computers used within the Skype service: a standard node, a super node, and the Skype server. A standard node is any workstation that has the Skype client software installed. Users are able to make and receive calls, send messages, and use all the Skype functionality through this workstation. The super nodes are similar in appearance and functionality to the end user, but these workstations have been chosen by the Skype service to handle much of the Skype system's work. If a workstation has a publicly addressable IP (Internet Protocol) address and extra bandwidth, it is capable of becoming a super node, and the end user has no control over whether their workstation is a super node or not.

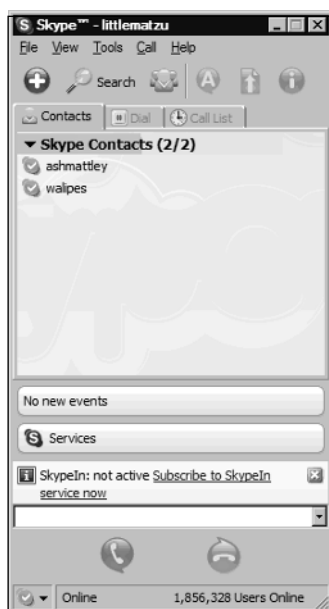
These super nodes do the heavy lifting for the Skype service, and the service relies on these super nodes, not a centralized server, for keeping track of other users in a directory (known as the Global index) and data from regular nodes. Workstations that are behind a firewall or a Network Address Translation (NAT) gateway will never be eligible to become a super node since the workstation's address its IP address is not public.

Figure 7.1 Skype Architecture



Since communications including text, voice, and files may be sent to other workstations before reaching the intended recipient, it is important to encrypt these communications so that users whose workstations are relaying this information are not able to spy on the information that is exchanged. Before messaging begins and two clients have established that they wish to transfer information between each other, an encrypted session begins. All data that is sent and received between two clients is encrypted using 256-bit encryption based on the Advanced Encryption Standard (AES). The key for this exchange is unique to that particular sessions and that particular set of workstations exchanging information. Once the session has been terminated, the key is no longer valid. Figure 7.2 shows the main window a user is presented with when first signing into Skype. According to Skype's website, "Skype uses 1024 bit RSA to negotiate symmetric AES keys. User public keys are certified by the Skype server at login using 1536 or 2048-bit RSA certificates."

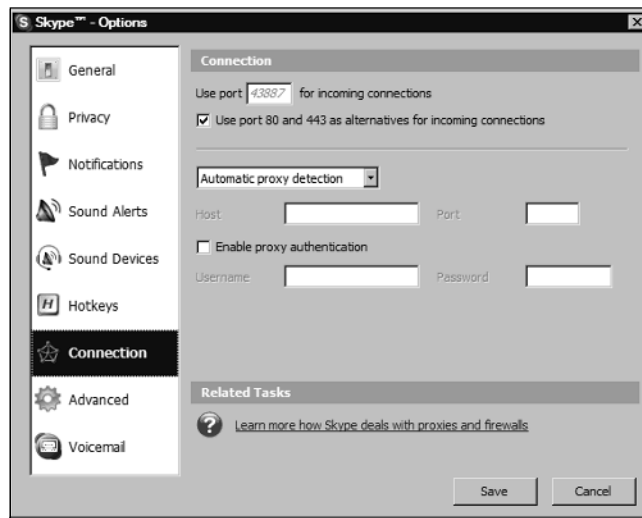
Figure 7.2 Skype Main Window



Skype communicates over a large range of ports with many different workstations and servers. The client utilizes both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) packets to relay information between these workstations. By default, the Skype client will listen over port 1387 for incoming connections, and if this port is unavailable it will use ports 80 and 443. Additionally, Skype provides support for several proxies, including HTTP (Hypertext Transfer Protocol), HTTPS

(HTTPS over Secure Sockets Layer), and SOCKS5. This gives an advanced user the option to use a proxy to connect to the Skype server for authentication if there are any connection issues with the native network, as you can see in Figure 7.3.

Figure 7.3 Skype Connection Settings



Skype uses a proprietary protocol for communications with other users. Not much is known about this protocol since it is closed and is encrypted. More information on the Skype architecture can be found in a paper published by Salman A. Baset and Henning Schulzrinne from Columbia University. This paper is available in PDF format at <http://arxiv.org/ftp/cs/papers/0412/0412017.pdf>.

Features and Security Information

Although Skype is well known for its voice communication, it is a very functional client for instant messaging via text and file transfers. Since it encrypts its information natively, this is a good tool to use for online communications.

Instant Messaging

Even though Skype has concentrated on voice communication, it includes the capability to communicate with other users through text messaging, called *chat*. Figure 7.4 shows an example of a chat session in Skype. This feature may be considered a security issue due to the inability of a user to truly identify the person he or she is messaging. The person on the other end of a conversation may be a malicious user

who has stolen an identity of a user. This person may also have merely sat down at the user's workstation or used a username that is somewhat similar to a known contact. This type of communication can lead to social engineering attacks, where a malicious user can convince another user to divulge sensitive or confidential information such as credit card numbers, or social security or bank information. It is recommended that personal, confidential or other sensitive information is not discussed via Skype's chat feature unless you are positive of the identity of the person you are communicating with.

Figure 7.4 Skype's Chat Feature



Encryption

Skype is one of the few instant messaging utilities that include free encryption for communication. Unlike Trillian, Skype's encryption is enabled by default and is always used for all information sent to another Skype user. Encryption is necessary for Skype since data sent from one user may be routed through a third party's workstation functioning as a surr node.

Chat History

Message archiving allows you to record instant messaging activity automatically and store the conversation to a file on your hard drive. This feature provides some set-

tings to determine how long messages should be saved locally. The history is saved at the following location (this location is not configurable):

C:\Documents and Settings\

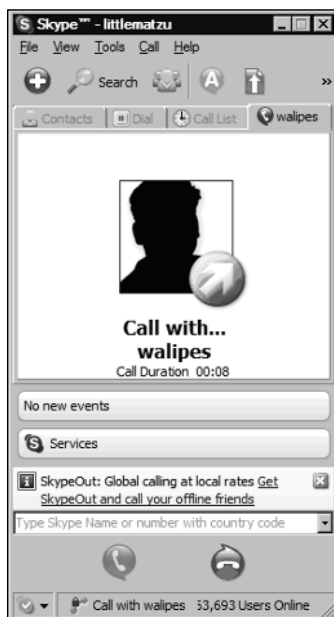
This file is not encrypted, giving anyone with access to your workstation the ability to read or copy the file. Additionally, vulnerabilities that compromise a workstation may allow for a malicious user to gain access to the file. It is recommended that this feature be disabled. Figure 7.5 shows the settings that can be defined for this feature.

Figure 7.5 Skype's Chat History



Skype Calls(Voice Chat)

The main feature of Skype is VoIP, where users can communicate via voice. Obviously, in order for this feature to be useful, both users need to have microphones and speakers on their workstations. This feature can provide a user with the ability to bypass restrictions on communication, including phone, e-mail, and other possibly regulated communication systems, allowing sensitive information to be communicated to people outside an organization without an administrator's knowledge. Figure 7.6 shows a Skype call in session.

Figure 7.6 Skype Call

Group Chat

There are two types of group chat available to Skype users. Users can opt to invite others to a chat session that is already taking place. Multiple users, once invited into the group chat, can send messages that can be viewed by all participants. This is very similar to a chat room, which broadcasts all information to those who are present in the chat. An example of this type of group chat is shown in Figure 7.7. Skype also allows multiple participants in voice calls, which are known as *conferences*. The originator of the call, or host, is able to add more people to the call up to a maximum of five people. This allows users to communicate in a way very similar to conference calls over standard telephone lines. Skype recommends that the host have a very fast Internet connection with available bandwidth. This is due to the host having to combine all the data sent from multiple users and resend them to the entire conference so all participants can hear the conversation. In order to ensure high quality sound, the conference call is limited to five people. Figure 7.8 shows a user setting up a conference, while Figure 7.9 shows a group of users in a conference.

Figure 7.7 Group Chat Session



Figure 7.8 Setting Up a Conference

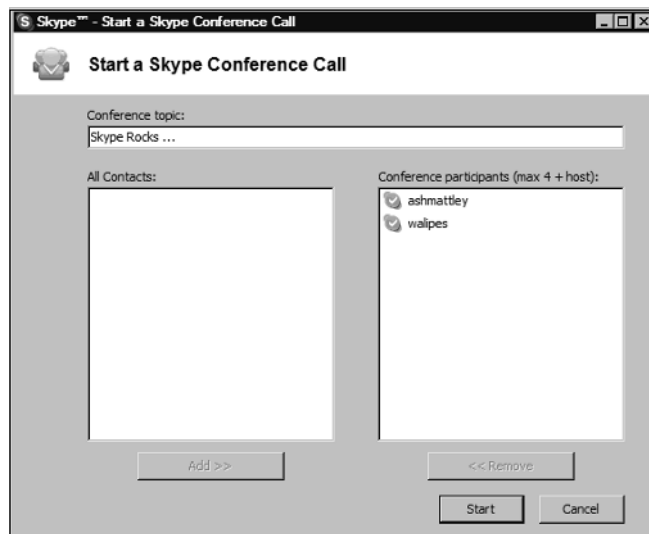
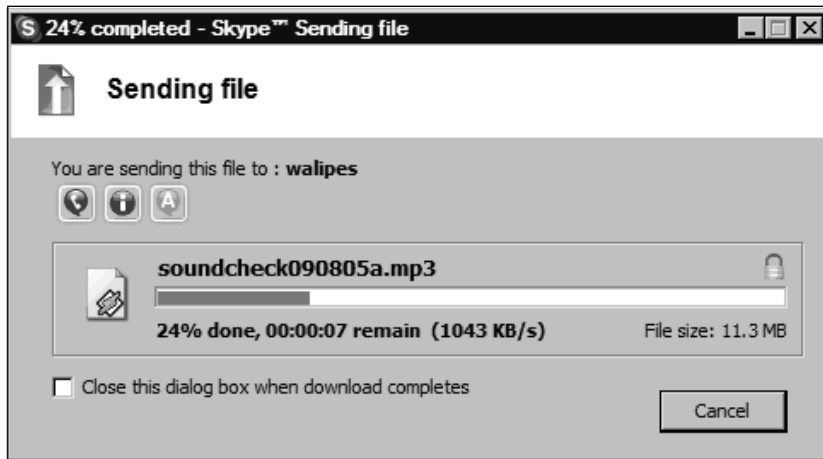


Figure 7.9 Conference In Session

File Transfer

Skype offers the functionality to send and receive files between users directly. The recipient must accept the file transfer before it begins, and specify a location for the saved file. This file is sent encrypted like all other communication between Skype users. This feature seems to be very slow, and can vary depending on the route it takes to get to the intended workstation. This feature allows for users to exchange files in an unregulated fashion, bypassing any controls or logging functionality that may be implemented. By utilizing the file transfer feature, users can bypass restrictions placed on large files through e-mail systems or that block users from utilizing FTP (File Transfer Protocol) servers. This feature does not have any restrictions on file types, allowing users to exchange large files such as confidential documentation and copyrighted material such as music and movies even though security measures may be in place to prevent file sharing and P2P services. This may also allow a user to distribute malicious code such as a virus. Unless a user can authenticate the person initiating the file transfer, it is recommended to deny all file transfers. Figure 7.10 shows a dialog box for accepting a file transfer in Skype.

Figure 7.10 Skype File Transfer Dialog Box

Malicious Code

As of this book's publication, Skype is not known to be the target of any malicious code, including worms or Trojans. However, as Skype's popularity increases, it is possible that this may become a more attractive target for users who create malicious code. To protect yourself and your workstation from attack, it is recommended that you configure Skype to accept messages and files from known users only. Skype requires all users to request to communicate with someone else, giving you the ability to restrict communication with unknown or unwanted users. It is important to ensure that only users who you know the identity to are accepted. If there is a problem with a user who is already on your contact list, Skype provides the ability to block the user so that the user is no longer capable of communicating with you or even seeing whether or not you are online. Figure 7.11 shows the privacy settings that can be configured so only users who are on your contact list have the ability to communicate with you.

Figure 7.11 Privacy Settings

Client Security

Skype has been relatively free from vulnerabilities. To ensure protection against malicious attacks, you should always upgrade to the latest version of Skype, which may include security-related fixes to its code.

Skype maintains a security site, which can be referenced at <http://skype.com/security/>. This site contains a security advisory section, which details the security issues the client has had since June 2004. There have been only three security issues from June 2004 to August 2005:

- SSA-2005-01 (Apr 20): Skype API Access Grant Revocation Failure
- SSA-2004-02 (Nov 17): Callto Handling Buffer Overflow
- SSA-2004-01 (Jun 15): Callto Handling Range Check Error

The Callto Handling Buffer Overflow may allow a malicious user to execute arbitrary code on the affected workstation. Skype's security website details this vulnerability as follows:

November 17, 2004

SKYPE SECURITY ADVISORY

SSA-2004-02: CALLTO HANDLING BUFFER OVERFLOW

Overview

Certain versions of Skype for Windows contain a buffer overflow vulnerability that could possibly allow a remote attacker to execute arbitrary code with the privileges of the user running Skype.

Systems Affected

Microsoft Windows systems running

Skype for Windows versions 1.0.*.94 to 1.0.*.98

I. Description

A buffer overflow vulnerability exists in the way Skype parses command-line arguments. If Skype is executed with a command line longer than approximately 4096 characters, Skype would report an Access Violation and terminate. However, an attacker could use this vulnerability to overwrite the program stack with data given in the command line, thus giving rise to the possibility of injected code execution.

This vulnerability could be exploited in conjunction with the Skype-specific callto: URL. Once registered, Windows passes any callto: URL to Skype as a command-line argument. Therefore, if the user follows a specially-crafted long callto: URL, the victim instance of Skype could execute arbitrary code supplied by the attacker in the URL.

II. Impact

By inducing a user to click on a specially crafted callto: URL on a web page or in an HTML e-mail message, an attacker could possibly execute arbitrary code with the privileges of the user. The attacker could also cause Skype to crash.

III. Solution

Upgrade to Skype for Windows version 1.0.0.100 or higher (<http://www.skype.com/download/>).

IV. Credit

Skype thanks Fabian Becker for discovering and reporting this issue.

Contact

The security of users is Skype's highest priority. You can contact Skype Product Security Incident Response Team (PSIRT) by e-mailing security@skype.net. Past advisories and the Skype PSIRT PGP key are available at <http://www.skype.com/security/>.

Skype remedies vulnerabilities by releasing a new version of the client. Unless there is a specific policy related to the installation of instant messaging clients, users and administrators must be aware of security issues and what steps have to be taken to lessen the exposure. If software is centrally managed and updated for an organization, it is critical that an administrator take responsibility for updating instant messaging clients when a security issue is found in order to protect against malicious users and vulnerable code. If instant messaging clients are not a critical piece of software in an environment, it is recommended that it is removed or disabled.

A Word about Network Address Translation and Firewalls

When the Internet began, the creators didn't envision the type of growth that we are experiencing today. During the last 10 years, the number of hosts on the Internet increased by more than 50 times.¹ In order for each Internet device, or host, to communicate on the Internet, it must have a unique internet protocol (IP) address. The addressing scheme for the Internet allowed for billions of IP addresses, but now most of them are allocated.

The Internet's popularity results in a maximum number of available IP addresses. Homes and offices around the world are now connecting many hosts at a single location and it is not possible for every single device to have its own public IP address. To accommodate the limited amount of addresses, a new standard called IPv6 has been developed. Until IPv6 is finalized, other methods are needed to allow for allocation of public addresses to more people. The most effective solution is called network address translation (NAT), defined in the request for comments 1631 (RFC 1631).

NAT is a special type of router that has several different implementations. One popular method of implementation allows for the use of special, unroutable IP addresses on private or internal networks. The private addresses are translated to a public host address, which allows communication over the Internet. Three blocks of the unroutable, or private, IP addresses are defined in RFC 1597 and RFC 1918. The private addresses are reserved by the Internet Assigned Numbers Authority (IANA), the organization that is responsible for all IP addresses. The private addresses are represented in Classless Inter-Domain Routing (CIDR) notation as:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

These address blocks cannot communicate directly with public addresses on the Internet and must be translated.

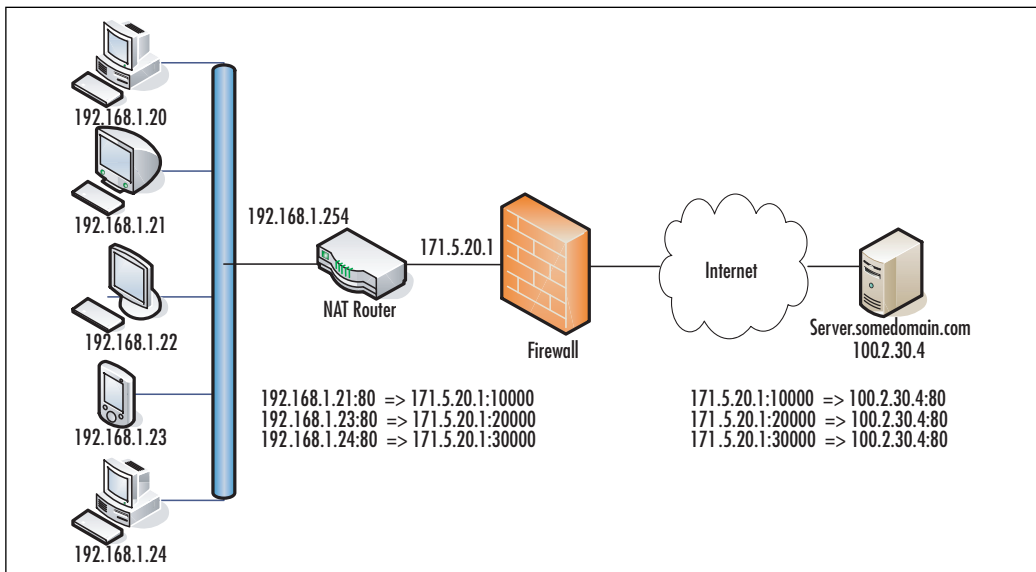
NAT utilizes a mechanism in the Transmission Control Protocol/Internet Protocol (TCP/IP) stack called multiplexing to enable these private addresses to establish communication over the Internet. Multiplexing makes it possible for a single device to establish and maintain several simultaneous connections with one or more hosts using different TCP and User Datagram Protocol (UDP) ports. This architecture allows an implementation where a single public IP address can service the needs of an entire network of hosts, a many—to-one relationship.

NAT routers keep a table of internal address and port combinations, as well as the public (global) IP address and port used to establish the remote connection. External hosts do not see the internal address, but instead use the public IP address to respond to requests. When responses are sent back to the external IP address and port of the NAT router, it translates the response and relays it back to the internal address and port that originated the request.

Firewalls are Protocol layer rules engines. A firewall can be hardware or software based, and many routers include basic firewall functionality as an additional feature. A typical firewall provides a list of rules that are evaluated sequentially against the header data in the packet being processed. As each rule is examined against the packet header, the packet will be blocked, or the next rule will be evaluated. This process continues until the packet is blocked or all rules have been examined.

A proxy server is similar to a firewall, but it works at the Application layer. Proxy servers have packet-filtering features. Packet filtering allows examination of the actual data being transmitted within the packet itself. Packet filters are available on Windows XP, Windows 2000, and Windows Server 2003 products as part of the advanced features of the TCP/IP configuration. However, because Skype encrypts the data it transmits, packet filtering is an ineffective means of managing Skype traffic. Proxy servers handle the requests for each protocol, whereas firewalls merely forward the traffic. If the proxy server is disabled, no traffic is allowed to pass. If you disable a firewall, you are turning off all rules processing and allowing all traffic to pass, which is not a recommended practice.

In the Figure 7.12 a single external IP address is exposed to the Internet. When hosts on the private network make a request, the following occurs:

Figure 7.12 Single External IP Address Exposed to the Internet

1. The host initiates a request for the remote destination address and port.
2. Since the address is remote, the router handles the request.
3. The NAT router adds the entry for the internal host IP address and port to the translation table.
4. The NAT router assigns a new port on the external interface IP address for the internal client and adds it to the translation table.
5. The NAT router then initiates a connection to the remote host on the external network, through the firewall, substituting a new source port and IP address in the IP packet header.
6. The remote host responds to the request to the external address and port.
7. The firewall compares the IP address and port with the list of firewall rules. If the IP address passes the IP address test, the port is checked. For Skype, this would be a UDP port, or if UDP is blocked, TCP port 443 or TCP port 80.
8. The router uses the translation table to translate the response from the remote host from the external address and port to the original internal address and port of the host that initiated the request.

Home Users

We strongly recommended that home users obtain a basic peer-to-peer-friendly, broadband router with firewall capabilities. In addition to a hardware-based router/firewall, you should always use a software-based firewall on each client machine. Windows XP has built-in firewall software that is enabled by default after you install Service Pack 2. Other options for software-based firewalls include products by McAfee, Symantec, and Zone Alarm. Skype should work right out of the gate on most home networks without requiring any further configuration. For home users, no modification is needed.

Later in this chapter, we discuss how to improve the quality of the communication, which could require minor configuration settings on your firewall.

Small to Medium-Sized Businesses

Small to medium-sized businesses must use discretion to determine whether to use a simple implementation, as discussed for home users, or to provide a more robust firewall solution, such as the Symantec Firewall/VPN Appliance, Cisco Pix, or other SOHO solution. Regardless, we suggest that small and medium-sized businesses use software-based firewalls on each network client to provide an additional layer of security.

Large Corporations

Larger corporations must ensure that the many routers used on the LAN allow Skype traffic over UDP to pass to other clients on the LAN if they want to use Skype effectively.

To better understand how Skype communicates, you need to get a picture of how the Skype network is organized. There are three basic roles in the Skype communication infrastructure. The roles consist of the following:

- Skype client or peer
- Supernodes
- Login servers

A Skype client is your computer running the Skype software. Supernodes are just Skype peer nodes that are not behind a firewall or have unrestricted access to the Internet. Supernodes come and go depending on the needs of the overall network. Any Skype client node can become a supernode if it is not behind a NAT router or blocking firewall and has sufficient CPU and bandwidth capacity.

Notes from the Underground...

Avoid Becoming a Supernode

To avoid a Skype client from becoming a supernode all that is required is for the client to be behind a NAT device or a corporate firewall device.

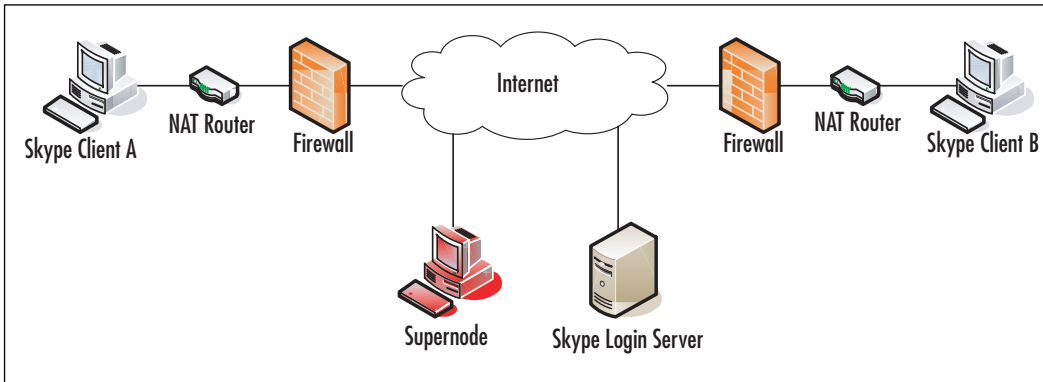
If a Skype client is behind a NAT router or firewall, the Skype client cannot establish a direct connection to another peer. In these situations, the supernode peers act as relaying agents to help Skype peers behind firewalls or NAT routers establish connections to other peers that are behind firewalls or NAT routers. Skype peers tend to connect to supernodes that are in relative proximity to their locations on the Internet. By connecting to nearby supernodes, Skype reduces utilization and decreases the latency in response times, thus providing a fast and scalable communication network.

Notes from the Underground...

Avoid Relayed Calls or File Transfers

To avoid a Skype call or file transfer from being relayed, the firewall must allow a P2P connection.

When Skype starts, it determines whether the client is behind a firewall or NAT router. If there is a firewall or NAT router, Skype determines the best method for communication via the firewall or NAT router using various UDP mechanisms. If no UDP ports are open, Skype will attempt to use TCP port 80, then TCP Port 443. Refer to the basic topology to get a picture of what happens next.

Figure 7.13 Skype Navigates the Firewall or NAT Router

After Skype Client A determines how to navigate the firewall or NAT router, Skype contacts a supernode peer from its supernode list to attempt to log in. If for some reason there are no supernodes listed for the client, the client attempts to log in to the Skype login server. Once the client logs in, the supernode list may be updated with the current active list of supernodes.

Once the connection is established, you can place a call, begin to instant message, or transfer a file. The call starts with a search of the Skype Global Index to locate the target Skype user. Skype Client B will follow the same process to log in. If the target user, Skype Client B, is behind a firewall or non-P2P-friendly device, the supernode acts as the liaison to direct TCP traffic from client A to Client B and vice versa, thus allowing Skype Clients A and B to find and communicate with each other using one or more supernode peers to relay messages.

What You Need to Know About Configuring Your Network Devices

We'll now discuss configuring network devices in various environments.

Home Users or Businesses Using a DSL/Cable Router And No Firewall

To use Skype typical home users will not need to configure anything on their DSL/Cable routers with or without wireless unless they have an older DSL/Cable router that is not P2P friendly. Running NAT Check, discussed later in this chapter, and enabling the Technical Information in Skype's Advanced options will help you determine if your router is capable of a Skype P2P connection.

Small to Large Company Firewall Users

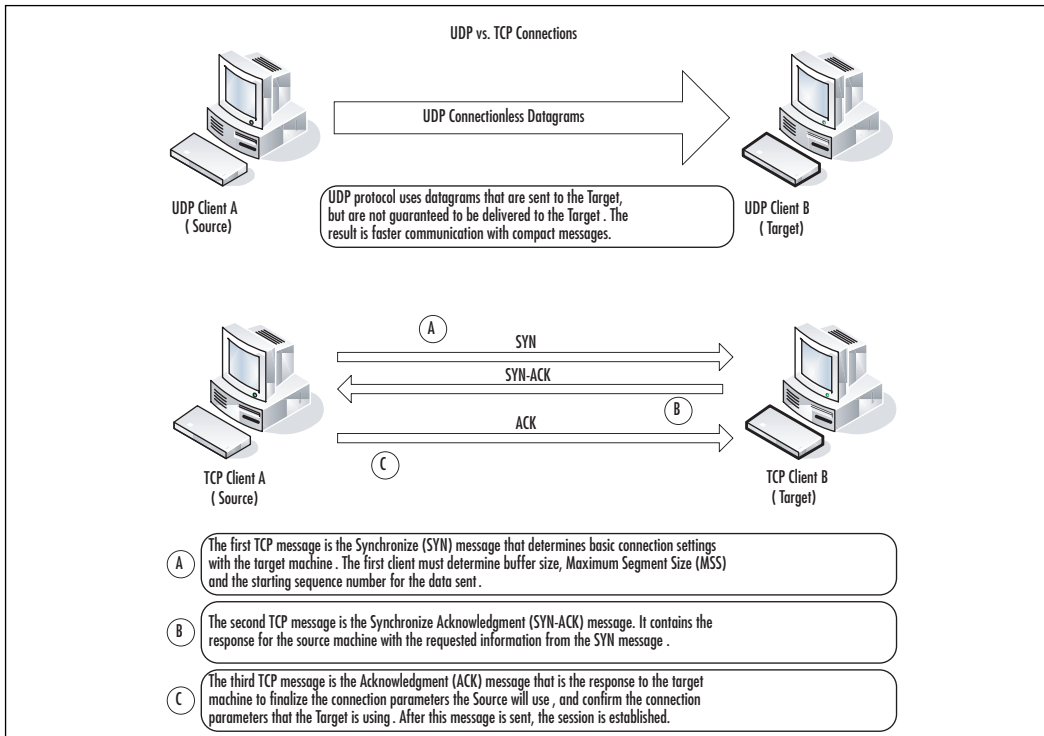
To provide the best performance on your network, you will need to tune your network to optimize handling of the Skype traffic. Skype leverages the use of UDP extensively to provide the best possible connection quality with its peers. The NAT translation table is a volatile table that ages old connections to free up room in the routing device's buffer for new connections.

It is important that the NAT routers hold the definition for UDP datagrams sent from the internal network for at least 30 seconds. The delay ensures that there is ample time provided for a response to the original request initiated from the client. The translation table should consistently map the internal host address and port number for UDP traffic in order to be reliably translated from the external address and port used to establish the communication. UDP has very little overhead, but it is prone to loss because it is not guaranteed to be delivered to the destination. Because it has little overhead, UDP is a faster method for communications.

TCP and UDP Primer

TCP requires a three-way handshake to verify that data reaches its destination, whereas UDP just sends that data and does not require acknowledgment of delivery. Because UDP does not require all of the overhead in the message structure, the messages are smaller, and UDP headers are always the same size. The UDP message structure makes the delivery much faster. Establishing communication sessions over TCP takes three trips instead of the one trip UDP requires. The TCP headers are much larger and vary in size, so there is more overhead to process each TCP message as well.

Figure 7.14 UDP vs. TCP Connections



NAT vs. a Firewall

Remember, a NAT device just translates many internal IP addresses to one or more external routable Internet addresses. A firewall can also provide NAT functionality and includes additional intelligence to apply rules to the traffic that passes through the firewall. NAT devices such as a DSL/cable router may or may not have firewall functionality.

Skype also recommends that the firewall or Internet gateway support IP packet fragmentation and reassembly. Fragmenting the packets allows the stream of data to be broken into smaller packets that can be sent simultaneously over multiple ports to the destination. This packet fragmentation can dramatically improve quality and performance by allowing higher throughput, which in turn allows for more effective bandwidth. Some firewalls detect this type of parallel UDP communication incorrectly as port scanning and will block the host traffic. The result could be a degradation of Skype performance.

Skype references a tool called NAT Check by Bryan Ford. The tool can be located at <http://midcom-p2p.sourceforge.net>.

The tool can be used to determine how P2P friendly your network is. Ford has described the details on UDP communications over the Internet using NAT in an Internet draft. The paper is located at <http://mirrors.isc.org/pub/www.water-springs.org/pub/id/draft-ford-natp2p-00.txt>.

The following example shows the output from NAT Check for a relayed call:

Figure 7.15 NAT Check Output

```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\ddouglass\Desktop\WIP\Skype>natcheck.exe
Request 1 of 20...
Request 2 of 20...
Request 3 of 20...
Request 4 of 20...
Request 5 of 20...
Request 6 of 20...
Request 7 of 20...
Request 8 of 20...
Request 9 of 20...
Request 10 of 20...
Request 11 of 20...
Request 12 of 20...
Request 13 of 20...
Request 14 of 20...
Request 15 of 20...
Request 16 of 20...
Request 17 of 20...
Request 18 of 20...
Request 19 of 20...
Request 20 of 20...

TCP RESULTS:
TCP consistent translation:      NO <BAD for peer-to-peer>
TCP simultaneous open:         NO <BAD for peer-to-peer>
TCP loopback translation:       NO <BAD for P2P over Twice-NAT>
TCP unsolicited connections filtered: YES <GOOD for security>

UDP RESULTS:
UDP consistent translation:      YES <GOOD for peer-to-peer>
UDP loopback translation:       YES <GOOD for peer-to-peer>
UDP unsolicited messages filtered: NO <BAD for security>

C:\Documents and Settings\ddouglass\Desktop\WIP\Skype>

```

Ports Required for Skype

We'll now discuss the ports that are required to use Skype.

Home Users or Businesses Using a DSL/Cable Router and No Firewall

To use Skype, typical home users will not need to configure anything on their DSL/cable routers or within the Skype software.

Small to Large Company Firewall Users

Skype uses UDP and TCP to communicate with other Skype clients. UDP is primarily used to establish connectivity and perform global directory searches. If the UDP ports above 1024 are open outbound, and you allow UDP replies to return through the firewall, you can improve Skype's voice quality and performance.

Opening UDP ports could allow peers on your network to connect more efficiently

by providing closer neighbors on the P2P network, thus reducing latency and improving call quality. Allowing more UDP ports also prevents internal contention of port translation in the NAT translation table.

In a perfect world, all outgoing TCP ports would be open through the firewall or Internet gateway. If it is not possible to open all outgoing ports, TCP port 80 should be opened. Using port 80 is a standard practice. When Skype attempts to log on, it first tries to connect using random ports. If Skype cannot connect, it attempts to connect via port 80. If port 80 cannot be opened, Skype attempts to use port 443. There is no guarantee that Skype will work through port 80 if the firewall or proxy server is restricting traffic to the HTTP. By restricting traffic to HTTP, the proxy server or firewall can scan the packets to ensure that the data is actually HTTP data. Skype does not use HTTP and will not function correctly through port 80 if traffic is restricted to HTTP traffic. If you receive errors #1101, #1102, or #1103 the firewall may be blocking port 80.

When Skype installs, it will select a random UDP port to communicate. This port setting is found in the Connection tab under Options and is an adjustable setting and stored in the shared.xml file on each computer and could be set the same for all users of Skype. If you want to avoid relayed Skype calls and relayed file transfers, you can open up the UDP port on your firewall that is specified in Skype to allow for better voice call quality and faster file transfers.

Understand that opening these UDP ports changes the normal corporate security policy, and proper approval and risks associated with opening anything on your firewall should be weighed prior to opening these settings. Discuss this issue thoroughly with your information security team on the impacts and what additional layers of security could be implemented to mitigate any risks, such as enabling a client-side personal firewall solution discussed earlier in this chapter. You could allow TCP and/or UDP inbound on the ports listed in Skype options for all clients internal to the firewall. If necessary, Skype will use TCP ports 80 and 443, respectively, to communicate with other Skype peers, and this will create relayed Skype calls and slow file transfers..

Skype's Shared.xml file

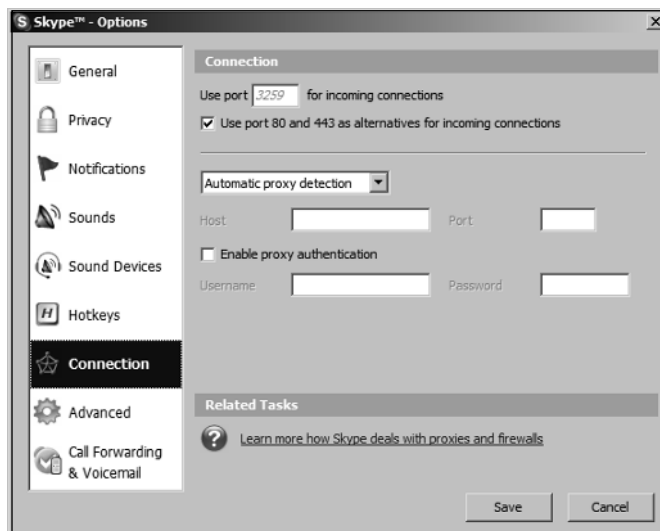
In a larger network, you can control the port for incoming connections by modifying Skype's shared.xml file in the following location:

- <Drive>\Documents and Settings\<UserName>\Application Data\Skype folder

The setting is found toward the end of the file under **config/Lib/Connection/**

ListeningPort. By configuring all users to use the same UDP port, you can improve the quality of Skype conversations by opening a single inbound UDP port, if your network security policy permits this. If the traffic inbound on that port is high, you could logically segment the traffic by setting different groups of users to use a specific UDP port and opening multiple UDP ports inbound, while still maintaining some control over what ports are opened and to whom. Visit Dan Douglass's Web site at the following URL for scripts and utilities to help modify the `shared.xml` setting in a business environment: www.codehatchery.com/skype.html.

Figure 7.16 Improve the Quality of Skype Conversations with a Single Inbound UDP Port

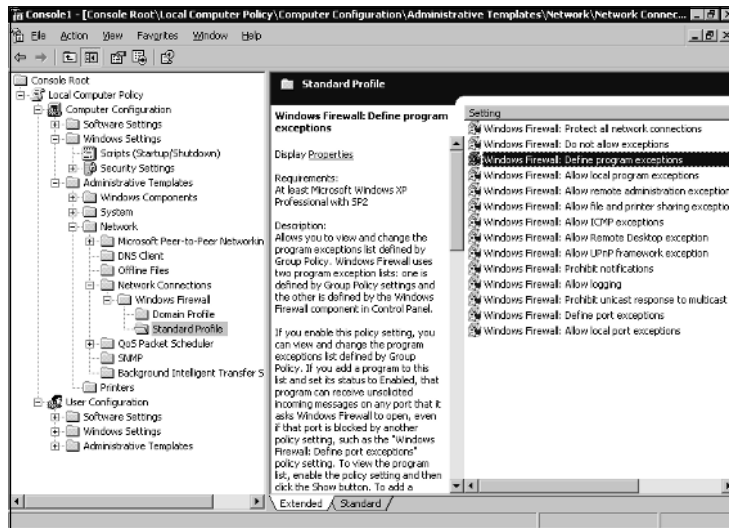


Microsoft Windows Active Directory

In a typical Windows Active Directory-based enterprise, with clients running Windows XP Service Pack 2, you can set a Group Policy that allows you to enable the Skype traffic through the Windows Firewall on all client machines with little effort. This can be achieved via the following steps:

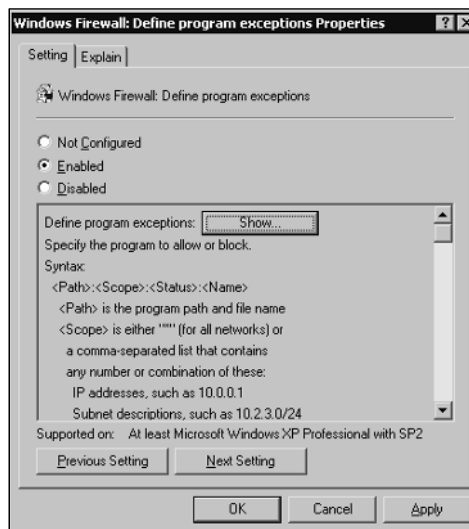
1. Open the **Group Policy Object Editor** console on the Active Directory Domain controller.
2. Locate the **Group Policy** setting found in **Computer Configuration\Administrative Templates\Network\Network Connections\Windows Firewall\Standard Profile**.

Figure 7.17 Group Policy Setting



3. Select the **Policy Setting for Windows Firewall** to enable the Define program exceptions policy.

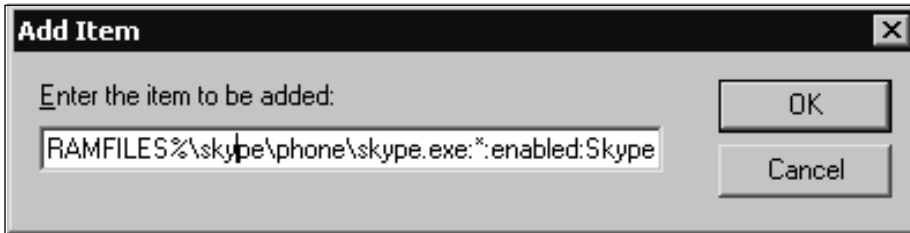
Figure 7.18 Define Programs Exceptions Properties



4. Next, click the **Show Button** that was enabled by the previous step.

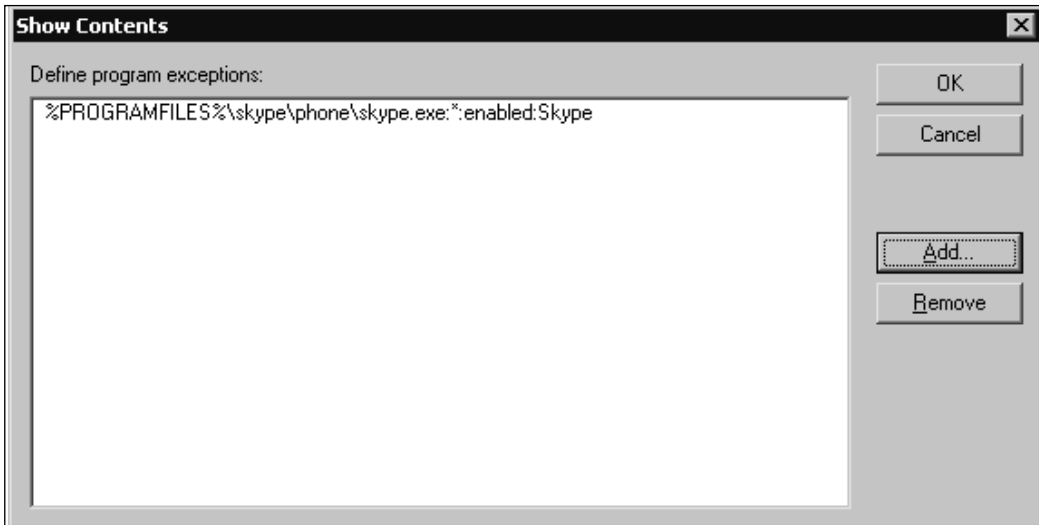
5. Add a definition for a Program Exception as
`%PROGRAMFILES%\skype\phone\skype.exe:*:enabled:Skype`
and then click **OK**.

Figure 7.19 Adding a Definition for a Program Exception



6. Click **OK** to close the Show Contents dialog box, then click the OK button to close the **Windows Firewall: Define program exceptions Properties** dialog box.

Figure 7.20 Show Contents Dialog Box



7. Allow time for the Group Policy to be refreshed. The time varies depending on the network settings. Allowing exceptions for Skype and opening up the recommended ports make it easier for Skype to establish reliable communications outside of your network. Other products, such as Norton Internet Security, McAfee Firewall Pro, and Zone Alarm Pro, have

similar functionality. Visit Skype's Web site at http://web.skype.com/help_firewalls.html for the specific configuration of your product.

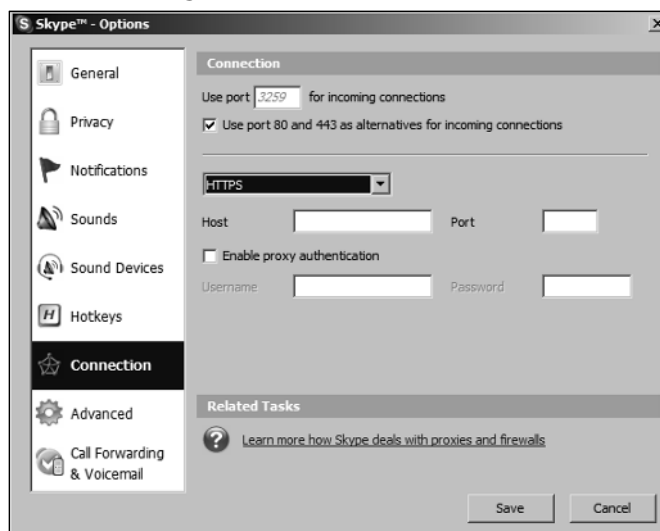
The same option can also be manually configured on each workstation in the enterprise by using the Windows Firewall applet in Control Panel.

1. Open **Control Panel** and double-click the **Windows Firewall** icon.
2. Click the **Exceptions** tab.
3. Tick the box next to **Skype**.

Using Proxy Servers and Skype

Many popular proxy servers are available on the market today. Skype supports HTTPS, SSL, and SOCKS5 proxy standards. Skype can optionally include authentication over proxies if the proxy server requires it. On Windows clients, Skype automatically uses the connection settings in Internet Explorer to identify the proxy settings that may be defined for that user on that computer. It is possible for the user to set Skype to use a manual configuration in the **Tools** menu, **Options**, and **Connection** tab settings.

Figure 7.21 Manual Configuration



If you are using a SOCKS5 proxy server, it must allow unrestricted connections to the ports discussed in the “Ports Required for Skype” section of this chapter.

Most proxy server solutions provide packet-filtering features. As previously mentioned, enabling packet filtering and restricting traffic over port 80 to only HTTP could cause communication problems for Skype.

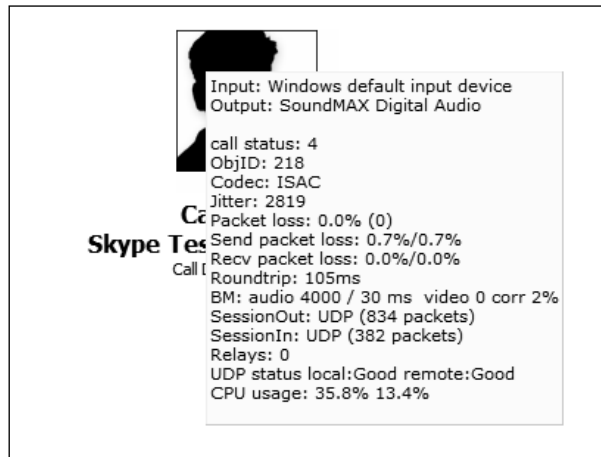
Many companies implement a wireless network, preferably using 802.11G, that directly connects to the Internet. If you want to then connect to company resources, you would VPN back into the corporate network just as you would from home or a hotel over the wireless network. The wireless network could allow for fewer restrictions on traffic for wireless clients while still allowing for stricter security on the wired devices. You should read the papers on wireless infrastructures at http://ciscu.org/bench_wireless.html for more information on implementing wireless in your enterprise.

If you are experiencing , high latency or poor voice quality with Skype, you can troubleshoot your connection quality by using NAT Check or Skype's Display Technical Call info feature found in the Advanced options tab. To enable the tech support feature or edit the Config.xml file manually:

1. Exit Skype.
2. Locate the **Config.xml** file located in the **<Drive>\Documents and Settings\<User Name>\Application Data\Skype\<Skype user name>** folder and open it with Notepad.exe or a similar text editor.
3. Find the setting **config/UI/Messages/DisplayCallInfo**
4. Change the value from 0 to 1 and save the file.
5. Launch Skype.

Visit Dan Douglass's Web site at the following URL for scripts and utilities to modify the config.xml file setting in a business environment:
www.codehatchery.com/skype.html.

Once you have enabled the **Display Technical call info** feature, you can make a test call to the Skype Test Call user. Once you have established the call, simply hover the mouse cursor over the user's avatar (picture), and you will see a tooltip-style popup with connection information:

Figure 7.22 Test Call

Note that in this scenario, the relays count is 0 and the roundtrip time is 105ms (1000ms = 1 second). Since the Skype answering machine is open, the connection is very clean, and there is very little latency.

Display Technical Call Information

The following is detailed information about the Technical Call Information popup items shown in the preceding and following examples.

Call Status

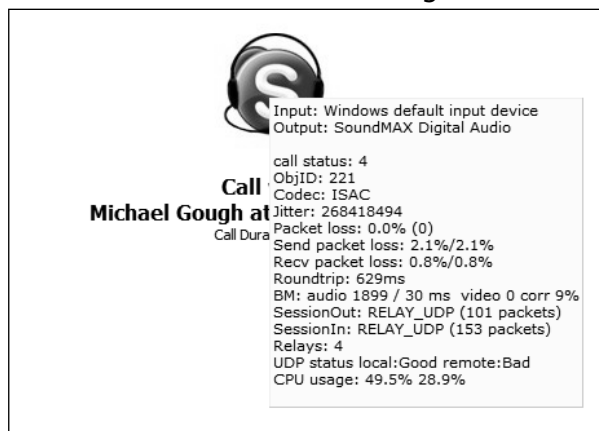
- 0 = Hosting conference.
- 1 = ROUTING - call is currently being routed.
- ?? EARLYMEDIA - with the pstn there is possibility that before the call is actually established, the early media is being played. For example, it can be a calling tone, or it can be some waiting message (all operators are busy, hold on for a sec) etc.
- ?? FAILED - call failed. Try to get FAILUREREASON for more information.
- 3 = RINGING - currently ringing.
- 4 = INPROGRESS - call is in progress.

- 5 = ONHOLD - call is placed on hold by you.
- ?? FINISHED - call is finished.
- ?? MISSED - call was missed.
- 8 = REFUSED - call was refused.
- 8 = BUSY - destination was busy i.e. pressed hang up button.
- 10 = ONHOLD - call is placed on hold by other party.
- 13 = CANCELED (Protocol 2)
- ObjID: Ignore this information as it is not important.
- Codec: ISAC is always the codec in use
- Jitter: Network administrators need to look at jitter. Jitter is the variation in the time between each of the delivered packets of data arriving from the source to the destination. This could indicate a bandwidth bottleneck or heavy traffic from the source to destination causing some packets to arrive sooner than others. The common method for reducing jitter is to buffer data at the destination.
- Packet Loss: Network administrators need to be aware of packet loss. This is the total percentage of the packets of data that don't make it to or from each party in the conversation. This should be low, but will be something if you are using UDP, since delivery is not guaranteed.
- Send packet loss: Network administrators should pay attention to this setting. This indicates how much data is not making it to the destination party in the call. If the Send packet loss is high, it means that something is causing the packets from getting to the remote client.
- Recv packet loss: Network administrators should pay attention to this setting. This indicates how much data is not making it from the other party in the call. If the Receive packet loss is high, it means that something is preventing the packets from getting to you from the remote client.
- Roundtrip: Normal users and Network administrators can get information from this. The higher the number is, the longer it takes for your voice to get to the other party and back. This should be low, and anything about 300ms starts to get choppy, reducing call quality. Look at SessionOut and SessionIN, or run NAT Check to determine why you are relaying.

- **BM:** This is related to the bandwidth and quality of the audio and is not important.
- **SessionOut:** Network administrators should look at this if roundtrip values are high. This should say UDP. If it says TCP or RELAY_UDP, then you are not operating at the best performance. In this case look at UDP status remote. If it says remote:Bad, then the remote party is behind a firewall and cannot receive UDP traffic inbound from the supernode.
- **SessionIn** Network Admins should look at this if roundtrip values are high. This should say UDP. If it says TCP or RELAY_UDP, you are not operating at the best performance. In this case look at UDP status local. If it says local:Bad, you could, at your discretion, open up the UDP port as discussed earlier in this chapter to allow inbound UDP traffic through the firewall inbound from the supernode.
- **Relays:** This is almost always 4, but 0 when you call Skype voice mail.
- **UDP status:** should always be local:Good remote:Good. If either are Bad, look at SessionIn/SessionOut to remedy.
- **CPU usage:** 35.8% 13.4% Total CPU usage of each processor by all running applications on the local machine. If this is too high, then the machine may be too overloaded to allow Skype to operate efficiently. Other applications will most likely be suffering as well.

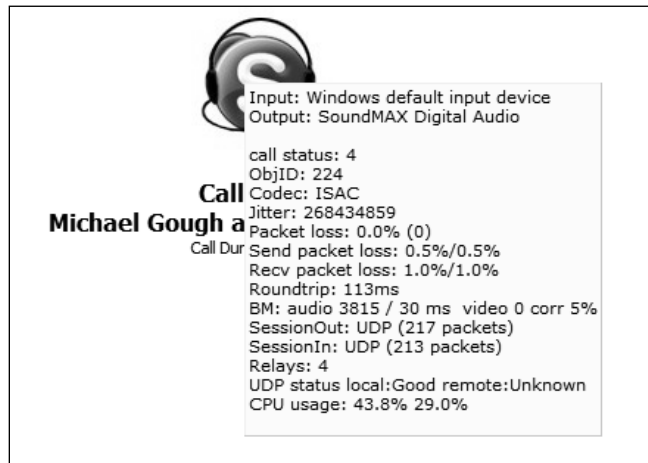
The next example is a call to a user on large corporate network where no inbound UDP is allowed back in through the firewall, and there is a very complex network infrastructure.

Figure 7.23 No Inbound UDP Permitted Through the Firewall



Note the difference in the SessionOut and SessionIn results. *RELAY_UDP*, and the UDP status *remote:Bad* show us that the remote location is the problem and that the UDP traffic is using a supernode to relay UDP information for each of the clients. The result of the relays is the long roundtrip time of 629ms, and therefore, there is a delay in transmitting the voice data to the remote client. Basically, it takes more than half a second for everything you say to get to the remote client, so the conversation is choppy and degraded. To improve this connection, the callers can use NAT Check to see if they are able to use UDP and troubleshoot the connection. If it is possible to open the UDP port inbound to the remote client in this scenario, the sessions can use a direct UDP or peer-to-peer connection, and the communication will be improve almost tenfold. See the following example, to the same caller, without the firewall restrictions:

Figure 7.24 Same Call, No Restrictions



To summarize, if you have a bad connection, each client can run NAT Check and the Display Technical info to see who is having difficulty communicating. The findings can be confirmed with the configuration demonstrated in the previous section. To correct the issue, determine the UDP port the trouble client is listening on. Open that port inbound by defining a firewall rule. The rule should be specific to the client, so it might be something like *Allow: WAN * to LAN 192.169.1.21 UDP: 3259*, which allows all WAN IP addresses to communicate inbound to the private LAN address 192.168.1.21 over UDP port 3259.

Small to Large Companies

In most large companies, this will not be feasible and may possibly be against the corporate security policy and allowable network practices, but this does remain an option for small to medium-sized businesses that desire better communication quality and have the flexibility to modify their firewall rules. Some firewalls allow rules to be enabled during a specific time frame, and outside of that time window, the rule is disabled. If you are using Skype only during business hours, this type of feature would provide better security than leaving the port open all the time. With any modification to your firewall rules, be sure to check your corporate security policy and with corporate security and your network team to gain approval and to understand the potential risks that are associated with opening any ports on a firewall to an internal client. Additional layers of security should be implemented if this configuration is to be used. If any peer-to-peer communication is allowed, it is recommended that the clients have a personal firewall solution to further protect the systems from malicious activity.

How to Block Skype in the Enterprise

From a security or network administrator's point of view, the very same features that make Skype connect reliably through a restrictive firewall present a challenge to preventing or blocking Skype traffic on a network. Skype is very robust and can function with access to only port 80. Most corporations allow outbound Web traffic, so port 80 (HTTP) must remain open. Port 443 is the SSL port (HTTPS), and secure Web sites require this port to remain open. It is not as simple as blocking ports to prevent Skype from functioning.

Several tasks must be completed to block Skype in your enterprise. The first step is to block access to the Skype downloads to prevent the executable from even being installed on your client machines. This practice is referred to as *black listing*. This step is not entirely effective by itself, since some users might already have the Skype client installed or could bring the installation package from home on a CD or thumb/flash drive.

It is good practice to prevent unnecessary applications from accessing the Internet. The best way to achieve that is by blocking all ports on the firewall and then selectively allowing known traffic to pass, the "deny all unless explicitly allowed" mentality. In addition, you may choose to restrict access to all Internet sites except those that have been approved by your organization. This is referred to as *white listing*, and although it requires more maintenance, it is much more secure.

Another method used to prevent communication over the Internet is to use packet filters. Packet filters examine the data inside the headers of transmitted packets. This information can be used to create rules to dump messages that contain headers that meet the filter criteria. Unfortunately, Skype data is encrypted, so packet filters are unable to examine the information in the data packets; therefore, packet filtering is useless. However, a new hardware device is purported to identify the signature of Skype communication and block Skype traffic based on that identification.

In a corporate enterprise environment, you may have other software solutions that allow the use of application filters on the desktops. This is another effective way to block Skype. The method of policies depends on the platform, but essentially, the concept is the same. When a user attempts to execute a program that is defined as disallowed, the process that monitors the client will prevent the program from executing. An example of this would be to use Microsoft Systems Management Server and define a restriction on the Skype.exe executable. Network Associates and Symantec have similar features built in to their groupware products.

Skype is very effective at finding ways to communicate with other Skype peers. There is no straightforward way to block Skype in the enterprise. The most effective method is to prevent the program from running at all or scan for it on all systems that are not approved and delete it from each system.

Endnote

1. “Number of Hosts Advertised in the DNS.” *Internet Domain Survey*, July 2005, www.isc.org (accessed October 4, 2005)

Summary

Skype markets itself differently than a standard instant messaging client. Skype emphasizes its voice capabilities, which provide Skype users with the ability to call another Skype user for free. Skype also includes features that allow it to call or receive calls from people on standard telephone lines. The rates for the calls are a fraction of what a standard telephone provider would charge, and are contributing to the growth of this client. Skype's voice quality is excellent, especially when compared to its competition, and this has allowed it to become one of the most popular communication tools currently available, with over 154,000,000 downloads of its client software. Besides voice communication, Skype includes features that are standard for other instant messaging clients, namely text messaging and file transfers. Since Skype encrypts all the data sent through its service, it can be viewed as a secure instant messaging platform. However, this security may be disconcerting to some users, since it is impossible to audit. Skype's protocol is closed, meaning that no one has access to the technical details of how it works, and due to its encryption, it is nearly impossible to analyze completely. Its encryption can be an issue for organizations that are required to monitor correspondence, since the data that is captured is encrypted and unreadable. Additionally, some organizations are required to monitor phone conversations; this is not possible with Skype, which may provide an alternative method for communication that circumvents standard systems and controls.

The features that provide communication cannot be regulated or monitored in ways that other systems such as e-mail can. Communications that are not monitored may result in a user sharing confidential or sensitive information to unknown or unauthorized individuals. This information is not limited to a conversation, but may also include files. Social engineering may be used by a malicious user to encourage data leakage, by convincing users to dispense with sensitive information or files. By posing as a known contact, a malicious user may be able to obtain information easily. Skype provides users with an option to store usernames and passwords on the client, and it is recommended that this feature be disabled to reduce the risk of a user operating your workstation to deceive your contacts and obtain information from them. It is recommended that you eliminate or at least limit the discussion of sensitive and personal information with others online. Credit card, bank information, or other personal and sensitive information should not be discussed with anyone online. Skype, along with other instant messaging clients, can be used to transfer copyrighted materials into an organization. File transfers may allow users to share movies, MP3s, or other restricted files.

For most users, Skype's security and encryption are beneficial in that it removes a security issue that all unencrypted instant messaging clients possess. Malicious users or rogue system administrators employing packet-capturing utilities are unable to decipher any communication between users of the Skype service. Since Skype's encryption is mandatory for all communication, all network traffic between Skype clients is unreadable.

Solutions Fast Track

Skype Architecture

- ☑ Skype has seen rapid growth in the adoption of its client software. From its beta release in August 2003 to August 2005, Skype has amassed over 150,000,000 downloads of its clients.
- ☑ Skype is similar to P2P networks in that most network traffic is handled by other clients. The Skype server is responsible for authentication of users, while routing traffic and other functions of the service are handled by clients.
- ☑ There are two types of clients: nodes and super nodes. Nodes are standard clients that are able to send and receive calls and other information. Super nodes are clients that have a large amount of bandwidth available and are using a public IP address (not behind a firewall or NAT gateway). These super nodes transfer data to other nodes and perform location services (through the Global Index) to ensure messages are sent to the proper recipient.
- ☑ Skype uses a wide range of ports (1024–65535), for both UDP and TCP. Skype's listens over a range of ports as well, including TCP/80 and TCP/443.

Features and Security Information

- ☑ Skype encrypts all of the information it sends over its service. This is due to the likelihood that it is relayed to one or more intermediate workstations before reaching its destination. Skype employs 256-bit AES encryption on all communication. Encryption is not optional, and is included in every client.

- ☑ Chat history is enabled by default and stores information for all text conversations. This information is stored as an unencrypted HTML (Hypertext Markup Language) files on the local drive. This feature can be disabled.

Malicious Code

- ☑ As of August 2005, there is no known malicious code that targets Skype clients.

Client Security

- ☑ Skype has had three vulnerabilities from its first release through August 2005, and maintains a Web page devoted to security information, available at <http://skype.com/security/>
- ☑ MSN Messenger is the largest target of all instant messaging clients. Worms and other malicious code have been created to take advantage of its wide distribution.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: What are the bandwidth requirements needed to use Skype?

A: According to Skype, the bandwidth requirements are minimal. Skype requires a 33.6 Kbps modem or higher to operate. Skype’s architecture is able to offload much of the data transmission to other workstations that are connected to the service, minimizing the load necessary on a single workstation. If there is a slow connection to the Internet, Skype would ensure that the workstation is not a super node, and would not handle data for other workstations. Skype also has multiple codecs that can be used to compress voice data. Based on the speed of the connection, Skype would simply choose the best codec based on the speed of the Internet connection and processing

power of the connected workstations. Skype itself uses between 3 and 16 Kbps during a voice call.

Q: How many people can use Skype at the same time for optimal sound quality?

A: Skype limits conferences to five people at a time. The reason for this is that all voice communications need to be combined by a single workstation and retransmitted to all the conference participants. This ensures that all parties are heard at the same time. There is a substantial amount of processing power required for all these conversations to be recorded, combined, and retransmitted. This also creates a larger burden on the bandwidth for all users in the conference.

Q: Can Skype be configured to prevent sharing of movies and other large files?

A: Skype does not have any limit to the size of files that can be sent or received. Depending on the configuration of a Skype client, file transfers can be very slow. If Skype users are unable to connect with each other directly (due to a firewall or router) to initiate a file transfer, the file is broken up and transmitted among many different workstations that are able to connect directly to the receiving workstation. Skype calls these transfers relayed transfer and limits the transfer rate to 1.0 Kbps, which makes this type of transfer quite slow and inefficient for larger files.

Q: Are there any security issues with Skype or potential vulnerabilities?

A: Since Skype is an encrypted protocol, all communications are hidden from malicious users who may be using packet capturing utilities. Skype has security issues that are similar to other instant messaging clients and services. Social engineering, data leakage, and malicious code and vulnerabilities are all security issues that affect Skype. However, Skype is primarily used for voice communication, which is less likely to be misused since a user can be relatively sure of the identity of the party at the other end of the conversation. At the time of this writing, Skype has not been affected by a worm on the service, and has had very few security vulnerabilities.

Part II

Peer-to-Peer Networks

Introduction to P2P

Solutions in this chapter:

- Welcome to Peer-to-Peer Networking
- The Next Step: Swarming
- Other Networks
- P2P Concerns
- The Future of P2P Networks

Introduction

As we have seen in instant-messaging clients, technology is well suited to satisfying certain needs. There is rapidity to IM, which even beats out the modern e-mail junkie's responsiveness. It is well suited to "dynamic content," which for participants flows from the whims and intents of the moment. And it is perfectly suited for "directed content," in which your IMs are "narrowcast" to (hopefully) your intended audience.

But what if the content isn't dynamic? What if the content is essentially fixed and has a life span of greater than one IM or e-mail exchange? What if you have a piece of art—a written story, for example—that you want to communicate? You certainly could type it up in an IM, but if it is intended to have a life outside of that communication and be read again by others, this isn't the best use of technology. Better to save it to an external file, say, a word-processing document, and then call up the file as needed. You could copy and paste this file into a message or even just transmit the whole file inside an IM or attached to an e-mail. This will work to get your story to a few people that you want to direct it to.

What if you want it to be available to more people and potentially even unknown recipients? You would then have to look at alternative technology to distribute your file. Certainly, a Web page could be created, but for the sake of argument, let's say you want to keep the integrity of the file as a file. File Transfer Protocol (FTP) would be a logical choice because it is designed to handle exactly this kind of task. On a server, your fixed content can be held for any number of users with whom you don't need to be familiar; all they need to know is how to connect to the FTP server via an FTP "client" on their machine. Of course, this still doesn't address how you can connect these unknown people to your file in the first place.

And this client/server solution has other difficulties. FTP has to be running on a server for people/clients to be able to connect. This means high availability; you can't just run this off a machine that is turned off when you disconnect from the Internet. It also implies reliability, since a machine that is your FTP server cannot be brought down, either accidentally or intentionally (by an outside attack, for example) or the availability disappears. For these reasons, most users cannot be bothered with setting up their own FTP servers; they instead use a centralized one provided by their ISP or some other service.

This, in turn, might incur charges for the user in terms of bandwidth, which is a shame, since most people have a broadband connection that sits around relatively unused in terms of bandwidth. It might not be important for the user to personally distribute the file in question, as long as he or she can make it available by some means and it doesn't get the location (or existence) of the file into the hands of a

much wider audience. This is where the strength of peer-to-peer, or P2P, networks comes into play. But let's take a step back for a second.

For most people using the Internet, it is easy to get the impression that most of the Internet's architecture is based on the client/server model. The first settings new users on the Internet have to configure with new broadband accounts are for the POP and SMTP server settings. And they are most likely directed to their ISP's Web server with completion of their sign-on. Even though they might not really grasp the concept of "server" or "client," still the impression is seeded that there are much "bigger" computers on the Internet to which their "little" computer needs to connect to accomplish tasks. The notion of peer-to-peer by contrast seems to be something much more modern.

It is easy to overlook the fact that P2P technologies have been in place since the earliest days of the Internet and that the concept is deployed in both the well-seasoned (Usenet, for example) and the fundamental (DNS is a prime example here) technologies of the Internet.

Welcome to Peer-to-Peer Networking

The modern boom in P2P deployment and technologies really came at a "sweet spot" along several curves in the timeline spanning the early to mid-1990s: the dropping cost of storage, the increasing processing power of home computers, the widening adoption of faster Internet connections, and compression technologies that, although "lossy," still maintained a reasonable resemblance to the original in the resultant compressed file.

Hard drives had been getting large enough to accommodate the storage of much more data than ever before, to the point that people could begin "warehousing" more and more complex files, which had larger space requirements than simple word-processing or spreadsheet documents.

Broadband connectivity was beginning to grow, putting better than measly dial-up connections in enough hands to establish a "critical mass" of well-connected users. Too few users on broadband would have precluded the adoption of P2P, because the barrier to entry (upload and download speeds) would have simply been too high.

The increased processing power of home computers allowed them to take on more complex tasks. Real-time decoding of compressed files was becoming possible, even if real-time encoding wasn't. (Although the decoding is more important, since the file would need to be decoded with every use and therefore it occurs more often, if the time cost of encoding is too high, this would slow mass adoption of compression.)

And, finally, new compression technologies, specific to audio, were reaching the general public. A single raw audio file could take upward of 30MB to 40MB on a hard disk that had precious little space for files this big. And files this large were unthinkable for transmitting across the Internet in an age where 1MB could easily take more than 10 minutes to upload. However, by leveraging newer compression technology, a user could, at the cost of a user-selectable loss of quality (hence “lossy”), easily take these files down to 10 percent of their size or less. MPEG-1, Audio Layer 3 would eventually reach the masses by its much more pronounceable extension: .mp3. (Although there was a lively debate back then as to which retained better fidelity for the compression, Layer 2—which began to take on popularity in roughly 1993—or Layer 3—which began gaining popularity around 1995—Layer 3 won out. I performed side-by-side tests back then and felt that Layer 2 did indeed sound better. Some of these files are still in my collection to this day.)

Larger storage space available to the average user, greater processing power to encode and decode compression of the files on the hard drive, a faster connection to other users on the Internet, and the adoption of MP3 by enough people all tilled the Internet landscape. It was people’s desire for easy access to music that became the seed. The explosion that was the modern P2P network burst forth from this user interest. So, let’s take a step forward.

We have an issue of distributing a file that is fixed content, like our story in the previous example or the example of an MP3 file. We are not dealing with the issue of directed content, since either of our examples is intended for consumption by anyone in the general public. We could upload our file to an FTP site, but that could incur costs for us when we otherwise have free bandwidth. It doesn’t need to be distributed from a site related to us (and we might not want it to be), and there is still no easy means to inform the general public of this file and where to get it.

Notes from the Underground ...

Other Classic File Swapping

Although the example of FTP is used here for simplicity, there are many more ways of distributing files on the Internet. Usenet was a common way of distributing all manner of files, including pictures, (very rough) video, applications, and other files. Certainly the early warez boards and other BBSs that predate the mass adoption of the Internet were a gathering point for people who wanted to swap files. IRC depot channels were another common solution. “Underground” networks like Hotline were common sources for the latest files in circulation. There were many other methods of trading files, but they all had something in common: For the most part they all “flew under the radar” and didn’t raise enough attention to draw too much unwanted interest. There were occasional minor busts, but it wasn’t until the P2P networks came to the fore, with their ease of use and the ability to put a user in contact with content the user wanted, that there came to be a specific targeted effort to stop users trading files.

Enter Napster

In 1999 Shawn Fanning released a program he had spent months working on in response to a problem his friend was having. MP3s had been around for a while by then, the hard disk space and processors of computers were up to the task, and the Internet connection at Northeastern University, where Shawn went to school, was more than adequate—the “sweet spot” on all the curves were in place. The one missing ingredient was that it was still difficult to find the files that a user wanted, or even find out if they existed. If you were interested in collecting MP3s, it often took a lot of your time to track down what you wanted.

Fanning wrote a program that could keep track of precisely that information. On centralized servers, Shawn’s application could keep track of the files that were out there and where they could be found. As users connected using his software, the servers would be updated and a current index of all the available files could be searched by any participant. The files themselves would still be kept on the end client’s machine (called a *peer*), thereby leveraging the individual storage capacity of all the combined users, and the server could facilitate the transfer of a requested file to another end client’s machine (another *peer*), thus leveraging the otherwise unused broadband on each client’s connection. Thus modern P2P networking was born.

Napster was an unqualified success, with almost 30 million users sharing 2.8 billion files.

By keeping all but the indexing and searching off the servers and relying on P2P for the distribution and exchange of files, Napster was able to facilitate the exchange of massive amounts of information, becoming the “killer app” that drove a whole new segment of the population online and spurred further P2P development (including an open-source response called OpenNap).

It was through this division between the actual content being exchanged and information about this content (where it was and how to get to it) that Napster thought it would keep its software safe from legal attacks. After all, it was the end clients who were exchanging files and possibly violating copyright, if copyright applied. All Napster was doing was facilitating communication between the peers; it had little control over what the users chose to do with that facility. As we will see later in this chapter, this division wound up not being as safe as Napster thought and would eventually lead to the company’s downfall.

Notes from the Underground ...

Copyright and the P2P Networks

Although there is some dispute as to exactly how much of the music being swapped on Napster was covered by copyright, it is worth noting that a vibrant selection of public domain works was made available and being consumed by a vast audience that otherwise would not have had access to it. In many respects, this is where the hidden strength of P2P is seated. Although the record companies were focused primarily on protecting the latest hits from the latest stars and didn’t want to see that revenue whittled away, the argument can be made that the value in P2P wasn’t the fact that a user could get the latest Brittany Spears single. Many people, of course, did do just that, but at the same time it would be hard to turn on a pop radio station or a music video channel and not hear the same song for free (and possibly copy it from these sources). There might be a value to acquiring the file off P2P networks, but that value is marginal.

Of greater value was the wealth of harder-to-find music, most of which was in the public domain. Suddenly a whole new segment of the population had access to this music in a way they never had before. Many new or smaller bands without big record contracts leveraged this tool to grow a fan base. And if a user had a sudden urge to find an obscure recording by an early Czech proto-punk band or a rare big-band recording from the 1930s, P2P gave him or her that

Continued

opportunity. Many works “on the fringe” of modern music could be found, whereas before it would have been almost impossible to reach many of these users.

The violation of copyright is certainly a much more tangible topic and has reached the ears of the general public, but it shouldn’t be overlooked that there is still a lot of value in P2P networks that doesn’t involve illegal file swapping.

Gnutella and a Purer P2P Network

In 2000, a response to Napster’s dependence on centralized servers was released. Napster was in the process of being brought down because its attempt to isolate its network function of locating files from the act of files being transferred by users was not enough in legal eyes. If Napster was to be held accountable for providing the servers that facilitated the exchange of files, what if the “servers” were not apart from the nodes on separate servers but were rather part of the client? In that way every client could shoulder a little of the responsibility and there would be no centralized servers for any legal action to attack.

The Gnutella network is based on this idea. It decentralizes all the servers’ functions out to all the clients, and the clients take on all the responsibility. A given node, upon launching, must connect to other nodes via a list (generated by a variety of means) of available other nodes. The new node “pings” its known nodes, and active nodes respond with information about the files they are sharing. A query can then be sent to the other peers and propagated as needed until the query’s condition is met and a reply is returned. From this point a simple HTTP *GET* request is initiated, and the file is on its way. (Again, the network primarily facilitates the search and find capabilities for the users.) A more purely P2P network took hold, and Gnutella skyrocketed in popularity as Napster collapsed in on itself.

The client software for the Gnutella network, such as BearShare, LimeWire, Gnucleus, and Morpheus, became some of the most popular tools for P2P file sharing. (More information on these clients can be found in the chapter on Gnutella; also see Table 8.1 later in this chapter.)

But there also is a problem with this purely P2P approach. The authors of Gnutella, Justin Frankel and Tom Pepper of Nullsoft (makers of Winamp), didn’t get a chance to fully develop the program. It had been released as an intended beta version on Nullsoft’s servers, unbeknown to Nullsoft’s new owner, AOL, which was in the midst of an ongoing merger with Time Warner. The software was removed from the servers, but not before thousands of copies had been downloaded, along with the source code. (It was released under the GPL as open source.) The program had only really been tested in a lab environment with a couple of hundred users, but then was

suddenly released “into the wild” with tens of thousands of active nodes. Although it is fine for a limited number of peers, the network quickly grew to a size where the technology just couldn’t scale appropriately. There were issues of disparity in bandwidth between users, and some technical problems, such as nonexpiring TTLs, compounded the overall demand on the network. Having vast numbers of clients all at the same peer level just led to chaos and slowed the network down.

Gnutella would eventually be able to adapt and change its technology in an important manner, with the advent of the ultrapeer.

The Rise of the Ultrapeer

For obvious reasons, all the file sharing we have been discussing can’t take place on centralized servers. As Napster learned, even just facilitating this sharing on centralized servers won’t work, because the network can be taken down by legal means. And the reaction—removing all the centralization and pushing it all out to the clients—only resulted in a system that quickly became overwhelmed, and that too resulted in an unusable network.

What then is left if we can’t centralize the network but at the same time we can’t completely decentralize it? Is there a middle ground? Yes, and that is the approach that Gnutella had to adopt, and it is the approach of a new network: FastTrack.

FastTrack developed software that, provided it was running on a client with ample processing power and had the benefit of a fast Internet connection, the software could promote itself from peer (or node) status to ultrapeer (or supernode). A supernode could then effectively become an indexing server for the peers connected to it. This allowed it to more efficiently process search requests than if every peer was allowed to talk with every other peer. The ultrapeer communicates with other ultrapeers until a file is found. At that time the ultrapeer then facilitates the communication between the requesting peer and the peer with the file, and a normal (HTTP) file transfer happens peer-to-peer.

The FastTrack protocol became immensely popular, having at one point more users than all Napster at its highest point. The software itself, as opposed to Gnutella, is closed source, and it wasn’t until more recently that portions of the communications were reverse-engineered and clients were developed for it outside the licensed clients. KaZaA (now Kazaa and all its variants), Grokster, Morpheus, and iMesh are the most popular and are addressed later in this book.

Certainly by this point in time, copyright holders’ attention had been drawn. A lot of the works being traded on the FastTrack networks were being traded in viola-

tion of copyrights. As we saw before for Napster, it wouldn't be long before court cases were filed, as we discuss later in this chapter.

Here also we saw a new “attack” against the P2P networks by the copyright holders. Legal avenues were being explored fully, but in an effort to stem the tide of downloading, record companies, artists, and labels tried seeding the networks with “poisoned files.” These files contained bogus information in an attempt to raise the noise-to-signal ratio on the P2P networks and make it more troublesome for P2P users. More on this topic is discussed later in the section on problems with P2P networks.

The Next Step: Swarming

With a strong network of distributed nodes in place, the next obstacle became how to increase the throughput of the P2P networks. With the size of the files being transferred, it might be some time before a new node would be able to share a new file it is acquiring. A new approach, called swarming, cleverly sidestepped this issue. If a file can be downloaded in small chunks and then immediately be shared by new nodes, this bidirectional transfer reduces the load on nodes that have the complete file, thereby distributing not only the file but its access as well. This can lead to an increased availability of high-demand files; the greater the demand, the “faster” the download. There are two different P2P networks that use this technique: eDonkey and BitTorrent.

eDonkey (Kademlia/OverNet)

eDonkey shares a bit in common with Napster due to the fact that it relies on centralized servers to facilitate file location and transfers while leveraging the power in swarming between the peers. But, taking a lesson from Napster, rather than one company housing the servers in a particular jurisdiction, which might make a juicy target for legal action, the eDonkey servers were released into the wild, where the community could maintain them. Anyone with enough processing power and bandwidth can set up a server and add it to the network.

However, even though so many servers scattered around the world are a much more difficult legal target, this P2P network does suffer from some of the downside of having to rely on centralized servers. In part, this is why the eDonkey network is shifting to the OverNet network. Much more like the Gnutella network, OverNet (based on the Kademlia protocol) shifts the communication and searching back to the nodes.

There are a few clients for the eDonkey network. eDonkey2000, the original client, has been outstripped in popularity by eMule. Other clients such as MLDonkey round out the list. For the time being, it seems that eDonkey is the most popular P2P network, but recent occurrences might change this situation dramatically.

BitTorrent

BitTorrent rose to the top of the P2P networks as Kazaa tumbled down. At one point, it was estimated that BitTorrent traffic accounted for 30 percent of all traffic on the Internet (www.cachelogic.com/research/). Its popularity was due in no small part to its swarming ability, which made its transfers very fast. It, like eDonkey, does have a reliance on centralized servers, trackers, but its nodes in a group put an emphasis on a smaller set of files, which affords it better bandwidth utilization (at least in terms of those files), and thus it allows for more rapid downloads.

Unlike the other P2P networks, BitTorrent does not offer the capability to search within the application. Instead, users must have prior knowledge of tracker sites and know where to look for the torrents they want to download. As with Napster, this makes for ready legal targets, and within this year major tracker sites have been taken offline, dramatically affecting the use of the BitTorrent network.

There is hope on the horizon for BitTorrent, however. Owing to BitTorrent's efficiency at rapidly delivering large, high-demand files, many companies are now looking at BitTorrent for legitimate distribution of files, such as upgrades and ISOs. At the end of September 2005, BitTorrent received an infusion of US\$8.75 million (<http://arstechnica.com/news.ars/post/20050929-5363.html>) to foster this development.

Some of the more popular BitTorrent clients include BitTorrent, BitTornado, Azureus, and BitComet, all of which are discussed in the chapter on BitTorrent.

Other Networks

A few other networks merit mention. WPNP, Freenet, and MUTE are fine examples of mainstream P2P network offspring. Table 8.1 also details some popular clients and the networks they run on:

- **WinMX Peer Network Protocol (WPNP)** WinMX was born out of the time of Napster and ran off the OpenNap network for some time until those servers were brought down. In response to that problem, WinMX developed its own protocol, WPNP. WinMX enjoyed quite a bit of popularity, but as of September 2005, its servers are down. It is understood that a

cease-and-desist letter was delivered to WinMX, but not much is known after that. There is some speculation that a move to the Pacific Island of Vanuatu, not unlike the move that Sharman Networks, owners of Kazaa, made, might be enough to put the company outside the legal arm of the Recording Industry Association of America (RIAA), but only time will tell on this matter.

- **Freenet** In response to many legal concerns, a new generation of P2P networks is being developed, offering some level of anonymity to the participants. Freenet is a fully decentralized system that is designed to be a cooperative, encrypted, distributed file system, resistant to outside attack and/or censorship. As such, the network is designed to protect the contents of it and the participant in it rather than designed as a file-swapping network per se. Some factors, like the overhead involved in encryption, would be an impediment to adoption for file swappers, but this network style is worth watching over time, because it may indeed be adopted as legal pressure increases on other P2P networks.
- **MUTE** Mute is another approach to anonymity that merits observation. It is a friend-to-friend (F2F) network that establishes a network of trust first. All exchanges are done directly with trusted peers and indirectly with peers who are further removed. This way it becomes much more difficult to track the behavior on the network. Again, this network style is worth examining for future possible adoption.

Table 8.1 Some Common Clients and the Networks on Which They Work

	Napster/ OpenNap (1999)	Gnutella (2000)	Gnutella 2 (2002)	FastTrack	WPNP	eDonkey2000/ OverNet	BitTorrent
Napster	X						
Napigator	X						
TekNap	X						
SunshineUN	X						
Lopster	X						
Xnap	X	X					
WinMX	X (until 2000)				X (after 2000)		
Bearshare		X					
LimeWire		X					
Morpheus	X	X (2003)	X	X (2001)		X	X
Shareaza		X	X				
Gnucleus		X	X				
Phex		X					
iMesh		X	X	X		X (removed)	
Grokster				X			
KaZaA				X			
eDonkey						X	X (plugin)
eMule		X				X	
MLDonkey	X		X	X		X	X
iSwipe	X	X		X		X	X
BitTorrent							X
BitTornado							X
Azureus							X
BitComet							X

Concerns with Using P2P Networks

Now that we have seen an overview of the history and basic architecture of some of the more common P2P networks, let's turn to some of the concerns with using P2P software. Some concerns have been alluded to in this chapter, and they all will be explored more fully in the chapters to come, but to get you thinking about these concerns now, let's look at three different areas: general concerns, infected or malicious files, and legal concerns.

General Concerns

Here we address some of the concerns that are necessarily tied into using P2P software and that should be in the forefront of any user's mind before he or she installs any P2P software.

Keep in mind that if you install a P2P client to connect to one of the networks, you are setting up software on your machine that is bidirectional. (It is, after all *peer-to-peer*.) In the excitement of finding all manner of wonderful files to download, it can be easy to forget that, as you do so, you are also sharing these files back out to the world. This means open ports, open services, and open shares. Quite often, in installation the P2P client will attempt to open ports for itself on your computer's firewall. You might have once thought your machine was locked down reasonably well, but you could fail to notice ports being opened up. In conjunction, the P2P client will have services for these ports running on your system. Each of them is one more exploitable vector for your machine that potentially opens it up to attack from the outside. There are known attacks in the wild, and more are discovered every day. Keep a close, rather than a blind, eye on activity in this range.

More common is the concern about open shares on your machine. By running the P2P client, you are now sharing a portion of your hard drive with potentially every other node on the network. It is not uncommon for a user to mistakenly allow too wide a scope for a share. A quick scan of the P2P networks turns up a treasure trove of files from people who have configured their clients to share their entire machines. This includes every file you have, including financial information, passwords, and files that you might not want to see the light of day.

Infected or Malicious Files

It is almost more common than not these days to find on P2P networks some kinds of files that have been tampered with, including files infected with viruses, worms, or Trojan horses. You might think you are downloading some great new program when

in reality you are just infecting your machine. Exercise more caution than you do in opening attachments to your e-mail.

Another kind of tampering, although not directly harmful, should be mentioned here. In an attempt to stem the tide of copyright violations, the P2P networks are being deliberately seeded with “poisoned files”—files that might look legitimate but in reality are not what they purport to be. The notion here is to discourage file swapping by raising the barrier to getting a good file. The people behind this practice reason that if the noise-to-signal ratio can be raised high enough, it will either discourage file sharing entirely or at least slow it down. One of the most famous early cases involved music tracks that seem to have been released into the P2P networks prior to the album even being out. In retaliation, the artist, Madonna, counter-seeded files of her own containing a loop of Madonna saying, “What the fuck do you think you are doing?” where music was otherwise expected to be heard.

More generally, it is not just the files on the P2P networks that you must concern yourself with; sometimes it is the client software. An early occurrence happened in early 2002 when it was discovered that Kazaa included in its install software called Altnet Secureinstall from Brilliant Digital Entertainment. The software could, via an independent connection, attach to other networks and add features, including, it seemed, the ability to use the client’s computer for Brilliant’s purposes. The company planned to sell the services of thousands and thousands of Kazaa users’ computers, without their express knowledge. (Language buried in the legalese of the end-user license agreement stipulated this but relied on the fact that most people will not sift through the pages and pages of dense text in the EULA when all they want to do is to run a program. This is a technique that is becoming more common with time. For a comparison of EULAs for some P2P applications, see www.benedelman.org/spyware/p2p/. For readers interested in evaluating the content of any EULA, a free program, EULalyzer, is available for download at www.javacoolsoftware.com/eulalyzer.html. This tool can highlight some of the common questionable area of a EULA, but users should be aware that there is no substitute for reading the EULA on their own.)

Unfortunately, this quickly became a revenue stream for P2P networks that did not have a direct way to otherwise profit from the networks they developed.

Other P2P software contains adware, spyware, or most recently, Trojans. Listing all the offending clients and their payloads here is just not possible. The list can change so rapidly that by the time this book is in your hands, new packages could be added, old ones removed, and clients that used to be clean might now have spyware in them. At one time or another, all the following have contained spyware: Grokster, Kazaa, BearShare, eDonkey2000, LimeWire, and Morpheus. The latter two

are the only ones on this list that are thought to currently be free of spyware. The reader is recommended to do his or her own research prior to installing any application. Two sites that might be of assistance in this effort are <http://www3.ca.com/securityadvisor/pest/> and www.spywareinfo.com/articles/p2p/, although it must be noted that this information can change rapidly and that what is safe today might not be tomorrow.

Also the reader must be cognizant of where his or her copy of the P2P client comes from. It is not unheard of for someone to infect a client with malware, spyware, viruses, or Trojans and then release the infected copies, as though they were the original. Try to get clients from a safe and known source.

Legal Concerns

No discussion of P2P networks can be complete without delving into the legal cases that have plagued the technology. As mentioned, most of the P2P networks have met their demise on the wrong side of a gavel. The legal landscape is changing, and it is important to know where the cases have been and potentially where they are going.

Sony Corp v. Universal City Studios

The first case that is important actually predates any of the modern P2P networks (and most of the Internet) by some time. Movie studios brought suit against Sony for selling a device that would allow people to duplicate aired television shows. In *Sony Corp v. Universal City Studios*, 464 U.S. 417 (1984; www.justia.us/us/464/417/case.html) the studios argued that the Sony Betamax VTR encouraged the violation of the studios' copyright on works they aired. The U.S. Supreme Court eventually ruled that the technology was not inherently illegal if there could be established a "substantial noninfringing" use for the technology. In the words of the court:

The sale of copying equipment, like the sale of other articles of commerce, does not constitute contributory infringement if the product is widely used for legitimate, unobjectionable purposes, or, indeed, is merely capable of substantial noninfringing uses the business of supplying the equipment that makes such copying feasible should not be stifled simply because the equipment is used by some individuals to make unauthorized reproductions of respondents' works.

As long as there were legitimate purposes for which the technology was predominately used, there was no infringement. This is what Napster was counting on.

A&M Records Inc. v. Napster Inc.

In 1999, a lawsuit was filed in U.S. District Court in Northern California on behalf of the Recording Industry Association of America (RIAA) against Napster. Napster's network that facilitated the swapping of MP3 files, argued the RIAA, violated state and federal statutes by "contributory and vicarious copyright infringement"—*A&M Records Inc. v. Napster Inc.*, 239 F.3d 1004 (9th Cir. 2001; www.ce9.uscourts.gov/web/newopinions.nsf/0/c4f204f69c2538f6882569f100616b06?).

Napster, in part emboldened by the ruling in the Sony case, argued that it did not violate copyright, since "fair use" precluded this, there was a misrepresented use of copyright, and First Amendment concerns took precedence.

Napster's fair-use argument rested on several points: Napster did not profit directly, therefore it was noncommercial activity; it offered the "sampling" of music; it allowed "space-shifting" (a nod to the "time-shifting" that was deemed permissible in the Sony case); and rather than cutting into music industry profits, Napster's actions increased the demand for CDs.

The court found that, although there was not a direct profit from the action of Napster, its actions were indeed directed at raising advertising revenue, because Napster did have "the right and ability to supervise the infringing activity and also has a direct financial interest in such activities." Napster's users didn't "sample" the music in any traditional sense; rather, they obtained complete and permanent copies of the music. Further, the evidence presented to the court indicated that the music files were not "space-shifted," but instead users were collecting new files, previously not owned, and that therefore users' demand for the CDs in question would not increase because they were being given them for free.

Napster was found to have contributed to third-party direct infringement of copyrighted material, and the Napster network and its servers were brought down.

We saw earlier what effect this had on the P2P networks. Outside of BitTorrent, most modern networks move to decentralize their servers so as not to directly provide the same legal target that Napster did.

MGM Studios Inc. v. Grokster Ltd.

The next big case against P2P networks was initially against both Grokster and Kazaa. The case against Kazaa was dropped because the company was based in Vanuatu, well outside the jurisdiction of U.S. courts. Again, the argument was made that the P2P networks were facilitating infringement of copyright. But in this situation, the U.S. District Court in California, guided by the Sony Betamax decision, dismissed the case. On appeal, the 9th Circuit Court of Appeals stated that P2P net-

works did indeed have legitimate uses. However, by the time the case reached the Supreme Court in June 2005, the court had reached a different decision in *MGM Studios Inc. v. Grokster Ltd.*:

We hold that one who distributes a device with the object of promoting its use to infringe copyright, as shown by the clear expression or other affirmative steps taken to foster infringement, is liable for the resulting acts of infringement by third parties.

This decision was to have a chilling effect on the P2P networks. In September 2005, appearing before the Senate Judiciary Committee on Protecting Copyright and Innovation in a Post-Grokster World, Sam Yagan, president of MetaMachine, the company that is responsible for eDonkey, said:

I am not here as an active participant in the future of P2P, but rather as one who has thrown in his towel and with no interest in replaying past issues ... The Grokster standard requires divining a company's "intent," the decision was essentially a call to litigate. This is critical because most startup companies just don't have very much money. Whereas I could have managed to pay for a summary judgment hearing under *Betamax*, I simply couldn't afford the protracted litigation needed to prove my case in court under *Grokster*. Without that financial ability, exiting the business was our only option despite my confidence that we never induced infringement and that we would have prevailed under the *Grokster* standard.

Although the loss of eDonkey and other P2P networks is a blow to the technology, it is by no means the end. Every time the networks get squeezed, a new network with new solutions pops up elsewhere. Perhaps organizations like the RIAA are aware of this capability for rebirth, and this is the reason they have taken a new tack.

RIAA vs. The People

Starting in 2003, the RIAA took a new approach to attempting to control P2P networks. After a loss in Federal Appeals Court when ISP Verizon refused to turn over information on its customers, the RIAA began to accumulate information on the behavior of—not those responsible for the software or networks of P2P, but the users of the networks. On September 8, 2003, the recording industry filed 261 lawsuits against individual file sharers. Many of these cases were settled directly with the RIAA by paying the damages demanded of the P2P users. Since then, over 10,000 lawsuits (read: over 13,000 but still waiting to get verification on this figure; also read 1,500 two months after the *Grokster* ruling) have been filed. Very few P2P users

have been willing to take their cases to court, but as time goes on more people are taking a stand.

As more cases of deceased grandmothers and 14-year-old kids being sued reach the nation's media, it becomes more apparent that in many cases inadequately researched “boilerplate lawsuits” are quickly filed, ostensibly for effect. A backlash against the RIAA begins to grow, and some people refuse to settle, instead insisting on their day in court.

This is just the beginning, and it is too early to tell how the situation will resolve itself. For more information on the current state of legal issues surrounding P2P networks, visit the Electronic Frontier Foundation at www.eff.org.

The Future of P2P Networks

As with any prognostication, it is impossible to predict exactly where the P2P networks will go, but by taking a look at what has happened historically and the trends that are on the horizon, we can get a general feel for the future of P2P networks.

The popularity of P2P networks is indisputable, with some estimates asserting that P2P is responsible for a major portion of the traffic on the Internet. Although specific P2P networks have been brought down by legal action, the resilience of the technology and the demand for its services keep P2P reincarnating. P2P will be difficult, if not next to impossible, to squash.

Already we have seen shifts to anonymity and encryption to hide users and content as well as relocation of vital components to jurisdictions outside prosecution. The chilling effect of the *MGM vs. Grokster* case is already pushing networks out, and although some may shut down, others using different approaches will replace them.

However, there is a shift in the zeitgeist for P2P's end users. In part due to better education of end users and in part due to newer and legal downloading services, some percentage of the P2P user space is no longer swapping copyrighted works. Although by no means a complete substitute for all the file swapping on P2P networks, the response to consumer demands for a reasonable way to get online music has created a whole new selection of legal sources for MP3, such as the wildly popular Apple iTunes Music store. Some users, as with anything in life, persist in swapping, regardless of the history of this practice. But if solutions like iTunes continue to develop, the portion of end users who simply want easy and affordable access to online music will have little need to swap copyrighted materials, and the value of P2P for them will be access to the wealth of uncopyrighted media that otherwise would be prohibitively difficult to access.

P2P has demonstrated itself to be a more than viable means of transmitting files, so it should continue to see commercial use for legal distribution of all manner of files. Many companies are turning to solutions like BitTorrent for release of patches and ISOs. In addition, media content creators, once thought to be the enemy of P2P networks, are turning to P2P to give their works life for a much larger audience. Already the British Broadcasting Corporation (BBC) has announced iMP (www.bbc.co.uk/imp/), a means for distributing its over-the-air content to Internet users. Other networks are looking to follow, and P2P will become mainstreamed as the legitimate uses for it overtake more questionable ones.

The dust has yet to settle on the P2P controversy. Exactly how the content creators can generate revenue from their products has not fully been explored. The hot debate over the inclusion of Digital Rights Management (DRM), which content creators want to maintain some control over their works but which frustrates end users due to its complexity to use, is still far from conclusion. But one thing is certain: P2P networks will continue to be a fundamental component of the Internet experience.

Gnutella Architecture

Solutions in this chapter:

- Gnutella Clients and Network
- Using LimeWire
- Using Morpheus
- Other Gnutella Clients

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

Now that you understand the basic concepts of how peer-to-peer (P2P) software file-sharing programs work, let's look at some of the most commonly used networks and available clients, examining how they can be configured for a higher degree of security. This chapter focuses on the architecture of the Gnutella network, concentrating on the LimeWire and Morpheus clients. LimeWire is one of the most commonly used file-sharing programs because of its ease of use and the availability of files on the Gnutella network. This chapter also looks at LimeWire in depth, including how to search the network for files to download and what security risks it may lead to.

LimeWire is one of the best examples of P2P software. It explains how your workstation (or entire network) can become infected with malicious software (malware) from software downloads via the network, and discusses the precautions you can take to avoid the dangers associated with P2P networks and clients. This chapter examines the fundamental process of downloading and installing a P2P file-sharing program. You will install, configure, and use LimeWire. You will see how a P2P file-sharing program opens your workstation to attack, and you will learn how to protect yourself. The chapter ends with a discussion on why it is important to secure P2P architectures within your corporate environment or personal PC.

Gnutella Clients and Network

The Gnutella network is a P2P network that was designed to allow users the ability to share files without worrying about network access being blocked by system administrators. Targeted towards university students, it instantly became a popular network because of its many built-in privacy measures. Over the course of its continued use, it has gained a lot of attention from developers who have produced dozens of client applications that could be used to access the network.

Gnutella

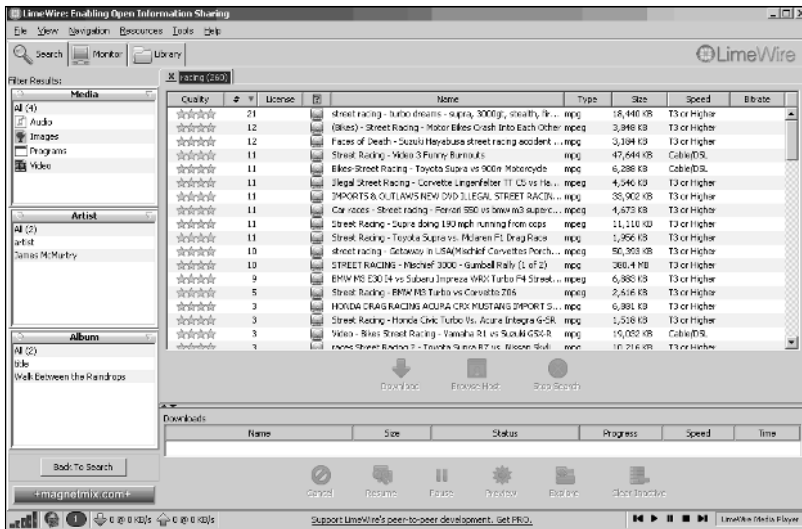
Gnutella was designed and created by the two founders of Nullsoft, the company that also created Winamp, SHOUTcast, and WASTE. Justin Frankel and Tom Pepper designed the protocol and its related client application to help user's easily share data, while allowing for total control over how the data was shared. This control allowed knowledgeable users to bypass firewalls and enter networks without centralized servers. Nullsoft started out as a small business; however, America Online purchased it in June 1999, with the founders staying on as full-time developers.

The protocol and the original Gnutella application were unveiled in March 2000. It was first publicized on the Web site *Slashdot.org*, where tens of thousands of users downloaded it within the first day. There were also plans to release the source code to the Internet public, but these plans were short lived. Within a day after the original program was announced, America Online forced the removal of the source code from Nullsoft's Web site. This move was not unexpected; the program was originally described as being an alternative service for transferring copyrighted material and any other forms of data across the Internet. This raised too many legal concerns for America Online, and the developers were ordered to stop work on the project. However, the closure was not quick enough to prevent the source code from ending up in the hands of thousands of users. Some of the more enterprising computer experts immediately began reverse-engineering the protocol used by the application, to design their own clients (known as *servents*) that could be used with the network.

LimeWire

LimeWire is a free, open-source Gnutella client located at <http://www.limewire.com>. It is programmed in Java, which allows it to run on any operating system with a Java engine (e.g., Microsoft Windows, Linux, and Mac OS X), and is the most popular Gnutella client currently used (see Figure 9.1). In late 2005, under pressure from the U.S. legal system and the Recording Industry Association of America (RIAA), LimeWire started altering their client to reduce the sharing of copyrighted material on the Gnutella network. This announcement caused the LimeWire source code to be forked, which means that the developer used existing code to build a spin-off of the original application. The new application is named FrostWire and can be located at <http://www.frostwire.com>.

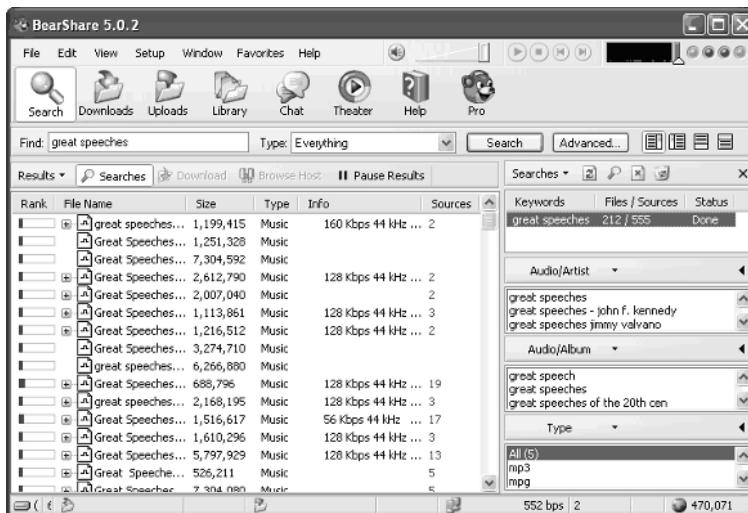
Figure 9.1 LimeWire client



BearShare

BearShare is another popular Gnutella client that was developed by Free Peers, Inc. and released as a closed-source application. It is currently provided in different variations, from an ad-supported free version to an ad-free commercial version. BearShare can be downloaded from its Web site at <http://www.bearshare.com>, which also provides screenshots (see Figure 9.2).

Figure 9.2 BearShare client



Gnucleus

Gnucleus is one of the oldest and free open-source Gnutella clients. Located at <http://www.gnucleus.com/Gnucleus>, it has implemented many new concepts and technologies into the Gnutella network. It introduced the concept of an UltraPeer, which is a peer that stores file information on other peers that are connected to it. This way it responds to search queries, saving the slower peers from the burden of bandwidth-intensive searches.

Morpheus

Morpheus is a P2P client that has traveled a variety of P2P networks. Though originally designed for the OpenNAP network, it was moved to the FastTrack protocol after OpenNAP's RIAA-induced shutdown. Morpheus' stay with FastTrack was short lived, though, as it was barred from the network in February 2002. It then recreated itself as a Gnutella client, using existing code made available by Gnucleus.

Gnutella Architecture

The Gnutella network is based on a decentralized architecture, making it a true P2P network that does not involve central servers for authentication, indexing, or assisting in file searches. This architecture eliminated the problems associated with a centralized server, including traceability and scalability, thus allowing the network to grow as users connect to it.

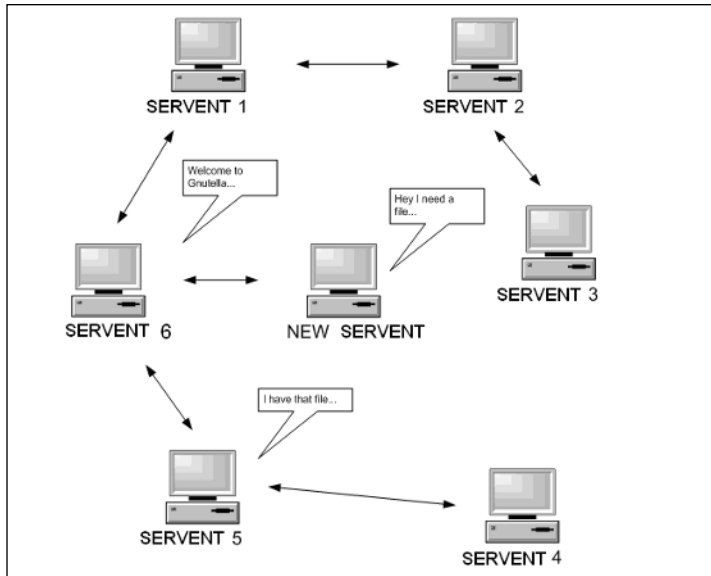
When connecting a Gnutella servent to a network, it must have some knowledge of which systems to connect to in order to join the network and begin sharing files. Gnutella servents connect to several Internet Protocol (IP) addresses that were installed with the client software. IP addresses are aware of other servents on the network, and transfer information to them with the one that has just connected.

NOTE

A "servent" is a workstation with client software installed, which functions as both a client and a server for the network. All peers on the Gnutella network are servents, and are considered equals, capable of transmitting searches throughout the network and sending and receiving files. There is no central server for authentication and aiding in searches in this architecture.

After the servent has connected to the Gnutella network and is aware of the IP addresses of nearby peers, it is capable of utilizing the features of the network, such as searching for files and downloading shared files (see Figure 9.3).

Figure 9.3 Viewing the Gnutella Architecture



When a servent initiates a file search, it begins by querying the servents it is connected to. In turn, those servents send the query to the servents they are connected to. This process continues as search results are sent to the requesting servent. For downloading files from the Gnutella network, there is a similar process that occurs for transferring files. As the process for downloading a file begins, the servent requesting the download contacts the servent that answered its search request. If that download does not start, the download request is passed from the reporting servent to the other servents that it is connected to. Eventually, the download request will be answered by a servent, which contacts the servent that forwarded the original download request.

Notes from the Underground...

Gnutella Communication

To understand the Gnutella architecture, you should familiarize yourself with the terminology of its components. For example, most times when looking at a network map, you will see something labeled a “server” or “client.” A peer is considered one of two computers (or systems) that communicate with each other as well as playing similar roles. When working with a central server solution, the systems are not peers, since the central server plays a unique role in the architecture. Gnutella’s P2P model does not use servers; therefore, the network is composed entirely of peers. Computers connected to the Gnutella network are usually referred to as *servents*, but may also be called “clients,” “nodes,” or “hosts.”

Servents connected to the network communicate via *pings* and their responses known as *pongs*. As a servent connects to the network, its presence is announced on the network as a broadcasted ping request. All peers that receive this ping request respond with a pong, which is an acknowledgement that the ping request was received. The acknowledgement also contains information about the responding servent, including communication information such as its IP address and the port it communicates on, as well as file-sharing information such as the total number of files being shared on the network and their size.

UltraPeers

A further improvement of the Gnutella protocol introduced the concept of UltraPeers. Styled similar to the SuperNodes of the FastTrack network, UltraPeers are regular clients that collect connection and file-sharing information from peers that are connected to it, and storing these values in a local database. This way, when a search query is passed along the network, the UltraPeer will respond back to it in place of a client beneath it. This allows servents with additional available bandwidth to handle much of the regular traffic across the Gnutella network, saving the traffic from inundating servents that are on slower connections. The clients that are connected to an UltraPeer are known as *leafnodes*. Many modern Gnutella clients are configured to connect users to an UltraPeer first. While the ability to become an UltraPeer is automatic and usually not noticeable to a casual user, it can cause a client’s bandwidth use and processing power to increase, due to the need to manage

up to 75 other servents. However, clients that implement this ability also have the option of the user disabling it within the application.

Gnutella Protocol

Gnutella features a well-documented protocol design, which has been used in a variety of client applications. The latest documentation on the Gnutella protocol can be found at <http://rfc-gnutella.sourceforge.net/developer/stable/>. The Gnutella protocol is broken down into a few key areas to handle the management of traffic. The areas covered here are: peer connections, descriptors, and file transfers.

Peer Connections

To connect to the Gnutella network, a client must first make connections to servents that are already on the network. This is made by sending customized User Datagram Protocol (UDP) ping packets to a list of IP addresses that it is aware of. This list is usually included within the client application. The ping packets and their responding pong packets are sent using descriptors, a specialized packet that Gnutella uses for administrative functions.

After performing a ping sweep of known IP addresses, the client attempts to connect to each address until it gathers a sufficient number of peers. It does this by sending a single Transfer Control Protocol (TCP) packet to the servent, and waiting for an appropriate response. Each packet contains a 22-byte payload with a hex code and the American Standard Code for Information Interchange (ASCII) equivalent of:

```
47 4e 55 54 45 4c 4c 41 20 43 4f 4e 4e 45  47 4e 55 54 45 4c 4c 41 20 43 4f 4e 4e 45  GNUTELLA A CONNE
43 54 2f 30 2e 36 0d 0a                      CT/0.6. .
```

This packet is the initial connection packet sent by the client to a Gnutella servent. The packet includes the static string *GNUTELLA CONNECT/* followed by an ASCII string of the protocol version used (e.g., 0.6). The packet is then terminated with a carriage return and line feed, which is generally displayed as `\n` or as `0x0d0a` in hexadecimal.

A servent must then reply to this request to notify the client of whether they will be allowed to connect. The reply is structured similar to a Hypertext Transfer Protocol (HTTP) connection reply (e.g., an approved connection will contain the following payload: *GNUTELLA/0.6 200 OK*). Similar to the connection request, the “0.6” refers to the protocol version and the packet terminates with `0x0d0a` values. In cases where a servent refuses a connection, the payload may be similar to the following: *GNUTELLA/0.6 503 Service Unavailable*.

Table 9.2 Payload Descriptor Types

Value	Type of Data	Description
0x00	Ping	Checks to see if another peer is online
0x01	Pong	A ping response
0x80	Query	A search query
0x81	QueryHits	The results of a search query, if matches were found
0x40	Push	A message sent to a firewalled peer to allow files to be transmitted past a firewall

Depending on the type of Payload Descriptor specified in the header, the actual descriptor payload could vary wildly. Most payloads are variable length, and thus require knowledge of the payload structure to be able to determine their contents. Some popular network analyzing tools, such as Ethereal, are not able to see these packets as Gnutella transmissions; instead, they are shown as normal UDP packets. Many payloads also contain optional Gnutella Generic Extension Protocol (GGEP), which is used for extensions to the original Gnutella protocol. The structure and format of GGEP data can be found at <http://rfc-gnutella.sourceforge.net/developer/testing/GGEP.html>.

Ping/Pong Descriptor Packets

In the case of ping packets, the payload is not required for it to function. Instead, the payload is usually omitted or contains optional GGEP data. In response to a received ping packet, a peer will reply back with a pong packet. This packet contains vital information about the servent and its capabilities on the network. The length of the pong payload is determined by the Payload Length byte in the descriptor header. A pong descriptor message will contain a hexadecimal and ASCII structure similar to:

```
ca 18 42 cf 4b 20 a8 00 00 00 00 10 00 c3 02 ..B.K .. .....
44 55 42 c0 a6 03 47 55 45 41 01 02 49 50 46 41 DUB...GU EA..IPFA
cf 56 ba 02 3f 03 49 50 50 7c a5 a5 85 5c ca 18 .V...?.IP P|...\..
ac d6 92 c3 ca 18 40 cb 3e 57 ca 18 43 bc 5d 17 .....@. >W..C.].
ca 18 45 98 90 3b cc 18 56 8a 74 04 cc 18 81 44 ..E...;... V.t....D
07 fb ce 18 d8 d3 57 18 bb 33 ac c9 1c 85 12 66 .....W. .3.....f
43 b9 29 40 9d 6b 03 4c 4f 43 43 65 6e 02 02 55 C.)@.k.L OCCen..U
50 43 01 1c 00 82 56 43 45 4c 49 4d 45 48 PC....VC ELIMEH
```

This particular packet had a payload length of 0x73, which equals 126 bytes (see Table 9.3).

Table 9.3 Pong Descriptor Payload

Data Size in Bytes	Hex Value	Description
2	0xca18	The port that the peer listens on. To determine this, reverse the two bytes and convert to decimal. 0x18ca is port 6346.
4	0x42cf4b20	IP Address of the peer, with each hex byte corresponding to a single octet. 0x42.0xcf.0x4b.0x20 is IP 66.207.75.32.
4	0xa8000000	Number of files shared by the peer.
4	0x00001000	Amount of Data shared by the peer in Kilobytes
Variable	Omitted for brevity	Optional Data that could be included in the Pong packet. It takes up the size of the rest of the packet, up to the previously defined Payload Length.

Query Descriptor Packets

A Query descriptor packet is one that allows searches to take place on the Gnutella network. When a client enters in a keyword for a file they want to search for, a Query packet is created and transmitted to every server that the client is connected to. These servers check to see if they are sharing the requested data, and pass along the query packet to every node that they are connected to. This relaying of the packet is performed until the Time to Live (TTL) is reduced to 0. The length of the query payload is determined by the Payload Length byte in the descriptor header. A Query descriptor message will contain a hexadecimal and ASCII structure similar to:

```
e0 00 72 61 63 59 6e 67 00 75 72 6e 3a 00    ..racing .urn:.
```

This particular packet, a search for the word “racing”, had a payload length of 0x0e, which equals 14 bytes. The hexadecimal data within this payload is broken down as shown in Table 9.4.

Table 9.4 Query Descriptor Payload

Data Size in Bytes	Hex Value	Description
1	0xe0	The minimum connection speed of a peer, in kilobits/sec, that the client is willing to download from.
Variable	0x00726163596e6700	The search string that was sent by the client. It begins and ends with a NULL character (0x00). It is also referred to as a NULL-terminated string.
Variable	0x75726e3a00	Optional Query Data. This normally contains a URN (Uniform Resource Name) which provides additional ways of identifying a file. For example, some queries include SHA-1 hash values as "urn:sha1:YNCK-HTQCWBTRNJIV4WNAE52SJUQCZO5C"

QueryHits Descriptor Packets

As Query packets traverse the Gnutella network and each servent that receives one, checks to see if they are sharing the requested data. If not, they discard the search query. However, when a servent is sharing requested data, it sends a QueryHits descriptor packet back to the requesting peer. This packet includes information about the servent sharing the file and the file itself, to allow the client to decide if they wish to download it. Upon receiving a set of QueryHits packets, a client chooses a file to download and the transfer begins, as these packets contain all of the vital connection information. The length of the QueryHits payload is determined by the Payload Length byte in the descriptor header. A QueryHits descriptor message will contain a hexadecimal and ASCII structure similar to:

```
01 59 0e c0 a8 01 21 5e 01 00 00 08 02 00 00 3c .Y....!^ .....<
1d 3c 00 43 61 72 73 20 2d 20 4d 75 73 74 61 6e .<.Cars - Mustan
67 20 76 73 2e 20 43 68 65 76 65 6c 6c 65 20 64 g vs. Ch evelle d
```

```

72 61 67 20 72 61 63 69 6e 67 20 28 31 29 2e 6d rag raci ng (1).m
70 65 67 00 75 72 6e 3a 73 68 61 31 3a 4c 52 49 peg.urn: sha1:LRI
57 49 45 4c 4a 4a 49 44 33 35 43 4d 5a 41 47 4b WIELJJID 35CMZAGK
51 4a 32 36 44 36 34 36 47 54 47 4c 56 1c c3 82 QJ26D646 GTGLV...
43 54 44 94 f7 f4 3e 00 4c 49 4d 45 04 3d 21 01 CTD...>. LIME.=!.
00 01 c3 02 42 48 40 84 50 55 53 48 52 18 a8 6e ....BH@. PUSHR..n
58 ca 18 44 47 e2 23 ca 18 44 77 42 71 ca 18 00 X..DG.#. .DwBq...
d4 8d ad 01 e6 08 f4 21 91 7f 2e 58 ad 81 2d 00 .....! ...X...

```

This particular packet, a query result for the word “racing,” had a payload length of 0xb0, which equals 176 bytes. The hexadecimal data within this payload is broken down (see Table 9.5).

Table 9.5 QueryHits Descriptor Payload

Data Size in Bytes	Hex Value	Description
1	0x01	The number of results that the servent is sharing responding with.
2	0x590e	The port that the servent listens on. To determine this, reverse the two bytes and convert to decimal. 0x0e59 is port 3673
4	0xc0a80121	IP Address of the servent, with each hex byte corresponding to a single octet. In this case, 0xc0.0xa8.0x01.0x21 is IP 192.168.1.33.
4	0x5e010000	The connection speed of the servent, though in extended formats this could also indicate if the servent is behind a firewall.
Variable	0x080200 ... f43e00	The actual set of results from the peer. This structure is broken down in Table 9.6.
Variable	0x4c494d ... ca1800	Optional extended data that is used for protocol extensions.
16	0xd48dad ... 812d00	A 16-byte Servent Identifier that uniquely identifies a servent on the Gnutella network.

You can see exactly what download choices were offered to a Gnutella client to pick from. Below is an extract of just the set of results from the QueryHits response:

```
08 02 00 00 3c 1d 3c 00 43 61 72 73 20 2d 20 4d ....<.<. Cars - M
75 73 74 61 6e 67 20 76 73 2e 20 43 68 65 76 65 ustang v s. Cheve
6c 6c 65 20 64 72 61 67 20 72 61 63 69 6e 67 20 lle drag racing
28 31 29 2e 6d 70 65 67 00 75 72 6e 3a 73 68 61 (1).mpeg .urn:sha
31 3a 4c 52 49 57 49 45 4c 4a 4a 49 44 33 35 43 1:LRIWIE LJJD35C
4d 5a 41 47 4b 51 4a 32 36 44 36 34 36 47 54 47 MZAGKQJ2 6D646GTG
4c 56 1c c3 82 43 54 44 94 f7 f4 3e 00 LV...CTD ...>.
```

In the case of multiple search results from the same peer, this result set would include multiple entries of the same structure (see Table 9.6).

Table 9.6 QueryHits Results Set Format

Data Size in Bytes	Hex Value	Description
4	0x08020000	The file index number, a number generated from the servent for the purpose of sharing a file in this instance.
4	0x3c1d3c00	The size of the file in bytes.
Variable	0x436172 .. 656700	The name of the file, as stored on the servent’s computer. This is a NULL (0x00) terminated string.
Variable	0x75726e .. f43e00	Optional data about this particular result; usually contains the SHA-1 URN of the file. This field is also NULL terminated.

File Transfers

After the user chooses an appropriate file, they request a download of that file by sending a TCP HTTP GET request to the client that is hosting the file. The requesting client already has the IP address and port number of the hosting client, because it was supplied by the QueryHit descriptor packet. A file download is started by requesting the file index and the name of the file directly from the host computer. Along with the request, the client sends its own IP address and information about itself.

There are two forms that this request can be made in, depending on the data that the client receives from a server. The normal method is to request the file by its index number and file name (e.g., *GET /get/<file index>/<file name>/*). However, if the client receives a URN SHA-1 value in the optional data portion of the result set (see Table 9.6), it will retrieve the file using that value. This is transmitted in the form of *GET /uri-res/N2R?<urn value>* (e.g., *GET /uri-res/N2R?urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C*). A complete HTTP GET request would look similar to:

```
GET /uri-res/N2R?urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C HTTP/1.1
HOST: 184.125.10.16:4680
User-Agent: LimeWire/4.9.37
X-Gnutella-Content-URN:urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C
Range: bytes=0-
```

In the example above, an HTTP request is made to the server defined by the *HOST:* line, 184.125.10.16 at port 4680. The file being requested is the one specified by the URN in the initial *GET* line. The client also specifies its software application using the *User-Agent*, and the exact range of bytes it wishes to receive from the client with the *Range:* line. The use of the *Range:* command allows a client to download a file from multiple servers by requesting only specific portions of the entire file from each server. A range of *0-* specifies that the entire file should be transmitted.

The hosting server then responds back favorably and starts sending the requested data. The response can take one of two forms, depending on if the file is requested from a single server or multiple servers. In the case of a single hosting server, the TCP packet sent will look similar to the following:

```
HTTP/1.1 200 OK
Server: iMesh 0.0.0.1 (GnucDNA 1.1.1.5)
Content-type: application/octet-stream
Accept-Ranges: bytes
Content-Range: bytes 0-19632/19633
Content-Length: 19633
Connection: Keep-Alive
X-Gnutella-Content-URN: urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C
X-Filename: Filename.jpg
<Blank line (0x0d0a)>
[Data contents of the file will be appended to the packet here]
```

However, if you are downloading from a set of multiple hosting computers, the HTTP header will be displayed differently. Each sender will send only a partial amount of the entire download at one time. The specific portion of the data transmitted will be specified within the header, under Content-Range. An example of this is:

```
HTTP/1.1 206 Partial Content
Server: LimeWire/4.9.37
Content-Type: application/binary
Content-Length: 131072
Date: Fri, 28 Oct 2005 17:21:46 GMT
Content-Disposition: attachment; filename="Cars%20-%20Mustang%20vs.%20Chevelle%20drag%20racing(1).mpeg"
Content-Range: bytes 131072-262143/11500292
X-Gnutella-Content-URN: urn:sha1:LRIWIELJJID35C6D646GTGLVBEBCTD12
X-Create-Time: 1498556000
X-Features: fwalt/0.1, chat/0.1, browse/1.0
<Blank line (0x0d0a)>
[Data contents of the file will be appended to the packet here]
```

Regardless of the download method, it ended like any other HTTP TCP connection. The hosting computer sends the data transfer with a FIN packet sent to the user's client. The client then responds with a FIN/ACK to acknowledge the end, and the connection is severed.

Features and Related Security Risks

Due to some of the many features and design implementations of the Gnutella network, it is open to a wide variety of security risks. Many of the issues that arise from using the Gnutella network are due to the nature of the content that is traded within it. The main issues covered here are the type of data that could be downloaded from the FastTrack network, and the type of data that could end up being shared across the network.

Problems Created by P2P in the Enterprise

Imagine spending your entire budget this year on security software. You purchase a firewall to protect the edge of your perimeter as well as Intrusion Detection Software/Intrusion Prevention Software (IDS/IPS), an upgrade to your current version of enterprise AntiVirus software. You go so far as to implement an anti-spam application for your e-mail server. You think you are bullet proof. The next thing

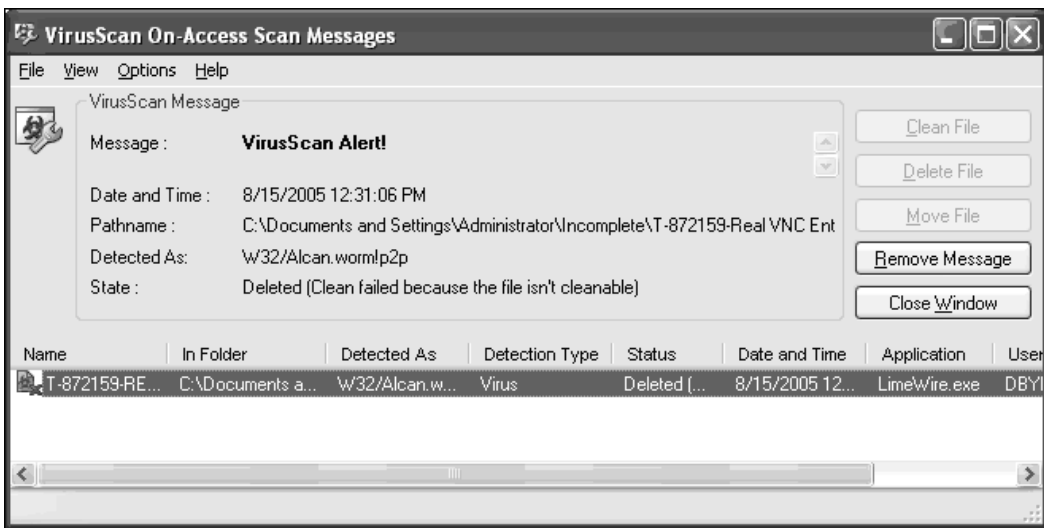
you know, you are being sued for copyright infringement. How did this happen? Why you? What happened? What you may not have considered was spending some time updating your security policy, which would have stated what your users could or could not do on the Internet, because now all of the security you implemented has been squashed by a single download. This section examines how it happened, but first, let's take a brief look at what you could be in store for if you allow for a program such as LimeWire (or other P2P software) to run on your network.

NOTE

If you are in control of security on your network and its systems, you should not be the exception to the rule. In other words, if you are a systems administrator and have the ability to divert from the current security posture, its recommended that you do not use P2P software to download music and other items you may want, you should set the example and enforce the rule. P2P software in the enterprise is "dangerous" and unnecessary unless your company decides to use it for business purposes. If so, it is suggested that you limit the exposure and set up a designated system outside the company firewall.

Infected Files: Trojans and Viruses

You will definitely see more malware if you allow P2P software to run rampant in your organization, or if you run it on your own personal system. Consider how P2P works. You have a Gnutella client you install on your PC that then connects you to this huge network of other PCs all sharing files. The sheer odds of it will almost guarantee a virus. There is no way to determine what could be a virus-infected file or what could not be unless you are running antivirus software on your system currently. There have been many reported cases of fraudulent, trojaned, and virus-infected files downloaded using Gnutella. Many new viruses are spread this way; this is becoming a very popular vector for the spread of new malware. It is imperative that you secure this threat or you will unavoidably have to deal with it eventually. Viruses can infect nearly any type of data format, and can easily be downloaded and infect a workstation without the user being aware of their presence (see Figure 9.4).

Figure 9.4 Getting a Virus from Using LimeWire

Misconfigured File Sharing

If you were using a Gnutella client at home on your personal PC or a PC inside a corporation, you face the same dangers, but to different degrees. If you are using a Gnutella client you are sharing your system with others on the Gnutella network. The way the clients work is to share a directory on your local system to be used to upload and download files. You are connected to this network and many others have access to your system. If you configure your Gnutella client incorrectly, you could possibly have configured more than a default directory to be shared.

NOTE

Sharing directories other than the one you want for the public to see is dangerous. Private information can be lost due to file-sharing applications with little security applied to them. If you have any folders on your system that you do not want to share, make sure that you remove them from your shared folder directory.

Copyright Infringement

Copyright infringement is a serious issue, especially with P2P and the Gnutella network. Just about anything can now be shared: voice recordings, pictures, lists, ideas, music, software applications, and movies, the list never ends. If you can get it in file format, it can be shared. This strikes a chord with those who for so long controlled the distribution of shared data (e.g., music). Copyright infringement, especially in the music business, is a big deal, with major acts (e.g., Metallica) coming down on their fan base for distribution of their music. Either way, it is imperative that you are not caught in the crossfire. By disallowing or completely controlling P2P software, you will keep yourself clear of having to deal with it all of the stuff coming into your network that you cannot have or distribute.

File Transfers Reveal IP Address

While you spend much of your life securing your networks and systems as well as your home, your wallet, your pocketbook, and so on, why would you install a P2P client that basically helps expose your system to others and also your IP address, which can help pinpoint you further down to the Internet Service Provider (ISP). Engaging in a file transfer or a file-sharing session reveals your true IP address. This information may allow someone to concentrate on your system for the purpose of cracking it. It may also make your system a target of a Denial of Service (DOS) attack

Technical Countermeasures for Gnutella

The best way to prevent problems from the Gnutella network is to simply never connect to it. It is recommended that you designate a single system in your home or on your network that can be completely protected and only used for P2P, to help you use the program while still keeping your data secure. Separate the application from any systems you do not want exposed.

If you do choose to secure Gnutella clients, be forewarned that all of them are different and require different configurations. Gnutella can use any port for communication on a network, including those that are generally open on most firewalls such as port 21 (FTP), 25 Simple Mail Transfer Protocol (SMTP), 80 (HTTP), and so on. File transfers utilize port 80; therefore, even if an administrator blocked Gnutella's default port (6346) both ingoing and outgoing, there is still a method to circumvent security controls. Due to the highly customizable capabilities of Gnutella clients, a network IDS may be the only way to detect users of these clients. Port 80 is open on just about every network that has and utilizes an Internet connection. If

you decide to use Gnutella clients, be aware that you will find it difficult to secure them. Using Netstat (a Windows-based utility that shows you network statistics from on the local PC) you can see how you become exposed when using a Gnutella client.

Most Gnutella clients give their users a variety of control over how the client can communicate across the network. The most important part of this is being able to change the default listening port that the client uses (see Figure 9.5). Users also have the ability to transmit their Gnutella traffic through a proxy server. Proxy servers are computers that relay traffic between two computers so that they do not have to directly communicate with each other. The use of a proxy server allows a client workstation to bypass filters and rules that block their connections to specific computers. Many Gnutella clients offer the ability to configure traffic to flow through a Windows Sockets (SOCKS) or HTTP proxy (see Figure 9.6).

Figure 9.5 Listening Port Configuration

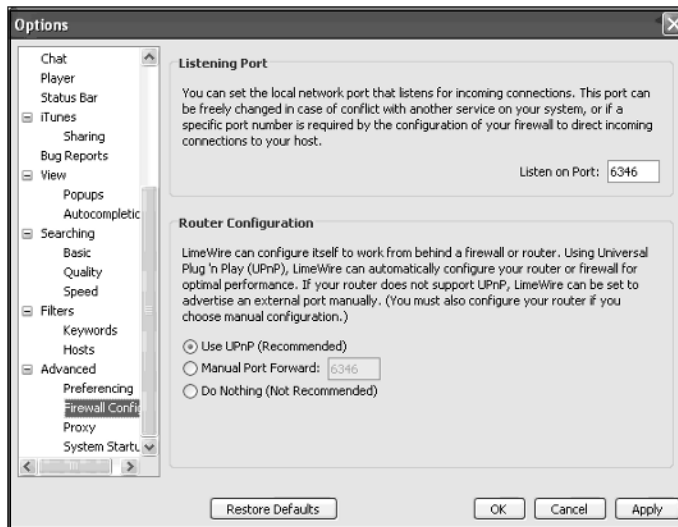
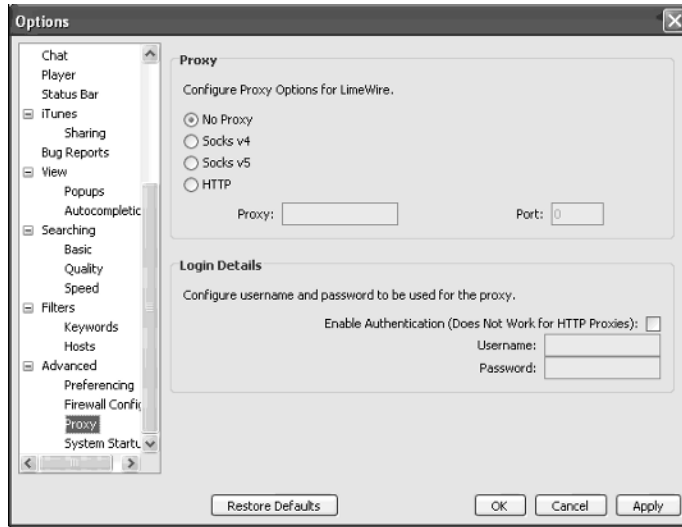


Figure 9.6 Proxy Server Configuration

Firewall Rules

Unfortunately for many system administrators, the Gnutella network was designed to easily bypass control mechanisms placed against it. By default, Gnutella uses UDP and TCP ports 6346 and 6347 for all communications, however, this setting can be altered by any client on the network for their local machine. While it may seem trivial that a client inside your network can change this value when they cannot receive incoming connections, it explains how internal clients can connect to the rest of the network. Many networks will implement a port block of 6346 and 6347, as they are the most common Gnutella ports. However, this will not stop an internal client from connecting to an external server that changed their port number to another value such as 80.

With that said, a port blocking UDP and TCP 6346 and 6347 will block a majority of Gnutella traffic, but not all of it. To effectively block all of the material, a firewall with the ability to inspect data packets may be required. Otherwise, an IDS may be implemented to monitor the usage of Gnutella, which will then allow a systems administrator to follow more administrative procedures for reducing the problem, such as computer use policies and employee reprimands.

To implement a basic port blocking with the Linux-based iptables firewall, dropping any TCP and UDP packet that is destined for, or has come from, those ports. For example:

```

iptables -A OUTPUT -p tcp --dport 6346 -j DROP
iptables -A OUTPUT -p udp --dport 6346 -j DROP
iptables -A OUTPUT -p tcp --dport 6347 -j DROP
iptables -A OUTPUT -p udp --dport 6347 -j DROP
iptables -A INPUT -p tcp --sport 6346 -j DROP
iptables -A INPUT -p udp --sport 6346 -j DROP
iptables -A INPUT -p tcp --sport 6347 -j DROP
iptables -A INPUT -p udp --sport 6347 -j DROP

```

However, as mentioned earlier, this implementation will not completely eliminate Gnutella traffic from entering your network. It may also be possible to search for strings that are contained within each packet. This could be done by using a network-based IDS, but that will only alert its presence, not stop it. Searching for strings is also possible with iptables, but requires the string match module, a module that is not normally installed by default because it is considered experimental.

IPTables String Match Module

Though considered an experimental module, the string match module is a very powerful tool that can help sculpt efficient firewall rules. With this module installed, more exotic firewall rules can be set that do not need to be based off of a port. Instead, each packet can be delved into to determine if it is unwanted traffic, not just traffic flowing on an unwanted port. For example:

```

iptables -A INPUT -p tcp -m string --string "X-Gnutella" -j DROP
iptables -A INPUT -p tcp -m string --string "urn:sha1:" -j DROP

```

Combinations can be made to limit the usage of Gnutella to certain types of data, to avoid more bandwidth-intensive material like movies and audio. This can be done by combining a string that is found in all Gnutella packets, such as “X-Gnutella,” and the “HTTP Content-Type” field. However, as powerful as the module may be, it is not installed by default in many Linux distributions. If it is not, it will have to be downloaded and installed as a kernel patch, which would require recompiling the operating system kernel. To install the string match module, download the latest version of iptables from <http://www.netfilter.org>. With it uncompressed, the string match can be installed by first running Patch-O-Matic by typing the following command lines. Ensure that the KERNEL_DIR variable is pointing to your pre-established kernel source code directory.

```

make pending-patches KERNEL_DIR=/usr/src/linux
make patch-o-matic KERNEL_DIR=/usr/src/linux

```

This will begin an interactive install where you can specify to install the string patch. With the patches in place, iptables must then be installed by typing:

```
make KERNEL_DIR=/usr/src/linux
make install KERNEL_DIR=/usr/src/linux
```

With iptables installed and the required kernel patches in place, the Linux kernel must then be set up and installed, which is done by changing to the kernel source directory (generally `/usr/src/linux`), and running a configuration program. You can then run either “make xconfig” or “make menuconfig.” Once in the kernel configuration, select the *IP: Netfilter Configuration* and locate the entry for *String match support (EXPERIMENTAL)*. Select this to be installed as a module by selecting “M” for this entry. At this point, back your way out and make any other changes you normally would for your specific kernel configurations. Upon completion, save your results and return to the Linux command line. Install the new kernel using your usual method or by typing the following command lines:

```
make dep
make bzImage
make modules && make modules_install
make install
```

When the install has completed, reboot your computer and the new kernel should be loaded into memory. With it installed, you will have the ability to set string-based firewall rules.

Tools & Traps...

Firewall Port Configuration

LimeWire will attempt to bypass your firewall security. Each application on a computer that communicates on the net has a specific port number assigned to it. On most servers, the default port for Gnutella is 6346. This can be found online at www.iana.org in the default ports section: <http://www.iana.org/assignments/port-numbers>.

Continued

The listing is as follows:

```
gnutella-svc 6346/tcp gnutella-svc
gnutella-svc 6346/udp gnutella-svc
gnutella-rtr 6347/tcp gnutella-rtr
gnutella-rtr 6347/udp gnutella-rtr
```

This means that a servent running Gnutella software is listening on port 6346. If that port is already in use, LimeWire scans for a free port incrementally from 6346 to 6356). Only one TCP port can be used for listening at a time. However, the user can change the port that is assigned to LimeWire on his or her computer, and will still be able to communicate with other servents that are listening on port 6346.

Snort IDS Rules

Snort is the most common IDS used today. It is a free open-source application that is available for Linux, Windows, and Mac OS X, and can be found at <http://www.snort.org>. Snort can be implemented into a network design to provide early detection of unauthorized traffic in order to provide administrators the ability to curtail it before it gets out of hand. Snort rules for most common applications can be found across the Internet, with some provided below for integration into existing Snort sensors. The rule below is based on the previous protocol analysis, and can be implemented by customizing the alert message and class type to match the rules of your IDS. It is designed to prevent internal clients from attempting to make TCP connections to servents on the Gnutella network, effectively blocking access before a client has the ability to search for and share files.

```
alert tcp $HOME_NET 1024:65535 -> $EXTERNAL_NET any (content:"GNUTELLA
CONNECT/"; nocase; offset:0; flow:from client; classtype:bad-unknown;
msg:"Gnutella Network Connection Attempt";)
```

Summary

This chapter covered some of the basic features and aspects of the Gnutella P2P network and protocol. The architecture for the Gnutella network is decentralized, and is a true P2P network where all servers connected to the network are considered equals and are capable of providing Gnutella services to other connected servers. With this implementation, search requests and file databases are not handled by a central server. Instead, every client on the network bears a portion of this responsibility. To connect to the Gnutella network, it is necessary to connect to several predetermined IP addresses, which are able to relay information about other server's IP addresses to a newly connected node. You will then be able to share data and upload and download data based on searched queries.

It covered the basic history of the Gnutella network, and how its release caused a stir with its corporate owner, which officially shut down the project. The network lived on, though, as hundreds of developers quickly reverse-engineered the protocol and released clients of their own. The most popular clients in use were LimeWire, BearShare, Gnucleus, and Morpheus.

The chapter also covered the fundamentals of P2P and how it relates to the Gnutella network. As a systems administrator, it is important to make sure that P2P software is controlled so that it does not bypass your firewall or any other security measure you have in place. Always monitor for the use of P2P software. Make sure you analyze your current network and systems to see if P2P have been installed on them, and if so, take action to limit your security exposure.

Finally, this chapter covered some of the security risks that may occur when a network such as Gnutella is used within a corporate environment. These risks range from malware to information leakage. The use of Gnutella clients within a network can be monitored and blocked, although the client makes most forms of blocking ineffective.

Solutions Fast Track

History of Gnutella Clients and Network

- ☑ Gnutella is a P2P file-sharing network that is used for sharing and exchanging of data. You need a Gnutella client that uses the Gnutella protocol to access and utilize the Gnutella network.

- ☑ Gnutella is a true P2P network that operates without a central server. Files (such as music, films and software) are exchanged directly between users.
- ☑ Gnutella clients are available for a number of platforms such as Windows, Apple, and Linux/Unix.

Gnutella Architecture

- ☑ Gnutella clients handle search queries by passing them from one node to another in round-robin fashion.
- ☑ Every search query on the Gnutella reaches a majority of the servers, who must respond back to the requesting client if they have files that match the query.
- ☑ Some nodes can become UltraPeers and have leaf nodes. The UltraPeers act as hubs for helping leafnodes complete search queries.

Gnutella Protocol

- ☑ The Gnutella protocol uses a combination of UDP and TCP packets for communication. UDP packets are used to detect operational servers by “pinging” them, and also for the transmission of search queries and search results.
- ☑ For UDP communications, Gnutella uses a standard, structured packet called a descriptor packet, which handles all of the administrative roles of the network.
- ☑ Connections to other servers on the network are made through TCP sessions. These are performed for initial connections to the Gnutella network, and for the transfer of data between peers. The actual file transfers are made using standard HTTP GET requests and sends.

Features and Related Security Risks

- ☑ Users can improperly configure file sharing, which puts the system at risk by sharing confidential or sensitive files, or possibly an entire hard drive.
- ☑ The use of P2P applications such as Gnutella also welcome possible viruses and trojaned software into a network.

- ☑ Be aware of copyright infringement and how it directly relates to you and your organization.
- ☑ While blocking and monitoring Gnutella traffic can be difficult, there are a few basic firewall and IDS rules that could be implemented to do so.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Where can I learn more about LimeWire?

A: <http://www.limewire.com/english/content/faq.shtml>.

Q: Where can I learn more about the Gnutella protocol and network?

A: <http://www.gnutellaforums.com>.

Q: What are the benefits of going to a “Pro” version of any of the available Gnutella clients?

A: As you can see with LimeWire (or any other Gnutella client) <http://www.limewire.com/english/content/pro.shtml>, going Pro allows you extra features and an “ad-free” experience when using the applications you desire. For example, if you receive a free version of Morpheus, you will be shown ads that generate pay for clicks for the company in which you are using the free software. If you purchase a professional copy, you do not have to see the ads and receive a few extra features. The choice is up to you whether you want more functionality and fewer ads, or a free copy of the software to use.

Q: What is a quick way to connect to and disconnect to the Gnutella network if you are inactive?

A: Use the “connect” and “disconnect” options on most of the clients available to quickly disconnect your Gnutella client from the network. This can offer you security if you only want to use your client to download for your own

needs, and then disconnect the client in times of inactivity. This will not allow any other users to use your system as a node to gather data from.

Q: Can I use LimeWire on other platforms, such as Mac or Linux?

A: Yes, absolutely. LimeWire is available for Windows, Mac or Linux. There are also many other clients that can be used for each platform. Besides for LimeWire, Morpheus, BearShare and Gnucleus, there are Windows clients such as Swapper, XoloX Ultra, and Phex. Other Linux/Unix based clients you can use include Gtk-Gnutella, Mutella, Qtella, and Phex.

Q: What port numbers should I control access to on my firewall, whether inbound or outbound?

A: Gnutella uses 6346 and 6347 (TCP and UDP), although it should be noted that ports can be changed both statically and dynamically, so if you have a Gnutella client that has the ability to search for a firewall-friendly port (such as port 80), as long as a server exists out on the network utilizing port 80, your Gnutella client most likely will eventually find a host to connect to and download data from.

Q: How can I secure LimeWire and other Gnutella completely?

A: Do not use them in corporate networks, outside the firewall, on a test system, or on a system set up only for P2P. In all reality, you really should not be using a freeware file-sharing program inside your company's network, unless it is for business reasons and purposes. If you have to use a file-sharing program, make sure you go through the default configuration settings and set up security to your personal needs and wants.

eDonkey and eMule

Solutions in this chapter:

- History of the eDonkey and eMule Clients and Networks
- Features and Related Security Risks

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

By some accounts, eDonkey (<http://edonkey2000.com>) is the largest and most active peer-to-peer (P2P) network. CacheLogic (<http://cachelogic.com>), which provides services to Internet Service Providers (ISPs), published a study in August 2005 that states that BitTorrent client usage has substantially declined. The study showed that traffic from P2P networks accounted for between 50 and 70 percent of the total traffic seen on ISP networks, with eDonkey being the most popular network accessed. The official client used to access eDonkey—eDonkey2000 (published by MetaMachine)—is a fairly popular client. Newer clients that can attach to the eDonkey network have become more popular than the original client. One client—eMule—(<http://emule.sourceforge.net>) is considered the most popular eDonkey client.

These clients along with the eDonkey network are responsible for the transmission of copyrighted material, including large files such as movies. The eDonkey network is an excellent source for movies and other large files due to its efficient architecture for transferring large files. Although extremely popular in Europe, eDonkey is not a household name in U.S.; however, it is used by more advanced users due to the large amount of files hosted on the network. eDonkey uses a semi-centralized network architecture where users of the service are encouraged to set up servers to aid in connection, locating files, and file transfers. A large amount of the responsibility for controlling this network is in the hands of the users.

For users looking to remain undiscovered while transferring copyrighted files, the eDonkey, Overnet, and Kademia networks are safer than the FastTrack network. This is due to the fact that the Recording Industry Association of America (RIAA) and other copyright holders do not police these networks as diligently as the FastTrack and other networks. This situation could change, so it's generally a good idea not to exchange any copyrighted works if you are concerned with breaking the law. Remember that it is entirely possible for users to obtain your IP address when you are sending or receiving files on a peer-to-peer network. This information can lead to your identity being revealed for the purposes of a lawsuit. Besides the threat of copyright enforcers, users of all P2P networks face the threat of malicious users seeding poisoned files on the network, which will appear to be desirable files, but contain viruses, worms, and spyware.

History of the eDonkey and eMule Clients and Networks

eDonkey was released in 1993 by MetaMachine, which improved on earlier P2P networks with newer technologies (since adopted by competing clients). eDonkey

was designed to be a semi-centralized network with an unlimited amount of servers for clients to connect to. This was accomplished by releasing the source code for servers, which allowed many people to set up servers for searching and transferring files. There are now many implementations of the eDonkey servers, all compatible with the network. This placed control of the network out of the hands of MetaMachine, and made it difficult to control because there was no central server used to authenticate clients. To the detriment of its users, MetaMachine distributed software published by other companies, including spyware, with its eDonkey2000 client. This practice served to scare off potential users of this client, who began looking for alternatives to connect to the increasingly popular network. MetaMachine also began offering a “Pro” version of the software (available at the time of this book’s publishing). The difference between the Pro and free version of the product is that in the free version a window containing advertising was placed within the product, thereby displaying advertising from sponsors while the program was open. The Pro version removed this window and did not interfere with the user experience. Figure 10.1 shows the install process where users may choose between the Pro and free versions.

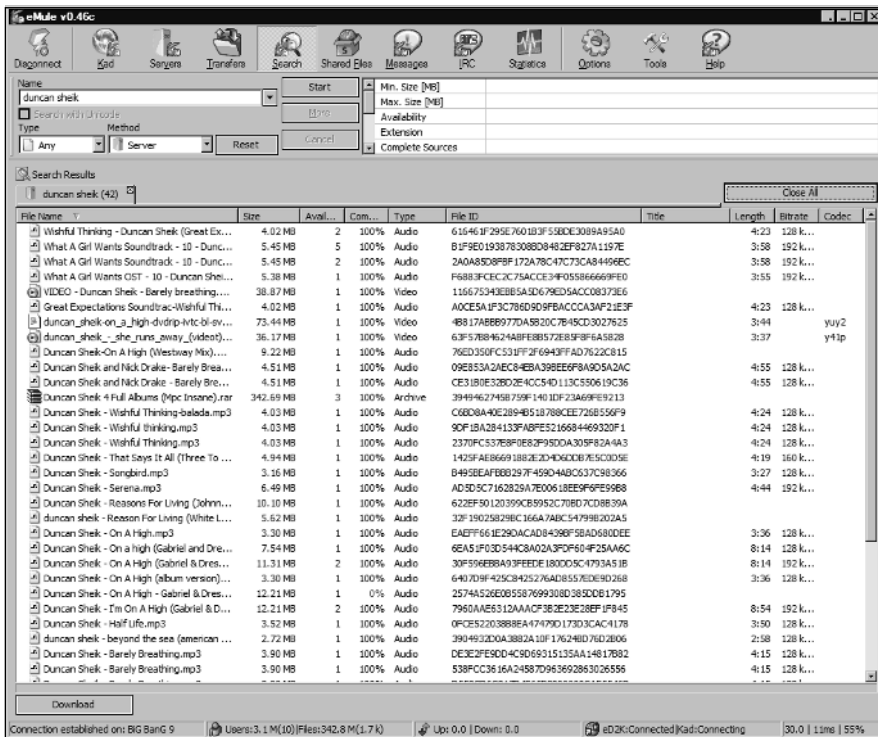
Figure 10.1 eDonkey Installation



The eDonkey network is highly popular in Europe and is becoming very popular in the U.S., especially as large files become common. This popularity, along with

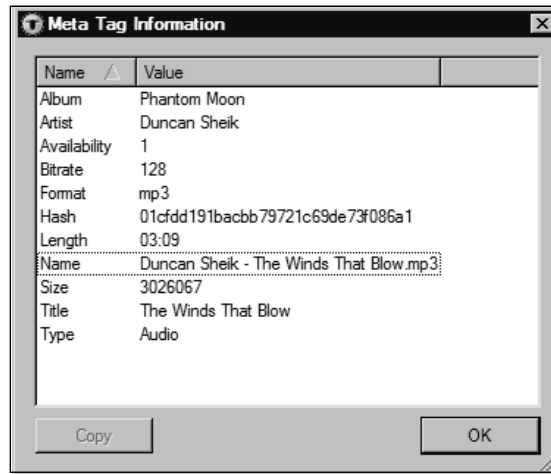
Metamachine's decision to bundle spyware and adware with its client, has spawned several clients compatible with the eDonkey network, the most popular being eMule. eMule is compatible with the eDonkey network, and can also connect to its own network, Kademlia, which is a server-less network (discussed later in this chapter). This network is similar to eDonkey's own implementation of Kademlia (known as the Overnet network), in that servers are not required to track file or client information. There are several other clients capable of connecting to the eDonkey network, but eMule is the most popular. Figure 10.2 shows the main eMule window, which is populated by search results. By many reports, the majority of users on the eDonkey network are using eMule to connect and share files. Much of this can be attributed to the open-source development of the eMule client. Soon after eDonkey was released, MetaMachine began bundling spyware applications with the official client. This became a trend when many P2P clients began distributing adware and spyware as a way to make money. As this practice began to take off, many users began looking for alternative clients that did not bundle adware or spyware. eMule, an open-source program, was not only free, but also well coded and lacking any additional programs. This made eMule a popular client for connecting to the eDonkey network. The beginnings of eMule are summed up on the Web site, which states: "At dawn of May 13th, 2002, a guy called Merkur was dissatisfied with the original eDonkey2000 client and convinced he could do better. So he did. He gathered other developers around him, and the eMule Project was born. Their aim was to put the client back on track where eDonkey had been famous before, adding tons of new features and a nice GUI."

Figure 10.2 eMule Search Results



The eDonkey and eMule Networks

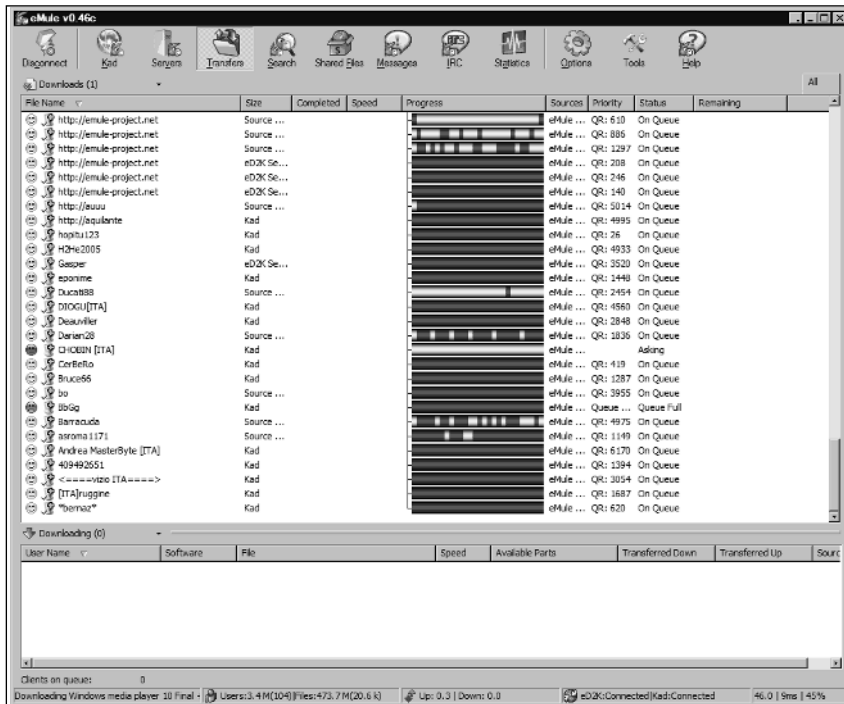
eDonkey shares some similar concepts with other P2P networks. Its architecture uses servers to connect users to the network. These servers also contain a list of other servers on the network that are responsible for tracking files available for download, and locating those files using searches. eDonkey uses Message Digest 5 (MD5) hashes to identify files, and the servers keep track of these files for all clients within close proximity. Figure 10.3 shows the Meta Tag information of a shared file, with an example of the information eDonkey uses to identify files on the network.

Figure 10.3 Meta Tag Information

When a user searches for a file, the client contacts the server and queries it to determine which clients have portions of that file available for download. If a user wishes to locate more sources for that file, he or she can extend the search, which causes the client to contact the server, which will query other servers with the same search request. The eDonkey network uses the Multisource File Transfer Protocol (MFTP). For more information on MFTP, visit www.edonkey2000.com/documentation/mftp.html. This protocol is well documented and has allowed others to create clients that are compatible with the eDonkey network.

Another feature that contributed to the popularity of eDonkey was known as “swarming,” which made transferring large files more efficient. When a user downloads a file, it is usually transferred in small pieces. eDonkey and eMule not wait for users to download a complete file in order to begin sharing it. The client is aware of portions of files downloading and automatically makes those portions of the files available for download to other clients. This makes it hard for users of the network to “leech” files from other users, since they are forced to share files. Additionally, this makes the network highly efficient, and provides users with a large amount of sources for file downloads, which is especially useful for large files such as movies. Figure 10.4 shows the download screen of eMule. Note that there are multiple sources that are being used for downloading a specific file. Although many sources for this file are not entirely solid, these clients will still allow you to download the portions of the file that are available.

Figure 10.4 eMule Download Sources



Although eDonkey has proven to be effective and popular, MetaMachine developed another client and network architecture that does not rely on centralized servers to assist in client connections and searching. This architecture is similar to the “gnutella” network in terms of how files are found, distributed, and shared. The Overnet network, based on the Kademlia protocol, has been built into the eDonkey 2000 client, with the hope of phasing out the server-based eDonkey network in favor of Overnet. (At the time of this book’s publishing, the eDonkey client is still a hybrid and contains the ability to connect to both eDonkey and Overnet). The ability to connect to both networks is transparent to the user, who needs to configure the client in a single location. The defaults are set during the installation process, and can be changed at any time. eDonkey2000 and eMule connect to clients via TCP and UDP. eDonkey 2000 uses TCP/4662 and UDP/12980 by default, while eMule uses TCP/4662 and UDP/4662. These values can be changed to any available port, making it difficult to block these clients from transferring files without using a firewall capable of packet inspection. Figure 10.5 shows the network configuration options available to eDonkey users, while Figure 10.6 shows these options for eMule users.

Figure 10.5 eDonkey2000 Network Options

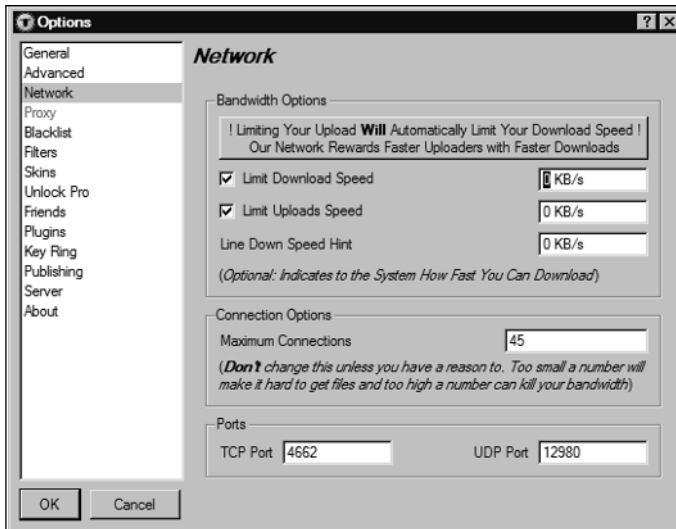
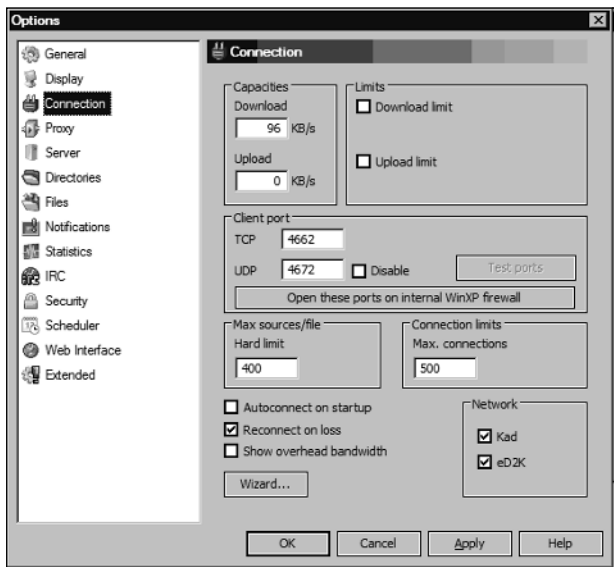


Figure 10.6 eMule Network Options



eDonkey has an additional input for file transfers, with the enhancement of accepting input from BitTorrent files as well. A feature of this integration is the ability of the eDonkey client to treat the BitTorrent data as another source of the downloaded file, which can be combined with other sources from the eDonkey or

Overnet networks. BitTorrent is generally able to download files faster, which speeds up the entire download process.

Features and Related Security Risks

Using eDonkey and eMule clients can create security implications on a network just by being installed. Since their primary function is to send and receive files (particularly the exchange of files that would violate copyright laws), much of the security risks revolve around the type of files that are shared and what folders are made available for sharing on a particular workstation. Many of the issues listed below deal with misconfigured sharing settings, the possibility that users may mistakenly download malicious code, and other similar issues.

Copyright Infringement

The eDonkey, Overnet, and Kademia networks all exist for sharing files. In particular, these networks are popular with users who want to share large files including complete albums encoded on MP3s, movies, and applications. Its notoriety as a network for the exchange of copyrighted works has not gone unnoticed by copyright holders and law enforcement agencies. The RIAA, the Motion Picture Association of America (MPAA), and other agencies also know about these networks and the files that are shared. These agencies and associations routinely police these networks, attempting to uncover the users responsible for sharing files. Remember, anyone downloading a file while using these clients also shares these files. Other tactics used to prevent the exchange of copyrighted materials include sharing incorrect, garbled, or useless files with the same names and characteristics as the files that users are looking to download. The users who share these files are often the target of investigations, as some of these networks do not have a server and the servers that do exist are scattered throughout the network and do not actually host the files. Additionally, eDonkey2000, eMule, and other clients that connect to these networks may display the IP addresses of the users are sharing the files. This makes it very easy to track down users who violate copyright laws. Lawsuits based on sharing files are not uncommon, and if the files are being exchanged in a corporate environment, the company itself may be liable for fines and damages associated with the illegal activity.

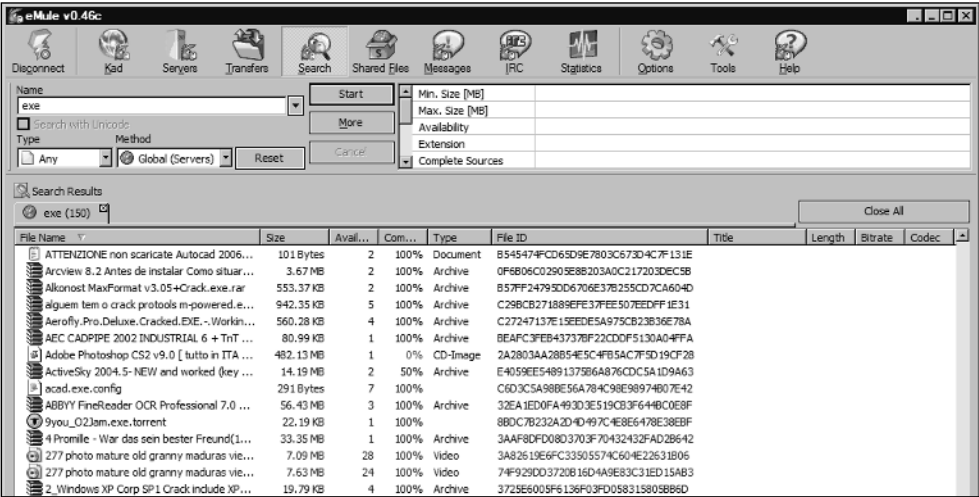
Malicious Software

Malicious software (malware) is downloaded accidentally by unsuspecting users of P2P networks; eDonkey and eMule clients are no exception. Malicious users often seed P2P networks with spyware, worms, viruses, and other forms of malicious code

that can be downloaded to infect a system. According to Secunia (<http://secunia.com>), 137 pieces of malware, including viruses, worms, and bots, have affected the eDonkey network. This is a large security risk, because users often execute the downloaded files to ensure that the desired files have successfully downloaded. Differences in file sizes are regularly ignored due to the differences in encoding bit rates for audio files and the different codes that can be used for video. Users do not suspect that the files with different sizes are malicious code, just different versions of the same file they are looking for. It is not uncommon for the viruses or spyware to install a backdoor to allow remote access, disable antivirus protection, or change the configuration of the P2P client software to share out the entire hard drive onto the network. This is especially dangerous if the users regularly download compressed files (such as .zip, .rar, or .ace) or applications.

Malicious code has the ability to spread throughout the network, because the files are automatically shared with others. As previously mentioned, as users download small portions of a particular file, other users wishing to download the same file can begin downloading even though the file is incomplete. Additionally, users can leave the client open for downloading, which can take several days for large files. Both of these options would allow malicious code to spread rapidly before the files were checked by the users or by antivirus software. Figure 10.7 shows the results of an eMule search for .exe files, which are executable files. These files execute code on a workstation, but may be malicious in nature or self-extracting compressed files that contain legitimate and malicious files in a single package. All files that are downloaded from a peer-to-peer network, especially executables, should be scanned by an anti-virus program before being used.

Figure 10.7 eMule Executable Search



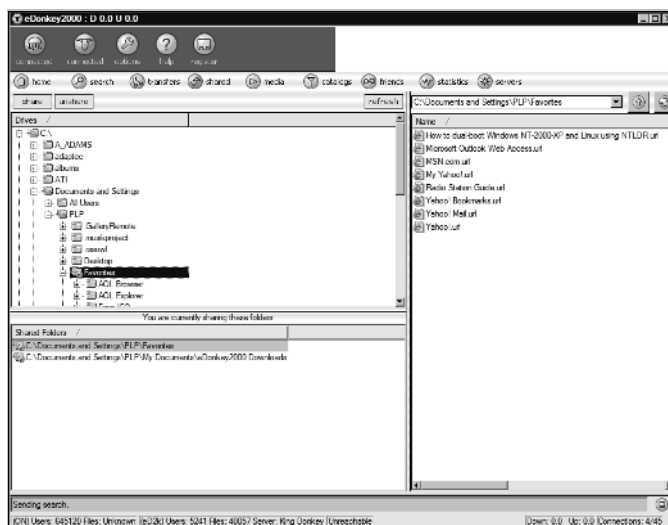
Poisoned Files

A recent development regarding P2P networking is the emergence of “fake files” that contain malicious code, also known as *poisoned files*. As stated previously, users of P2P clients are not surprised to see files with different file sizes sharing the same name. These files may be portions of a song that may be silent, or mixed and garbled. These files may also contain a malicious payload, as malicious users attempt to spread code on unsuspecting users. This is often accomplished by creating a self-extracting executable which, when executed, unpacks the desired file while installing a piece of malicious code.

Misconfigured Sharing

The main function of P2P clients is the ability of these programs to exchange files. These clients also give users the ability to share files and directories on their workstation with other users connected to that network. If improperly shared, these files would be searchable within the eDonkey network. For home users, the type of data that would be shared includes passwords, bank records, and other personal and sensitive data. Corporate users and system administrators have more to lose, because corporate documents and sensitive files such as source code, e-mail, spreadsheets, and diagrams could potentially be shared. A large amount of this data may be confidential, and a careless or uneducated user could create a large amount of financial damage. Figure 10.8 shows the configuration options available to eDonkey2000 users for configuring folders to share.

Figure 10.8 eDonkey2000 Sharing Options



Vulnerabilities

Seemingly, all popular software packages have several software vulnerabilities, and eDonkey2000 and eMule are no exception.

eDonkey2000 appears to be the less vulnerable piece of software, with only one security vulnerability, published in March 2003, and listed by Secunia as the following:

Vulnerability Description

- eDonkey2000 and Overnet have been found vulnerable to a resource exhaustion issue.
- By sending malicious chat requests as different users, it is possible to spawn new dialogs.
- By sending a large number of forged chat requests, a large number of dialogs are spawned and will eventually consume all memory and/or central processing unit (CPU) resources, rendering the system useless.

Vulnerability Solution

Version 0.46 limits the number of message dialogs that are spawned.

Vulnerability Provided and/or Discovered by PivX Bug Researcher

Six vulnerabilities have been found in eMule according to Secunia (see Table 10).

Table 10.1 eMule Software Vulnerabilities

Title	Date
Denial of Service (DOS) and <i>zlib</i> Vulnerabilities	2005-07-27
Web Interface Negative Content Length DOS	2004-05-11
“DecodeBase16()” Buffer Overflow Vulnerability	2004-04-05
Long Password DOS Vulnerability	2003-10-22
Multiple Vulnerabilities	2003-08-19
DOS	2003-03-26

The latest vulnerability, disclosed in July 2005, is classified as highly critical and listed by Secunia as the following:

Vulnerability Description

Two vulnerabilities have been reported in eMule that can be exploited by malicious people to cause a DOS or potentially compromise a vulnerable system. An error in eMule can be exploited to crash the client via a specially crafted *Kad* packet.

Successful exploitation requires enabled *Kad* support. eMule uses a vulnerable version of the *zlib* library.

For More Information

- SA15949
- SA16137

The vulnerabilities have been reported in versions prior to 0.46c.

Vulnerability Solution

Update to version 0.46c.

Vulnerability Provided and/or Discovered By

Reported by vendor.

Other References

SA15949: <http://secunia.com/advisories/15949/>

SA16137: <http://secunia.com/advisories/16137/>

Summary

This chapter examined two clients capable of connecting to the eDonkey network—eDonkey2000 and eMule. eDonkey2000 was released in 1993 by MetaMachine, as a semi-centralized network that gave users the ability to set up their own servers. These servers are responsible for locating clients on the network and facilitating searches. Servers may be beneficial to the network by speeding up searches, but they also have drawbacks regarding legal issues, and often create a bottleneck for the network. The eDonkey network became popular quickly, especially for the transfer of large files such as movies and applications, although transfer speeds are often slower than competing services. MetaMachine addressed these concerns with a newer, server-less network called Overnet. Another feature that made transferring files more efficient was the ability to use BitTorrent as another source for files. In general, BitTorrent transfers are faster than many other P2P networks. Combining these transfers with those on the eDonkey network create a very fast environment for downloading files. The eDonkey network is highly efficient because it is not necessary to download an entire file before sharing it with the rest of the network. eDonkey has the ability to share incomplete files, making it possible for all users to download files, and to automatically become sources for other users to download from. This creates a large amount of sources for downloading. eMule is a compatible client for accessing content on the eDonkey network, and has quickly become the preferred client for accessing the network, because it was not bundled with adware and spyware like previous versions. eMule is open-source software, and its source code is available for download.

Both clients have the ability to create security issues once they are installed. Copyright infringement is a major issue for users of P2P software, as users often exchange copyrighted material. File sharing in general is dangerous, due to the amount of malicious files that are available for download. Unsuspecting users may download and share viruses, worms, or spyware by downloading files that appear to be benign but are disguised malware. Often, malicious users bundle their code with legitimate files in an effort to trick users.

Solutions Fast Track

History of the eDonkey and eMule Clients and Networks

- ☑ eDonkey2000 was released in 1993 by MetaMachine, as a semi-centralized network with an unlimited number of servers set up by users of the system.
- ☑ eDonkey2000 is available in both a free and a paid “Pro” version.
- ☑ MetaMachine developed a new server-less network called Overnet, which did not rely on servers to find files on the network.
- ☑ eMule is a P2P client capable of connecting to the eDonkey network. eMule was released in May 2002.
- ☑ “Swarming” is a feature that allows eDonkey to share portions of downloaded files with other users. Instead of waiting for a user to download the entire file, eDonkey will exchange portions of the file.

Features and Related Security Risks

- ☑ Copyright infringement is an issue faced by most users of P2P networks. Users of these networks often exchange files that have copyright restrictions (e.g., movies and music). This illegal activity is policed by several law enforcement and industry agencies. Users who exchange these files may reveal their IP addresses, making it easy to track down users who engage in illegal activity.
- ☑ Malware is readily available on P2P networks. These files are seeded on P2P networks since they have the ability to spread quickly. eDonkey is especially efficient, because all of the users downloading files are also required to upload and share those files automatically.
- ☑ Poisoned files may present a security risk for all P2P network users. Unsuspecting users may accidentally download files with the same name they are looking for, but that are bundled with malicious code. Tactics such as self-extracting executables often deliver the intended file along with spyware or viruses.

- ☑ Misconfigured file sharing is an easy mistake to make that can have dangerous consequences. Users may change settings within their client for which folders are shared on a specific workstation. Users may choose a wrong directory or share their entire drive with users of the P2P network. Users may be unknowingly be sharing their sensitive and confidential data with millions of users.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: What are the main differences between eDonkey2000 and eMule?

A: eDonkey and eMule are very similar in that they both connect to the eDonkey network. The official and original client, published by MetaMachine, is known as eDonkey2000. The differences in these clients are that eDonkey2000 is based on proprietary code and comes in free and “Pro” versions. eMule is an open-source project, with source code available to everyone. Other differences include the secondary server-less networks that both of these clients connect to. eDonkey2000 connects to the Overnet network, while eMule connects to the Kademlia network. Although Overnet is based on Kademlia, these clients cannot connect to each other’s networks.

Q: How can you protect yourself from worms and viruses when using a P2P network?

A: The easiest and most effective way to protect your network from malicious code is *not* to download unknown files or files from unknown sources. This usually involves restricting the usage of P2P networks, whose main purpose is to download files from multiple unknown sources. If you are unwilling to restrict usage of P2P clients, it is a good idea to scan all files with an anti-virus program. Scanning these files before executing or opening them will usually ensure that these files are safe. A plug-in is available for eDonkey2000, which enables users to automatically scan all downloaded files with an installed anti-virus client.

Q: Why is eDonkey2000 less susceptible to vulnerabilities?

A: eDonkey2000 is probably not any more secure than eMule, even though less vulnerabilities have been made public in the client. A possible reason for the disclosure of more vulnerabilities in eMule is that the client is more popular. Researchers often tend to look for vulnerabilities in more popular programs, concentrating on programs that have a wide distribution.

BitTorrent

Solutions in this chapter:

- History of the Network
- Network Architecture and Data Flow
- Protocol Analysis
- Features and Related Security Risks
- Bandwidth Issues and Mitigation Steps

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

History of the Network

The BitTorrent network was an innovative new approach in the peer-to-peer community. BitTorrent was designed and developed by Bram Cohen, who envisioned a network where the client hosting a popular file doesn't need to be punished by high bandwidth. It is also a network that allows the content to be managed through the service by use of upload and download file ratios, a technique mainly found in archaic bulletin board systems (BBSs) and FTP sites. No one expected the BitTorrent network to take off the way it did, though. At one point it accounted for over one third of all Internet traffic and has currently received millions of dollars in funding.

The concept behind the BitTorrent network is a slightly new approach in the peer-to-peer community. A Web host sets up a tracker, which lives up to its name of tracking the various users who are transferring files. A user decides to share out a file onto the Internet and proceeds to create a torrent file, which includes the URL of the tracker that will be used to track it. This torrent file is uploaded to a Web site for other users to download, and its creator begins sharing out the file for others to download.

When another client downloads the torrent file and opens it with a BitTorrent client, the client will connect to the tracker and request a list of peers that are sharing the requested file. It will then directly request data from the peers it has been given. As soon as any single segment of the file, known as a *piece*, is downloaded, it immediately becomes available for sharing to other peers. In this process of constantly giving and taking, each peer receives more and more of the data, until they eventually receive it all.

Apart from being a medium for the exchange of data between users, BitTorrent has also been used as a distribution method for many large software packages. Many software publishers are finding that spreading content through BitTorrent networks can reduce strain on Web servers and allow users to receive their software more quickly. The first real-world demonstration of this technique was performed in March 2003 when Red Hat released its Red Hat Linux 9.0 via BitTorrent, posting a link to the well-visited Web site Slashdot.org. Within four hours, 1.5 Terabytes of data had been transferred to downloaders. Three days after its release, 21.15 Terabytes of data had been transferred, with very little originating from Red Hat's Web servers. Currently, virtually all major distributions of Linux can be downloaded through BitTorrent, which allowing users to be able to download their favorite software immediately after it is released while avoiding busy file servers and mirrors.

Blizzard Entertainment, makers of the award-winning *World of Warcraft* game, has used BitTorrent technology to distribute its entire software package and all its

patches to hundreds of thousands of users. Sun Microsystems has also utilized BitTorrent to help spread its Open Solaris operating system. BitTorrent has been used to distribute independent films, such as *Star Wars: Revelations*, for which the torrent can be downloaded from www.panicstruckpro.com. However, this capability has also been clouded by equal attempts to disrupt the sharing of files, such as when Microsoft forced the removal of torrents sharing its Windows XP Service Pack 2 (SP2), relegating all users to download the software solely from Microsoft's Windows Update site.

BitTorrent

The BitTorrent network was designed by an eccentric developer named Bram Cohen. As is the case with most peer-to-peer networks, the network's designers also release an official client that is used to connect to the network. In the case of BitTorrent, the client shares the same name as the network. The client is commonly referred to as the *mainline client* and can be located at www.bittorrent.com.

The BitTorrent network was first designed in early 2001. After some initial development it was officially unveiled at the 2002 CodeCon, a programming conference that is also organized by Bram Cohen. It took a number of months before the concept took hold with the Internet public, but eventually torrent sites began appearing in late 2002. Starting in late 2003 and continuing to the present, various distributed denial of service (DDoS) attacks began targeting major torrent sites. It is still unknown where many of the attacks originated, though there is speculation that some may be from rival sites, piracy groups, and even the Motion Picture Association of America (MPAA). Such attacks led to a greatly diminished number of torrent servers on the Internet. The servers that survived the attacks soon began implementing various new features and forms of security onto their sites. Many sites became open only to registered users, with some of these adopting site membership caps to keep from growing too large. Others firmly implemented ratio bans, where users would be barred from using the tracker if their ratio of upload bytes to download bytes was too small, which removed many leechers from tracker Web sites.

Bram Cohen went on to design implementations of the BitTorrent network into specialized software. In 2003, he was temporarily hired by Valve Corporation, a large videogame producer. His work there involved creating a distribution method for Valve's online Steam application, which allows gamers the ability to download and update entire purchased games.

In recent times, many changes were made to the official client that have also been carried out to other BitTorrent clients. One such change was to flag BitTorrent TCP packets as bulk data, which allows BitTorrent traffic to mesh cleanly with more

time-sensitive data across the Internet. One large change made in May 2005 involved adding trackerless support to the official client. This change allowed a client to be able to seed and spread files without having to first create torrent files and upload them to a tracker.

However, a real change in the public's perception of BitTorrent came in September 2005, when the company secured \$8.75 million to help fund the project. The company announced that much of the funding would be used to improve the network's infrastructure and design models, allowing for more legitimate implementations. As a gentle nod to this funding, in October 2005 the BitTorrent monetary donation window, which has long been a facet of the software, was removed.

BitTornado

BitTornado is a BitTorrent client that can be located at www.bittornado.com. The interface is very similar to the standard BitTorrent client but also features additional abilities such as bandwidth throttling, prioritized download of individual files within a batch, and the ability to view detailed information about connection peers. Similar to the BitTorrent client, BitTornado allows only a single download per instance of the application.

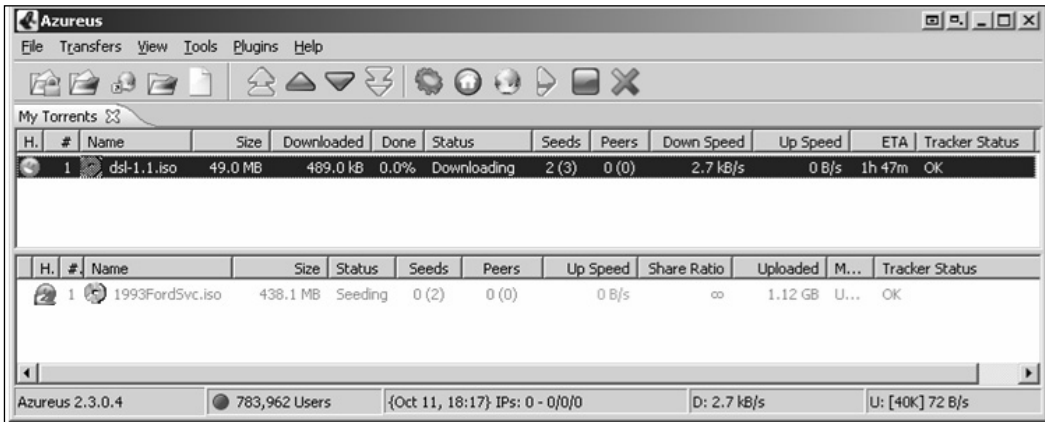
Azureus

Azureus is a Java-based graphical client for the BitTorrent network that can be found at <http://azureus.sourceforge.net>. Though initially designed as a plain graphical interface for the network, it has gained much popularity and is currently one of the most commonly used clients, with versions available for Windows, Linux, Mac OS X, and Solaris UNIX. It was voted the SourceForge Project of the Month for September 2004 and has continued development with the goal of being the best BitTorrent client. Azureus initially implemented many features not found in other clients, such as a trackerless distribution method. However, many clients now offer trackerless connections, using distributed hash table (DHT) technology. Azureus has received quite a few negative remarks because its DHT implementation is incompatible with every other BitTorrent client. An example of the Azureus client is shown in Figure 11.1. Azureus currently offers the following features and abilities:

- Multiple, simultaneous uploads and downloads
- Bandwidth throttling for all or individual torrents
- Proxy support for anonymity
- Automatically imports torrent files from a specific directory

- Embedded tracker, to remove the need for specialized tracker software and a Web host
- Support for magnet links
- Automatic port forwarding (UPnP)
- NAT traversal

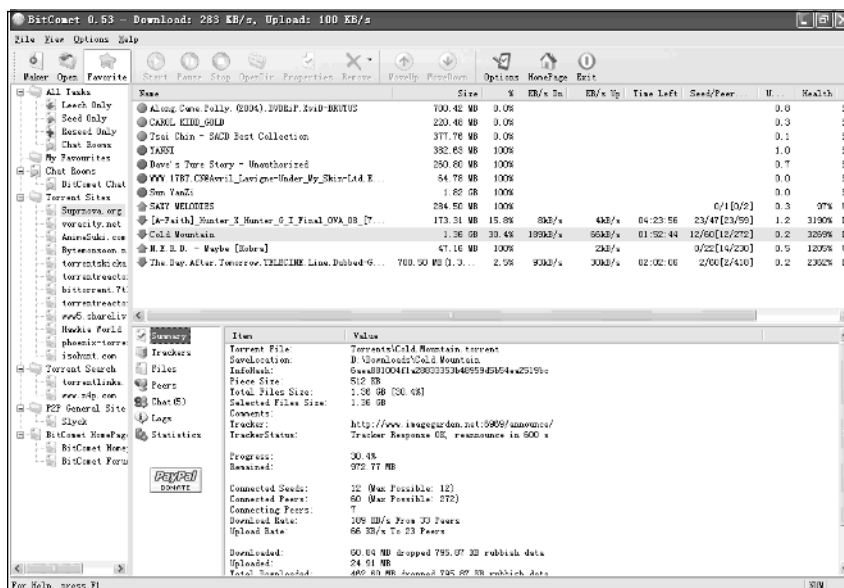
Figure 11.1 The Azureus Client



BitComet

BitComet is a free Windows client for the BitTorrent network that can be found at www.bitcomet.com. BitComet is one of the more feature-filled clients that exist, allowing for the ability to have multiple uploads and downloads, bandwidth throttling, video previewing during downloading, trackerless torrents using DHT, automatic port forwarding (UPnP), and NAT traversal. An example of the BitComet client is shown in Figure 11.2, as a screenshot released by BitComet's creators.

Figure 11.2 The BitComet Client



Other Clients

Dozens of BitTorrent clients are available to be used with the BitTorrent protocol. Many of these clients were based on the original BitTorrent client, for which Bram Cohen released the Python source code. Even though all clients share much the same functionality, many clients were written with specific goals in mind. A few of the less common clients are featured in this section with descriptions of the features that make them notable.

ABC

ABC, short for Yet Another BitTorrent Client, is a client that can be found at <http://pingpong-abc.sourceforge.net>. ABC is a modified version of the original client, with its greatest modification its ability to download multiple files within the same instance. It is also capable of queuing downloads and supports many additions that were featured in BitTornado.

μTorrent

μTorrent is a basic BitTorrent client that features some modifications, such as being able to download multiple files within the same program. It also supports bandwidth throttling and UPnP support. However, its selling feature is that it is easier on

resources than other clients, designed in Java and Python. It is a miniscule application that uses less than 4MB of RAM, which is small in comparison to some clients that can use up to 100MB of memory during operations. μ Torrent can be found at www.utorrent.com.

G3 Torrent

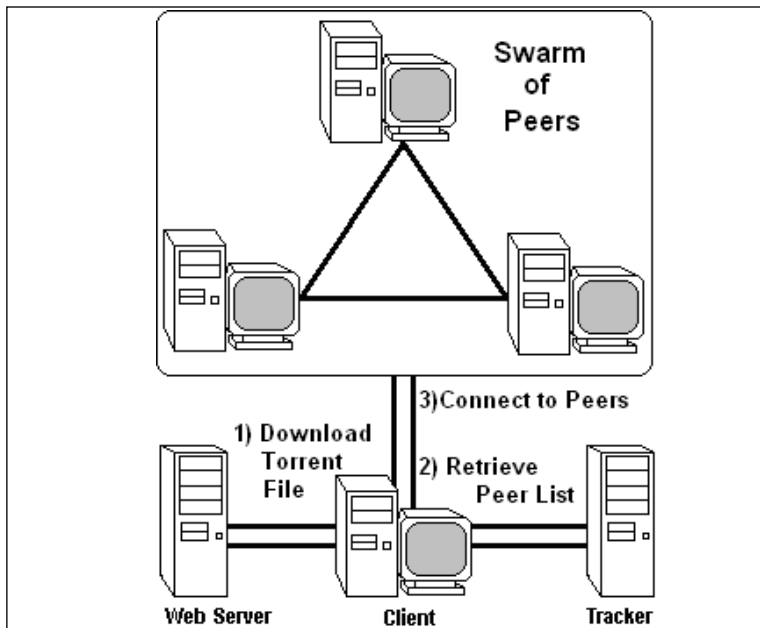
G3 Torrent is a popular client that is designed to both be visually appealing and to provide the most amount of data and information about connections that have been made. It does so by providing many areas of information where data is shown about connected peers and connections, such as the IP address, host address, and country of origin of each peer. It also provides graphing that plots transfer speeds over a period of time. One other notable feature is that G3 Torrent can spoof its *User-Agent* tag to resemble other clients. Some tracker sites may require only specific clients to be used when accessing it, and G3 Torrent can pretend to be those clients. G3 Torrent can be found at <http://g3torrent.sourceforge.net>.

Shareaza

Shareaza is technically a multinet network client, with access to the Gnutella, Gnutella2, eDonkey, and BitTorrent networks. It has also been translated into 18 languages. It can be found at <http://shareaza.sourceforge.net>.

Network Architecture and Data Flow

BitTorrent focuses around a centralized peer-to-peer network design. Similar to Napster's network design, BitTorrent allows a central server to connect peers to share files. However, instead of just connecting two peers for a single file transfer, it connects dozens of clients to transfer small pieces of a file at a time. This central server is known as a *tracker* because it keeps track of the various clients that are connected to it. A basic diagram of the BitTorrent network is shown in Figure 11.3.

Figure 11.3 The BitTorrent Network

Torrent Files

Before a connection to a tracker is made, a torrent file must be activated by a BitTorrent client. This torrent file acts as a key to allow a client to download a specific file or set of files. The information within the torrent file is hashed, using SHA-1, short for the Secure Hash Algorithm 1, to provide a value that can uniquely identify the data it represents. The torrent file also includes the full URL to the tracker that will be used for the connection. A full breakdown of this torrent file can be found within the “Protocol Analysis” section of this chapter. This torrent file is then posted around the Internet or placed within specific torrent search sites and databases. This allows users to be able to search for particular files or applications and find its corresponding torrent file. This file is also used to determine the tracker used for the transfer and make the connection to it.

Trackers

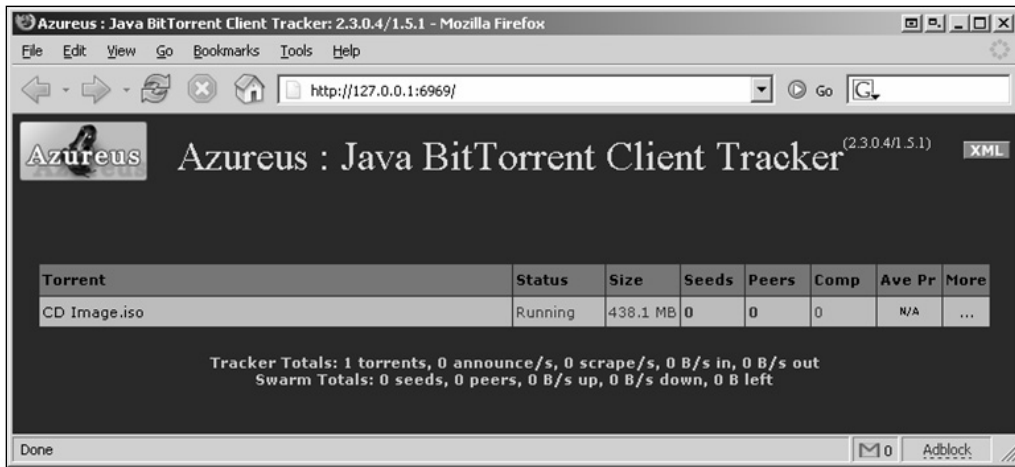
Trackers are the keystones to the entire BitTorrent network. The trackers act very similarly to the centralized servers that Napster was using on its peer-to-peer network. They accept connections from various clients and link them for data sharing.

Once a torrent file has been downloaded and opened by a client, the client software makes a connection to the tracker site specified within the torrent and requests the data specified by its hash value. The tracker constantly updates an internal database with all the clients that are sharing a particular file. The tracker will acknowledge the connecting client and transmit to it a peer list, which is a list of the IP addresses and ports of other clients that are sharing out the file. At this point, the client will make a connection to a peer IP address and request the data directly from it.

As the client downloads data from its peers, it will also send regular requests to the tracker for more peers. These requests also include statistical information about the client's connection, such as how much data it has uploaded and downloaded. Because the tracker is aware of all uploads and downloads, it can maintain an upload-to-download ratio for each client. For example, if a client has downloaded 2 Megabytes of data and has only shared 1 Megabyte with peers, it will have a 0.5:1 sharing ratio. If it has shared 100 Megabytes and downloaded 50 Megabytes, it will have a 2:1 sharing ratio. These ratios can be used to determine whether a client should be allowed to continue receiving peer lists from the tracker, and if so, which peers should be sent to it.

Users who are unable or unwilling to host a tracker on a Web site can run a tracker from within a BitTorrent client. Azureus is one such client that offers this capability. The client will then act as an HTTP hosting application, running a basic Web server from within it. By default, the client will listen on port 6969 for HTTP connections. Other users can then browse available torrents by connecting to the client's IP address at port 6969 with a regular Web browser. A typical internal tracker display is shown in Figure 11.4. The use of an internal client tracker within a network might not work in a default network configuration. The network's firewall has to be configured to forward data from the HTTP port and the data transfer ports to the specific client's IP address. It is very useful, however, within a large internal network.

Figure 11.4 Azureus Internal Tracker



Of Leechers and Seeders

Clients on the BitTorrent network fall into two general groups: seeders and leechers. A *seeder* is a client that is sharing out a file onto the network. A *leecher*, or normal peer, is a client that is downloading the file from others. The entire virtual network of seeders and leechers for a single file is called a *swarm*. For a transfer to begin, at least one client must be actively “seeding” the file. The original seeder is a person who created the initial torrent file and specified the tracker that would be used. However, once more seeders begin sharing out the file, the original may stop sharing it.

A leecher—a client that is currently downloading the file’s data—does not yet possess the complete file or set of files, so it depends on the tracker to connect it to seeders and other peers to download pieces of the file. Once a leecher has obtained 100 percent of the file’s data, it immediately becomes a seeder. At that point, it can continue seeding the file for other clients or it could stop the seed.

When a client is sharing a file, it makes a few vital decisions on how it wants to share the data out. A peer can choose which peers it wants to send data to, and it places higher priority on peers that have a higher upload rate. This is not just because the peer needs information from the faster peer, but instead it provides a reward system for sharing data, known as “tit for tat.” Clients that transmit data at a fast rate will be given higher priority by neighboring peers, with higher transfer rates. A seed will also give priority to new clients that have joined the swarm, to quickly give them enough data so that they can begin sharing with other peers.

During the process in which a leecher is downloading pieces of the file, it is also sharing out pieces to other peers. This way, the bandwidth required to transfer a file is spread among many peers, not just a single set of seeders. Under normal means, there is no way for a client to disable the sharing of data while they are actively downloading it. The sharing of a file may be canceled after the transaction is complete, but in some instances the amount of data shared at that point could be equal to or greater than the amount of data downloaded. It is very bad netiquette to not seed a file immediately after downloading it, though.

There are a few accepted standards of conduct with seeding, some of which say that a file should be seeded out for at least 72 hours after its download is complete. Other social circles believe that a client should at least share three times the size of the file before they stop seeding. There are very few social controls in place, though. The only effective means is the upload-to-download ratio, but that normally records a client's overall ratio on that tracker. So a client could rack up a large ratio on some files and choose not to seed others, yet still be allowed use of the tracker. Although generally the terms *leecher* and *peer* are interchangeable, in some circles a true leecher is one that maintains a very low ratio and tries to avoid sharing data as much as possible.

Basically, the difference between a seeder and a leecher is that a seeder already has the complete file and a leecher does not. They are both sharing data to other peers, but a seeder does not need to request pieces from peers.

Trackerless

Establishing and maintaining a tracker for a transfer can be very difficult for normal computer users. However, the trackers that do exist might not allow the transfer of the type of data that a user wants to share. Or a user might only want to share a file between others in his social circle, such as fellow members of a Web forum or IRC channel. In these cases, it is possible to share a file without using a tracker or even a torrent file. BitTorrent accomplishes this by implementing a protocol based on Kademlia's DHT. The trackerless data transfers work outside the normal BitTorrent guidelines, and as such they don't offer any statistical information to trackers for monitoring, such as upload-to-download ratios. Be aware that BitTorrent's official means of trackerless communications differs from the one built into Azureus. If an Azureus client is using trackerless mode, also known as *decentralized* mode, the data can only be accessed by other Azureus clients.

Regardless of the client used, a torrent file is made that is designed for the trackerless connection. This file is then shared with friends and family with whom you want to share the data. Instead of using a regular tracker URL, they use a DHT URL, such as

dht://C7084BEB94E52DCBEB73ABEB3FE4FBEB430BEBAB.dht/announce. It may also be possible to create a *magnet link*. This is a single line that can be entered into a client, such as Azureus, to download a specific torrent file directly from a DHT network, bypassing the Web host. This magnet link can then be sent via e-mail or posted onto Web sites and chat rooms for immediate connections. An example of a magnet link is `magnet:?xt=urn:btih:YLAUR5OBGPA7KRXTXF24Q7RC2ENGINAH`.

Protocol Analysis

BitTorrent uses normal data connections over a few well-known ports for a majority of its traffic. Most of the network architecture seems very straightforward, but quite a few extra steps and decisions are made by the client to take a peer list and use only the most optimal peers from within it.

Bencoding

Most metadata that is sent with BitTorrent is encoded in a format called Bencode. This data includes the actual torrent files and the peer lists that are sent from the tracker and between clients. Understanding Bencode formatting is crucial to reading the data contained within these packets.

Bencoding allows multiple data variables to be stored within a single line of text. It offers support of character strings, integers, lists, and arrays. Each one is stored in a particular fashion within the text.

String values begin with a base-10 number that defines the length of the string, followed by a colon, and finally the actual string value. For example, `10:BitTorrent` would be read as a string value of *BitTorrent*. The text `10:BitTorrent4:file` is read as the string values of *BitTorrent* and *file*.

Integer values begin with the lowercase letter *i*, followed by the base-10 integer value and ending with the lowercase letter *e*. For example, `i672e` is read as the number 672. The text `i672ei123e` would be read as the numbers 672 and 123.

A list is a set of further bencoded values that are grouped together. A list begins with a lowercase letter *l*, followed by the bencoded values contained within it and ending with the lowercase letter *e*.

Dictionaries are arrays of data that store a variable name and the variable's value. A dictionary begins with the lowercase letter *d*, followed by the bencoded values that alternate between a variable name and its corresponding value. It ends with the lowercase letter *e*. For example, `d4:Type9:classical4:Sizei3123467e` reads as *Type: classical, Size: 3123467*.

Lists and dictionaries can also be combined within each other, which could make deciphering the text difficult. For example, *6:MyListId4:Type9:classical4:Size3123567eed4:Type3:pop4:Size1902125eee* is read as a list of dictionaries. It is broken down as:

- **MyList**
 - Type: classical, size: 3123567
 - Type: pop, size: 1902125

Torrent Files

The torrent file is a vital component of the BitTorrent network; it is the key that allows clients to be able to access files from the network. In most cases, a client must be in possession of a torrent file to begin downloading or uploading data to other peers. The torrent file is a normal ASCII text file that is hosted by various Web sites and can also be e-mailed between users. The text contained within a torrent file is all encoded using Bencoding and includes information about the tracker and the file itself. Information included within this file is required for a client to be able to connect to a tracker and begin downloading data. In the BitTorrent network, all data is seen as just a stream of data, so the torrent file is required to determine the metadata of the shared files, such as their filenames and file sizes. Without a torrent file, it normally wouldn't be possible to determine the filename of the shared files or their size. The torrent file is made up of a single bencoded dictionary string.

A torrent can accompany two styles of files: single files and a set of files. Examples of each are displayed in this section. An example of a torrent file for a single file transfer is:

```
d8:announce30:http://tracker.prq.to/announce13:creation
datei1128708386e4:infod6:lengthi550062e4:name15:filesharing.png
12:piece lengthi262144e6:pieces60:R{grÆžÄÄ^¥Ä+ 1a¹fU'„£Ä□İ$*M
'İ*□V□` ôµ/]1[*"D{ □ç2Ä³éšee
```

Because the torrent file is a large bencoded dictionary, every value has a corresponding key. Each key and its value are shown in the following examples. Using the example shown previously, with the text being decoded, the information stored within it is as follows:

- **announce:** “**http://tracker.prq.to/announce**” The URL of the tracker that will be used for this file transfer.
- **creation date:** “**1128708386**” The creation time of the torrent, in number of seconds since 1-1-1970.

- **info** A dictionary contained with the torrent that includes information about the files being shared. It includes the following values:
 - **length:** “550062” The size of the file in bytes.
 - **name:** “filesharing.png” The name of the file.
 - **piece length:** “262144” The size of each file piece.
 - **pieces:** “*reduced for brevity*” A concatenation of each piece’s 20-byte SHA-1 value. The number of pieces in the stream can be determined by simply dividing the length of this value by 20.

When you use a torrent file that contains multiple files, the torrent file structure is slightly different.

An example of a torrent file for a multiple file transfer follows. It has been reduced for brevity and also includes some nonstandard values:

```
d8:announce32:http://69.25.27.21:6969/announce18:azureus_propertiesd17:dht_backup_enablei0ee7:comment0:13:comment.utf-80:10:created by15:Azureus/2.3.0.413:creation datei1129492122e8:encoding5:UTF-84:infod5:files1d6:lengthi281600e4:path19:setup.exe10:path.utf-819:setup.exe4:name6:cygwin10:name.utf-86:cygwin12:piece lengthi32768e6:pieces500:Û$5|087:privatei0eee
```

Using this example, with the text being decoded, the information stored within it is as follows:

- **announce:** “http://69.25.27.21:6969/announce” The URL of the tracker that will be used for this file transfer.
- **azureus_properties** A dictionary of additional values used by the Azureus client.
- **dht_backup_enable “0”** An integer value that determines whether Azureus should use DHT as a backup in case the tracker becomes inoperable.
- **comment:** “null” An optional value that the torrent’s author can include to describe the file.
- **comment.utf-8:** “null” A deprecated value that normally includes a UTF-8 encoded comment. However, currently all strings in the torrent file are UTF-8 encoded.
- **created by:** “Azureus/2.3.0.4” An optional value that specifies the software used to create the torrent file.

- **creation date: “1129492122”** The creation time of the torrent, in number of seconds since 1-1-1970.
- **encoding: “UTF-8”** A value that specifies the encoding method for strings within the torrent.
- **info** A dictionary contained with the torrent that includes information about the files being shared. It includes the following values:
 - **files** A dictionary containing information about each file in the set.
 - **length** The size of the file in bytes.
 - **path** The path and name of the file. The path is omitted if the file is located directly under the parent directory.
 - **path.utf-8** A deprecated value that includes the path and name of the file.
- **name: “cygwin”** The name of the directory under which the files are stored.
- **name.utf-8: “cygwin”** A deprecated value that is equivalent to the name field.
- **piece length: “32768”** The size of each file piece.
- **pieces “reduced for brevity”** A concatenation of each piece’s 20-byte SHA-1 value. The number of pieces in the stream can be determined by simply dividing the length of this value by 20.
- **private “0”** An integer value that defines whether the tracker is a private tracker. If enabled, the client will only accept peers directly from the client, not from DHT or from other peers.

Tracker Connections

After retrieving the particular tracker to use, the client will make a connection to the tracker to receive a peer list. This connection is made by sending a TCP packet to the tracker’s IP address and specified port, using a simple HTTP *GET* request for the specific hash value of the file. An example of this request follows, with a backslash representing a line continuation. This request will detail the SHA-1 hash value of the file that the client is attempting to download, the host name and connection port of the tracker, and the specific client software used by the user:

```

GET \
/announce?info_hash=%C&j%C4%BB%D9q%D3%90%295%DB%CF%C2%D3%EA \
%25%B4%28%A5G&peer_id=%2DBC0060%2D%90%2B8%BC%3F%97WJ1%90%5F \
%7D&port=23645&uploaded=0&downloaded=0&left=2133210 \
&numwant=200&compact=1&no_peer_id=1&key=23765&event=started \
HTTP/1.0
User-Agent: BitTorrent/3.4.2
Connection: close
Accept-Encoding: gzip, deflate
Host: www.legaltorrents.com:7070
Cache-Control: no-cache

```

The initial *GET* line of this request sends vital statistical information about the client to the particular tracker. The tracker uses this information to determine the user's upload-to-download ratio and which peers they should receive. The following variables, or keys, are sent in this request to the tracker:

- **info_hash** The 20-byte SHA-1 hash value of the file, as retrieved by performing a hashing algorithm against the “info” section of the torrent file. The data are encoded using URL Escape codes, which allow the transmission of ASCII characters in hexadecimal. For example, & is escaped to %26, and = is escaped to %3D. The hex value is the character's hexadecimal equivalent of its ASCII code.
- **peer_id** A 20-byte string that the client randomly generates to use as a unique identifier. This string is also encoded using URL Escape codes. Some clients will include a static portion into the value. For example, Azureus begins the *peer_id* with -AZ2304-, where 2304 refers to the application version 2.3.0.4.
- **port** The port number that the client is listening on for BitTorrent data. By default, this normally ranges from 6881 to 6889.
- **uploaded** The total number of bytes uploaded by the client to other peers, represented in base-10 ASCII, since the event started.
- **downloaded** The total number of bytes downloaded by the client from other peers, represented in base-10 ASCII, since the event started.
- **left** The number of bytes that the client is lacking in the file download, represented in base-10 ASCII.
- **ip** An optional key specifying the IP address of the client. Not normally needed, as the tracker can determine this from the packet address.

- **event** An optional key that announces the client's connection phase for this particular torrent. If it is not specified, the packet is treated as a routine request for peers. It can be one of the following values:
 - **started** A value that is required for the initial peer list to be sent and is sent in the very first tracker connection.
 - **stopped** A value that should be sent when a client stops downloading information, such as when the application is closed. This notifies the tracker to remove the client from the peer list.
 - **completed** A value that should be sent when the client has downloaded a file in its entirety, thus becoming a seeder. It is not sent when a client is launched with the file already completed.
- **numwant** An optional key that specifies how many peers the client wants to have in their requested peer list.
- **no_peer_id** A deprecated, optional key that asks the tracker not to send *peer_ids* in its response. It has since been replaced by the compact key, but it might still be in use by some clients.
- **compact** An optional key that requests the tracker to use compacted replies. When set, *peer_ids* for each peer will not be sent back, and IP addresses will be sent in binary instead of ASCII. For example, 81.12.101.14:6881 will be sent back as five total bytes: one for each octet and one for the port. 51 0C 65 0E 1A E1.
- **key** An optional key that provides a unique identifier to the tracker that is not shared with other peers.
- **trackerid** An optional key that returns a unique identifier specified by the tracker in previous connections.

The tracker will then respond to this *GET* request with the peer list in a return. This data will be transmitted as a single bencoded dictionary string. This string contains at least two portions: the interval value and the peer list that the client should use to download data from the network. An example of this response follows, with the bencoded text one continuous line:

```

HTTP/1.0 200 OK
Content-Length: 494
Content-Type: text/plain
Pragma: no-cache

d8:intervali1800e5:peersld2:ip11:84.12.11.147:peer id20:-
AZ2304-zJaMauZXHzYC:porti50008eed2:ip12:84.44.14.2177:peer
id20:-AZ2304-RiW9msFbd0h4:porti6881eed2:ip11:72.25.3.2257:peer
id20:M4q0a1zzdc8928c6 b2b4:porti6881eed2:ip13:21.246.12.1527:
peer id20:exbc:jBjl qYòD□0à
4:porti26034eed2:ip12:84.19.48.1017:peer id20:-BC0059-
VoB1{Ì"Hkà4:porti6882eed2:ip12:69.25.24.2217:peer id20:-BC0060-
+S¼?-WJ1□_}4:porti23645eed2:ip11:21.32.86.907:peer id20:-
lt0700-0UçÒGE#Ä1-2|4:porti9108eeee

```

With the previous line decoded, the information stored in it is as follows:

- **interval:** “1800” The duration of seconds before the client should send another peer list request; 1800 seconds corresponds to 30 minutes.
- **peers** A value that signifies that the peer list is enclosed in the following list:
 - **ip:** 84.12.11.14, **peer id:** -AZ2304-zJaMauZXHzYC, **port:** 50008
 - **ip:** 84.44.14.217, **peer id:** -AZ2304-RiW9msFbd0h4, **port:** 6881
 - **ip:** 72.25.3.225, **peer id:** M4q0a1zzdc8928c69b2b4, **port:** 6881
 - **ip:** 21.246.12.152, **peer id:** exbc:jBjl qYòD□0à, **port:** 26034
 - **ip:** 84.19.48.101, **peer id:** :-BC0059-VoB1{Ì"Hkà4, **port:** 6882
 - **ip:** 69.25.24.221, **peer id:** :-BC0060-+S¼?-WJ1□_}4, **port:** 23645
 - **ip:** 21.32.86.90, **peer id:** lt0700-0UçÒGE#Ä1-2|4, **port:** 9108

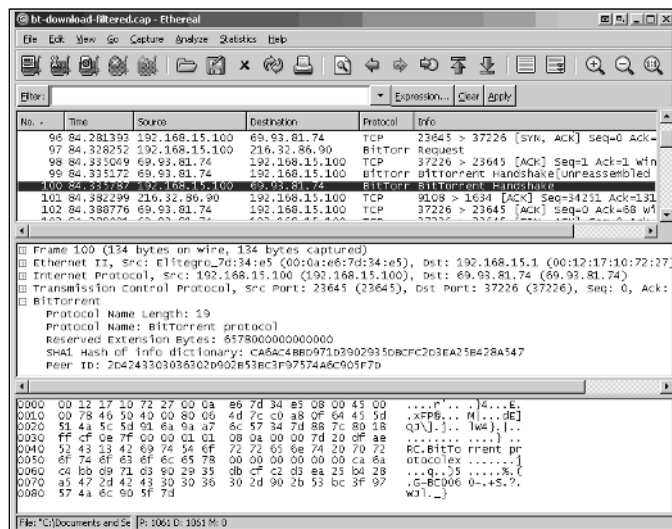
Peer Connections

Once a BitTorrent client has received a peer list from a particular tracker, it then must make connections to the provided peers to transmit and receive data. These connections are made in BitTorrent using Peer Wire Protocol (PWP) messages. By default, TCP port 6881 is used for peer-to-peer connections. However, this port does vary, since each client can specify their own port, even a randomly created one. The specific port for each client is included in the peer list, though, so the client only has to blindly connect to the data fed to it. Peer connections are also symmetrical, which means that all packets are identical, regardless of whether they’re incoming or outgoing packets.

Before any data is transmitted between peers, they must negotiate a handshake between themselves. This is accomplished by the client sending a TCP packet to a peer's open port, as shown in Figure 11.5. This packet contains a specific 68-byte data payload that has an ASCII equivalent of:

```
13 42 69 74 54 6f 72      72 65 6e 74 20 70 72      .BitTor rent pr
6f 74 6f 63 6f 6c 65      78 00 00 00 00 00 00      otocol x.....
ca 6a c4 bb d9 71 d3      90 29 35 db cf c2 d3      .j...q. .)5....
ea 25 b4 28 a5 47 2d      42 43 30 30 36 30 2d      .%. (.G- BC0060-
90 2b 53 bc 3f 97 57      4a 6c 90 5f 7d          .+S.?.W J1._}
```

Figure 11.5 Ethereal Screen Showing BitTorrent Handshake



This packet is the official handshake request. The hexadecimal data within this packet is broken down as shown in Table 11.1.

Table 11.1 Peer Handshake Request

Data Size in Bytes	Hex Value	Description
1	0x19	Specifies the length of the following field
19	0x426974546F7272656E742070726F746F636F6C	The protocol name; it must correspond to <i>BitTorrent protocol</i> or else the connected will be dropped

Continued

Table 11.1 continued Peer Handshake Request

Data Size in Bytes	Hex Value	Description
8	0x6578000000000000	Reserved for future use
20	0xCA6AC4BBD971D3902935 DBCFC2D3EA25B428A547	The 20-byte SHA-1 hash value of the file's torrent
20	0x2d4243303036302d90 2B53BC3F97574A6C905F7D	The 20-byte Peer ID of the client

After the connection is made with surrounding peers, communication can then take place directly between the peers. This communication is not just pure sharing of data. It can be affected by various states placed on peers and clients.

Peer States

For all BitTorrent connections, there are two states that each client maintains about its peers: choked and interested. For each client, a peer's state is either enabled or disabled. These are basically just labels applied to peers; either he is choked or he isn't, and either he is interesting, or he isn't.

When a client is *choked* by a peer, it is notified that it is no longer allowed to request data from the peer. When a client has choked a peer, it severs the transfer of data between itself and that peer. The ability to choke goes along with the implementation of “tit for tat,” where a client will reward more generous peers with faster connections. If User A notices that User B has a very slow upload rate or is not transmitting enough upload packets, User A may choke User B. It will continue choking unsuitable clients so that the unchoked clients that remain can take advantage of its bandwidth and data. Because a client may continually monitor peers around it, it could decide in the future to unchoke a previously choked peer. By default, as soon as a client receives a peer list, it automatically views all the peers as choked and must decide which peers to unchoke.

When a client sets a peer to an *interested* state, it notifies the peer that it would like to receive data from the peer. If the peer does not have the client choked, it will regard the interested state and allow the client to download data from it. If a client decides that it no longer needs any data from a peer, it sets the peer to *Not Interested*. By default, all clients are set to *Not Interested* at the start of the connection. The client then must communicate with the peers to determine which peers have data pieces that it is interested in and set the *Interested* bit on those peers.

Each peer maintains the state in which it sees other peers as well as the state in which each peer sees it. For example, if User A is transmitting data between itself and Users B and C, it can change how it views each individual peer. If User A wants

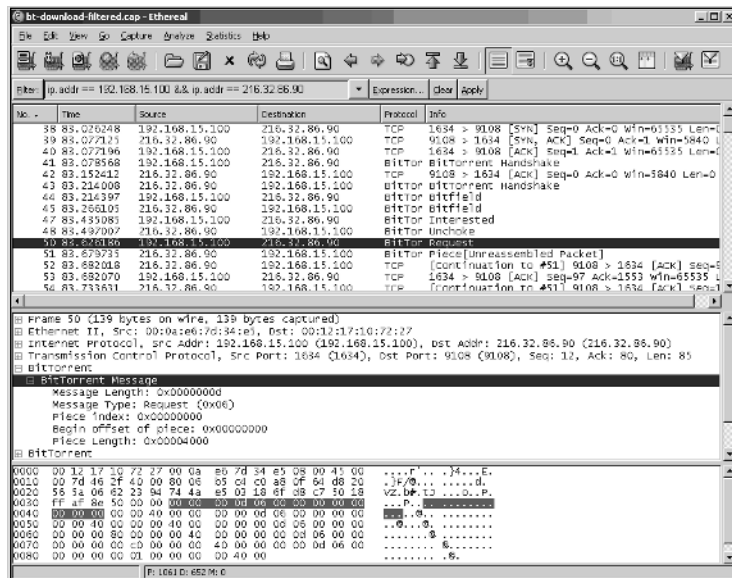
to stop receiving data from User B, it will choke User B. This is done by sending a PWP message to User B notifying them that they are choked by User A. User A's client will then see that it is choking User B but is not being choked by User B. If the peers are not interested in each other's data, User A's client would see User B as Not Interested as well as know that User B sees it as Not Interested.

Peer Wire Protocol Messages

Peer communications take place via PWP messages. These messages include the state that the client views the peer as or data that is being requested or sent to other peers. Each of these message packets has structured data payload content, but also one that varies in size based on the information being transmitted. These packets are TCP packets that are sent to a peer's listening port, as shown in Figure 11.6. An example of this packet, an *Interested* packet, contains a data payload with an ASCII equivalent of:

```
00 00 00 01 02 .....
```

Figure 11.6 Ethereal Screen Showing BitTorrent Handshake



This packet contains structured PWP information. Each packet has a set header in the payload, with variable amounts of payload trailer data, depending on the type of message sent. The header begins with a 4-byte hexadecimal value that specifies the length of the message. This is immediately followed by a 1-byte hex value that specifies the type of message that is being transmitted. The various message types are detailed in Table 11.2.

Table 11.2 Peer Wire Protocol Message Types

Message Type	Description	Data Payload
0 Choke	Notifies peer that it is choked	None
1 Unchoke	Notifies peer that it is unchoked	None
2 Interested	Notifies peer that it is being set to Interested	None
3 Not Interested	Notifies peer that it is being set to Not Interested	None
4 Have	Notifies peers that the client has just downloaded a specific data piece	Index number of the file piece
5 Bitfield	A bitfield that is sent to peers to notify them of data pieces that the client currently has	A bitfield that is the size of the message length minus 1; "0x00" entries refer to missing pieces
6 Request	A request for data, with a fixed length of 13 bytes	4-byte value of the requested piece index + 4-byte value of byte offset that data begins at + 4-byte value of block size
7 Piece	A data piece; this is the actual file data being transmitted between peers	4-byte value of the piece index + 4-byte value of byte offset that the data begins at + the file data that is the size of the message length minus 9
8 Cancel	Notifies a peer to stop sending a particular block of data; used to nullify a previous Request	Identical structure to a Request packet
9 Port	Specifies the client's listen port, used for DHT tracking	2-byte port in hexadecimal format

Many message types are self-explanatory. However, two of the more important messages, *Request* and *Piece*, deserve further analysis and are covered in the following sections. It is important to note that a PWP packet may include multiple messages, even of different message types. It is common to find packets that contain multiple *Requests* as well as packets that contain both a *Have* and a *Request* for the same peer.

Peer Requests

The request for data is one of the main components of the BitTorrent protocol because it allows a client to specifically request the data that it needs to complete a file. In looking at the previously defined data payload for a request packet, we see that it contains the requested piece index, the byte offset where the data begins, and the block size, all in 4-byte hexadecimal values. An example of this request block is shown in Figure 11.6. This request is sent to a peer to request a specific portion of a piece of data. Each data request asks for a block of data, which is but a portion of the overall file piece. Generally, the size of a file piece is no larger than 512KB, whereas a block is sized around 16KB. For each PWP packet sent, there may be multiple requests attached within the payload. Each request is made for a different block of data from a single peer. In response to this request, the peer will send each block of data to the client using the PWP *Piece* message.

Peer Data Transmission

Obviously, the most important component of the BitTorrent protocol is the data that is sent between peers. This data is sent in response to a specific data request by another peer. A peer will request a particular block of data that is contained within a piece of the file. This block of data will then be sent using a PWP *Piece* packet. As explained previously, the *Piece* packet contains a payload of the piece index, the byte offset where the data begins, and finally the actual requested data. Since a block size is normally larger than a packet size, the file data will be spread over several continuous TCP packets. The piece index and byte offset should coincide with the information contained within the corresponding *Request* message.

DHT Connections

Trackerless connections through DHT have been a fairly new implementation into the BitTorrent client. The official BitTorrent mainline client supports a form of DHT called Khashmir, which is based on Kademlia. Kademlia is referred to as an overlay network protocol, in which its usage can coincide with other protocols used within a client, such as BitTorrent. It enables each peer on the network to act as a simplified tracker, notifying a client as to what peers are sharing a particular file.

With DHT-enabled software, each client creates a random Node ID that is used to uniquely identify it within the network, which is generally a 160-bit SHA-1 hash of the client's IP address. A client then makes its connection to the network using a preset IP address of a peer within the network. All data on a Kademlia network is stored with keys and values, similarly to the dictionary implementation in Bencoding. A key is used to specify what a set of data is, where the value actually contains the data. In this implementation, a key that is normally used is a SHA-1 hash value of the particular shared content. Each key is assigned to particular peers on the network. Each peer also maintains a set of links to other nodes around it. When a client requests data from the Kademlia network, it submits the 160-bit SHA-1 hash value of the file it is attempting to download. This can be done using a set DHT value, which can be shared over e-mail and Web pages, such as `dht://C7084BEB94E52DCBEB73ABEB3FE4FBEB430BEBAB.dht/announce`. The obvious set of hexadecimal data within this field is hash value of the content, which should correspond to the *info_hash* field of a normal torrent file. It makes this request to the peers that are closest to it on the Kademlia network. If they do not possess the information for the hash, they will then pass it along to other peers, until the peer that possesses that data can be found. This data could be composed of a copy of the torrent file needed to start the connection or a set of peers that are hosting the data.

Features and Related Security Risks

Due to the design of the BitTorrent network, various security risks accompany its use. Some of the dangers are standard with most peer-to-peer networks, such as the authenticity and validation of the shared files. Although this is not as large an issue as it is on other peer-to-peer networks, such as FastTrack, it is still present in BitTorrent. For example, many files on other peer-to-peer networks are found by regular keyword searches and can be shared out indefinitely from peers that refuse or are unable to delete offending material. With BitTorrent, because the torrent files have to be hosted on a known Web site, corrupted or offensive material can be removed from the network once it has been located, simply by removing the torrent file. However, many other issues still come into play with the BitTorrent network.

Copyright Infringement

As with all peer-to-peer networks, a large legal issue for BitTorrent is the type of data that can be traded over it, such as copyrighted materials. Since the protocol is designed for the transfer of large files, as in files over 100MB in size, it is not common to find

the presence of small files being traded. Therefore, the BitTorrent network hasn't received the same attention as networks like FastTrack from organizations like the Recording Industry Association of America (RIAA). However, its intended data usage has made it a target of other organizations, such as the Motion Picture Association of America (MPAA). The trading of very large files, such as those over 1GB in size, has made the BitTorrent network ideal for sharing motion pictures. Although much of the network is used for legal purposes, many users do use the network to download and share copyrighted materials. The presence of such material on workstations and computers could lead to potential liabilities and embarrassment for corporate organizations as well as fines levied against the actual downloader.

However, there are still quite a few large torrent sites that flourish with access to copyrighted material, even with ever-increasing legal pressure. One such site is www.thepiratebay.org, a torrent site run by Pirat Byran, a Swedish P2P advocacy group. Torrent sites themselves do not possess any copyrighted materials, just a torrent used to download materials, but they can still be held liable by the Digital Millennium Copyright Act (DMCA). This is not the case with many foreign countries, such as Sweden, where the Pirate Bay resides. Details of legal threats made against the Pirate Bay as well as off-color responses can be found at <http://thepiratebay.org/legal.php>.

Instead of targeting torrent sites and trackers, many copyright holders have begun targeting actual users. Although these actions aren't as large-scale as the RIAA's and don't include heavy fines, they have been useful in dissuading users from downloading certain content. For example, recently Home Box Office Inc. (HBO) joined in the transmission of some of its television shows to gather IP addresses of peers that were sharing the file. These addresses were traced back to the user's Internet service provider, who was issued a DMCA complaint against the user. In October 2005, the first BitTorrent user was found guilty of using BitTorrent for copyright infringement of popular Hollywood movies. The individual, from Hong Kong, uploaded the torrents for the movies to popular Web sites, which allowed countless other peers to download the content.

Poison Peers

Although many peer-to-peer networks deal with the issue of poisoned files, which are corrupted files shared purposely to peers, this is not normally found within the BitTorrent network. Instead, the concept of poison peers has been introduced, as evidenced by recent activity by HBO. In response to growing download activity over recent episodes of a hit television show, HBO began implementing fake clients to flood actual peers with corrupted data. These clients would connect to a known

tracker and notify it that the client is seeding the complete file. The tracker would pass along the client's IP address to other peers who would soon begin requesting and downloading data from the poisoned peer. Once the data were transferred, the receiving peer's client detected that the data were corrupted and dropped it. However, with a large amount of poisoned peers, quite a few users were unable to download valid data from valid peers. Eventually the affected peers receives a new peer list that would have authentic peers on it and retrieve their requested data. The overall effect, though, is the severe delay of downloading material and the large amount of bandwidth that is pushed to clients, who promptly discard it as corrupted.

Automatic Sharing of Data

The automatic sharing of data is one of the most important features of the BitTorrent network, but it also has some serious implications for peers on the network. BitTorrent has the unique feature of sharing data while it is in transit to your computer, which is wildly different from other peer-to-peer networks, which begin sharing data out only after the data have been completely downloaded. What this entails is a peer not being able to visually inspect the contents of a file to verify their authenticity before sharing out the information to countless other peers. All that is necessary is for one online prankster to post a torrent that describes itself as useful data but instead contains offensive materials. Dozens, if not more, clients could then be in near complete possession of a file that does not resemble its description.

This problem becomes even worse when copyrighted material is in use. Although many legal courts make a distinction between users who are downloading material and users who are actively uploading it, that distinction becomes blurred on the BitTorrent network. Any user who is currently downloading material is also uploading it, which could hold that user accountable for the distribution of copyrighted material.

Bandwidth Issues and Mitigation Steps

Due to the design of the BitTorrent network, it is possible for a single client to amass a large amount of bandwidth. This can occur quite easily if a client downloads a popular file and proceeds to seed it indefinitely. Sometimes this occurs because a user simply doesn't realize that his or her BitTorrent client is uploading data to dozens of clients continuously over days or weeks. This bandwidth consumption could easily require a company to make higher-than-expected bandwidth payments.

Bandwidth Scheduling

In instances where the use of a BitTorrent client is acceptable for a network environment, the issue of bandwidth could cause problems during busy hours. It is possible, though, to schedule the use of BitTorrent traffic to off-peak hours, normally from midnight to 8:00 A.M., when user activity is very low. This can be done with certain BitTorrent clients, such as Azureus, through the use of a Speed Scheduler plug-in. It allows specific bandwidth speeds to be set at defined times of the day. It can be set to reduce the upload and download speeds to a slow rate during business hours and remove all speed caps during off-peak hours. It could also be configured to completely stop BitTorrent traffic during business hours.

Although the use of Speed Scheduler prevents a specific client from using too much bandwidth during busy times of the day, it could also be used by a corporate employee to hide activity of unacceptable BitTorrent use. By hiding transfers during normal business hours and transferring data only in the late evenings, the client's presence could go unseen by network administrators.

Trackers

Connections to the BitTorrent network require the use of a tracker of some sort. Participation in the network can be effectively reduced or eliminated by simply blocking access to external trackers on the Internet. Although a tracker could listen on any port, 6969–7000 is the most common range of ports, with 6969 one of the ports most used by trackers. However, there is no standard rule for this, since a tracker can use any port it wants to, as long as it is explicitly specified in the torrent file.

Because a tracker is based on a simplified HTTP server, it does allow the ability to filter out traffic based on ASCII text sent over the connection. For example, when requesting a peer list from a tracker, the user's client sends a *User-Agent* value that identifies the software it uses—for example, *User-Agent: BitTorrent/3.4.2*. By implementing string searches on packets to filter for known BitTorrent clients in this value, packets that are meant for trackers could be discarded before they leave the network. Other values that exist within this initial tracker message that could be used for identification include *info_hash=* and *event=started*.

Sharing of Data

The sharing of data is a very important portion of the BitTorrent network. Using the tit-for-tat algorithm, a client that is unable to share data to others will be ignored and snubbed by a majority of its peers. This would cause download activity to either cease or travel at an extremely low rate. Blocking a client's ability to share

data can then effectively reduce or eliminate its ability to download data from the BitTorrent network. Most peers on the BitTorrent network use a standard range of ports for data connections with other peers, ranging from 6881 to 6999. However, in many clients, the user has the ability to specify whatever port he or she wants to for connections. A standardized port isn't required for use, since the peers simply need to report their chosen port to the tracker, which relays it along to other peers.

Before data can be sent between peers, the peer connection must be established by use of a specific TCP handshake. This handshake does include standardized text contained within its data header. This data is sent over Peer Wire Protocol (PWP) packets, which all must contain the text *BitTorrent protocol* at offset 1 in the data payload. Therefore, a firewall that can perform string matching can detect the text contained within the packet on incoming and outgoing packets and drop them to prevent peer connections from being made. Additionally, IDS rules could be written to detect such text traveling out of the network for alerting measures.

Snort IDS Rules

Snort is the most common IDS in use today. A free, open-source application that is available for Linux, Windows, and Mac OS X, it can be found at www.snort.org. Snort can be implemented into a network design to provide early detection of unauthorized traffic, giving administrators the ability to curtail it before it gets out of hand. Snort rules for most common applications can be found across the Internet, with some provided below for integration into existing Snort sensors. These rules are based on the previous protocol analysis and can be implemented by customizing the alert message and *classtype* to match the rules of your IDS. The first two rules use the PWP data fields of *Message Length* and *Message Type* to determine whether they are BitTorrent packets.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (content:"|0000000d06|";
offset: 0; depth: 5; flow: established; classtype: bad-unknown; msg:
"BitTorrent Data Request");
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (content:"|0000400907|";
offset: 0; depth: 5; flow: established; classtype: bad-unknown; msg:
"BitTorrent Data Transfer");
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (content:"/announce"; flow:
to_server,established; classtype: bad-unknown; msg: "Possible BitTorrent
Tracker Connection");
```

Summary

In this chapter, we examined some of the basic features and aspects of the BitTorrent peer-to-peer network. We started with an overview of the BitTorrent network and its origins on the Internet. The BitTorrent network was designed in early 2001 by its single creator, Bram Cohen. It has since evolved to become of the most well-used peer-to-peer applications in current history. Cohen maintains the official BitTorrent client, but its released source code has inspired the creation of dozens of spin-off clients, each offering its own additional features. One of the most popular of these is the Azureus client, an open-source application written in Java.

We then discussed the basic design of the BitTorrent network architecture, which covered the topics of torrents, trackers, and peers. For a file to be shared on the network, it must initially be shared by a single peer, which creates a torrent file to share with others. This torrent file contains metadata about the data content to be shared, as well as information about the specific tracker that will be handling connections for it. This tracker is a specialized software application that listens for connections from BitTorrent users. Each user requests a peer list from the tracker for the particular file he or she wants to download, specified by the SHA-1 hash value of the torrent file. The tracker responds with a list of peers who are currently sharing or downloading portions of the data. The peer then begins making connections to the peers that it has received and requests data directly from them.

We delved into the BitTorrent protocol data and the information that is contained within them. Much of this information is encoded using a format called Bencode, whereby complex data structures can be stored in a single line of a flat text file. This format is used to store information within the torrent files that are shared about the Internet, which includes vital information about the data hosted on BitTorrent networks. We also observed the data contained within tracker connections to determine information regarding the data requested, as well as information about the client requesting the peer list. We then discussed the various Peer Wire Protocol (PWP) messages that peers on the BitTorrent network use to communicate with each other. These packets are sent in a very structured form that allows for multiple types of data to be sent simultaneously and for peers to be able to constantly update each other about data connections.

Finally, we discussed some of the major issues that affect the BitTorrent network, such as the widespread sharing of copyrighted materials, corrupted material on the networks, and the inherit behavior in the client to automatically share data with others. We also mentioned the bandwidth issues that come into play when sharing data with thousands of peers, as well as ways of either reducing or eliminating them.

For example, bandwidth use for BitTorrent could be scheduled for an off-peak time, when bandwidth use is cheaper and less taxing on active users. Furthermore, data sent to trackers and other peers can be detected and discarded based primarily on static information contained within them, although simple port blocking can effectively reduce a good portion of the data.

Solutions Fast Track

History of the Network

- ☑ BitTorrent is a popular peer-to-peer application that has been in widespread use since 2002.
- ☑ BitTorrent's creator maintains an official client but also releases the source code, which has allowed dozens of other clients to appear over time.
- ☑ One of the most common customized clients is Azureus, which provides countless features and abilities not found in the official client.

Network Architecture and Data Flow

- ☑ The use of the BitTorrent network hinges around using a torrent file to make the connections to the appropriate network of peers.
- ☑ A tracker is also a very integral part of the network; it maintains a list of all clients that are sharing a particular file and passes this information on to other peers.
- ☑ All actual data are transferred directly between clients, which also manage which peers they want to communicate with and in what way.

Protocol Analysis

- ☑ Torrent files, which allow initial connections onto the network, are Bencoded ASCII text files that include information about the file and the tracker used for the connection.
- ☑ When connecting to a tracker, a client uses a standard HTTP *GET* request, combined with the SHA-1 hash of a file and information about itself, to gather a list of peers.

- ☑ Peers communicate directly with each other using TCP Peer Wire Protocol messages, a format used exclusively by the BitTorrent protocol.

Features and Related Security Risks

- ☑ One of the largest concerns with using the BitTorrent network is the issue of copyrighted material, which is a large part of the data on peer-to-peer networks.
- ☑ In battling the sharing of copyrighted material, organizations have begun targeting individual users by legal means and by sending large amounts of corrupted data to their clients.
- ☑ Another concern with the BitTorrent network.

Bandwidth Issues and Mitigation Steps

- ☑ The use of BitTorrent can be scheduled via a plug-in to off-hour times to reduce bandwidth cost, but this method can also be used to hide activity from network administrators.
- ☑ Blocking standard BitTorrent ports can be effective at reducing the amount of traffic on the network, but it will not eliminate the problem, because new clients and trackers are implementing random and changing connection ports.
- ☑ The most effective way of detecting and reducing BitTorrent traffic is to perform packet analysis and detect known ASCII strings within the packets.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: Is it possible to preview a file before it is finished downloading?

A: In most cases, no. Because BitTorrent downloads file pieces in random order, it might not be possible to view a file until the entire download is completed. This will vary based on the type of file format, though, because some audio songs may be previewed whereas some videos may not.

Q: Is it possible to download only a few files from a multiple-file torrent?

A: This capability does not exist in the official client. However, some third-party applications such as Azureus and BitComet allow separate priority control over files within a torrent. This allows specified files to be downloaded first or not at all. It is possible to download a torrent of 20 files and only download 10 of them. Your client will simply refuse to request data for the files you specified it not to download.

Q: Is it possible to hide a BitTorrent application on a computer?

A: In many cases it is very easy to hide a BitTorrent application, depending on the operating system in use. Many Windows-based applications offer the ability to minimize to the system tray. There are also Linux and UNIX-based clients that are designed for “headless” computers, which are computers without monitors. These clients can be installed and placed into a background process to allow them to run even when their user is logged off.

Q: What's the difference between BitTorrent and Avalanche?

A: Avalance is a Microsoft peer-to-peer concept that has recently received press attention. However, at this time, Avalanche is not available for public use and is far from becoming a usable protocol, something known as vaporware. Avalanche shares many of the traits of BitTorrent, with minor modifications, such as tit-for-tat. It is claimed to be more efficient than BitTorrent, with faster download speeds. At this time, Avalance is not planned for release or implementation into any product.

Q: Can BitTorrent traffic be transmitted through a proxy?

A: Although the official client does not have built-in support for proxy connections, third-party applications such as Azureus do have the capability to transmit data through SOCKS proxies. There are separate options for transmitting just tracker information and information to peers.

FastTrack

Solutions in this chapter:

- History of Clients and Networks
- Spyware Bundling and Alternative Clients
- Network Architecture
- Protocol Analysis
- Features and Related Security Risks
- Bandwidth Issues and Mitigation Steps

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

FastTrack is one of the largest peer-to-peer networks on the Internet and one that not many people have ever heard of. However few people know of its name, the clients that are used to access FastTrack have gained global infamy. Kazaa, a product of the two creators of the FastTrack network, has been the subject of a number of lawsuits over its ability to allow users to easily share, and thus infringe on, copyrighted material. The FastTrack protocol, accompanied by the Kazaa client software, was released in early 2001 by the company Consumer Empowerment. Because the once-famous Napster service was on its way to closing its doors due to court-ordered injunctions, the concept behind the FastTrack protocol became the new hope for millions of users on the Internet. Unlike Napster, which was based on a centralized server that stored each user's hosted files, the FastTrack protocol was built with a partially centralized network design.

However, with time, it quickly became a haven for the trading of copyrighted music, movies, and pornography. This same content has become one of the most pressing issues facing the network and its users today. At one time, the network was the largest peer-to-peer network in the world. The FastTrack network, though, wasn't the safe haven that many of its users hoped it would be. It soon became a hunting ground for members of the Recording Industry Association of America (RIAA), as well as other entertainment publishers, to target sharers of copyrighted material. Regardless of the amount of illicit material on its network, Kazaa is actively pushing licensed material on the FastTrack network, through its AltNet associated service.

History of Clients and Networks

The FastTrack network has had its share of clients over its short life span. Due to FastTrack's closed design, many new clients were developed by performing unauthorized modifications to existing clients. As the owners of such valid clients made modifications to bar unwanted clients from their network, lines were drawn and new networks were created. One of the best ways to know about a software product is to know its history. Through following the history of an application, you can learn its purpose and why it is continually developed.

The FastTrack Network

The FastTrack network was unveiled to the world in March 2001 as a combination of an established network and client software named KaZaA. The network and its client were designed by two Scandinavian entrepreneurs, Janus Friis and Niklas

Zennström, owners of the Consumer Empowerment company. The FastTrack network was designed at a time when decentralized networks were very young and immature. At the time of FastTrack's release, the only decentralized network in use was Gnutella, which was still in its infancy and problematic in use. FastTrack was very similar to Gnutella's fully decentralized network, but it differed with the introduction of supernodes.

A *supernode* is a network client that facilitates search requests for other clients; a connection to a supernode is required for access to the FastTrack network. A supernode maintains an index of all the files shared from the peers below it, acting as a pseudo-central server of the network. The implementation of the supernode concept changed the way that the entire network could be controlled. No longer could a single, central server be taken down, as was the case with Napster. Now many computers, stretched across the globe, could take up the role of central servers at any time.

The FastTrack network was designed using a closed-source, proprietary protocol. It uses encryption for much of its internal network communications, intended to keep access of the network confined to only licensed clients. This is exactly the opposite of Gnutella's network design, which is open-source and to which access is made available for any client. However, due to extensive experimentation, major portions of the network communications were unraveled by the Internet public. In particular, the communications between nodes and supernodes were the largest bits that could be determined. The discovery of this data led to the creation of unauthorized clients that could communicate with the FastTrack network, many of which have added capabilities that the basic Kazaa client lacked.

Control over the licensed protocol has been kept tight by its owners. Slight modifications were made along the way, such as in February 2002, when the protocol was changed to prevent a third-party client, Morpheus, from being able to access the FastTrack network.

Kazaa

The KaZaA client, now known as Kazaa Media Desktop or simply Kazaa, started life along with the creation of the FastTrack network in March 2001. The current software product, Kazaa Media Desktop, offers many features that can be useful for a file searcher, such as:

- The “Search More” function, that will extend an existing search to other supernodes, resulting in more results
- An integrated Search Agent, which will perform automated searches repeatedly for 24 hours.

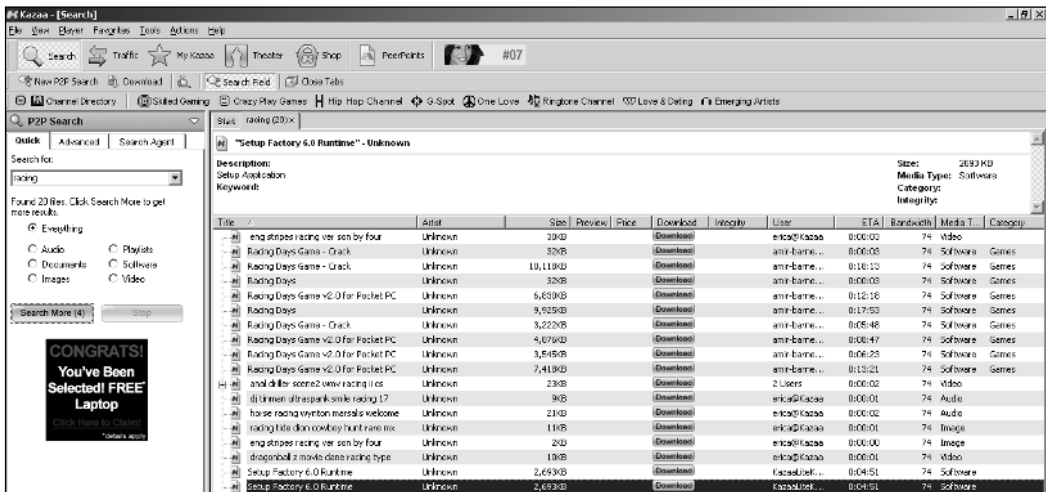
- The ability to perform up to 24 simultaneous searches
- Built-in antivirus protection
- The ability to filter out files from your results based on extension, or keywords
- Magnet links, allowing a user to post direct links to their shared files on web sites
- Built-in Media player that plays most music and video formats

Along with all of these features is the ability to search for files amongst millions of peers. The Kazaa client supports the sharing of all file types, but has specific categories for Audio, Documents, Images, Play lists, Software, and Video, as well as the all-encompassing Everything category. These categories are generally defined just by the file extensions. For example, when searching for images, the results will be limited to files such as JPEG, GIF, PNG, TGA, and bitmap files.

The Kazaa client is offered in two forms from its current owner, Sharman Networks: Kazaa Media Desktop (KMD), as shown in Figure 12.1, and Kazaa Plus. While Sharman Networks maintains a web site at www.sharmannetworks.com, they offer their Kazaa clients at www.kazaa.com. The former of which is a free client that offers basic access and usage of the FastTrack network. However, it is technically defined as an “ad-supported” application, which is a technical term referring to its inclusion of spyware applications. For a nominal fee, though, Kazaa Plus is available for purchase. Kazaa Plus offers the same access as the base KMD application, but does not include extraneous advertising applications. It also features additional functionality over the base client, such as:

- The ability to download a file from up to 40 sources, instead of 8 with KMD
- The ability to perform a “Search More” multiple times, offering a total amount of 3,000 possible results
- Skype Voice-Over-IP (VOIP) client, which allows free Internet-based phone calls to other Skype users

Figure 12.1 Kazaa Media Desktop



History of Kazaa

While the Kazaa client was introduced at the creation of FastTrack, it has had a long history under multiple owners, each facing legal troubles due to the functions of the client, and remains one of the very few official FastTrack clients. The client, originally named KaZaA, was designed and released by the Consumer Empowerment company. It provided the basic means to search the FastTrack network for various types of files, and also included the ability to be automatically updated from its developers.

Almost immediately, though, legal troubles began for Kazaa, as well as the entire FastTrack network. In October of 2001, the owners of the three major FastTrack clients, KaZaA, Grokster, and Morpheus, were sued by the Recording Industry Association of America (RIAA) and the Motion Picture Association of America (MPAA) for contributory and vicarious copyright infringement. At the same time, Consumer Empowerment was sued in a local Netherlands court by Buma/Stemra, a Dutch copyright watchdog organization. A court ruling declared that the creators of KaZaA would either have to take drastic steps to prevent its users from being able to share copyrighted files, or be forced to pay large fines. This ruling started a chain of events that changed the FastTrack network's landscape, and eventually the entire peer-to-peer community.

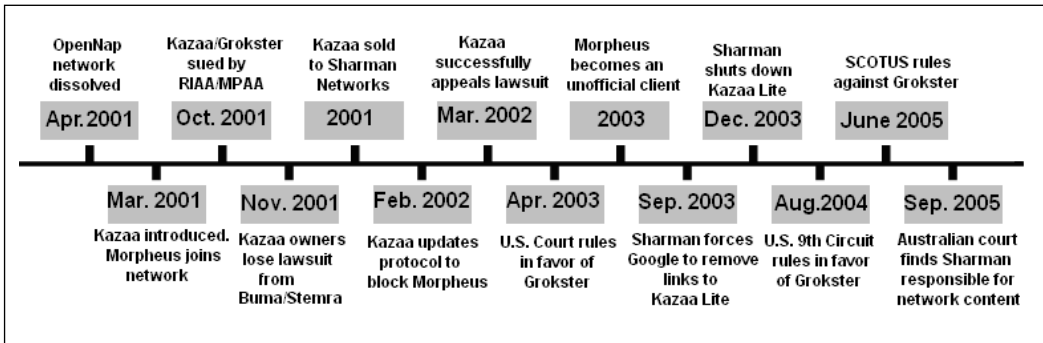
In response to the court ruling, Consumer Empowerment sold its KaZaA application technology to various foreign businesses, most notably the Australian-based

Sharman Networks. In a sign of signifying new ownership, Sharman Networks renamed the KaZaA client to a less eccentric spelling of Kazaa. At the same time, Consumer Empowerment changed its business name to Kazaa BV, while still holding control over the FastTrack network intellectual property. It was a business decision that seemed to place much of the court-mandated control of the network in the hands of other businesses, while letting Kazaa BV profit from licensing out the network architecture.

Kazaa finally met with good legal news when, in March of 2002, they were able to successfully appeal their original lawsuit by Buma/Stemra. The judgment made mention that providing “the means for publication or reproduction of copyrighted works is not an act of publication or reproduction in its own right.” The FastTrack network was free and clear to operate normally again. The text of this judgment can be located and reviewed at: http://www EFF.org/IP/P2P/BUMA_v_Kazaa/. Nevertheless, there were still pending suits against Kazaa BV from American media conglomerates, which had deep financial pockets and strong perseverance in seeing peer-to-peer networks disappear from the Earth. Even though many of Kazaa’s cases are still pending as of this time, a successful RIAA suit against Grokster, a fellow FastTrack client, created a strong precedent in the legal system that would be hard for Kazaa to fight against.

In fact, demonstrating that shaking the various lawsuits held against the network will be difficult work; Sharman Networks faced the losing end of multiple lawsuits in 2005. First, in June of 2005, the United States Supreme Court found that the owners and operators of peer-to-peer networks are ultimately responsible for the material that is passed within their networks. This major ruling was held against all makers of peer-to-peer software, not just Sharman Networks. More trouble brewed when in September of 2005, in a court ruling in its home country of Australia, a judge of the Federal Court of Australia ruled that Sharman Networks was liable in the spreading of copyrighted materials on the FastTrack network. Simply posting a message to users to not trade copyrighted material was not a sufficient control of the wide-spread sharing. It was mandated that changes be made to the network to reduce the illicit sharing, while not intruding on legal file transfers.

A graphical timeline, representing the major stages of the FastTrack network, is shown in Figure 12.2.

Figure 12.2 FastTrack Network Diagram

Morpheus

Shortly after FastTrack was released, another music-based peer-to-peer network was facing troubles of its own. The OpenNap network, upon which the MusicCity service was based, had come under the scrutiny of the RIAA. Faced with charges of spreading copyrighted music files, the OpenNap network was dissolved in April 2001, leaving tens of thousands of users homeless. However, the owner of MusicCity, Steve Griffin, saw the FastTrack network as a way to continue his business. Thus, Morpheus was born from MusicCity's ashes, using licensed FastTrack protocol information, and the company name changed from MusicCity to StreamCast Networks. However, its life on the FastTrack network was short-lived, and it eventually became a Gnutella network client, and eventually a client used to access most peer-to-peer networks.

The Morpheus client was very similar in appearance, and operation, to the original KaZaA client. It provided the ability to search the FastTrack network for various types of files, including music, videos, and software. It also featured a built-in media player that could organize and play music and video libraries. It featured one great benefit that was not found in the KaZaA client, though: the inclusion of high-quality music files. KaZaA's client was designed to disallow the sharing and downloading of audio that had a higher than 128 Kbps (Kilobits per second) bit rate. In comparison to other audio qualities, 128 Kbps audio files featured greater compression that could deaden many sounds and tones of the music. The Morpheus client was well used due to its lack of a bit rate filter, allowing users to trade music encoded at higher qualities, such as 160 Kbps or 192 Kbps.

Morpheus's initiation into the network brought tens of thousands of users that helped the network grow large and strong. Conversely, its later forceful exit from the network took hundreds of thousands of users away from FastTrack, and into competing networks.

This forceful exit came about in February of 2002 when the Morpheus client was barred from the FastTrack network by its owners, Kazaa BV. The official reason given by Kazaa BV was that Morpheus' owner, StreamCast Networks, Inc., had failed to pay their quarterly licensing fee. In a statement made to CNET News, Niklas Zennström, who was one of the original creators of FastTrack, and the founder of Kazaa BV, wrote:

"MusicCity (also known as StreamCast Networks) has failed to pay any amounts due to Kazaa BV under the parties' license agreement. As a result of MusicCity's breach, Kazaa BV did not provide version 1.5 to MusicCity. Kazaa has also terminated MusicCity's license."
(<http://www.mp3newswire.net/stories/2002/pulledplug.html>)

StreamCast insiders rejected this reasoning, hinting over a squabble that had taken place between the two companies. It was alleged that Kazaa BV had implemented certain controls in files used by Morpheus, which would allow access to the FastTrack network to be blocked completely. Ultimately, this issue became moot, as Kazaa BV sent an automatic update out to all of its official clients, with exception to Morpheus, which updated the network protocol from version 1.3 to version 1.5. This move permanently blocked Morpheus clients from being able to connect to the FastTrack network.

In late 2003, though, Morpheus integrated FastTrack support back into their client, making it a multi-network client. The support and access are unlicensed from the owners of the network, and were based off of another multi-network client, MLDonkey. It joins the ranks of the many unlicensed clients on the FastTrack network, such as K-Lite, Kazaa Lite, and giFT.

Grokster

Grokster is another original FastTrack client, and one that has gained much notoriety due to its long-running battles in the U.S. court system. Founded and operated by owner Daniel Rung, Grokster has faithfully stayed with the FastTrack since its inception. The Grokster client is nearly identical in form and function to the Kazaa client. Grokster's official web site is <http://www.grokster.com>.

Grokster's popularity stayed high through its repeated exposure in the news media, due to its ongoing lawsuit from the Recording Industry Association of America and the Motion Picture Association of America. The lawsuit, filed against Grokster, Ltd. In October of 2001, remained an ongoing battle for many years. The first few legal rounds rewarded Grokster with great victories. In April of 2003, a federal court judge ruled in favor of Grokster, noting that the use of Grokster was not

illegal. In August of the same year, the RIAA and MPAA appealed the decision. However, victory was Grokster's again the following year when, in August of 2004, the Ninth Circuit Appeals Court sided with the peer-to-peer client's owner.

The legal battle didn't end there, though, as the Supreme Court of the United States decided in December of 2004 to hear the case. Additional financial help in the Supreme Court case came from the eccentric billionaire, Mark Cuban, through the Electronic Frontier Foundation (EFF). The EFF itself represented Grokster for free. Unlike previous cases, Grokster soon found itself at the losing end of this battle. In June of 2005, the Supreme Court rendered their decision in support of the RIAA and MPAA. While the long term effects of this ruling are still being worked out, many peer-to-peer clients are now requiring users to affirm to a statement that they will not use the software to download copyrighted materials.

Grokster currently vies for a reputable market by offering legitimate offerings with its client software, such as its fairly recent partnership with Mercora, Inc to deliver radio music transmissions to its users.

iMesh

The iMesh client has the distinction of being one of the oldest peer-to-peer applications, and can be downloaded from <http://www.imesh.com>. It was originally released late in 1999, months after Napster was introduced. It was also one of the most problematic, and has been shunned by many members of the peer-to-peer community. However, iMesh did bring about major innovations to the community. For instance, it was the first client to offer the ability to download a single file from multiple sources, whereas other clients only allowed downloads from a single hosting computer. This feature has grown to be a required standard in every modern client. It was also the first client to offer support for sharing of any type of file, not just audio files. Now, users had the ability to not just share music, but also full-length movies and documents. Of course, with access to larger files to download, another important feature was implemented into iMesh: the ability to resume a download.

A mere two years after its introduction to the world, iMesh dropped its native network and signed up to the FastTrack network, where it has remained ever since. Recently, iMesh has been one of the first clients to attempt to legitimize their client, by signing up with the Sony-BMG group to sell music tracks to users.

Spyware Bundling and Alternative Clients

All of the available FastTrack clients have free editions, or at least that is what users are led to believe. As is the case with most free, popular software on the Internet, it is released as “ad-supported”. This term can mean a variety of things, but at the very least the software is included with Adware applications, and in some instances even Spyware applications. Adware is simply software that is installed into an application and displays advertisements in a variety of ways, such as pop-up windows and graphical banners. Spyware is a more dangerous sort of application, one in which unique, identifying data about a computer is transmitted to a central server. This server, usually in the control of an advertisement organization, collects data to build profiles about individual user’s Internet habits, for future advertisement opportunities.

While many of us feel that we are simply too smart to be affected by spyware, this is not the case for most of the Internet public. Years of Internet experience have taught many of us that if a web site says that it requires a free plug-in to operate correctly, that it is not truthful. We’ve learned that free, closed-source applications cannot stay in business without having some sort of monetary income, with most of it gathered by embedded advertisements and spyware. However, millions of people that come in contact with peer-to-peer networks do not have this knowledge, and will click any “Yes” or “OK” prompt that appears, without hesitation. What is worse is when these applications are installed on corporate workstations, making their inevitable repairs very costly and time-consuming.

One of the few defining differences between most FastTrack clients is simply the different advertising software that they implement into their clients. For instance, the Kazaa Media Desktop currently uses Adware and Spyware such as the Gator Advertising Information Network (GAIN) and Cydoor applications to provide advertisement data to client software. Additional applications that it may also install include: PerfectNav, MyBar, SaveNow, and WhenU. Kazaa Media Desktop will also install the TopSearch application, used in conjunction with AltNet. And, as time has shown, Sharman Networks is only more willing to include more advertisers within their product. In October of 2005, they inked a deal with a company named Direct Revenue to include The Best Offers adware application in the KMD bundle.

AltNet

One application hidden into Kazaa and some other FastTrack clients which has raised many concerns is Brilliant Digital Entertainments AltNet software. AltNet is a

product that was released with a credible mission: to allow to the distribution of copyrighted material, such as music and movies, securely to paying consumers. Through encryption, only authorized purchasers of the data are allowed to download it, and transmit it to other authorized users. The first inclinations of trouble came when the software would be installed without even being activated. The application would lie dormant on a computer, because the needed infrastructure and design were not yet complete. Installing unknown software on your computer is one thing, but installing software that doesn't currently operate, but may do so in the future to do unknown commands proved to be very scary among many Kazaa users.

Another issue with AltNet that frightened Kazaa users was an optional portion of the AltNet client that would install additional applications to use system processing power, and bandwidth, to perform data calculations for corporations. This model is similar to the ones used by Folding@Home and SETI@home for distributed computing power. Although the implementation is an "Opt-In" choice, the potential for misuse weighed heavily on some users minds. The details of this application are officially detailed on Brilliant Digital Entertainments website at www.brilliantdigital.com/content.asp?ID=779.

It was mainly because of these software additions that modified versions of the clients soon started appearing. Marketed as "Lite" versions of the originals, these clients were offered without the inclusion of Spyware applications. Some were even modified to bypass limits imposed within the clients, such as the maximum number of "Search Mores" available to a user.

Kazaa Lite Client

Kazaa Lite is a FastTrack client derived from the original Kazaa software. The major modifications made to the software were the removal of all adware and spyware applications, including BDE's AltNet software. Additional enhancements were also made to the software, including the inclusion of a basic firewall application named PeerGuardian. PeerGuardian is an application that blocks the retrieval of files from previously specified IP addresses. This feature has become well touted as lists of known RIAA and MPAA sponsored sharers of "poisoned" files become known and cataloged. These are companies that exploit the file hashing calculations within Kazaa to release fake files onto the network.

The development of Kazaa Lite was unique in that the developers did not create a new client, as they did not have access to the source code of the original. Instead, the Kazaa Media Desktop client was modified by use of plug-ins, external utilities, and by manually editing KMD's binary files. The modifications were very effective and the Kazaa Lite client gained a lot of popularity during its time.

Its popularity grabbed the attention of Sharman Networks, however, who vowed to stop the spread of the unauthorized client. Sharman Networks initially used portions of the Digital Millennium Copyright Act (DMCA) in August and September of 2003 to have search results intentionally removed from Google's search engine. Furthermore, in December of 2003 they targeted all of the known sites hosting Kazaa Lite, forcing them with threats of legal action to either shut down or remove information and files pertaining to the Lite client. Under pressure from these legal threats, development of the Kazaa Lite came to a complete stop, and the sites that officially hosted it are no longer operational.

Due to Kazaa Lite being effectively shut down, other developers took over the reigns to create their own versions of Kazaa Lite. Sharman Network's squashing of one client just ended up creating more clients for them to chase after. Many of the upcoming clients, being based off of Kazaa Lite, included the existing modifications that Kazaa Lite introduced.

Kazaa Loaders

The question is then asked, "How was it possible for a closed-source Kazaa client to be modified?" Due to limitations in modifying existing Kazaa files, and not having the luxury of the client's source code to read and make changes to, creative measures had to be taken to make radical design changes to the way that a Kazaa client operates on the FastTrack network. To facilitate this need, many Kazaa loaders were released. A loader is a software application that executes the normal Kazaa Lite executable, and then proceeds to modify known data locations in memory that Kazaa Lite produces. Because no original files were disassembled or reverse engineered, this process was useful in bypassing computer Intellectual Property laws in some countries.

K++, which is sometimes referred to as Kpp, is one such external loader utility that makes dramatic modifications to the Kazaa Lite application. K++'s original author, Random Nut, eventually stopped development on the project, though the source code was released to the Internet. The last official Kazaa Lite with K++ was 2.4.3. Starting with Kazaa Lite 2.6 a replacement of K++ was used, named KHancer. KHancer is normally not downloaded by itself, instead it is included with more complete packages such as K-Lite. Over time, other loaders have also been developed, such as Diet K, which can be found at <http://www.dietk.com>. The use of a loader like K++ and KHancer has become a standard with modified versions of Kazaa's client. The use of them adds many additional functions, such as:

- The ability to install Kazaa without any adware or spyware
- Removing references to purchasing music, such as the Shop tab

- Providing the PeerGuardian firewall for IP address blocking
- Disabling port 1214 to prevent TCP scans from finding your client
- Allowing unlimited uses of the “Search More” feature
- Setting your client’s participation level to 1000, the highest amount
- The ability to jump to a different supernode on-the-fly

External Utilities

To provide even further functionality to Kazaa clients and to alter the entire FastTrack network experience, multiple utilities were created for FastTrack users. These utilities are usually included with modified Kazaa clients to accompany existing modifications made to the clients. These applications extend a client’s capabilities, such as being able to access FastTrack from behind a router using Network Address Translation (NAT) and being able to jump to another supernode immediately. Examples of Kazaa utilizes include:

- **K-Dat** A tool used to view data stored in your local Kazaa .DAT files. K-Dat could be used for forensic purposes to determine what files were downloaded, where they were downloaded from, and where they were locally saved to.
- **K-Sig** A replacement for the Sig2Dat tool, K-Sig creates a hash value of a specific file that can be shared with others. This value can be used to create a magnet link that can be shared to others on Web sites and forums.
- **AVI Preview** Allows a user to view partially downloaded video files.
- **KaZuperNodes** Displays a listing of all FastTrack supernodes and allows a client to jump to a specific supernode.
- **KaNAT** Allows Kazaa to operate behind a router or gateway that uses NAT.
- **MP3 Shield** Verifies all MP3 audio files that are downloaded to ensure that they are authentic, and are not “poisoned” or fake files.

Kazaa Lite Resurrection Client

Kazaa Lite Resurrection, also referred to as KLR, is a client that emerged very shortly after the original Kazaa Lite was forced into closure by Sharman Networks. KLR started out as very similar to the old Kazaa Lite, with just additional tools, but

its continuous development has helped the client stay current and well used. Although it does not maintain its own official Web site, it distributes official copies of the software at http://filesharingplace.com/downloads/kazaa_lite_resurrection.php as well as offering a support forum at <http://filesharingplace.com>.

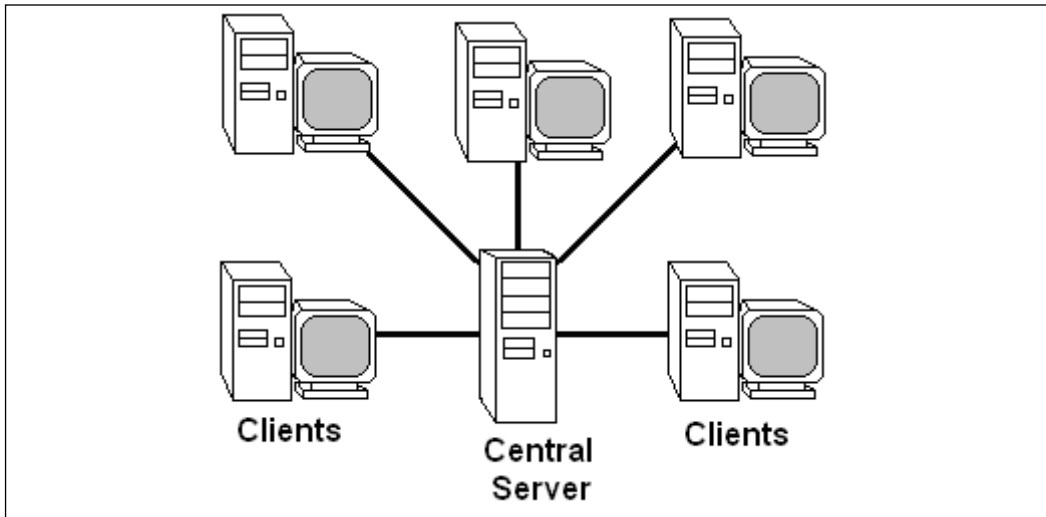
K-Lite Client

K-Lite was another Kazaa-modified client that arose after the departure of Kazaa Lite. K-Lite currently is one of the most commonly used unauthorized FastTrack clients. Downloaded from its Web site, www.my-k-lite.com, K-Lite features many new capabilities that are not found on other Kazaa clients. For instance, K-Lite features a fake file filter. Similar to the MP3 Shield utility, K-Lite can compare a set of files from a search result, based on their MD5 hash value, to determine whether they are authentic files or fake files.

Network Architecture

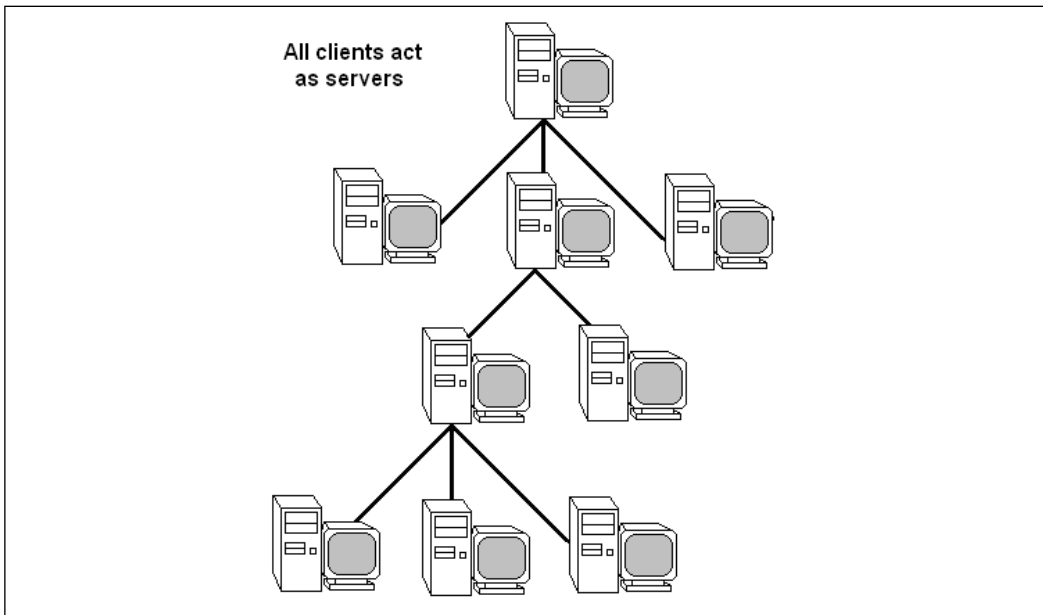
FastTrack features a network architecture that was relatively uncommon at its conception—one technically referred to as a *hybrid*, or *semicentralized*, network. The hybrid network design is currently one of the most efficient designs being implemented. In comparison to other networks, the hybrid design reduces the legal exposure of the centralized network, but it does not suffer from slow searches, as the decentralized network does.

The centralized network design, famous for its use by Napster, used a central server to store a listing of the music catalogs hosted by its members (see Figure 12.3). When a user made a search for a particular music file, the server would create a list of users who were hosting the file; this list would then be reported to the searching client. Having a central server made for very quick searches and high-speed downloads. However, this design eventually created legal troubles for Napster, so much so that the network was shut down permanently for basically making it too easy for users to trade copyrighted music and audio files.

Figure 12.3 A Centralized Network

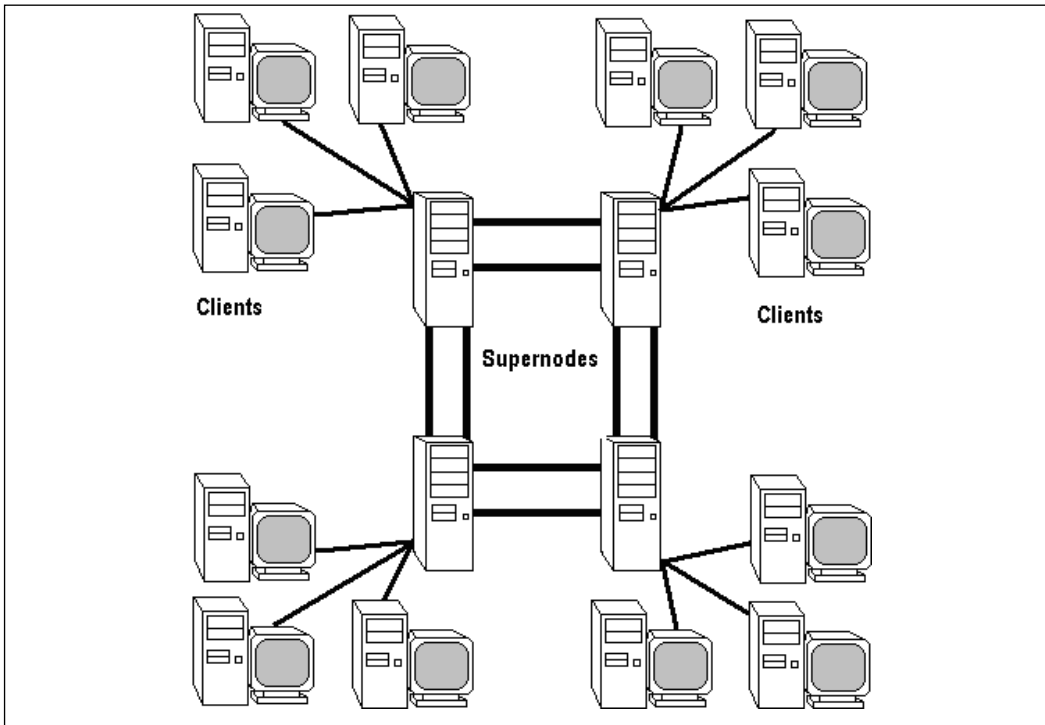
At the other extreme is the decentralized network design, used by networks such as Gnutella. In a decentralized network, there are no central servers used to store data about traded files. Instead, each client that connects to the network makes a connection to other existing clients, creating a layout very similar to a mesh network (see Figure 12.4). Although the lack of a centralized server means that there is very little control over the network and that it would be difficult to shut down, it also places all the searching tasks onto other computers. This method ends up creating large amounts of traffic flooded across the network every time a client searches for a file.

The hybrid network design shown in Figure 12.3, used by the FastTrack network, was a compromise of the two network designs. Instead of a central server governed by the operators of the network, FastTrack implemented the concept of supernodes.

Figure 12.4 A Decentralized Network

Supernodes

Supernodes are normal clients that are the equivalent of a central server, offering the ability to index the file contents of various clients while not being in control of a single entity. Due to their nature, supernodes cannot be controlled, nor can they be disabled, by the owners and operators of the network, even under legal pressure, without resorting to protocol changes that would significantly alter the network design. For an example of a typical FastTrack network layout, refer to Figure 12.5, where supernodes create the central ring of the network, each supernode with its own collection of connected clients.

Figure 12.5 A FastTrack Network

Each supernode starts out as a standard client on the FastTrack network. Within the client software is an algorithm used to determine whether the computer should become a supernode, based on the amount of hard drive space, RAM, and bandwidth available to the client. The exact specifications of how this is determined are not disclosed and are among the many secrets stored within Kazaa's proprietary software library used by FastTrack clients. However, thankfully, each client contains the capability to disable itself from becoming a supernode. What is known is that a supernode can allow at least 100 individual clients to connect to FastTrack. The number of supernodes that exist is itself astounding; an estimated 30,000 supernodes are formed at any one time on the FastTrack network. As clients connect to a supernode, they will transmit a listing of all files that they are sharing, and this list will be stored within an index on the supernode. This index contains information about each shared file, such as the filename, file size, type of file, and file description.

When a client performs a file search, the search query is initially sent to the supernode that the client is connected to. The supernode will search its own index to determine whether any other connected client, of which there could be 100 for a

single supernode, contains the appropriate files. The supernode will then send the same search request to other supernodes across the network. As these supernodes find appropriate results in their indexes, they will send information about the file to the supernode that requested it, which will then be relayed back to the original client. At this point, when a client decides to download a file, it connects directly to the computer that is sharing out the file.

As a supernode, a computer plays a greater part in the overall FastTrack network. However, this comes at an expense to the resources available to the computer. Not only is the computer acting as a gateway for a plethora of clients, it is also actively maintaining a database of every file that each of those clients is hosting. The network connection that a supernode sits on will have increased bandwidth usage from both sides of the network. The clients that connect through the supernode will use it to send queries out to the network and to store information about files that they are sharing. Also, other supernodes on the FastTrack network will continually send search queries to the supernode, asking for IP addresses and information for files that clients below it contain.

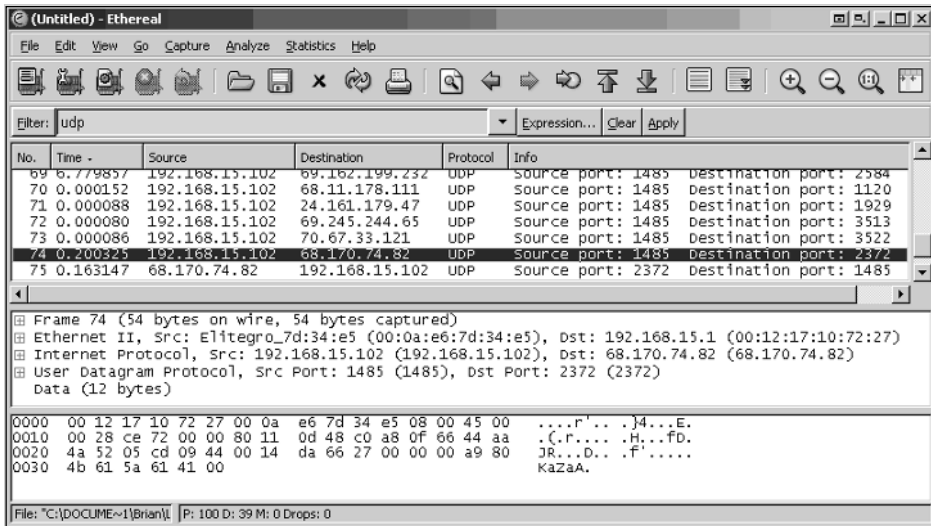
A supernode-related packet can be determined by viewing the data payload contents of the packets on a network, based on the protocol information. This information can be gleaned by viewing the latest protocol revision at <http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL>, a site for the giFT client. The site describes various supernode-to-supernode packets that are not used for client connections. The data in these packets begin with known hexadecimal bytes, such as 0x01, 0x0A, 0x0B, and 0x0C. It is also possible to detect incoming supernode pings from external clients, which are packets destined for supernodes. More information about these specific FastTrack packets can be found in the next section.

Protocol Analysis

In the earlier days of FastTrack, all the basic search data and connection information was transmitted and received primarily on port 1214. However, as the network and its clients evolved, they implemented the use of other ports to bypass firewall rules. This section covers some of the actions taken by the network in certain situations so that you may be able to learn how to minimize or completely block such traffic. This traffic can be viewed and analyzed using any normal traffic analyzer, such as Ethereal (see Figure 12.6). Ethereal is a free, open-source application that can be downloaded from www.ethereal.com. Due to the fact that the protocol is closed-source, much of it has had to be discovered through experimentation. The devel-

opers of the giFT-FastTrack plugin, located at <http://developer.berlios.de/projects/gift-fasttrack>, have discovered and disclosed many of the details of the FastTrack protocol. giFT-FastTrack is a plugin for giFT, which is a recursive acronym for *giFT: Internet File Transfer*, a free, open-source daemon that allows connections to multiple peer-to-peer networks.

Figure 12.6 Ethereal, Showing Supernode Connections



Connecting Clients

When a client attempts to connect to FastTrack, it connects using a set of supernodes that were collected and stored locally. When a FastTrack client is started up, it initially sends a UDP packet to various supernodes to request a connection to the network, as shown in Figure 12.6. This is a normal UDP packet sent from a local port, usually 1214. The local source port may differ, since some clients use port 80 and others use randomly assigned ports. The destination port fluctuates depending on the supernode you're communicating with. Each packet contains a 12-byte payload with a hex code and ASCII equivalent of:

```
27 00 00 00 a9 80 4b 61 5a 61 41 00 '.....KaZaA
```

This packet is technically known as a *node ping* because it is used to first ping available supernodes before creating a connection. The hexadecimal data within this packet is broken down as shown in Table 12.1. Much of these data have been researched and believed to be true by the developers of the FastTrack plugin for the open-source giFT client.

Table 12.1 Node Ping Payloads

Data Size in Bytes	Hex Value	Description
1	0x27	A static value that signifies that this is a node ping
4	0x000000a9	The minimum encryption type used by the client
1	0x80	Unknown
0x00 terminated	0x4b615a614100	A string that represents the network name

When a supernode decides to accept a connection, it replies with a similar UDP packet that contains a 17-byte payload and a 1-byte trailer. This packet informs the client that they may use this particular supernode. In the event that multiple supernodes respond, the client will use the first one that it received a response from. The payload will appear similar to the following, with hex and ASCII shown:

```
fc 28 00 00 00 a9 00 2a 86 50 36 09 4b 61 5a 61 41 00  {...*.P6.KaZaA.}
```

This packet is technically known as a *node pong* because it is used to reply to a prior node ping. This packet lets the client know that the server is available for a connection. The hexadecimal data within this packet is broken down as shown in Table 12.2.

Table 12.2 Node Pong Payloads

Data Size in Bytes	Hex Value	Description
1	0x28	A static value that signifies a node pong; if the packet uses 0x29 in this value, it is disallowing a connection
4	0x000000a9	The minimum encryption type used by the client
1	0x00	Unknown but presumed to be a static value
1	0x2a	Unknown but increments by 1 approximately every 30 minutes
2	0x8650	Unknown
1	0x09	Percentage of load on supernode

Continued

Table 12.2 continued Node Pong Payloads

Data Size in Bytes	Hex Value	Description
1	0x4b	Unknown
0x00 terminated	0x4b615a614100	A string that represents the network name

With modern FastTrack clients, there could also be other channels of communication to enter onto the FastTrack network. By default, the previously defined UDP packets are used to create a connection. In the event that the client detects that they are blocked by a firewall, it will switch to encrypted TCP packets. The issues raised by this fact are addressed later, in the “Bandwidth Issues and Mitigation Steps” section.

Performing a Search

When a search query is entered into a FastTrack client, it is immediately encrypted and transmitted to the client’s assigned supernode from a local ephemeral port. The supernode then sends queries to other supernodes around it, gathering more results that eventually get transferred back to the client. All the packet transmissions during this phase remain encrypted, and their enclosed data cannot be viewed. Also of interesting note is that further searches will use the same local ephemeral port.

Transferring Files

When a user has chosen an appropriate file, he or she then requests a download of that file. This request is made by sending an HTTP *GET* request to the client that is hosting the file. The requesting client will already have the IP address and port number of the hosting client, since it was supplied by their supernode during the search process. A file download is started by requesting the hash value of the file or the name of the file directly from the hosting computer. Along with the request, the client will send its own IP address and information about itself. For example, a HTTP *GET* request would look similar to this:

```
GET /.hash=39728b66389d1745b0216fea6a1f7eb7f1e5ffff HTTP/1.1
Host: 84.15.12.6:2218
UserAgent: KazaaClient Jul 27 2004 21:14:16
X-Kazaa-Username: my-k-lite.com
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 192.168.15.102:1485
X-Kazaa-SupernodeIP: 69.81.20.135:2783
```

```
X-Kazaa-UserSite: www.my-k-lite.com
Connection: close
X-Kazaa-XferId: 11113283
X-Kazaa-XferUid: CxfaY6w4PIpTc+i/OBFr9sKM75cvLzHFbZxtUnixdEo=
```

The hosting client can then respond in one of a few ways. For one, if the client already has too many connections, it may respond that the connection is unavailable. In this case, the user's client will display that they are "queued" for download. An example of an unavailable response would look similar to the following TCP packet payload:

```
HTTP/1.0 503 Service Unavailable
Retry-After: 184
X-Kazaa-Username: robhorn
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 84.105.102.6:2218
X-Kazaa-SupernodeIP: 70.248.6.40:1698
```

Otherwise, the hosting computer will respond favorably and start sending the requested data. There are two situations in which this data could look different: when you request a file from only a single computer and when you request a file from multiple computers. In the case of a single hosting computer, the TCP packet sent to you will look similar to the following:

```
HTTP/1.1 200 OK
Content-Length: 8894
Accept-Ranges: bytes
Date: Fri, 07 Oct 2005 21:57:39 GMT
Server: KazaaClient Jul 27 2004 21:14:16
Connection: close
Last-Modified: Tue, 15 Mar 2005 12:57:44 GMT
X-Kazaa-Username: anonymous_user
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 65.32.71.174:3643
X-Kazaa-SupernodeIP: 65.35.122.231:2898
X-KazaaTag: 3==peT49yvylCGmNgmfCp++xUHd//8=
Content-Type: text/plain
<blank line>
[Data contents of the file will be appended to the packet here]
```

However, if you are downloading from a set of multiple hosting computers, the HTTP header will be displayed differently. Each sender will send only a partial amount of the entire download at one time. The specific portion of the data transmitted will be specified within the header, under *Content-Range*. An example of this is:

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 2475125-2593614/3062673
Content-Length: 118490
Accept-Ranges: bytes
Date: Fri, 07 Oct 2005 22:51:15 GMT
Server: KazaaClient Jul 27 2004 21:14:16
Connection: close
Last-Modified: Sun, 01 May 2005 19:17:29 GMT
X-Kazaa-Username: anonymous_user
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 65.24.219.80:1164
X-Kazaa-SupernodeIP: 65.25.150.31:2033
X-KazaaTag: 5=192
X-KazaaTag: 21=128
X-KazaaTag: 4=Title of Song
X-KazaaTag: 6=Song Artist
X-KazaaTag: 3==f54lghhZUfvtiCvWgCgc71FfJWA=
X-KazaaTag: 14=Alternative
Content-Type: audio/mpeg
<CR>
```

[Data contents of the file will be appended to the packet here]

Regardless of the download method, it is ended just like any other HTTP TCP connection. The hosting computer ends the data transfer with a *FIN* packet sent to the user's client. The client then responds with a *FIN/ACK* to acknowledge the end, and the connection is severed.

The X-KazaaTag

With music file downloads, Kazaa clients also include some basic information about the file along with the download. These are shown in the HTTP packets from the hosting computer as *X-KazaaTags*. These are predefined metadata of the file with information such as the artist, title, and genre of the audio file. Each tag begins with the text *X-KazaaTag:* immediately followed by the tag identifier and an equal sign (=). Behind each equal sign are the data associated with that particular tag. A list of

the most commonly used tags and their meanings is shown in Table 12.3. A systems administrator with a sense of humor could use firewall rules in conjunction with these tags to allow users to have the ability to download any genre of music they want, unless it's rap or hip-hop—or only allow music by Barry Manilow.

Table 12.3 X-KazaaTags

Tag ID	Type of Data	Description
1	4-byte integer	Year of release
3	Hexadecimal	Hash value
4	ASCII string	Audio title
5	4-byte integer	Duration in seconds
6	ASCII string	Artist name
8	ASCII string	Album name
10	ASCII string	Language
12	ASCII string	Keywords
13	4-byte integer x 2	Resolution (used for videos)
14	ASCII string	Genre
16	ASCII string	Operating system (for software)
17	4-byte integer	Bit depth
18	ASCII string	Type of file (e.g. "TV Shows")
21	4-byte integer	Quality (bitrate)
24	ASCII string	Version (sometimes used to store URLs)
25	4-byte integer	Unknown
26	ASCII string	File description
28	ASCII string	Codec (e.g. DIV3)
29	4-byte integer	Rating
33	4-byte integer	Size
37	Unknown	Unknown but sent with every Kazaa query
45	Unknown	Unknown but sent by iMesh with each hash query
53	Unknown	Unknown
65539	ASCII string	The <i>kzhash</i> value of the file, used for searching for magnet links

Features and Related Security Risks

Due to some of the many features and design implementations of the FastTrack network, it is open to a wide variety of security risks. Many of the issues that arise from using the FastTrack network are simply due to the nature of the content that is traded within the network. The main issues covered here are the type of data that could be downloaded from the FastTrack network as well as the type of data that could end up being shared across the network.

Downloading and Copyright Violations

Because the FastTrack network was designed for downloading files, it falls prey to the many problems that are contained within that process. During the early days of FastTrack, it was a haven for file traders. The newest popular songs and movies could be found and downloaded quickly. However, as it gained popularity, FastTrack also gained notice by the owners of various copyrights and, subsequently, the legal system. Initially the focus of these problems seemed to be on the network itself and the owners of the network, but in recent years some copyright owners have started targeting individual users. For a home user, this could mean the possibility of paying large settlement fines to copyright holders or facing a drawn-out court battle. For businesses where employees are downloading infringed material, it could place the actual business owners and administrators at risk of legal concerns for allowing employees the ability to participate in illegal activity.

Malicious Software

One of the largest threats related to downloading software off the FastTrack network is the likelihood of encountering and possibly running malicious software. This is a software category that includes viruses, Trojans, worms, adware, and spyware. This software could have various impacts on the client computer. It could simply install a virus onto the computer, institute a backdoor to allow remote access by evildoers, or even alter the configuration of the client software to share out the entire hard drive onto the network. For a variety of reasons, this software continues to flourish on the FastTrack network, along with other peer-to-peer networks. For one, users have come to expect that the same file will come in multiple sizes and different filenames. This is especially true with audio, since each file may be encoded at a different bit rate. It's also expected with applications and documents, because there is no means to verify the authenticity of a file before it is completely downloaded. This is another problem with downloading software—the fact that you cannot be sure what you've downloaded until the file is completely downloaded. This is not as big a problem

with audio and movies, which have indexed content that allows a user to be able to play portions that have been downloaded. However, with executable files, ZIP files, and Word documents, this might not be possible.

One of the greatest reasons for the spread of malicious software, though, can be found in the use of FastTrack clients. When downloading files, the common user expects that the file will not be immediately downloaded and could take some hours, even days, to complete. For this reason, many users select multiple files to download and simply minimize the application for long periods of time. That means that a file could be completely downloaded and sitting unchecked on a user's computer for a long while. Normally, this wouldn't pose any threat at all, except for the fact that, by default, FastTrack clients immediately start sharing out files that they have downloaded. So, before a user has checked for viruses in the files they've downloaded, the user is already sharing those files out to other members of the network. Another factor is that when a FastTrack client starts transmitting a file to another client, it places a write-lock onto the file, making it impossible to delete or move the file. The upload connections would have to be canceled prior to removing the file, which can be difficult because some clients will aggressively attempt to reconnect.

Finally, how many users actually have a functional antivirus solution on their computer? An antivirus program that has been regularly updated performs routine scans of the entire hard drive and monitors new files as they're created. Such a thing is still relatively uncommon. For many users already using FastTrack, if they wanted an antivirus application, they'd just download it using Kazaa—and they wouldn't know that the antivirus software they just downloaded was itself infected.

Fake Files

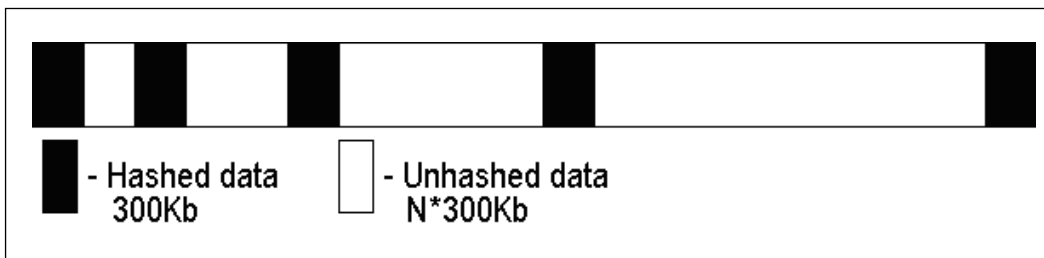
One of the growing problems on the FastTrack network lately has been the occurrence of fake files, also known as “poisoned” files. These are files that are seen as authentic files, which allows the network to let them be traded alongside perfectly normal audio and movie files. However, instead of playing an audio file, they normally contain a looped sample of the song or just pure silence. This process is done by modifying a file to overcome a very weak hashing algorithm used by FastTrack clients.

In normal situations, a hash is created to gain a virtual fingerprint of a file's content. Popular hashing algorithms include CRC, MD5, SHA1, and SHA256. In each, the data of a file is calculated and a set-length hexadecimal string is output as the hash value of the file. If any single bit within the file changes, a future hash will greatly differ from the original. For this reason, it is a popular means by which files can be authenticated and verified after they've been transferred or copied between devices.

However, the hashing solution implemented into FastTrack is one that is not meant for validating data. Rather, it was designed to allow clients to choose among multiple file sharers, where the data from each sharer has the same hash value. Because a full-length hash of a data file could take long periods of time, approximately 30 seconds for a 700MB file, a partial hash is created from the shared file in most circumstances. In performing a hash of the data, if the file is less than 300Kb in size, an MD5 hash of the entire file is created and is used for sharing the file. If the file is greater than 300Kb, a partial hash is created by first hashing the first 300Kb of the file using MD5. The hashing process then skips a set amount of data, calculated as the offset times 300Kb ($N \times 300\text{Kb}$). After each block of data is hashed, the offset undergoes a shift left, which is the same as multiplying it by two. The offset refers to the byte location in the file at which reading will begin.

For example, with a very large file, the offset begins at the first character, and the first 300Kb of the file will be initially included in the hash. The next 300Kb block will be ignored, but the following one will be included. Then a 600Kb block will be ignored, followed by a hashed 300Kb. Then a 1200Kb block will be ignored, followed by a 300Kb block. The ignored blocks will grow exponentially from 300Kb to 600Kb, 1200Kb, 2400Kb, 4800Kb, 9600Kb, and onward. For a graphical representation of this model, refer to Figure 12.7. In a typical MP3 that ranges around 3000Kb in size, only 900Kb of the file will be used for hash comparison. This means that 2100Kb of the file can be filled with corrupted data that will not be noticed until the download has completed. With full-length movies, around 600Mb in size, the hashed portion accounts for only about 3.5Mb of the data. This allows a large file to be hashed in mere milliseconds, but it also allows the file to be virtually filled with unwanted data.

Figure 12.7 FastTrack Data Hashing



Sharing

File sharing is the keystone of the entire FastTrack network and many peer-to-peer networks. If you download a file, you are generally expected to also offer up files in return, even if it's the same one that you just downloaded. It is through sharing that networks stay large and busy, as thousands of new users join at a regular rate and start performing searches.

At the heart of the sharing problem is the sensitivity of the data that could potentially be shared across the network. An improperly configured client or one that has had malicious software installed could allow an entire hard drive to be shared out onto the FastTrack network. This would allow a simple filename search to uncover documents that could be sensitive to your family or to your business. At home, many people store sensitive data in regular Word documents so that retrieval is convenient. This information could include phone numbers, bank account numbers, credit card information, and even Social Security numbers. For a computer located on a corporate network, the dangers are even greater because the client is indiscriminate of the data it shares. This includes Word documents, PDF documents, source code, Outlook e-mail personal folders (PST files), CAD designs, spreadsheets, and databases. A careless employee could, for example, end up leaking design specifications for a future product well before its announcement date.

Legal Threats

A topic that has been gaining much worldwide attention lately is the subject of legal notices sent to actual users of peer-to-peer networks, especially the FastTrack network, regarding copyright infringement. The RIAA has taken an aggressive stance in recent years to curb the amount of copyright infringement on the Internet. To date, well over 10,000 file sharers have been served with papers over civil charges resulting from their sharing of copyrighted material on the Internet, with the average settlement amount ranging from \$3,000 to \$4,000 in U.S. dollars. The charges have been filed against a wide range of personalities, from a 12-year-old honor student in New York to a 71-year-old grandfather in Texas—even a Yale University professor. Some users were downloading and sharing hundreds of popular songs; others were downloading as few as five songs in total. The vast majority of these cases ended with the consumer paying hefty settlement fines. However, in September 2005, some consumers started to fight the lawsuits. At this time, the cases are still in litigation.

The issue of legality is still a gray area with peer-to-peer networks, based largely on regional boundaries. The aggressive lawsuits are currently found only within the United States. Today, the downloading of copyright music is legal in many countries, even Canada, as long as the music is not redistributed or sold.

Vulnerabilities

The FastTrack networks business model is based on licensing portions of code that allow a client to be able to connect to the network. However, the client software itself is designed as whatever its creators want to make it. With a wide variety of client software available, generated by small companies or single programmers, the testing and validation procedures are not as rigorous as for a more professional product. This opens up the possibility of exploits and vulnerabilities within the clients themselves. To date, there have been multiple exploits against FastTrack clients, including a denial-of-service (DoS) attack that makes connections to a Kazaa client and utilizes all available bandwidth. Later exploits were found that could crash clients acting as supernodes.

The problem has grown even larger with the great number of “modified” FastTrack clients that have appeared over the years, such as Kazaa Lite and K-Lite. These clients employ techniques to modify existing client software in ways that it was not intended to be used, which has the potential to create resource and security issues on a local computer.

Bandwidth Issues and Mitigation Steps

In this section of the book, we cover how to reduce or eliminate the amount of bandwidth used by FastTrack clients such as Kazaa and Grokster. Many of the examples for bandwidth management contained here are based on the Linux operating system and applications produced for Linux. However, the basic rules are explained so that they could be implemented, if possible, within a Windows network implementation.

For networks where the use of a FastTrack client is acceptable, many instances of bandwidth control can be instituted simply by making basic changes within the client’s options. Additional controls could be made by written user policies that are continually enforced. For example, it is recommended that a FastTrack client be disabled from starting upon a computer’s boot-up. This prevents the software from running in the background, continuously sending out data, without the computer user’s awareness. To coincide with this, it is also recommended that users be taught how to properly shut down a FastTrack client. When closed, many clients merely hide within the system tray of Windows, continuing to run in the background. Many typical computer users would be completely unaware that the client is still running in this situation. To properly shut down the client, the user must select its icon in the system tray with a right-mouse button click and select the option to Exit or Shutdown.

Supernode Clients

Besides the actual transfer of data, the most bandwidth-intensive portion of the FastTrack network is the data that are sent between supernodes. These data, although bearing the overhead of encryption, contain search requests for any of the 3 million users on the FastTrack network. Sharman Networks claims that a Kazaa supernode will use less than 10 percent of bandwidth and hardware resources of a client, but it could still place a large strain on many networks. What's worse is that almost all FastTrack clients are optioned by default to allow the computer to automatically become a supernode, without any notice or warning to the computer's user. In network situations where a FastTrack client is acceptable, it is highly recommended that the supernode capability be disabled within the client configuration. Many large networks, such as those used by universities and ISPs, can and will block Internet access to clients that produce too much bandwidth on the network. Many acceptable-use policies (AUPs) specify that if a FastTrack client is installed, the supernode option is to be disabled. This option is found under the Options section of each client and is generally relegated to an Advanced tab that most users ignore.

Firewall Rules

Using a variety of the static values within FastTrack packets, blocking access to them is not very difficult. Many clients still use port 1214 for sending and receiving data, though others use more exotic ports. The client could also be configured to use a port that the user manually assigns. In instances throughout this chapter, my FastTrack client was set to use port 1485 for sending and receiving data. Barring any actual use of these ports by authorized programs within your network, a simple block on such ports may prevent most FastTrack clients from being able to connect to the network. With the Linux-based *iptables* firewall, this can be done by dropping any TCP and UDP packet that is destined for or has come from those ports. For example:

```
iptables -A OUTPUT -p tcp --dport 1214 -j DROP
iptables -A OUTPUT -p udp --dport 1214 -j DROP
iptables -A INPUT -p tcp --sport 1214 -j DROP
iptables -A INPUT -p udp --sport 1214 -j DROP
```

However, the problem with this approach is that modern FastTrack clients are no longer using fixed ports for communication. This method might not always work, since current versions of Kazaa will attempt to use hundreds of ports to establish a file transfer connection if they detect that 1214 has been blocked. It might also be

possible to search for strings that are contained within each packet by the use of a network-based intrusion detection system (IDS), but that will only alert to its presence, not stop it. Searching for strings is also possible with *iptables*, but that requires a module that is not normally installed by default, since it is considered experimental: the string match module.

IPTables String Match Module

Though considered experimental, the string match module is a very powerful tool that can help sculpt efficient firewall rules. With this module installed, you can set more exotic firewall rules that do not need to be based on a port. Instead, each packet can be delved into to determine whether it is unwanted traffic, not just traffic flowing on an unwanted port. For example:

```
iptables -A INPUT -p tcp -m string --string "X-KazaaTag" -j DROP
iptables -A INPUT -p tcp -m string --string "X-Kazaa-SupernodeIP" -j DROP
```

Combinations can even be made to limit the use of Kazaa to certain types of data, thus avoiding more bandwidth-intensive material such as movies and audio. This could be done by combining a string that is found in all FastTrack packets, such as *X-KazaaTag* and the HTTP *Content-Type* field. However, as powerful as the module might be, it is not installed by default in many Linux distributions. If it isn't, it will have to be downloaded and installed as a kernel patch, which requires recompiling the operating system kernel. To install the string match module, download the latest version of *iptables* from www.netfilter.org. With it uncompressed, you can install the string match by first running Patch-O-Matic by typing the following command lines. Ensure that the *KERNEL_DIR* variable is pointing to your pre-established kernel source code directory:

```
make pending-patches KERNEL_DIR=/usr/src/linux
make patch-o-matic KERNEL_DIR=/usr/src/linux
```

This will begin an interactive install where you can specify to install the string patch, by Emmanuel Roger. With the patches in place, *iptables* must then be installed by typing:

```
make KERNEL_DIR=/usr/src/linux
make install KERNEL_DIR=/usr/src/linux
```

With *iptables* installed and the required kernel patches in place, the Linux kernel must then be set up and installed. This is done by changing to the kernel source directory, generally */usr/src/linux*, and running a configuration program. You can

then pick your poison by running either *make xconfig* or *make menuconfig*. Once in the kernel configuration, select *IP: Netfilter Configuration* and locate the entry for *String match support (EXPERIMENTAL)*. Select this to be installed as a module by selecting the *M* for this entry. At this point, back your way out, and make any other changes you would normally have for your specific kernel configurations. On completion, save your results and return to the Linux command line. Install the new kernel using any method that you know or use the following command lines:

```
make dep
make bzImage
make modules && make modules_install
make install
```

When the install has completed, reboot your computer and the new kernel should be loaded into memory. With the kernel installed, you will be able to set string-based firewall rules.

P2PWall

To counter the ever-growing considerations with FastTrack traffic, P2PWall was released in July 2003. P2PWall is a collection of tools that are free and released as open source under the GNU General Public License (GPL). P2PWall can be downloaded from <http://p2pwall.sourceforge.net>. Its first tool released, FTwall, short for FastTrack Traffic Firewall, is also used to block FastTrack traffic. Unlike a normal string match firewall rule, which will not stop connections to a supernode, FTwall can stop any and all connections to the FastTrack network. FTwall works by interacting with an existing *iptables* installation on a Linux-based computer, which is configured to pass potential offending packets to a queue. The FTwall program then reads the packets from the queue and determines whether they are actual packets for the FastTrack network. Any packet destined for a supernode within the FastTrack network is dropped before it has a chance to leave the local network.

FTwall's initial ability is to scan all FastTrack UDP connection packets from internal client workstations that are trying to connect to supernodes. The IP addresses of each supernode are logged, and a spoofed packet is sent back to the client workstation. The spoof is important because it leads the client software to believe that it is not being blocked by a firewall. It then continues to connect to every supernode that it has listed, and each will be caught and logged by FTwall. The *iptables* rule used to establish this first phase of protection, assuming that the network interface used is *eth0*, is:

```
iptables -A FORWARD -p udp -i eth0 -j QUEUE
```

For those of you unfamiliar with *iptables*, this rule uses the *-A* option to append the rule to existing sets. The word *FORWARD* designates the *iptables* “chain,” which is the channel used for the connection. The *FORWARD* chain is used because this is traffic that will be forwarded through the firewall to another network. Other possible chains are *INPUT* and *OUTPUT*, which respectively refer to packets destined for the firewall and packets that the firewall initially sends out. The *-p* option refers to the protocol to scan for, such as TCP, UDP, ICMP, and so on. The *-i* option refers to the network interface that the packet is expected to be received on. Finally, the *-j* option specifies where the packet should “jump” to if it matches. A *-j DROP* tells the firewall to drop any packet that matches this rule. *-j QUEUE* tells the firewall to store the packet into a queue.

After this first phase of connection attempts, the client then attempts to contact each supernode requesting more information with an additional UDP packet, as well as attempting to make a TCP connection to the supernode. Using the list of IP addresses gathered during the first phase, FTwall can detect these connection attempts and promptly drop them from the network. This second phase is established in *iptables*, again assuming the interface to be *eth0*, by typing:

```
iptables -A FORWARD -p tcp -i eth0 --syn -j QUEUE
```

Although this might seem sufficient to stop most applications, some FastTrack clients have a few more tricks up their sleeves. They will withhold a set number of supernode IP addresses during the first two phases. If a connection could not be established in those phases, it will then use these hidden IP addresses with TCP encrypted packets to create a connection. In basic theory, FTwall would be unable to block these packets, because it is unaware that these are IP addresses of supernodes, since it did not see initial UDP packets for them. To get around this feature, FTwall allows a configurable duration where all packets from a particular client will be ignored after detecting TCP supernode connections, until the client software is shut down. By default, the time delay for ignoring packets is two minutes. Be aware that this will block all TCP connections from that particular workstation, even valid traffic such as Web traffic.

However, blocking packets in such a way was not foolproof, and FastTrack clients eventually found a way to bypass it. The client would continue trying to connect to a supernode but at a very slow rate and using only one IP address at a time. If this connection is made to an IP address that has been held back from the beginning, it will immediately bypass FTwall. To battle this attempt, FTwall will continually detect whether the FastTrack software is still running on the client computer. It does so by sending a spoofed UDP to the client computer, which prompts a

response to the firewall. FTwall listens for this response, and if it's received, continues blocking TCP packets originating from the client. This option is enabled on the firewall using the following *iptables* command:

```
iptables -A INPUT -p udp -i eth0 -j QUEUE
```

Note here that the *INPUT* chain is used rather than the *FORWARD* chain. That is because the firewall is the intended recipient of the packet, since it sent out the original spoofed UDP packet.

Snort IDS Rules

Snort is the most common IDS used today. A free, open-source application that is available for Linux, Windows, and Mac OS X; it can be found at www.snort.org. Snort can be implemented into a network design to provide early detection of unauthorized traffic, giving administrators the ability to curtail it before it gets out of hand. Snort rules for most common applications can be found across the Internet, with some provided here for integration into existing Snort sensors. These rules are based on the previous protocol analysis:

```
alert udp $HOME_NET any -> $EXTERNAL_NET any (content:"|27 00 00 00 a9 80
4b 61 5a 61 41 00|"; classtype:bad-unknown; msg:"FastTrack/Kazaa Supernode
Ping");

alert tcp $HOME_NET 1024:65535 -> $EXTERNAL_NET any (content:"GET /.hash=";
nocase; flow:from client; classtype:bad-unknown; msg:"FastTrack/Kazaa File
Request");

alert tcp $HOME_NET 1024:65535 -> $EXTERNAL_NET any (content:"X-Kazaa-
SupernodeIP"; nocase; flow:from client; classtype:bad-unknown;
msg:"FastTrack/Kazaa File Request");
```

Summary

In this chapter, we examined some of the basic features and aspects of the FastTrack peer-to-peer network. We started with an overview of the FastTrack network and its origins on the Internet. The FastTrack network was begun in early 2001 to allow the sharing of all types of files across the Internet. We went into the history of the major clients that are used to access the network, such as Kazaa, Grokster, and iMesh. Along with the history of each client, we reviewed some of the many unauthorized, alternative clients that could be used to access the FastTrack network. For the most part, these clients were introduced because of the amount of adware, spyware, and untrustworthy software found in the official clients.

We then described the basic design of the FastTrack network, a hybrid design that took the best aspects of centralized and decentralized networks. The network was centralized enough, using supernodes, to provide a backbone to the network for quick searches among millions of users. It was also decentralized to the point where the supernodes were not under control of the network operators. These supernodes were regular clients on the network that were automatically upgraded to be parts of the backbone, handling connections and searches from dozens of clients. Delving into the actual network traffic used by much of the network, we were able to identify the initial connections that were made to allow a client access to the network. Unfortunately, most of the network traffic is encrypted and proprietary, so many of the packets are difficult to identify. All file searches and index queries are performed using these encrypted packets. However, the actual file transfers are sent in cleartext using standard HTTP requests.

We demonstrated some of the inherent features of clients as well as the problems that they may pose on a workstation or an entire network. This could be due to the transfer of malicious files or the great strain that the client could place on the bandwidth available to a network. The ease of sharing files could also place sensitive data in danger of being spread widely throughout the network. Many of these issues can be handled with basic firewall rules, of which a variety were discussed, particularly for the Linux *iptables* firewall software. Rules to block clients from connecting to the network were demonstrated, as were some of the more advanced means for clients to connect and ways in which they can be blocked.

Solutions Fast Track

History of Clients and Networks

- ☑ FastTrack, a proprietary P2P network, was released in 2001 and supports only a handful of clients that license technology from the owners.
- ☑ Kazaa is one of the most common licensed clients for the FastTrack network, along with Grokster and iMesh.
- ☑ Kazaa has faced many legal battles through its existence, with some of the final blows determining that Kazaa is responsible for the content that flows across their network.

Spyware Bundling and Alternative Clients

- ☑ All licensed FastTrack clients offer a “free” version of their client software, which is financially supported by including adware and spyware applications.
- ☑ Adware and spyware are third-party applications forced onto a computer to display continuous advertisements and to collect personal, identifying information about the computer or use to send to advertisement agencies.
- ☑ Kazaa Lite was the first program to appear to load the Kazaa client without loading its supposedly “required” adware applications.
- ☑ Kazaa Lite was eventually replaced in the market by Kazaa Lite Resurrection and K-Lite.
- ☑ A “Lite” client works by using a loader utility to alter the original software after it loads in memory.
- ☑ Most modern “Lite” clients also include a variety of external utilities to provide additional services on the FastTrack network.

Network Architecture

- ☑ FastTrack is a semicentralized network design, a hybrid of centralized and decentralized systems.

- ☑ The FastTrack network still integrates centralized servers throughout the network in the form of supernodes.
- ☑ Supernodes are common client computers that have been given a higher role in the network, sometimes without the operator's knowledge.
- ☑ Supernodes handle connections from a plethora of clients, including file indexes for each.

Protocol Analysis

- ☑ Client negotiates with supernodes by sending and receiving specific UDP packets.
- ☑ File searches and all supernode communications are encrypted and undecipherable.
- ☑ File transfers are performed using standard HTTP requests.

Features and Related Security Risks

- ☑ Because their primary role is downloading files from unregulated networks, FastTrack clients can download malicious software, fake files, or illegal content.
- ☑ The sharing ability within a client could cause company documents and private material to be distributed across the Internet.
- ☑ The widespread development of many clients and client modifications could make FastTrack clients susceptible to future exploits.

Bandwidth Issues and Mitigation Steps

- ☑ A supernode could use less than 10 percent of bandwidth and hardware resources of a client, but it could still place a large strain on many networks. Almost all FastTrack clients are optioned by default to allow the computer to automatically become a supernode, without any notice or warning to the computer's user. In network situations where a FastTrack client is acceptable, it is highly recommended that the supernode capability be disabled within the client configuration.

- ☑ Port-blocking firewall rules could be initiated, but they are easily defeated by modern FastTrack clients.
- ☑ With appropriate software, it is possible to filter for packets with specific text contained within them.
- ☑ Special software, such as P2PWall, can be implemented to specifically defend against FastTrack clients.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q: What is the Kazaa Participation Level?

A: The Participation Level is a number that signifies a client’s upload-to-download ratio. Ranging from 0 to 1000, this ratio represents how much more data you share than download. It is only used for the number of Search Mores a client can do, and it is generally useless because customized Kazaa clients automatically set it to 1000, the highest level.

Q: How do I disable Kazaa from automatically loading on a Windows restart?

A: When Kazaa is configured to automatically launch, it adds a registry key labeled `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Kazaa`. This key includes the path to the client executable, such as `C:\Program Files\Kazaa\kazaa.exe /SYSTRAY`. Removing this key will prevent Kazaa from automatically launching.

Q: How do I prevent a computer from becoming a supernode?

A: Besides blocking supernode related packets from connecting a computer, there is also a setting within FastTrack clients that allow users to opt-out of becoming a supernode. With Kazaa, this value is stored in a registry key labeled `HKEY_CURRENT_USER\Software\Kazaa\Advanced`. The value is named *SuperNode*, and is set to 0 to indicate that the computer is eligible to become a supernode. Simply set this value to 1 to disable the computer from becoming a supernode.

Q: How anonymous is a computer on the FastTrack network?

A: A computer is not anonymous at all on the network unless it refrains from uploading and downloading data. As soon as a download is initiated, the file's sharer will have details of a computer's IP address, which they can use for their own purposes. Likewise, if a client is sharing out data, anyone who attempts to download that data will be able to retrieve the sharer's IP address. Besides offensive procedures being applied against the IP, it can be traced back to an Internet service provider, which can turn over the account's owner under a subpoena. However, a Bad IP Blocker in many customized clients can block connections to known networks that hunt for sharers of infringed material.

Q: Is there a way to avoid downloading fake files from FastTrack?

A: The occurrence of fake files has become a common sight on the FastTrack network. To effectively prevent downloading them, some clients offer the ability to analyze search results to determine if some are fake files. This is done by comparing the sharing computers against a list of blocked IP addresses and marking the results that match.

Q: Is it possible to proxy a FastTrack client connection?

A: Yes, FastTrack clients support the use of a SOCKS5 proxy connection.

Part III

Internet Relay Chat Networks

Internet Relay Chat—Major Players of IRC

Solutions in this chapter:

- History
- IRC Server Software Packages
- Major Networks

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

In this chapter we will explore the world of Internet Relay Chat, also known as IRC. IRC is a multi-faceted entity, geographically and technically spanning a wide area with many differences between networks, servers, and clients. Here, we will outline the differences between the major networks, and a little about the history of IRC in general.

Notes from the Underground...

Underground IRC

So, what exactly is IRC? To summarize, IRC is a text-based chat medium. It provides facilities for many users to chat together on a network of interconnected servers, which share common information. There are many thousands of IRC networks, ranging in size from tens of thousands of users to under ten users, and between fifty servers and just one. A wide variety of topics are discussed and of course, IRC has a secretive underworld, as does most of the Internet, where discussions about illegal activities occur, and illegal activities themselves (such as information theft) are carried out directly over IRC. This section of the book will document a large number of these issues and help you to be aware of them.

History

IRC was originally the brainchild of Jarkko Oikarinen. Wanting a better replacement for the UNIX *talk* program, with the ability to network servers together and hold group conferences (talk was limited to only two users at a time) he set to work in 1989 in creating the first IRC server and IRC client. Months later, this new system was released and users of the University of Oulu computer centre started to make use of it. Pretty soon, due to uptake of the Internet, this new program had spread across Scandinavia, and then to the United States.

As with all open source projects, it did not take long before others took up the baton and started to develop their own extensions and produce patches for the IRC server. Time passed by, and a second version of IRC was released. The first version had limitations that many wished to overcome, such as:

- Version 1 of IRC only allowed channels to be created that were numbered, for example “JOIN 1.”
- Version 1 of the program had no support for many features we take for granted (secret flags on channels, invite only channels, etcetera).

Various other features are now lost in the depths of time. Unfortunately, the first few versions of the IRC server software were not documented and very few if any copies remain (essentially, anything before version 2.6 is considered lost). One such feature is a built-in game of *Hunt the Wumpus* (a multi-player role-playing game), which was played by users on IRC. This is still available (download an old version of the IRC server from <ftp://ftp.funet.fi> to see), but no longer runs on modern IRC servers. In 1993, the second version of the IRC software (known in IRC circles as *irc2*) was released and became the defacto standard. In fact, the majority of IRC servers (also known as *ircd*, or IRC daemons) that are used today are based directly upon this branch of the original code. The RFC (request for comment) documents (RFC 1459, 2811, 2812, 2813) are directly based upon this current version of the protocol pioneered by version two of IRC. IRC server software is still actively developed today, and there are many branches of the initial program created by Jarkko Oikarinen. Each of these branches has its own quirks and design goals; some are superior for running a secure network than others. All IRC servers (known also as IRC daemons) must follow the RFC documents listed above to be compatible with IRC clients (such as *mIRC*, available at www.mirc.com), however some IRC servers bend or break the specification to add features or improve security and performance. The sections and chapters to follow will discuss the common types of IRC server software, and the types of security features they provide.

IRC networks have changed over time, moving from initially one network to the many thousands that exist today. Originally there was only the IRC network run by a committee that included Jarkko Oikarinen. Over time, as IRC grew, fractured and large networks such as FEFnet were born (which later fractured into other networks such as IRCNet, EFnet and AnarchyNet). The lineage of IRC networks, if documented properly, would easily consume a whole book on its own, and for the most part is past the scope of this chapter or even this book. It is the very nature of IRC and networks to fracture, split, and merge, establishing new ideas and new ways of looking at problems, and this is likely to continue for many years to come.

IRC Jargon

IRC is very popular on the Internet, and therefore over time it has developed a large amount of jargon terms. These jargon terms are mostly unique to IRC, and without

some understanding of these terms it is very difficult to understand IRC as a whole. The common terms are discussed below.

Nick

The word *nick* is used to refer to a user's nickname on an IRC network. Each user has a single nickname, which uniquely identifies that user upon that network (for example, there can be no duplication of nicknames). Nicknames are not persistent, so in most cases, if a client disconnects from IRC another client may connect and immediately use the same nick.

Ident or Username

On IRC, the word *ident* or *username* refers to the actual username that you use on your local machine to log in. Historically, IRC was relatively small (two or three servers within one university) and in cases of abuse or just wanting to know information on a user, these usernames were useful. In the modern day, these usernames (idents) are not as useful as they once were. Each user has an ident value along with a nickname, however many users may have the same ident or username, whereas they may not have the same nickname. These usernames may be verified by a special server called an *identd* (RFC 1413 contains details on how this works). Refer to Chapter 14 for a discussion of the security implications of running an ident server.

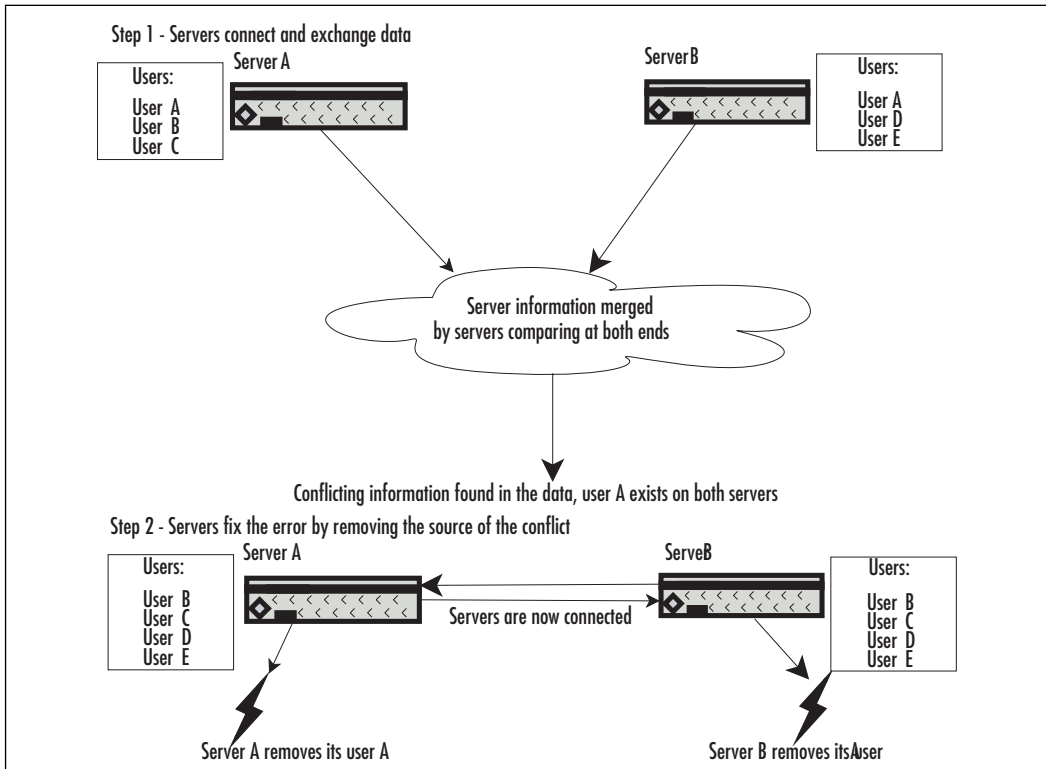
Channel Operator

An IRC network consists of many chat rooms known as *channels*. Each of these channels contains one or more users, and authority is delegated to users in this channel to manage it how they see fit. The first person into the channel (the person who essentially *creates* it) is granted a privilege known as *channel operator status*, or *Ops* for short. This status gives this user the ability to change the channel's settings (known as modes) and kick and ban users from the channel. This user can then give this status to others, each of whom can then in turn give or take this privilege from each other or other new users. People will and do fight for these privileges more than any other privilege on IRC, making this a serious security concern if you wish to use IRC for any serious purpose. Botnets (groups of programs designed to perform one common task) are often designed in such a way to facilitate stealing of channel operator status from channels, either by causing servers to disconnect from each other (see below to find why this works) or by directly causing a DDoS (Distributed Denial of Service) to a user who already has channel operator privileges, causing them to exit the IRC network and leave the IRC channel defenseless.

Nick Delay and Time Stamps

Time stamps and *nick delay* are systems whereby servers can determine which information is correct when two servers merge. Because the concept of two servers merging is a comparison between two different sets of data, conflicts in the data can occur. The connection of two groups of servers, known to IRC users as *sync*, involves both servers sending their entire state tables to each other. When this occurs, originally, both IRC servers would deal with conflicting information by removing both sources of conflict. For example, if two servers connected and both servers had the same user, both servers would disconnect their local instance of that user. This is known as a race condition, and is inefficient and intrusive. An example of this is shown in Figure 13.1.

Figure 13.1 Race Condition



Malicious users on IRC would (and still do) attempt to create these race conditions with the intention of changing their status information (for example, becoming a channel operator on a channel where they should not be) or with the intention of

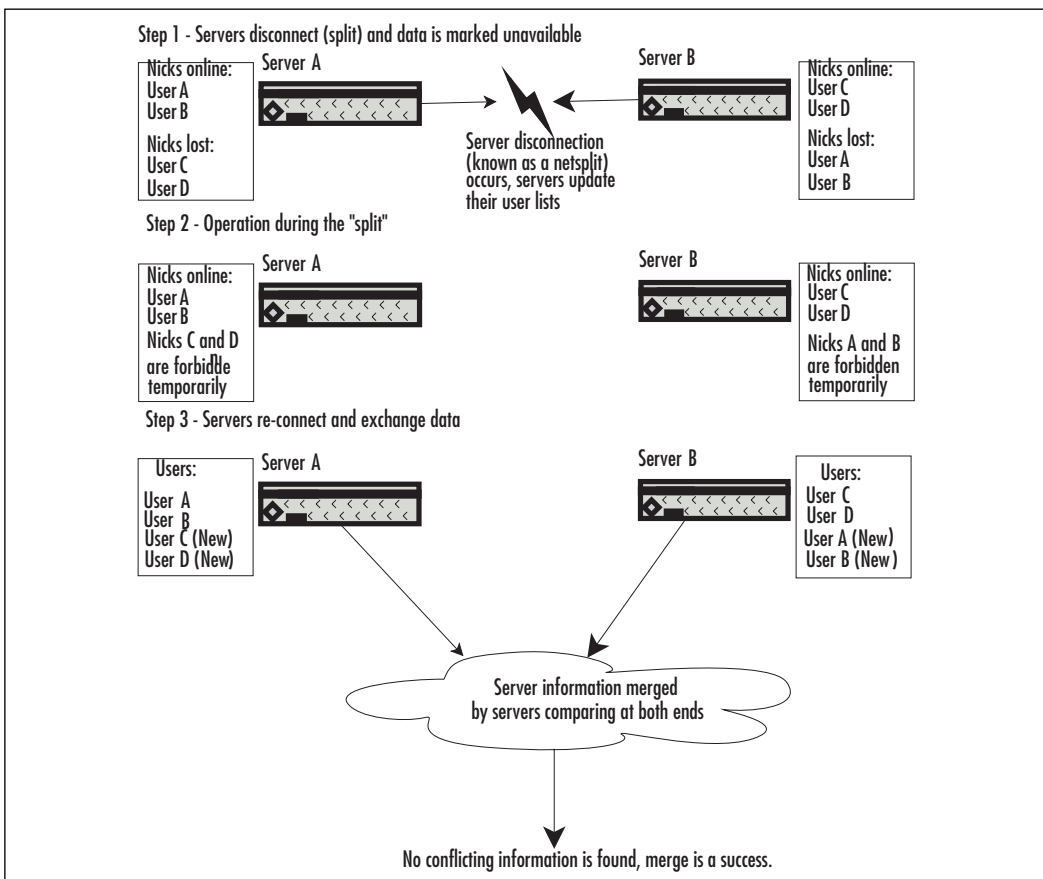
stealing another user's nickname or disconnecting him or her, so that they can simply take their channels from them.

Many discussions were held as to how to detect these conditions (known as collisions) and prevent them. The two methods that became the standards, nick delay and time stamps, work in completely different ways and are explained below.

Nick Delay

When two servers separate, all information that was part of the *lost* server's information set is marked "unavailable". Attempts to make use of this unavailable information are denied for a preset amount of time, usually ninety minutes. This is applied to both channels and nicknames, and in theory prevents conflicting information from entering the network when the servers re-establish their connections, as shown in Figure 13.2.

Figure 13.2 Nick Delay

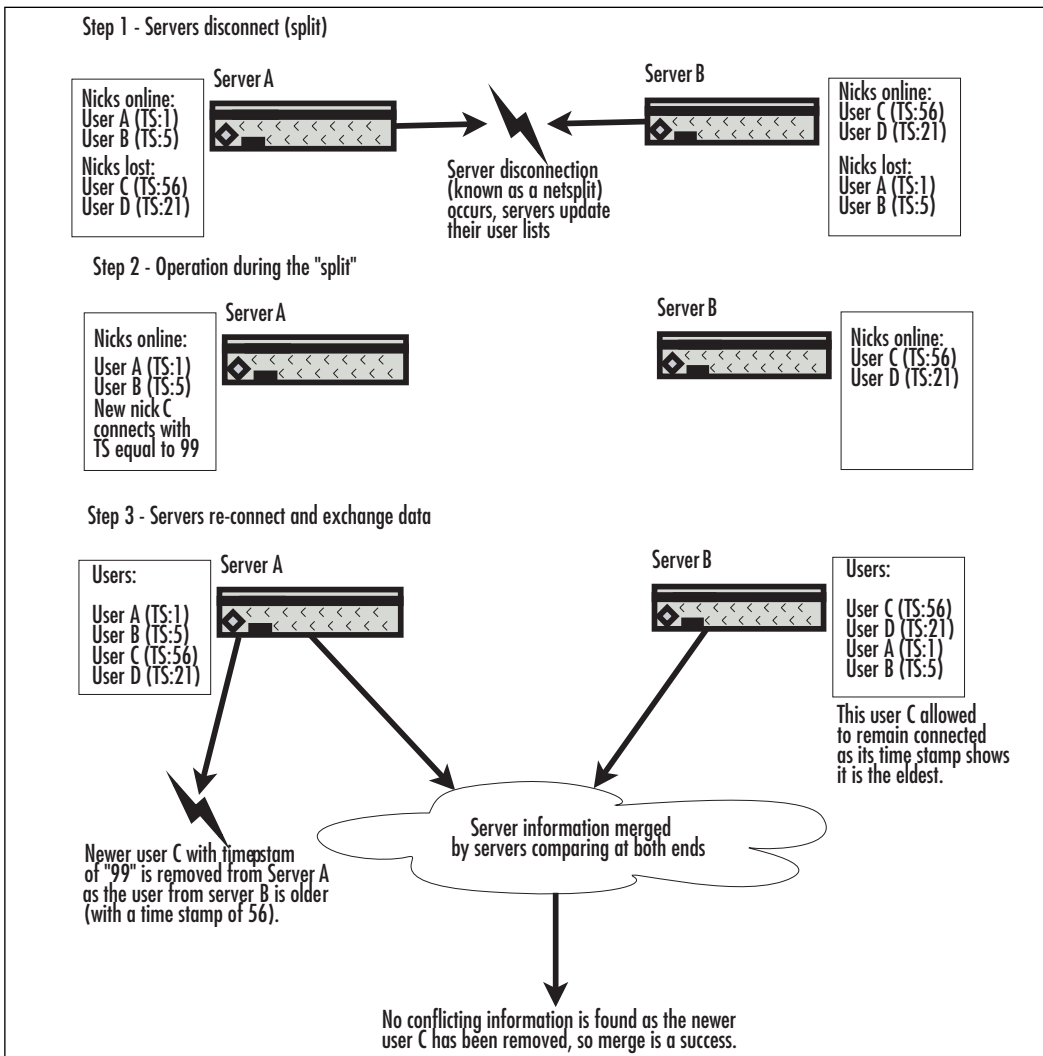


As shown in Figure 13.2, in practice, nick delay works fine. However, if the two servers in the figure above were to remain disconnected until beyond the ninety minute threshold, chaos will ensue as people took back their old nicknames, joined channels that were marked unavailable, and otherwise caused conflicts in the data stored on both servers. This is important to know from a security standpoint, as these issues *will* affect any channel you set up on IRC for any purpose (professional or otherwise) if you choose a network that uses this system (for example IRCnet).

Time Stamps

In the time stamp system, each object (for example a client, a channel, or a channel topic) has a time stamp (known usually by its shortened jargon term, TS). When two servers merge, the timestamps of any *conflicting* items are compared. Instead of removing both conflicting objects, the older of the two items is kept while the other is removed. This makes perfect sense, as the older item was clearly there first, and has the right to remain, while the other item is possibly an abuser trying to just cause a disruption. This is best explained with a diagram as detailed in Figure 13.3.

Figure 13.3 Time Stamps



IRC Server Software Packages

The following section discussed many types of IRC server software packages that can be used to run an IRC network. They are explored from the point of security and functionality, allowing you to easily judge their merits and downfalls. Because most corporate users who make use of IRC in any form will prefer to *set up their own* IRC servers, this section will be invaluable to you when it comes to making a decision about which software you will choose for the task if you decide to take this route.

ircd 2.11.x

ircd v2.11.x is available from <ftp://ftp.irc.org/>, and is the current version of the original IRC server by Jarkko Oikarinen. The development of this software has been taken over by other programmers, and it is used mainly by the IRCNet network (see the “Major Networks” section below). The developers of this software prefer to stick to the standards (in fact, they are even responsible for documenting and creating the current standards in RFC 2811, 2812, and 2813) rather than experiment and bring about changes, for security purposes or otherwise. The IRCnet software is one of the few pieces of IRC software to make use of the nick delay feature. Note that this IRC server’s use of nick delay to resolve race conditions is slightly and it provides very little in the way of security features, being one of the most basic IRC servers you can install. It is, however, simple to administer in comparison with the other software listed in this section, which in itself can improve security, as there are fewer issues to monitor for problems.

ircd-hybrid

ircd-hybrid, available from www.ircd-hybrid.com, is a branch of the ircd 2.11.x software used by EFnet. EFnet, unlike many networks, use many different IRC daemons, which are compatible with each other, even though they contain different code. As with ircd 2.11.x, the hybrid IRC server attempts to stay with the standards rather than implementing new features. It is maintained by many EFnet staff and committee members. This IRC server software was the first to pioneer the concept of time stamps, which makes it more secure than the IRCnet server software. However, there are no specific security-related commands beyond what is provided as standard (for example, the KLINE command, to ban a user).

bahamut

bahamut is the IRC software used by the DALnet network and is a high-performance IRC daemon designed to hold a large number of users efficiently. It attempts to add many behind-the-scenes features to assist operators and users, such as features that allow services (virtual servers that allow ownership of nicknames and channels) and operator features such as *akills* (network-wide bans to keep troublemakers out). The bahamut software is actively maintained by a large group of individuals and is documented at <http://bahamut.dal.net>. bahamut was designed to replace the now deprecated and no longer available dreamforge software. From a security standpoint, bahamut is a secure piece of software which provides many features for keeping troublemakers from a network and controlling their behaviour. Most of these fea-

tures only become usable by network administrators once an “IRC Services” package is coupled with the server (please see the following chapter for detailed description of what IRC services do), after which, this software provides many facilities for control of nicknames, channels and various other server resources, such as the ability to configure how many connections each user may have network-wide (known as sessions).

ircu (and Derivatives)

The ircu software used on Undernet is designed with security and stability in mind. Designed to hold very large amounts of users and generally to conceal information from those who do not need access to it, ircu is the Fort Knox of IRC software. Information that the RFC says should be displayed can be hidden from users in this software, following the security concept that users should only see information they need and nothing more. Whether this actually helps prevent abuse is unproven, as Undernet has a very large number of abusers, as with all IRC. The ircu software is available from <http://coder-com.undernet.org/> and is actively maintained by the Undernet coder community. Other large networks such as Quakenet also use a derivative of ircu known as *asuka*. As previously stated, this software aims for security foremost, so if you are intending to set up an IRC network for any purposes, this software is highly recommended. All the features listed above can be set in the extensive configuration files, which are too large to even summarize here; however you can find them online at <http://dev-com.b2irc.net/guide/ircd.conf>.

UnrealIRCd

UnrealIRCd is a very popular IRC daemon, available from www.unrealircd.org and designed with many nonstandard features. The goal of UnrealIRCd is to create an advanced IRC server (to add features which are not listed in the RFC, but are considered beneficial from a security point of view) such as spam filters, different styles of user bans, various channel modes to prevent abuse and flooding, SSL (Secure Sockets Layer) connection support, and compressed server connections. It is originally based on dreamforge, a now deprecated IRC server that was the predecessor to the actively maintained bahumut server.

As this server software possibly has the most security features of any IRC server, it makes sense to go into more detail here about what this server offers and some of the commands it provides:

- **Shun** Administrators running this software may prevent clients from talking on IRC at all by issuing the shun command, which blocks all outbound text from the user.
- **Spamfilter** Administrators may use this command to specify regular expressions (www.regular-expressions.info) that, when found in lines of IRC text, will cause actions to be taken (such as bans, shuns, and clients being disconnected). Because this software allows server side filtering of malicious messages, it can prevent many security issues from ever being a problem.
- **Dccdeny** Administrators may use this feature to block files of certain types being sent over IRC. As illegal files being transmitted over IRC can easily become a liability for the administrators dccdeny is a very powerful tool to have in your arsenal for use in securing an IRC network.

This list is only the tip of the iceberg of features available in the UnrealIRCd server. For more information it is highly recommended that you read the UnrealIRCd user guide at www.vulnscan.org/UnrealIRCd/unreal32docs.html.

Major Networks

There are many major networks, some of which are covered in the next chapter. Some of these have been around for since 1993, while others are relatively new. The searchirc website maintains lists of IRC networks and their rankings (<http://searchirc.com/network-size/70000>).

As of 2005, the top four networks are Quakenet, Undernet, IRCNet, and EFnet, which run the Asuka, ircu, ircd 2.11.x, and ratbox/hybrid software. At this writing, all four of these networks have 70,000 or more users. Next in the list are rizon (irc.rizon.net), DALnet (irc.dal.net), GameSurge (irc.gamesurge.net), and Freenode (irc.freenode.org). When deciding to use IRC for any purpose you have two choices; to set up your own network with all the associated costs of hosting and administration, or use one of these existing networks (some of which are discussed below) and put your trust into another organization to keep you secure. Knowledge of the popular IRC networks and their security implications can help you make a well informed decision about the route you take.

Quakenet

The Quakenet network (www.quakenet.org) is primarily a gaming network, which holds many channels intended for the discussion of computer games, their associated

groups, and discussion of game development as well as the actual playing of games. Recently, Quakenet has begun to allow registrations not related to games, showing an expansion into more generalized chatting. From a security standpoint, Quakenet is secure enough for most users. Quakenet does in fact host many IRC channels for small, medium, and large hosting companies who offer IRC- and gaming-related services, and therefore for them it is sensible to offer support on IRC itself.

Quakenet has a form of IRC services (Chapter 14), which allow you to keep channels and nicknames secured against unauthorized use, and the network can hide personally identifying information such as your IP (Internet Protocol) address from prying eyes.

Undernet, IRCnet, DALnet and EFnet

The Undernet, IRCnet, DALnet, and EFnet networks are discussed in detail in the next chapter, and so will not be covered in depth here. It is worth noting here, however, that these are the oldest and most established of the top ten networks.

Rizon

The rizon network (www.rizon.net) is similar in operation to EFnet, and holds a large number of file trading channels. File trading, and its associated risks are covered in the next chapter. Relatively new in relation to its peers, rizon has risen (bad pun and bad spelling intended) from nothing to the top ten in a relatively short period of time (a couple of years). It is *not* recommended that this IRC network be used for any professional reasons. It does offer security facilities such as IRC services, the ability to hide IP addresses, and time stamp-based servers, but because it attracts a large number of illegal and undesirable channels on topics such as software pirating, illegally copied music, hacking, and denial of service, it is simply an unsafe network on which to place any professional channel. *Chatters Beware!*

GameSurge

GameSurge (www.gamesurge.net) is also a games network, similar to Quakenet. Formed by a past merger of GamesNet and PGPn, it is primarily gaming users, and since its merger is the largest gaming network after Quakenet. It is one of the few networks to attempt to register itself properly as a non-profit organization. As with Quakenet, many larger hosting companies that offer IRC and gaming-related services have set up here knowing that such a move is sensible for their businesses. It is worth noting, however, that these companies set up on these IRC networks because it makes good business sense to offer support over IRC. In general, IRC is not a

good place to hold professional discussions, unless you run your own IRC servers, free from external influences. As with most of the major networks this network provides IRC services, but it does *not* provide any facility for hiding IP addresses from other users, which may put your corporate systems into the line of fire should you use this network.

Freenode

Freenode (www.freenode.net) was previously known as OpenProjects, and is a network dedicated to providing facilities for open source projects, run by the PDPC (Peer Directed Projects Centre). Their staff members operate a donation-based way of running the network, with various rewards for contributors such as virtual hostnames (cloaks). The way freenode is run is very different from the way other networks are run, with most channels not having users with channel operator status most of the time unless it is required. This network is secure and peaceful, and *highly recommended if you are developing an open source project*. It has very few of the problems found on other networks (such as denial of service attacks, malicious IRC bots, and stealing of channels). Many large and established open source projects such as GAIM (www.gaim.org) and GNOME (www.gnome.org) have set up a permanent channel here, where they offer support and discuss development of their software.

Summary

This chapter has covered the basics of IRC—What it is, where it came from, who created it and why, and the state it exists in today, and how this affects its security in the present day. We have discussed the major networks briefly and what their functions are, even a little about what kind of people you might meet on them. We have also discussed some of the deeper technical aspects of IRC such as time stamps and nick delay.

Solutions Fast Track

History

- ☑ IRC was originally created by Jarkko Oikarinen of the University of Oulu as a better replacement for the UNIX talk program.
- ☑ IRC has existed since 1989, but much less time in the form we know it today.
- ☑ The previous version of IRC, which existed before 1993, only had numbered channels and lacked many channel mode types. This version of IRC is no longer available.
- ☑ Many arguments about the development and future paths IRC should take have caused its development and the people who created it to splinter into many factions and produce wildly different programs. Despite their differences, all can generally be classed as IRC servers.
- ☑ There are many different IRC server packages, which have very different features, some of which may be of use to you and others which may not.
- ☑ There are projects in existence that plan to change how IRC works and bend or break the rules of the RFCs even though they are not really supposed to do so.

IRC Server Software Packages

- ☑ `ircd v2.11.x` (<ftp://ftp.irc.org/>) could be considered the original IRC and is responsible for creating current standards in the RFC 2811.

- ☑ Similar to ircd v2.11.x, the ircd-hybrid (www.ircd-hybrid.com) is an IRC pioneer, conceptualizing timestamps. This server sticks to basics and has made few feature developments since its inception.
- ☑ bahamut (bahamut.dal.net) is a high-performance IRC daemon designed to facilitate a large number of users. Unlike the ircd family, bahamut is always adding new features and extras to the client, and boasts a highly secure reputation.
- ☑ The ircu software used on Undernet is designed with security and stability in mind. Designed to hold very large amounts of users and generally to conceal information from those who do not need access to it, ircu is the Fort Knox of IRC software. Information that the RFC says should be displayed can be hidden from users in this software, following the security concept that users should only see information they need and nothing more.
- ☑ UnrealIRCd (www.unrealircd.org) is a very popular IRC daemon thanks to its unconventional features like spam filters, user bans, channel modes and compressed server connections.

Major Networks

- ☑ The rizon network holds a large number of file trading channels.
- ☑ The Quakenet and GameSurge networks are primarily gaming networks with channels centred on discussion and development of games.
- ☑ The freenode network is an open source developer network containing channels related to open source software projects.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

- Q.** Why didn't you cover network X in your listing?
- A.** The listings of the top ten networks change pretty regularly and by the time you read this book, the chances are that one or more networks listed will have changed positions.
- Q.** If the majority of a network's users are there for file sharing, are they most likely real people, or bots?
- A.** The majority of the users will be real people. It takes relatively few bots to serve a large number of real users who come to the network seeking files. There are exceptions to this however, for example on IRC networks that encourage hybrid IRC/P2P applications like eMule that can connect to IRC.
- Q.** Why didn't nick delay catch on?
- A.** The majority of developers decided that time stamps were better than nick delay, and basically abandoned nick delay, leaving just IRCnet to develop nick delay-based servers.
- Q.** What is channel delay?
- A.** Channel delay and nick delay are part of the same system. The nick delay part of the system applies to nicknames and channel delay applies to protection of channels. However, in this book we use the term nick delay to refer to both mechanisms to prevent unnecessary confusion.

IRC Networks and Security

Solutions in this chapter:

- IRC Networks
- File Transfer Protocols
- IRC Botnets
- Automated Shares/Fserve Bots

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

This chapter will take a deeper look into IRC (Internet relay chat), investigating its illegal and immoral underbelly. From a security standpoint, IRC can be a risk to your assets; however, if used effectively it can also be an asset in itself. By following sensible practices and staying in the know regarding new developments in IRC, you can prevent yourself from becoming a victim of IRC abusers, turn the tables, and make IRC a useful tool. In this chapter we will cover the security-related aspects of each network, what they provide to protect you, what to look out for, and where. We will also cover file transfers, how they affect IRC, and how they affect you as a user of IRC. We will learn how to detect and make use of file sharing and prevent illegal material entering your own network.

IRC Networks

As stated in the previous chapter, there are many IRC networks. To give a detailed summary of all networks is beyond the scope of this book, due to the sheer number of them; however, we will document the top four well-known networks. The networks selected are not listed here because they have the most users, but because the security implications of these networks are most well known out of all the available networks, as these are the oldest and most established networks still running today. While other networks may provide newer and more advanced security features, these security features are not yet researched in depth and their true benefits and issues are not fully known. Several online services such as SearchIRC (www.searchirc.org) and NetSplit (www.netsplit.de) offer comprehensive online indexes of IRC networks and are updated in real time.

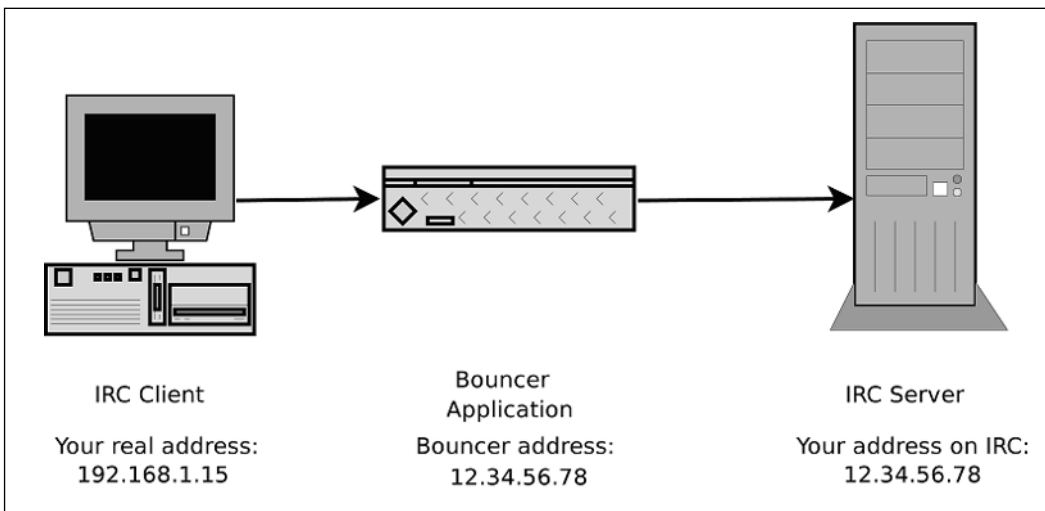
Most of the networks discussed below are maintained by volunteers. Servers are donated by large organizations, however, over the last few years there has been a steady decline in the number of servers donated for IRC use by large ISPs (Internet Service Providers) and universities. This is because of a steady rise in attacks committed against IRC networks (such as distributed denial of service (DDoS) attacks), which cost these businesses too much money to allow them to justify offering servers for IRC use. A large number of servers are therefore paid for directly from the pockets of IRC server administrators. There are a few notable exceptions to this rule however, such as the Freenode network, which operates as a registered business and also accepts donations.

EFnet

EFnet (www.efnet.org) is one of the largest networks in existence today, running the `ircd-ratbox`, `ircd-hybrid`, and `csircd` server software, which are compatible with each other (see the previous chapter). The servers themselves have very few security features, if any, so you are generally at the mercy of your own experience on EFnet. Your hostname or IP (Internet Protocol) address is visible for all to see, so by connecting to EFnet you expose your online location to the rest of your peers. If you consider this a risk, you should set up a *bouncer*.

Setting up a bouncer involves installing third-party software on a remote server, and proxying your session through the third-party server so that your connection to the IRC network is not directly made by your host. The software to do this is available at www.psychoid.net/psybnc.html and is called `psybnc`. Please note, however, that using `psybnc` has its own security implications. By connecting via another machine, this means there is another link in the chain to consider, and any chain is only as strong as its weakest link. Ensure that the server you select for your `psybnc` is secure, especially if you wish to discuss confidential topics over IRC, otherwise your conversations may be compromised. Use a trustworthy provider and where possible use dedicated instead of shared hosting solutions. Shown in Figure 14.1 is a diagram of a connection proxied via a bouncer showing what is visible from IRC and why.

Figure 14.1 Deploying a Bouncer



Because EFnet has no services (virtual servers designed to prevent abuse by stealing of nicknames and channels from their owners) this leaves users open to

abuse for ownership of nicknames. Abusers may collide and generally be a nuisance to users, parking clients on nicknames to hold them, or taking over channels to prevent users from gaining access to them. On Efnet, as with IRCnet, bots, or the previously mentioned *psybnc* are essential to prevent this. For more information on channel bots refer to Chapter 15.

EFnet does provide *chanfix*, which is a system whereby channel users who were previously operators can become operators again. Chanfix is not user accessible, and is completely automatic. It watches for people being given channel operator status on a channel, and times how long they have this status for. If the channel then later becomes op-less (nobody in the channel has operator status) then chanfix will pay a visit to the channel and restore channel operator status to anyone who has a certain score or higher. Points for this score are accumulated by being eligible (you must have a resolving static hostname and working *ident*) and by being on the channel a certain number of hours with channel operator status. Due to these reasons, it is mostly IRC bots and *psybnc* clients that obtain a significant number of points in the *chanfix* database.

Notes From The Underground...

identd

While using IRC you will notice many passing references to something referred to as *ident* or *identd*. In fact, even this book mentions it on more than one occasion. *identd* is a simple network daemon that allows remote systems to determine which username is responsible for each and every connection, inbound or outbound, on your machine. This is a major security risk, but some IRC networks such as EFNet and IRCNet require it.

When your computer connects to an IRC network, the IRC server will attempt to connect back to your computer on port 113, which is the TCP (Transmission Control Protocol) port used for *ident*. If it can ascertain a username from this service, the username is displayed in your connection information and will be visible to anyone who wants to see it. IRC administrators will use this information in the event there is abuse, in order to narrow down exactly which user on the system was the cause of the problem. You may find more information on the *ident* protocol from its RFC (number 1413) at [ftp://ftp.rfc-editor.org/in-notes/rfc1413.txt](http://ftp.rfc-editor.org/in-notes/rfc1413.txt).

DALnet

DALnet (www.dal.net) was the first network to pioneer IRC services. IRC services are, as previously mentioned, virtual servers that protect nicknames and channels. IRC services provide facilities that allow users to register nicknames and channels. This registration requires a valid email address, and once completed, allows you to identify for a resource, such as a channel or a nickname, which you are then the exclusive owner of. The services server represents these services via virtual users, known as NickServ and ChanServ. These virtual users are given commands to make them operate, and are explained below.

NickServ

This service provides many commands that are accessible to users, as shown in the log sample below:

```
-NickServ- ***** NickServ Help *****
-NickServ- NickServ permits users to 'register' a nickname, and stop
-NickServ- others from using that nick. NickServ allows the owner of a
-NickServ- nick to disconnect a user using the owners registered nick.
-NickServ- If a registered nick is not used by the owner for 30 days,
-NickServ- NickServ will drop it, leaving it up for grabs by another user.
-NickServ- Please do NOT register more nicks than you will actively use! =)
-NickServ-
-NickServ- For more information on a command, type
-NickServ- /msg NickServ@services.dal.net help <command>
-NickServ-
-NickServ- Core Commands:
-NickServ-     ACCESS      - Change the list of addresses allowed to use a
-NickServ-                  nick
-NickServ-     DROP        - Drop a registered nickname
-NickServ-     GHOST        - Terminate a ghosted nickname
-NickServ-     IDENTIFY     - Authorize yourself using a password
-NickServ-     RECOVER      - Stop someone from using your registered nick
-NickServ-     REGISTER     - Register a nickname
-NickServ-     SENDPASS     - Request that your password be sent via email
-NickServ-     SET          - Change settings, including the ENFORCE option
-NickServ-
-NickServ- Other Commands:
-NickServ-     RELEASE      INFO      ACC
-NickServ- ***** End of Help *****
```

To register a nickname, you should use the **register** command, as shown below:

```
/QUOTE NickServ REGISTER <password> <email>
```

You will then receive an e-mail with instructions as to how to activate your registration. Once your registration is activated you can identify for your nickname, making you the authoritative owner of it, thus preventing arguments over nickname stealing and also preventing collisions occurring, as happens on EFnet and IRCnet. Identification is done via the following command:

```
/QUOTE NickServ IDENTIFY <password>
```

Of course, NickServ has to give you time to identify, which means that there can be a delay of anything up to a minute between someone else taking your nickname (for example, to abuse it) and NickServ taking action. The **set** command can change these options, or otherwise, if you are on IRC you can use the **ghost** command to forcibly remove them:

```
/QUOTE NickServ GHOST <nick> <password>
```

When you issue the **ghost** command, the person using the nickname is forcibly removed from IRC, allowing you to change to that nickname and identify. **Recover** has the same effect, but after removing the impersonator, holds your nick against further use (by you or anyone else) for up to 60 seconds.

WARNING

Wherever there are passwords, there are those who will want to brute force, break, or otherwise obtain passwords by deception. Choose strong passwords for your NickServ details. There are abusers on DALnet who use armies of bots (see Chapter 15) to conduct activities such as brute forcing passwords. Fortunately, the DALnet staff is very active and watches out for this kind of offense, preventing the majority of these attacks from being successful.

ChanServ

The ChanServ service is also available for your use and is more advanced than NickServ. Where NickServ allows registration of nicknames, ChanServ allows registration of channels, ensuring that only registered users and those trusted by the regis-

tered user can get any privileges upon the channel, as shown in the log excerpt below:

```
-ChanServ- ***** ChanServ Help *****
-ChanServ- ChanServ gives normal users the ability to keep hold of a
-ChanServ- channel, without the need for a bot. Unlike other IRC networks,
-ChanServ- channel takeovers are virtually impossible, when they are
-ChanServ- registered.
-ChanServ- Registration is a quick and painless process. Once registered,
-ChanServ- the founder can maintain complete and total control of the
-ChanServ- channel. ChanServ will stop monitoring a channel if no Op enters
-ChanServ- the channel for 30 days or the founder's nick expires.
-ChanServ-
-ChanServ- For more information on a command /msg ChanServ@services.dal.net
-ChanServ- help <command>
-ChanServ-
-ChanServ-
-ChanServ- Core Commands:
-ChanServ-     REGISTER - Register a channel
-ChanServ-     SET      - Change various channel configuration settings
-ChanServ-     SOP      - Maintain SuperOp channel operator list
-ChanServ-     AOP      - Maintain AutoOp channel operator list
-ChanServ-     AKICK    - Maintain the channel AutoKick banned user list
-ChanServ-     DROP     - Drop a registered channel
-ChanServ-     SENDPASS - Request that your password be sent via email
-ChanServ-
-ChanServ- Other Commands:
-ChanServ-     IDENTIFY  ACCESS  OP      DEOP
-ChanServ-     INFO      INVITE  MKICK   MDEOP
-ChanServ-     UNBAN     COUNT   WHY     ACC
-ChanServ- ***** End of Help *****
```

To register a channel, you should issue the following command:

```
/QUOTE ChanServ REGISTER <channelname> <password> <description>
```

Once you have registered a channel, the channel is yours. Nobody else may have status in this channel of any description, unless you add him or her to the channel's access lists. These channel access lists are accessed through the **aop** and **sop** commands and via the **set** command. As with NickServ, pick a strong password for your channels.

You do not need to identify for your channels. You are also automatically given privileges in all channels you are a member of. You may call up the information on a channel registration, whether the channel is owned by you or not, using the following command:

```
/QUOTE ChanServ INFO <channelname>
```

DALnet's rules do not permit open file trading, as detailed later in this chapter. However, this does not mean file trading does not occur. With a network the size of DALnet to maintain, the staff are unable to police this rule efficiently, meaning a small number of file sharing channels still do exist on DALnet.

Undernet

Undernet's (www.undernet.org) security facilities could very easily occupy a book of their own. Luckily, however, most of these security features are internal to the servers and not visible to the users. In fact, very little at all is visible to users on Undernet. Undernet strictly follows the security mantra, "show users only what they need to see," hiding from users, among other things:

- The server list shown in /LINKS
- The server each user is connected to
- Each user's host (if the user is registered)

Undernet has its own services facilities, which are quite different from those of networks such as DALnet. Undernet's services package is accessed through a bot called X, which takes commands via messages. It is essentially a combined nickname and channel service. Nickname registration on Undernet is done via the cservice website at <http://cservice.undernet.org>.

Channel registration on Undernet is convoluted in comparison to DALnet, but prevents erroneous or inaccurate registrations. To register a channel on Undernet you must first register your interest in the registration on the cservice website, and provide a list of users who will back your registration. All users you list must agree to back your registration. If any of the users on your list reply negatively, your channel registration will be denied. Where DALnet's services favor protection of smaller channels, Undernet's services favor protecting larger established channels.

Many troublemakers and abusers congregate on Undernet to plan their malicious activities. In the past few years a large number of attacks on other networks have been orchestrated from Undernet channels (usually secret channels that nobody is aware of), such as the recent *anatoly/fyle* attack, which caused virus-infected clients

to connect to other networks for infection, but gave out free access to the machines to all Undernet users that knew how to find them.

IRCnet

IRCnet (www.ircnet.com) is one of the oldest and most established IRC networks. It provides no services of any kind (not even chanfix). However, recent versions of the IRCnet IRC server have what is referred to as the *reop mode*. The reop mode allows users to obtain their previous channel status if they lose it. A channel creator may use the modes `+r` and `+R` to add hostnames to a list of users that will be automatically given their status back should everyone lose status. Note that this feature only prevents loss of status in op-less channels, and will not prevent your channel being taken over by malicious users.

Because of the general lack of services of any description, IRCnet has more bots than most other networks. Bots and psybnc clients make up the majority of connections to some IRCnet servers, and are essential for holding a channel against persistent takeovers. Because IRCnet uses nick delay and channel delay rather than time stamps, any lengthy network split can cause chaos, especially if your channel is well known, as the abusers scramble to take it over before the split ends. IRCnet exposes your hostname to the network, and if your hostname is visible you are likely to receive various probes, scans, and all kinds of investigation of your network. It is common practice for servers and even bots to port scan your host looking for weaknesses, for both malicious and benign reasons.

IRC Servers in Sum

As you can see from the details given above, each network has its own benefits and pitfalls. It is recommended, however, that for actual business use, businesses set up their own IRC servers rather than use any of these networks. This is recommended simply because all the networks listed above are run with the goal of entertainment and are maintained as a pastime by volunteers. This means that you will not be able to obtain the same level of support for them that you would be able to get on a proper commercial service. The previous chapter outlines a number of IRC daemons that are recommended if you wish to take this route and set up your own IRC server for your business, which will give you much more control over what comes into your IRC network and how it operates. Table 14.1 is a summary of all the networks discussed above.

Table 14.1 IRC Servers Summary

Network	Services	Information hidden
IRCNet	None	None
EFNet	chanfix	None
DALnet	NickServ/ChanServ	None
Undernet	"X"	IP Addresses, Server Addresses

File Transfer Protocols

Originally, IRC was simply a text chat medium. Over time, the need arose for more features, and DCC was created. DCC stands for direct client connection, and is used for direct chats and file sends that bypass the server except for the initial one-line handshake. The authors of the `ircII` software package originally pioneered file transfers over IRC (see Chapter 16 for a list of client software).

The DCC protocol was never standardized as a proper RFC (standards document) so there is no RFC number for this protocol. However, there is a draft document available online (www.irchelp.org/irchelp/rfc/dccspec.html) which is usually adhered to quite strictly by most IRC clients. The DCC handshake and how to detect DCC transfers are discussed in the next chapter.

NOTE

The easiest way to detect DCC transfers using intrusion detection software is to watch for the text `"\001DCC SEND"` within TCP packets, which is part of the initial DCC handshake. The `\001` given in this example is not a literal `\001` with a backslash, but is the actual character value `"1"`. As the actual content of the DCC is just the file itself, there is no reliable way to detect the DCC from its content, so this initial handshake must be intercepted.

Today, DCC file transfers are prevalent on IRC. Many users of IRC see it as nothing more than one huge swap meet, rather like an e-Bay where you don't have to pay. All kinds of electronic property are exchanged, including software, music, movies, and sometimes even trade secrets and source code for proprietary applications. Some networks set themselves up solely to trade illegal files (usually these networks are also

setting themselves up for a fall, as many are subpoenaed by law enforcement asking for logs and permission to monitor channels, etcetera), while other networks actively track down and prevent illegal file distribution, either out of fear of law enforcement or out of a need to be good members of the Internet community as a whole. DALnet is an example of this. Other networks, such as EFnet and IRCnet (and also rizon) turn a blind eye to file trading but are fully aware it happens. This policy of non-interference allows these networks to operate without outside investigation for the time being.

DCC files are usually pretty small in size due to the fact that the DCC protocol is pretty limited, and usually groups of files will be zipped into archives because the DCC protocol does not support grouping of files, as is common in IM (instant messenger) applications. In fact, compared to modern IM applications, the DCC protocol is truly antiquated, and there has been little attempt to update it. There is one major exception to this however, and that is the XDCC protocol.

The XDCC protocol is an attempt to provide more file sharing features to IRC clients. It requires extra software on the system of the person sending the files, but it does not require the file's receiver to have any extra software. It implements a grouping system for grouping files together in batches and requesting and/or sending those batches to users. A user offering files via XDCC will commonly have a list of files on their client, and will advertise this list to the channel:

```
<E|Koyuki> ** ... New Manga: Iketeru Futari Volume 1: /msg E|Koyuki xdcc  
send #1 | Get Volume 2+ Chapters from #Nisei ... **
```

People in the channel can publicly request the files from this list, which will cause the sender to send the whole group of files associated with pack number 1, in this case, part of a set of image files. Usually, the automation goes no further in a standard XDCC system. If there is further automation to the system, it is more likely to be in the form of an automated share or fserve bot, discussed in the next section. Because the actual text and banners given out by an XDCC server can vary (for example, different colors, different advertisements for the files, and different filename formats) there is no reliable way to detect an XDCC server. Even the way an XDCC transfer is initiated can vary from program to program! However, its weak point is that it *still operates over the same DCC protocol that all other file transfers on IRC use*, and can be detected in the same way as any DCC file transfer (by looking for the initial handshake).

IRC Botnets

A group of bots with a common design is known as a *botnet*. Usually these bots share information among each other via some common connection or over IRC itself, actually defining the fact that it is a “bot net” (short for “bot network”). In addition to XDCC bots and fserve bots, there are many other kinds of IRC botnets. Some of these are malicious and others are benign. In fact, some of the more benign botnets are actually designed to keep malicious botnets out of IRC channels.

Originally, IRC bots were used for this purpose only. Because the original IRC networks had no concept of channel ownership at all, generally, anarchy reigned.

Automated Shares/Fserve Bots

Sometimes, grouping files into packs is not enough. People who trade in files over IRC, be they legal or illegal, want the process to be automatic. They do not want to have to sit at their PCs, and they want their bandwidth usage to be automatically managed. To this end, the fserve bot was developed. An fserve bot is an automatic file-sharing client with the sole aim of distributing files to groups of users on IRC. The functionality of an fserve bot is as follows:

- Idle on an IRC channel, occasionally putting out a notice that files are available.
- Responds to requests for the file list, either via message or via a DCC chat connection.
- Responds to requests for the files themselves, maintaining a queue of users who are waiting for the files.
- Only allows the queue of waiting users to grow to a certain length.
- Only allows one user to queue up a certain number of files, or a certain size of files (in megabytes).
- Only allows a certain number of sends to progress at once.
- Only allows each send to progress at a certain transfer speed.
- Usually terminates file sends if a user leaves a specific channel while he/she is receiving the file.
- Allows the bot owner to add files to the share list.
- Instructs the bot to fetch files from another authorized bots file list (in the case of mass sharing).

These automated bots usually provide all of the above facilities and allow them all to be configured by the bot owner. Often, these bots are placed upon compromised machines with high bandwidth connections, and entire collections of copyright or questionable material are uploaded for the bots to share. This can be done en masse with a large channel of file sharing bots appearing virtually overnight. Please note that these particular bots are not the same as other types of malicious bots (see Chapter 15) and are not designed to attack anything. They have one role and one role only—to share files, no matter what they are. There are publicly available fserve bots that are downloadable online, some licensed under the general public license. One of the most popular is a program called iroffer, which is available from the site www.iroffer.org.

Figure 14.2 File Sharing Bots In Action

```
mIRC - [Hmp3traderz [298] • ntl 308]: 100 serverz / 2,470,259 filez ThreeSaxMafia/Santana/SherylCrow/Shaggy/ToniBraxton/LizPhair/Ferdinand/PaulMcCartney/
File View Favorites Tools Commands Window Help
UnderNet M... Hmp3traderz Channels
20,788 List: Sep 28th Search: ON Mode: Servers Priority
[[Sasha]:Hmp3traderz SLOTS] 3 3 NOW 0 999 0 40820 187092140722 1 1127511298 70865 OneNServE v2.50
<DeviousD> @JeffGordon-que
<Stan_G> @BullDog
<Centiced> @12stepn
<ckline> Type: fckline phn0204-04 chesney, kenny - young.zip To Get This 3.87MB File
<dliberto> @Snoochie
<nasstjnlj> !wldflwr tony gayo - welcome back tony ya - 00 - live from rikers isl.mp3
<Cashew> Type: @Cashew- For My List Of: 9,611 Files Slots: 2/2 Queued: 0 Speed: 0cps Next: NOW
Served: 236 List: Sep 24th Search: OFF Mode: Normal
<In(Q)4U> New Music ALL the time, updated EVERY DAY, get my new list!!! Type: @In(Q)4U For My List Of: 25,810
Files Slots: 0/2 Queued: 0 Speed: 1,542cps Next: 12m Served: 77,652 List: Sep 26th Search: ON
Mode: Servers Priority
<NetHazard> I have just finished receiving nerfherdr-default(2005-09-02)-05.zip from nerfherdr 3
Slot(s) Empty! Grab one fast!
<Fly1> @KuNtRy_Metal
[Cashew:Hmp3traderz SLOTS] 2 2 NOW 0 999 7505 9611 51266041131 0 1127542347 15145 OneNServE v2.52
* R-m66iHhN has left Hmp3traderz
[In(Q)4U:Hmp3traderz SLOTS] 2 0 12n 0 999 1542 25810 125437004843 1 1127723643 28832 OneNServE v2.52
<DeliveryG> Type: @DeliveryG For My List Of: 4,691 Files Slots: 4/4 Queued: 3 Speed: 0cps Next: NOW
Served: 141 List: Sep 29th Search: OFF Mode: Normal
[JazzLvr:Hmp3traderz MP3] Dave Brubeck - Just You Just Me - 06 - Tribute To Stephen Foster.mp3
<JazzLvr> [128Kbps 44Khz Stereo]-[5m52s] [128Kbps 44Khz Stereo]-[5m52s] [128Kbps 44Khz Stereo]-[5m52s]
<NetHazard> @Cashew
* C14PT0N has joined Hmp3traderz
<westicle> @Find tool
[DeliveryG:Hmp3traderz SLOTS] 4 4 NOW 3 999 0 4691 25461640008 0 1128003377 1811 OneNServE v2.52
<FurbySlayer> New List! Lot's of AUDIOBOOKS Added!! Type: @FurbySlayer For My List Of: 2,587 Files Slots: 0/2
Queued: 2 Speed: 25,777cps Next: 3m Served: 1,226 List: Sep 29th Search: ON Mode: Normal
<StrayKat> wizzywiza has just received StrayKat-default(2005-09-28)-05.zip for a total of 41,966 file(s)
140 GB sent since 4th June 2005 KeepTrack 6.2 by "OneH"
<westicle> @C-Note-que
<NetHazard> I have just finished receiving Larry_the_Cable_Guy_-_Law_&Disorder_-_00_-_Bad_Dreams.mp3
from KuNtRy_Metal 3 Slot(s) Empty! Grab one fast!
<Ustwo> Conin' at ya at DSL Speed. Please close the door when you are done. Thanks Type: @Ustwo For My List Of:
9,024 Files Slots: 10/10 Queued: 0 Speed: 0cps Next: NOW Served: 122 List: Sep 11th Search: OFF
Mode: Servers Only
<westicle> fmerlin038 Tool - 03 - Sober.mp3
```

Tools & Traps...

The Magic Number 4

It is a limitation of the DCC protocol that it cannot handle files over 4 gigabytes in size. This is because internally it acknowledges each block of data with the current file position, and as this acknowledgement is four bytes wide, it cannot store a number higher than this, thus limiting the maximum size of any file sent over IRC.

File-Sharing Botnets

File-sharing botnets are designed to share files in large quantities. Not all file-sharing botnets share illegal material. Some botnets share open source files and abandonware, however this is not commonly found on IRC any more. As described above, file-sharing bots make use of DCC and XDCC protocols to send their files.

Channel Protection Botnets

Channel protection bots, such as WinBot (www.winbot.co.uk) and eggdrop (www.eggheads.net) operate together in networks to share channel state information. This channel state information includes lists of users who are authorized to use the bots, lists of blacklisted users who should be banned, and statistical information on who should get operator status, who has had operator status, and who should never obtain operator status. The bots use this botnet connection to communicate directly with each other, for example if one bot loses its operator status, it will signal to the other bots to ask for that status back. Similarly if one bot is kicked and banned, it will request from one bot that it be un-banned (and possibly re-invited to the channel), and request for a second bot, if available, to deal with the person who kicked the original bot out. These protection systems can become pretty advanced, as a lot of thought is put into how to protect a channel from being taken over by users who are not welcome in it (especially on EFnet and IRCnet).

The race between protection bots and channel takeover bots (see below) can be compared to the race between antiviral software and virus writers, and may possibly never be won. With channel protection, it is simply a matter of numbers and statis-

tics. The larger number of bots, for protection or abuse, will always prevail. This is coupled very loosely with the cleverness of the channel ops and takeover crew. For example, one bot can remove operator status from three clients in one line on IRCNet, with the following:

```
MODE #channel -ooo c1 c2 c3
```

Therefore, for each takeover bot that can possibly attack your channel you'll need four bots (one to watch the backs of the other three). However, this is only the tip of the iceberg, not even taking into account networks that let you do more than three de-ops in one line (which means all of them except IRCnet and EFnet) and other attacks such as DDoS. In short, there is no guaranteed successful way of defending a channel against being taken over by a third party without the use of services such as ChanServ and NickServ etcetera.

Channel Takeover Botnets

These bots are the exact opposite of channel protection bots described above. The role of a channel takeover bot is one hundred percent malicious with the one goal of causing chaos on IRC. The common tasks a channel takeover bot will perform are nickname collisions, channel split riding (trying to feed incorrect channel state data into one side of a sync to gain channel operator status), desync abuse (abusing invalid channel states, where one server has a different state to the other due to latency), and generally keeping the channel inaccessible to its owners after access is obtained. This usually takes significant research beforehand, such as obtaining lists of users hosts, nicks, and idents (usernames) so that a blacklist can be constructed in advance.

Channel Flooding Botnets

Channel flooding botnets are designed for only one purpose – to disrupt a channel or network by excessive scrolling of text, making it unusable and sometimes disconnecting users. This should not be confused with spamming (see next section). IRC servers maintain a buffer for each user, which has a finite maximum size. When a user's buffer is growing faster than it can be read (for example, when there is an excessive amount of channel text or messages) and the client is not reading the connection fast enough, the buffer reaches this limit. If the client exceeds this limit, the IRC server closes the connection and the client is disconnected, usually with a reason such as "SendQ exceeded". For this reason these types of bots are often combined with channel takeover bots, or bots are designed to do both actions in one program. These bots are often found on compromised hosts. These types of botnets

are best dealt with using specialist software that prevents them from being a nuisance. Some IRC servers have built in channel settings to prevent this abuse, whilst there are other programs available such as IRC Defender (www.ircdefender.org) and NeoStats (www.neostats.net).

Spamming Botnets

Channel spamming bots are designed to spam channels with website addresses and messages promoting commercial ventures. There are many of these on IRC, and recently they have started moving from the larger networks such as DALnet to smaller networks. The usual behavior of a spam bot is as follows:

- Obtain a list of all channels on the network.
- Join each channel, or the top few channels.
- In some cases, send spam to the channel (if the channel is not moderated).
- Some bots will send spam to all non-opped users in the channel (IRC operators not included).

The most efficient way to deal with these bots is via server-side filtering, for example the filtering systems of IRC server software such as UnrealIRCd and InspIRCd. This prevents the spam from reaching end users, but you must first obtain a sample of the spam to decide what to filter out.

DDoS Botnets

DDoS botnets are sets of virus- or Trojan-infected systems, awaiting an abuser's command on IRC. These types of bots are discussed in detail in Chapter 15 and are very different in behavior from other kinds of botnets found on IRC. While most other types of botnets are connected by a common connection, these types of bots are not, apart from their actual IRC connections.

Proxy Botnets

Flooding IRC with open proxy server connections is a common attack usually employed by people who are new to IRC and just want to cause issues. By obtaining lists of open proxies from Google or other search engines, the attacker can make dozens or hundreds of simultaneous connections to IRC, and instruct them all to do the same thing at the same time. Usually this will be to join a single channel, or just to idle and consume resources on the server or network. When they join a

single channel the effect can be similar to that of channel flooding botnets, and in fact they are generally the same thing with a different backend.

There are documented ways to prevent these bots from connecting to your network. The commonly accepted solution is to install a program called BOPM (blitzed open proxy monitor), which connects to the IRC network and scans clients that connect to your network, removing users who have open proxies. This software also queries a DNS (Domain Name System) blacklist (dnsbl) known as opm.blitzed.org, which contains a comprehensive list of compromised or open proxy servers. You can obtain this software from www.blitzed.org/bopm.

Other Uses for IRC Bots

If a program can perform an action, it is possible for an IRC bot to perform that action. There have been reports of botnets being used for many other weird and wonderful purposes, some clever and some not so clever, such as using IRC bots as DNS servers and using IRC bots as a distributed computing farm, usually for purposes such as crunching numbers, for cracking passwords, and generating usable credit card numbers. It is a certainty that as time progresses, more and more uses will be found for IRC bots, and they will become more of a problem for IRC administrators and IRC users alike.

A summary of botnet types is illustrated in Table 14.2.

Table 14.2 Summary Chart of Botnet Types

Botnet Type	Malicious?	Performs DDOS?	Attacks IRC?
File sharing	No	No	No
Channel Protection	No	No	No
Channel Takeover	Yes	No	Yes
Channel Flooding	Yes	No	Yes
Spamming	Yes	No	No
DDoS	Yes	Yes	Yes
Proxy	Yes	Yes	Yes

Summary

In this chapter, we have learned the various ways malicious users may harass and attack using IRC as a medium, and how to remain safe and avoid the problems associated with this. We have also discussed file sharing, how it affects IRC and its users, and how it works.

Solutions Fast Track

Networks

- ☑ There are many thousands of IRC networks that vary in size from tens of thousands to tens of users.
- ☑ Many smaller networks run extra security features but they aren't as tested as the measures used on the larger networks.
- ☑ EFnet leaves hostnames and IP addresses visible to all users.
- ☑ While using a network such as EFnet you should consider using a program called a psybnc to cloak your hostname.
- ☑ EFnet does not run services; rather it runs a program called chanfix, which determines who gets privileges based upon automatic scores.
- ☑ DALnet was the first network to pioneer the use of IRC services.
- ☑ DALnet provides two services for nickname and channel registration known as NickServ and ChanServ.
- ☑ There are malicious users on DALnet who make a habit of brute forcing NickServ registration passwords.
- ☑ Undernet can cloak your hostname for you if you register your nickname, removing the need for a bouncer program.
- ☑ Undernet's channel service, known as X, prefers larger channels over smaller ones, requiring the backing of several other users to get a channel registered successfully.
- ☑ A lot of abusive users call Undernet their home. As they never do anything to attract the attention of staff, nothing much can be done about this.

- ☑ IRCnet has a high ratio of bots to normal users, which are required to protect channels.
- ☑ IRCnet does not provide services or chanfix, however it does provide a special channel setting called reop.
- ☑ IRCnet is the only remaining larger network to use ND (nick delay) rather than TS (time stamps) to solve collision issues.

File Transfer Protocols

- ☑ Originally, IRC was simply a text chat medium. Over time, the need arose for more features, and DCC was created.
- ☑ All kinds of electronic property are exchanged over IRC, including software, music, and movies, and sometimes even trade secrets and source code for proprietary applications.
- ☑ Some networks set themselves up solely to trade illegal files.
- ☑ The DCC protocol is pretty limited, and usually groups of files will be zipped into archives because the DCC protocol does not support grouping of files as is common in IM applications.
- ☑ The XDCC protocol is an attempt to provide more file sharing features to IRC clients. It requires extra software on the system of the person sending the files, however it does not require the file's receiver to have any extra software.
- ☑ The XDCC protocol is simply standard DCC file sends combined with certain private and public “trigger” messages.

IRC Botnets

- ☑ IRC Botnets can be categorized into many different forms, with one similarity: most of them are used for malicious purposes.
- ☑ Of the few botnet types that are beneficial, the commonly appearing one is the channel protection botnet, used to protect against malicious IRC bots.
- ☑ Bots can be created to abuse open proxy servers in much the same way e-mail can be sent via open spam relays.

Automated Shares/Fserve Bots

- ☑ Automated Shares and Fserve bots are used when unattended file transfers have to be managed.
- ☑ Most Fserve bots are entirely remotely configurable, and contain many options to manage bandwidth and file collections.
- ☑ Fserve bots may sometimes be found installed on compromised high-bandwidth servers along with collections of pirate material.
- ☑ Some Fserve bots provide the facility to share amongst themselves like a peer-to-peer network.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

- Q.** Why isn't there a list of places where I can get pirated material in this chapter?
- A.** That would be against the law, wouldn't it?
- Q.** What is the best way to get file-sharing bots to leave my IRC network or server?
- A.** With file sharing bots, the owners may not perceive their own actions as immediately malicious, so the best approach is just to ask them nicely and give them a fair amount of time to relocate. The larger the channel, the more time you should allow them. Only if they do not agree to leave should you resort to banning these types of bots.
- Q.** Why do larger IRC networks turn a blind eye to file sharing?
- A.** The larger the network, the more time and effort it takes to police it. It is simply logistically impossible to watch all the channels in any sizeable IRC network, so the staff turns a blind eye to these activities rather than be involved. Some file sharers may also become aggressive if operators heavy-

handedly try to close them down. When DALnet stopped allowing file sharing channels, a few were banned as an example and the rest left peacefully over a longer period of time, as it would have taken them too long to ban them all and left them with very few real users.

- Q.** Will banning file sharing scripts and bots prevent illegal file sharing on my network?
- A.** No. Only filtering of the initial handshake messages between clients will prevent file sharing. To learn how to do this, read the next chapter.

Global IRC Security

Solutions in this chapter:

- DDoS Botnets Turned Bot-Armies
- Information Leakage
- Copyright Infringement
- Transfer of Malicious Files
- Firewall/IDS Information

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

As with all technologies, there are those who wish to abuse it in very dangerous ways. On IRC (Internet Relay Chat), a large number of abusers are involved in a method of abuse known as *botnets*. You may have heard of botnets from other security fields, but to summarize, in case you have not (and to clarify the differences between an IRC botnet and other forms of botnet), a botnet on IRC is *a set of automated programs (bots) operating in unison toward a common goal*. While many users have no malicious intent, there are others on IRC who are intent on stealing your information—be it your IP (Internet Protocol) address, your home address, your credit card details, you name it! This may be done using a variety of strategies such as Trojans and viruses. All of the common strategies are covered in this chapter, and you must be on the lookout for all of these activities if you wish to be secure whilst using IRC.

DDoS Botnets Turned Bot-Armies

Usually, a malicious user will compromise many machines (usually via Trojans or viruses), the result of which will cause all compromised machines to connect silently to IRC where they await commands from the attacker, who is usually referred to as a *botnet master*. For the rest of this section, the phrase *botnet master* will refer to the person who is abusing the bots over IRC.

The steps in such an attack are as follows:

1. Before infection, the bot program is configured to connect to a given IRC server. Many botnet masters will configure their bots to connect to a dynamic DNS (Domain Name System), such as the dyndns service provided by www.dyndns.org, so that if they are banned from an IRC network, they can change their own DNS to move their bots elsewhere quickly.
2. At the point of infection, and on startup, each infected machine will connect silently to the IRC server.
3. The bots on the infected machines will then usually join an IRC channel (which is usually keyed or otherwise restricted) and await commands.
4. To attack one or more hosts, the botnet master will join the channel where the bots are waiting, and after initially sending a password to the bots, will in most cases issue commands to start an attack. Such attacks may include:

- Flooding or otherwise causing inconvenience to other IRC servers or channels.
- Spamming (monetary goals), phishing (stealing personal financial details for fraudulent use), or identity theft.
- Distributed Denial of Service attacks (DDoS) against other sites on the Internet, either for fun or for blackmail.
- Distributed sharing of copyrighted works (*warez* sharing).

Notes From the Underground...

Benign Bots

Please note that not all programs referred to as bots are malicious, and not all malicious bots are actually IRC-based bots. It is a common misconception on IRC that all bots are bad. However, bots and other automatons are used on IRC for benign uses too, some of these being:

- Channel protection (actually helping to defend against the malicious types of bots).
- Entertainment (playing games and other such activities over IRC).
- File sharing (this can have malicious and not-so-malicious uses).

These non-aggressive bots were the original IRC automata, initially designed to keep channels secure. The IRC bot would keep the channel open, holding ops for authorized users, keeping channel state, and generally maintaining some degree of control. With the advent of services packages (such as those found on Dalnet) the use of bots for protection has somewhat declined on many networks. However, on networks that do not have services (such as IRCNet), bots are essential and very much a part of IRC life. For more information on benign forms of bots, visit the *WinBot* project at www.winbot.co.uk and the *eggdrop* project at www.eggheads.net. Both of these bots were designed with legal and friendly uses in mind.

Methods of Botnet Control

There are many ways of controlling a botnet. Understanding these methods will help you locate and remove such threats before they grow and become a problem, and

will allow you to keep your networks under your control, and not under the underhanded control of a malicious botnet master.

Traditional Denial of Service (DoS) bot programs such as TFN, Stacheldracht, and Trin00 share many features with malicious IRC bots, such as their capability to flood Internet sites, replicate themselves by attempting to exploit other machines, and allow the botnet master to execute arbitrary commands. In these environments, bots are usually controlled via specific TCP (Transmission Control Protocol) connections. However, on IRC most, if not all DDoS bots are controlled via IRC itself.

The IRC protocol is documented by the RFC (Request for Comments) 1459 and RFCs 2811 through 2813, listed below:

- RFC 1459, “*Internet Relay Chat Protocol*”: <ftp://ftp.rfc-editor.org/in-notes/rfc1459.txt>.
- RFC 2811, “*Internet Relay Chat: Channel Management*”: <ftp://ftp.rfc-editor.org/in-notes/rfc2811.txt>.
- RFC 2812, “*Internet Relay Chat: Client Protocol*”: <ftp://ftp.rfc-editor.org/in-notes/rfc2812.txt>.
- RFC 2813, “*Internet Relay Chat: Server Protocol*”: <ftp://ftp.rfc-editor.org/in-notes/rfc2813.txt>.

The primary method for controlling a botnet is via channel topics over IRC. This can easily be detected by sending a client into the channel:

```
[15:02:32] Topic for #botnet is: .advscan dcom135 200 3 0 -r -s
```

The topic shown above actually tells all of the bots in the channel *#botnet* to scan for vulnerabilities. Each vulnerable machine found during the scan will itself be infected with the bot executable, thus growing the botnet yet further. Some botnets are self-propagating in this way. Others (such as the one given in the example below) are not. It is relatively easy to watch for these patterns in topics, and when found, to remove the bots from the channel. Each bot joining the channel will follow the command that is set in the topic, so that commands may be left for the bots when their master is not online. To remove such bots, usually a similar command can be given in the topic, and upon joining, each bot will follow your instructions and exit.

These commands can, of course, be changed by the botnet master if he is smart enough to do so. However, most botnet masters leave these commands as default, so the following commands will usually succeed in removal of the bots:

- .rm
- .remove
- .uninstall
- @rm
- @remove
- @uninstall

Most commands will follow these patterns and can be figured out by observing the channel to figure out the format of the commands before you try.

The secondary method for controlling a botnet is to issue commands via the channel itself. This is similar in action to setting the topic, but in this case, to give commands the botnet master must reveal himself – if you can see the user issuing commands, you can identify who it is and take action against him or her. The commands given usually follow the same format as above, but whereas most bots that follow commands in topics *do not* require a password, those that take commands in channels usually *do*. The best ways to obtain such a password are via direct observation of the channel (for example, join the channel configured to look like one of the bots) or via packet sniffing or other such external monitoring. Note that where botnet masters use these types of bots, they are usually also configured to only respond to specific nicknames. The bots may only respond if you take the botnet master's nickname beforehand and issue commands via that nickname. An example of this is shown at the end of this section.

The least common method for controlling a botnet (mainly because it is difficult to multicast a command to many bots with this method) is by directly messaging each bot with commands. This has the positive effect of concealing the password and commands from prying eyes (which will prohibit observation of the channel as in the two examples above for obtaining botnet details) and forces you as a researcher to take other action to obtain the details. In these situations, the only real ways to obtain botnet passwords are:

- Install monitoring software on your IRC servers (which may be viewed as unethical).
- Use packet-sniffing software such as tcpdump or snort to obtain the details at the network stack level.
- Simply ask the botnet master (this actually *can* work—for more information refer to the discussion at the end of this section).

This method of controlling a botnet is usually used where each bot in the network is designed to act as a separate entity, for example in file sharing networks used to distribute warez (illegal software) where the data stored on each compromised machine will vary from host to host.

Reprisals

Be aware that by removing bots and attempting to fight botnet masters, you will undoubtedly annoy them. By doing so, you may cause them to attack your own networks. You should be prepared for this in the following ways:

- *Always investigate botnet channels in disguise.* For example, change your nickname, ident (username field), GECOS (real name field), and version reply, or whatever you can do to make yourself look as much like one of the bots as possible.
- *Avoid doing things that the bots don't do.* Such as (to the best of your knowledge) WHOIS-ing channel operators, speaking (or not speaking), quitting or parting with abnormal quit/part messages, etcetera.
- *Avoid making threats.* Annoyed botnet masters are most likely to cause damage via DoS attacks.
- *Wait until the botnet master is away (preferably has quit the channel) before making any attempt to remove the bots.*
- *Only when the threat is removed (when all the bots are uninstalled) can you rest on your laurels.* However, remember that the botnet master may own more than one botnet, and may attack you for revenge anyway. Always be prepared to absorb a DoS attack for your troubles, but remember that what you did, you did for the good of the Internet as a whole so it was worth it.
- *If you can obtain such information, obtain the server address the bots connect to so that you can take further action.* Where it is a dynamic hostname, consider reporting it to the provider of this service—such providers deal with problems like this on a daily basis and are more than happy to assist you with removal of malicious users. If it is a hostname under your control, you may be able to take other actions such as changing the hostname or moving the service to another IP address (which will usually cripple all existing bots in the botnet master's network as soon as the domain name changes propagate through the domain name system).

The ipbote Botnet: A Real World Example

The following is an example of a real world botnet, with a real botnet master. All information here is real, and was later submitted to antivirus researchers.

During the early months of 2005, our network suffered from a small number of bots connecting to it. Over time this small number grew and grew, until eventually at one point no less than fifty bots were connecting and joining the channel *#gfw*. These bots would come and go over time and would randomly output text to the channel referring to scanning various IP addresses on the Internet. An example of this (as well as other types of output from the bot) can be found below in the log extracts.

Further investigation did not reveal what types of bots these were, so we simply banned them from the network and waited. We did, however, have some idea of the author of these bots, a user who had been on our IRC network some time before, and had since quit.

A short while ago, this user reappeared on our IRC network on unrelated business, so we decided to have a quick conversation with him and determine what kind of bots he used, and maybe befriend him to have them removed. This quick conversation went much better than we could ever have expected.

As you can see from the log excerpt below, we not only found out how to remove his bots, but we also obtained source code for his bots (which was later e-mailed to Symantec). A little diplomacy can go a long way.

```
[17:51] --> You are now talking on #gfw
[17:51] --- Mode: Brain [#gfw +s]
[17:51] --> Joins: GermanME (GermanME@ChatSpike-ACDD51A9.dip.t-dialin.net)
[17:51] --- Mode: Brain [#gfw +o GermanME]
[18:02] --> Joins: [GzM]Sunny32 (Sido22@ChatSpike-71633939.ipt.aol.com)
[18:02] --- Brain has changed the topic to: .rm
[18:02] <-- Quits: [GzM]Sunny32 (Sido22@ChatSpike-71633939.ipt.aol.com)
(Client exited)
[18:02] <GermanME> [19:03] <GermanME> @remove
[18:02] <GermanME> [19:03] <[GzM]Sunny32> Recomoving Bot... you dumb asshole
:/
[18:02] <Brain> haha
[18:02] <GermanME> hehe, selfmade
[18:02] <Brain> works in pm?
[18:03] <GermanME> yes, but only if $nick == GermanME
[18:03] <GermanME> and some newer versions of the bot only respond to
"@removeX"
[18:03] <Brain> whats it called?
```

```

[18:04] <GermanME> the bot? i allways called it "ipbote", cuz it was
intentionally meant as ip-messaging-bot, but was mainly used for scanning
public ftp-servers useable for my former webwrez site
[18:04] <Brain> ah
[18:05] <GermanME> codet in visual basic, can log keystrokes to #kloggg,
connect up to 2 networks at the same time, execute dos-commands and return
the output, and newer versions even scan for exploitable ms-iis server


*proud*


[18:06] <GermanME> ya i know, i'm an evil, evil, baaaad, baad person
[18:07] <Brain> oh eye
[18:07] <Brain> :p
[18:49] --> Joins: markus36 (fgqndabzvf@ChatSpike-1D98D87.ipt.aol.com)
[18:49] <markus36> Scanning 123.*.*.*
[19:07] <-- Quits: markus36 (fgqndabzvf@ChatSpike-1D98D87.ipt.aol.com)
(Client exited)
[19:07] <GermanME> [19:50] <markus36> IP-BOTe: 18:48:56 - 172.213.205.143 -
mycomputer
[19:07] <GermanME> [20:07] <GermanME> @help
[19:07] <GermanME> [20:07] <markus36> Supportet Commands: @IP | @RECONNECT |
@REMOVE | @SHUTDOWN | @FTPGET [ip] | @RAW [command] | @EXEC [command] |
@LOCKINPUT | @ENABLEINPUT | @MONOFF | @MONON | @CDOPEN | @CDCLOSE |
@CLIPBOARD | @UPTIME | @SCAN | @STOPSCAN | @CURIP | @GETTHREADS |
@SETTHREADS [integer] | @MSGBOX [text] | @GETDIR | @GETOS | @GETVER |
@LISTRESULTS | @HELP
[19:07] <GermanME> [20:07] <GermanME> @getver
[19:07] <GermanME> [20:07] <markus36> I'm running 'IP-Bote' Version: 1.0.17
[19:07] <GermanME> [20:07] <GermanME> @getos
[19:07] <GermanME> [20:07] <markus36> Operating System: Windows XP
[19:07] <GermanME> [20:07] <GermanME> byebye *sniff*
[19:07] <GermanME> [20:07] <GermanME> @removeX
[19:07] <GermanME> [20:07] <markus36> Recomoving Bot... you dumb asshole :/
[19:09] <GermanME> those people must be wondering what happened to theyr
computers that made them so fast again, the scanner took ~40% cpu xD
[19:11] <GermanME> it's strange how boring people are when it comes to
computers, those "ipbote"-victims had nearly a year to learn how to use
theyr task-managers or msconfig
[19:24] <Brain> any chance you can send me the program? :D
[19:24] <Brain> i want a look :P
[19:27] <GermanME> mh it has no config file or something, if you've got visual
basic 6 .NET i can give you the source code, the .exe-file isn't much
interesting
[19:28] <GermanME> * visual basic 6 OR .NET
[19:45] <Brain> send me both please i want to play with it in vmware :D

```

From this short conversation, we knew how to remove the bots (simply change our nicks to that of the botnet master, and issue a command in a message) and also some of their capabilities (we knew that we should place a ban on the channel *#kloggg* for example). We can also summarize that the creator of these bots was very proud of his creation, and his willingness to share his source code for his “wonderful toy” was his undoing.

A few days later, we had removed all the bots from our network, and knew that all recent antivirus programs would be immune to this bot. Not only that, but because we were diplomatic regarding the removal of the bots and our requests for information, we avoided being attacked.

Information Leakage

It is hard to use the Internet without hearing about the dangers of identity theft. IRC is no different in this respect than the World Wide Web. The following points of advice will help prevent personal or corporate information loss:

- Remember that in most cases, IRC is a plaintext, unencrypted protocol. Where available make use of SSL-based IRC servers and use an SSL-based chat client. This will mitigate packet-sniffing attacks and other *man in the middle* attacks that unscrupulous third parties may try against you.
- Avoid giving away any personal details. Just because one of the fields in your IRC client is labeled **real name**, this does not mean your real name should be placed in it. The same goes for e-mail fields (unless of course you like to receive unsolicited e-mail).
- Where possible, block outbound DCC (Direct Client Connection) sends to prevent intentional or accidental leakage of files to third parties.
- Avoid visiting any URLs given to you on IRC. Even ones that look trustworthy could be designed to trick you into visiting them, which could then launch a Trojan or malicious script on your computer to steal information from you.
- Where possible, make use of a *bnc* (a program designed to cloak your IRC identity) to hide your hostname/IP address from other users. This costs extra money, but the benefits on larger networks outweigh the relatively small cost. Avoid the temptation to make use of *kiddie* virtual hosts, such as *my.momma.is.better.than.your.momma.com* and instead choose a virtual host that is more innocuous, such as *ppp94.someisp.com*, which will not draw the attention of malicious users.

Copyright Infringement

Because IRC allows users to transfer files to one another, it quickly becomes a vector for copyrighted materials (warez). These materials can be traded by large groups on even larger channels (easily noticed) or, more innocuously, person-to-person between friends. The majority of trading of warez on IRC occurs over DCC. DCC supports chat connections and file send/receive connections, the latter of which is used to transmit files. Because the nature of these protocols means the actual file data does not touch the server, only the initial handshake is sent across the network and can be detected (or blocked) by an IRC administrator.

Usually, the blocking of file types or file names does not prevent warez being propagated. To do so requires ingress/egress (inbound and outbound) filtering at the gateways of organizations that wish to prevent such activities. By blocking all but a subset of authorized inbound port numbers, file transfers can be prevented (See the “Firewall/IDS Information” section later in this chapter).

To prevent copyright infringement over DCC, we must understand to some extent the protocol being used so we can take action against it. A certain amount of information is given in the initial handshake for a DCC send, as shown below:

```
PRIVMSG Receiver :DCC SEND windowsxpro.exe 16777343 5627 651983911
```

The first parameter can be filtered at the server side, either by extension or by name, but as stated above, this should never be relied upon – as well as annoying users and making life difficult, it is easily circumvented. The second parameter to this command is the IP address where the send is coming from. If you were interested in determining who is distributing files on an IRC network you could very easily log these if your IRCD (Internet Relay Chat daemon) was correctly modified to do so. The IP address is in network byte order (usually reversed on Intel platforms). Converting this number to hexadecimal gives 0100007Fh, which is converted to the IP address 127.0.0.1. We can therefore determine that this send is occurring locally. The third parameter (5627) is the port number. This is randomly chosen by the client (usually within a port range). The final parameter is the file’s size in bytes. The DCC protocol only supports sending one file per request.

Other Forms of Infringement

IRC is also used as a swap meet for those wishing to infringe copyrights in other ways. Websites and FTP (File Transfer Protocol) sites trading illegal materials can be exchanged between users. The only way to prevent this is by setting up server-side filtering, a feature commonly supported by IRC servers. Two filters are UnrealIRCD

(www.unrealircd.com) and InspIRCd (www.inspircd.org). The filtering option is only available to you if you are the administrator of the server in question. If you are not the administrator of the server, the only way to detect such activity is by policing the IRC channels and watching for malicious activity. Be careful to follow the same methods of infiltration discussed earlier and hide your identity from the troublemakers to avoid future reprisals!

Where copyright materials are being traded via FTP sites and websites, and are not affiliated with the IRC server or network itself, it is usually not the responsibility of the IRC network to deal with removal of the illegal material. In this case, you must approach the netblock owners of the site's IP address or the domain technical contact. A good way to do this is via the website whois.sc or by issuing the UNIX WHOIS command, for example:

```
[craig@server:~]$ whois syngress.com
```

```
Whois Server Version 1.3
```

```
Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net for
detailed information.
```

```
Domain Name: SYNGRESS.COM
Registrar: NETWORK SOLUTIONS, LLC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS1.CONVERSENT.NET
Name Server: NS2.CONVERSENT.NET
Status: REGISTRAR-LOCK
Updated Date: 19-may-2005
Creation Date: 10-sep-1997
Expiration Date: 09-sep-2005
```

```
>>> Last update of whois database: Mon, 27 Jun 2005 16:13:10 EDT <<<
```

```
Registrant:
SYNGRESS Media, Inc.
145 Washington Street
Norwell, MA 02061
US
```


Domain Name: SYNGRESS.COM

Administrative Contact:

SYNGRESS Media, Inc. amy@syngress.com
 145 Washington Street
 Norwell, MA 02061
 US
 (617) 681-5151 fax: 999 999 9999

Technical Contact:

Network Solutions, LLC. customerservice@networksolutions.com
 13200 Woodland Park Drive
 Herndon, VA 20171-3025
 US
 1-888-642-9675 fax: 571-434-4620

Record expires on 09-Sep-2005.

Record created on 10-Sep-1997.

Database last updated on 28-Jun-2005 04:54:36 EDT.

Domain servers in listed order:

NS1.CONVERSENT.NET	216.41.101.15
NS2.CONVERSENT.NET	216.41.101.17

In this case, to contact someone capable of helping us at the domain, we could e-mail customerservice@networksolutions.com, or call 1-888-642-9675. It is the responsibility of domain owners to keep this information accurate. If this information does not help you, you should look up the netblock owner instead:

```
[craig@server:~]$ host syngress.com
syngress.com has address 155.212.56.73
syngress.com mail is handled (pri=20) by spool.conversent.net
syngress.com mail is handled (pri=10) by mailhost.syngress.com
[craig@server:~]$ whois 155.212.56.73
Conversent Communications CONVERSENT-155 (NET-155-212-0-0-1)
155.212.0.0 - 155.212.255.255
Syngress Publishing OEMN-155-212-56-64 (NET-155-212-56-64-1)
155.212.56.64 - 155.212.56.79
```

From this netblock owner (OEMN-155-212-56-64) we can determine contact details. Usually these details are listed within the WHOIS data, which makes the information easy to obtain.

If you do not have access to a UNIX system, there are many online WHOIS utilities that can be accessed using a Web browser. One of these (possibly the most well-known) is www.whois.sc (the WHOIS Source) which is provided by Name Intelligence, Inc. This service is free, but if you make a lot of queries you may be required to register (registration is also free).

Tools & Traps...

Trusting WHOIS

The methods discussed for obtaining contact details can be used just as effectively against users sharing files via DCC. Usually, you will need to initiate a DCC send from them to obtain their IP addresses for this activity (see above). Please be aware, however, that *information gathered via WHOIS queries is only as honest as the person that submitted the information*. False information can be entered into the WHOIS fields and correct information can rapidly become out of date on the ever-changing landscape that is the Internet. If you are in doubt as to the trustworthiness or accuracy of the information you are given, seek the information of the next person or group in the chain, such as a user's ISP (Internet Service Provider) or hosting provider.

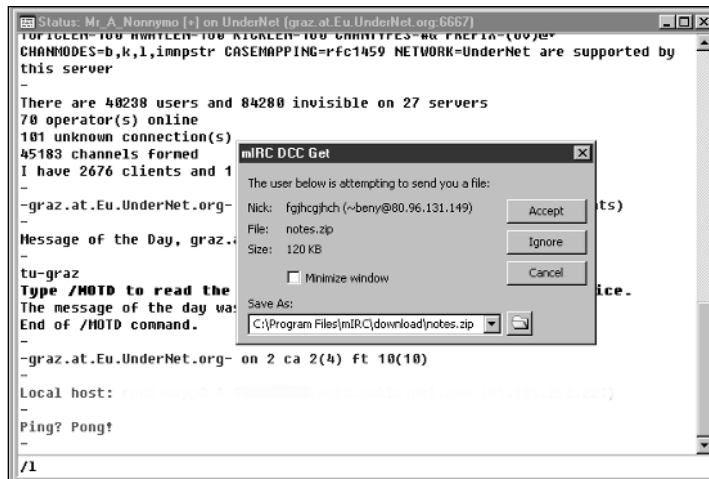
Transfer of Malicious Files

As IRC can be used to transfer files, it will probably not be any surprise to you that some of these files are intentionally malicious. From Trojans to viruses and joke programs, everything is out there and a lot of it is designed to do your network serious harm. Be on the lookout for executable files being transferred over IRC. As always, this is not as clear-cut as you would expect. For example, on the Windows platform (which is used by the majority of IRC users) there are several dozen types of executable files. These include:

- **Standard executables** These have the extension .exe, and contain directly executable code. These are difficult to audit.

- **Document files** Usually with the extensions .doc, .xls, or .mdb, these office files can contain viruses in the form of macros or even embedded executables and/or exploits.
- **Shortcuts** A shortcut can point at the command line interpreter, command.com or cmd.exe, and be given malicious parameters. They have the file extension .lnk.
- **URL shortcuts** These files (with the .url extension) can be made to point at malicious sites or local files.
- **INI files** Believe it or not, many clients such as mIRC use the .INI file type to contain their scripts. Such scripts can hold malicious code.
- **VBS files** These interpreted files can be executed by windows machines and can (in fact, usually do) contain malicious code.
- **BAT files** Batch files are an antique format, a throwback to the heady days of DOS (Disk Operating System), but they still have potential to wreak havoc. Any of these you receive over IRC are likely to contain malicious instructions.
- **SHS files** Once saved to disk on a normal Windows machine, these files will not show their file extensions, even if the file extensions are set to be displayed. They are equivalent to shortcuts and may (probably will) contain malicious instructions.
- **HLP files** These files contain Windows help data. They can also contain macros and native executable code.
- **CHM files** These next generation Windows help files are compiled and compressed HTML (Hypertext Markup Language). The compilation does not remove any of the nasty things you can hide within a HTML page.
- **HTM, HTML files** These pages can contain various exploits and scripts (such as JavaScript) that may do untold damage to your PC if they contain malicious data.
- **JPEG files** Believe it or not, recent Microsoft vulnerabilities in the JPEG code have resulted in the ability to manufacture JPEG images that can execute native code! So long as you keep your system updated, this should not affect you.
- **PLS files** These files are WinAmp (www.winamp.com) play lists. Several versions of WinAmp (a popular music player) have had vulnerabilities and these play list files can be malformed to execute native code and/or crash programs.

Figure 15.1 An Example of a Malicious File Transfer



Notes From the Underground...

Double Extensions

Be aware of double extensions. The double extension is a social engineering trick used to trick users into opening files they would otherwise ignore, for example a file called `mypic.jpg.shs`. As described above, the `.shs` part of the filename will be hidden from view, and the user will see `mypic.jpg`, and may access the file, hoping to see the picture contained within. Unfortunately, what awaits the user may be a less than pretty fate if they open this file.

How to Protect Against Malicious File Transfers

If you must allow file transfers (see the “Firewall/IDS information” section below for information on how to prevent them), make sure you always keep your antivirus software up to date. A good commercial antivirus solution such as Symantec Anti-Virus (or even a free solution such as AVG in a home user environment) will prevent the majority of IRC viruses from getting a foothold on your network.

What to Do if a Malicious File Infects Your Network

If you find that you have taken all possible precautions, but still a malicious file has infected your network, the most important thing to do is to not panic. Remain calm and cool, and follow the steps below:

1. Immediately unplug the machine from the network. Do not rely on any software controls to disconnect the machine. Only a physical measure such as unplugging the machine can be guaranteed to stop the spread of a virus or stop remote control programs from receiving signals.
2. Switch the machine into safe mode (or single user on UNIX) and attempt to clean the machine. You may wish to simply re-image the machine from a known clean source rather than risk further infection (in fact, this is highly recommended once you have ensured you have a current and reliable backup of the data on the machine).

Prevention of Malicious File Sends in the Client

There are many facilities for preventing the receipt of malicious files, but as these vary from client to client, we will focus only on the most popular Windows client, mIRC. The mIRC client has the ability to deny receives by extension and by nickname. By default it refuses most malicious file extensions (it implements a *white list* and only allows extensions that are on the white list). In its default configuration, mIRC is secure as long as no files you receive are designed to exploit bugs (for example malicious JPEGs designed to exploit recent Microsoft Windows vulnerabilities).

DCC Exploits

Recently there have been exploits within the most popular clients such as mIRC, that cause the clients to crash when sent certain DCC requests, as shown below:

```
PRIVMSG Receiver :DCC SEND crash.exe 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

By repeating the zeroes, a buffer within mIRC can be overflowed causing a crash. As always, the only way to prevent such attacks is to upgrade the software regularly.

Firewall/IDS Information

As with all modern Internet activities, IRC requires that you properly secure your system with a firewall, and optionally an IDS (intrusion detection system) program. Such programs have to be configured to accept IRC, as IRC has its many quirks and differences. The first of these is DCC.

If you wish to prevent users from sending files out over IRC, you should block DCC. DCC works by establishing an inbound connection, where the sender creates a listening socket and the receiver connects to that socket if he/she accepts that file. Therefore, blocking inbound connections prevents files being sent out. However, this will not stop receipt of files. The only safe and reliable way to block receipt of files is with ingress filtering (blocking outbound data). For use of IRC itself, the following firewall rules should be set:

- Outbound connections on port 6667.
- Optionally, outbound connections on other IRC daemon ports.

Usually it is required that IRC users run an ident daemon. IRC administrators use this daemon to identify users on multiuser systems. Although *largely deprecated* due to wide-scale use of single-user Windows machines, it is still widely used on IRC. If you do run an ident daemon, you should open port 113 for inbound connections at your gateway/firewall device. Most IRC clients that run on Windows (such as mIRC) come with a built-in ident server that serves requests for ident from IRC connections.

Port Scans

Whilst on IRC you will likely be subjected to a myriad of port scans (repeated TCP or UDP—User Datagram Protocol—probes designed to detect which daemons you are running). The large majority of these port scans will be harmless, and in fact, the majority of IRC networks will actually port scan you when you connect to them to detect open proxies. It is perfectly safe to block these port scans, but be aware that *adaptive* firewall technology such as that used by certain desktop products may accidentally block your access to the IRC server itself, so be sure to set the IRC server in your *trusted* list of sites.

IDS

IRC can pose a large risk of attack from third parties intent on breaking into machines just because they can. A good IDS utility, such as samhain (<http://la-samhna.de/samhain/>) can make the difference between total loss of data and an easy

recovery from an attack. IDS utilities such as these will notify you of changes to your files. Usually, when you receive an alert from such a utility, it is already too late and your system may already be compromised, so you should shut it down immediately. However, this extra time between the warning being sent and any real damage occurring gives you chance to mitigate the attack somewhat by disconnecting and quarantining the affected machine. Never ignore IDS warnings, via e-mail or otherwise. Instead, investigate the problem immediately and take action if any problems are found.

IRC causes no special problems to the operation of IDS and your IDS utilities require no special configuration. However, most people do not run IDS, and while on IRC you are encouraged to run such a utility.

Summary

In this chapter, we have introduced the basic concepts of IRC bots and malicious use of IRC, and how to prevent against it. We have discussed potential pitfalls that IRC users will encounter, such as viruses, worms, Trojans and DDoS bots, and how to be prepared for them.

We have also identified the difference between classic DDoS bots (such as Trin00) and IRC DDoS bots, which, while similar in malicious uses, function in a very different way by presenting an IRC-based front end to the attacker.

Solutions Fast Track

DDoS Botnets Turned Bot-Armies

- ☑ On IRC, a large number of abusers are involved in a method of abuse known as *botnets*.
- ☑ Before infection, the bot program is configured to connect to a given IRC server. Many botnet masters will configure their bots to connect to a dynamic DNS so that if they are banned from an IRC network, they can change their own DNS to move their bots elsewhere quickly.
- ☑ At the point of infection and on startup, each infected machine will connect silently to the IRC server.
- ☑ The bots on the infected machines will then usually join an IRC channel (which is usually keyed or otherwise restricted) and await commands.
- ☑ To attack one or more hosts, the botnet master will join the channel where the bots are waiting, and after initially sending a password to the bots will in most cases issue commands to start an attack.

Information Leakage

- ☑ In most cases IRC is a plaintext, unencrypted protocol.
- ☑ Avoid giving away any personal details. Just because one of the fields in your IRC client is labelled **real name** this does not mean your real name should be placed in it.

- ☑ Where possible, make use of a *bnc* (a program designed to cloak your IRC identity) to hide your hostname/IP address from other users.
- ☑ Avoid visiting any URLs given to you on IRC. Even ones that look trustworthy could be designed to trick you into visiting them.

Copyright Infringement

- ☑ Most copyright infringement occurs over IRC through the DCC send protocol, but it can occur off-IRC using IRC as a discussion medium.
- ☑ The best way to prevent warez from entering your network is by filtering.
- ☑ WHOIS lookup tools can be an asset when determining contact details when dealing with copyright infringement and other cases of abuse.
- ☑ The DCC protocol provides information to the IRC server, which can be used to selectively filter files.

Malicious files being transferred

- ☑ Malicious files on IRC use a variety of different file extensions. They are not all executables.
- ☑ When a PC is infected with malicious software, remove it from the network/Internet immediately.
- ☑ Most IRC clients have built-in mechanisms to prevent the transfer of malicious files.
- ☑ The DCC protocol itself can be exploited if your client is not kept current.

Firewall/IDS Information

- ☑ If you wish to prevent users from sending files out over IRC, then you should block DCC. DCC works by establishing an inbound connection, where the sender creates a listening socket and the receiver connects to that socket if he/she accepts that file.
- ☑ The only safe and reliable way to block receipt of files is with ingress filtering (blocking outbound data).
- ☑ Usually, it is required that IRC users run an ident daemon. IRC administrators use this daemon to identify users on multiuser systems.

- ☑ While on IRC you will likely be subjected to a myriad of port scans (repeated TCP or UDP probes designed to detect what daemons you are running).

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

Q. Why are there RFC documents for IRC, but none listed for DCC?

A. This is because the DCC protocol was not officially submitted to the RFC editor. The unofficial protocol specifications for the DCC protocol may be found at www.irchelp.org/irchelp/rfc/dccspec.html.

Q. What should I do if I find IRC DDoS bots or similar tools on a compromised machine that I am responsible for?

A. You should treat this the same way you would treat a virus infection and remove the tools immediately to prevent them being used any further. Refer back to the “What to do if a malicious file infects your network” section for more information.

Q. Why don't the larger IRC networks such as IRCNet and Efnet filter the types of files being sent across their network?

A. This is for many reasons. The first and most important reason is that each server is a separate entity, connected only by its TCP connections and only extremely loosely by online politics. This means that each server on the network would have its own opinions as to which file types and which names should be blocked. The IRC server software that these networks use is also incapable of blocking file transfers (they simply do not have the required feature set) and due to the massive number of clients upon these networks, there are valid concerns that such systems, while annoying many users, will actually increase processor load on servers to unacceptable levels.

- Q.** Given the choice of filtering at the server and filtering at the client, which should I choose?
- A.** This depends very much upon what you wish to filter and for what reasons. On IRC, most filtering is of malicious or illegal content, and wherever possible should be done at the server side. The main reasons for this are:
- Server-side filters cannot be overridden through user actions (accidentally or on purpose).
 - Server-side filters can be updated and managed centrally to deal with new threats.
 - Server-side filters are available no matter what client is in use by each user.

Common IRC Clients by OS

Solutions in this chapter:

- Windows IRC Clients
- UNIX IRC Clients
- Apple Macintosh IRC Clients
- Other IRC Clients

- ☑ Summary
- ☑ Solutions Fast Track
- ☑ Frequently Asked Questions

Introduction

This chapter will cover a sizeable number of available clients that can be used to access IRC (Internet Relay Chat). Each client has its own merits and benefits, however only security aspects of these clients will be discussed here, as anything else is not only off-topic, but also way beyond the scope of this book. As with most software, the majority of available IRC software is for the Windows platform, even though IRC can trace its roots deep within UNIX. Apple computers have the fewest number of native IRC clients (possibly due to the fact there are fewer Apple users).

Windows IRC Clients

As Windows is the most popular operating system available today, it is no surprise that this platform has the most IRC clients. This platform's clients range from the complex (X-Chat) to the very simple (Trillian). If you use Windows, you will have no trouble finding an IRC client to suit your individual needs.

mIRC

mIRC is the most popular IRC client for the Windows platform. With a multitude of features, this makes it more difficult to secure than its counterparts, however it also includes more capabilities with which to protect itself from attack. Its built-in scripting language is both a blessing and a curse for security, as it can be used to write scripts that will make the client more secure, while at the same time, there are viruses circulating in the wild that, while small and harmless-looking, can do some pretty nasty things, as shown below:

```
[00:15] <Wolfwood> I got owned by I got owned by
$(($decode(JGZpbmRmawxlKc4sKiwxLHNjAwQgLWF0MSAuYW1zZyBJIGdvdCBvd25lZCBieSAkIW
NikDEpKQ==,m),2)
```

These lines, when typed into the client (usually accompanied by some enticing reason to enter them, such as **shh, type this command and get ops**) cause insertion of the command into the client itself, perpetuating the virus. Such attacks are simple, but effective. In the past, mIRC has had several vulnerabilities that have all been patched in the next version of the software, some of these causing the client to crash.

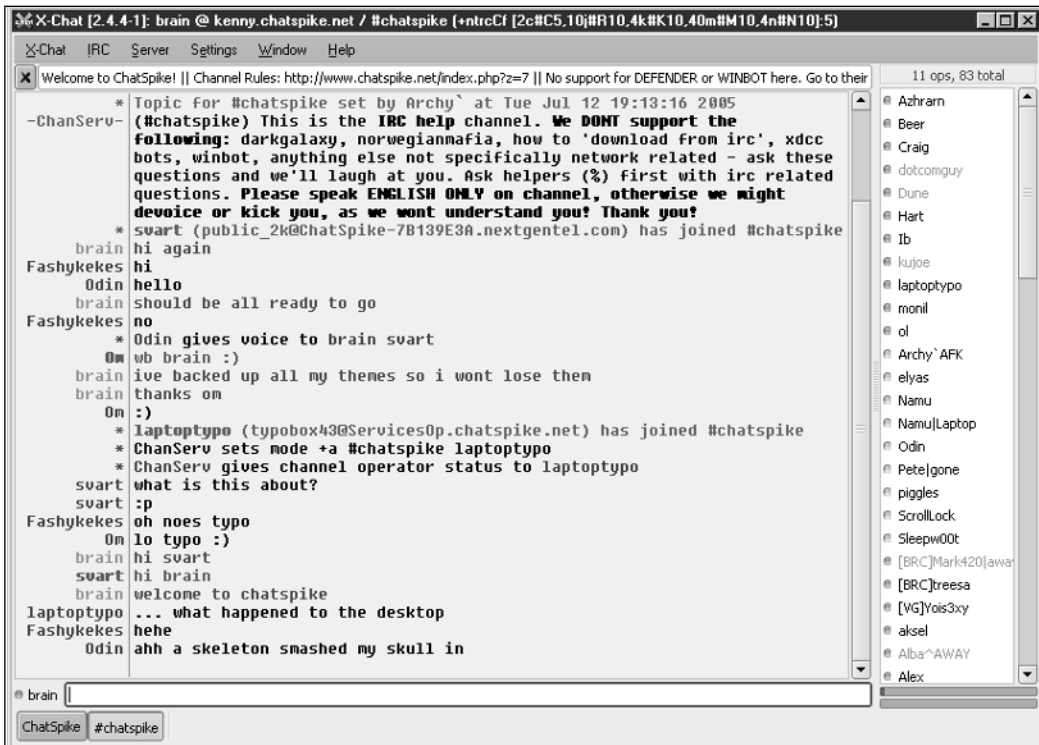
The large majority of DDoS (Distributed Denial of Service) botnets are constructed of bots built from mIRC, using its scripting. This is not necessarily due to the fact that mIRC's scripting language is designed for such activities (in fact the

X-Chat

X-Chat is possibly a close second behind mIRC in terms of popularity, but as it requires more patience and knowledge of IRC to set up effectively than its counterparts, its user base tends to have less security issues on IRC. Over the past few years, X-Chat has had relatively few vulnerabilities compared to other clients, however it has relatively few features in comparison. Supporting Perl, Python, and TCL for scripting, and with a plugin system, it can be expanded by a competent scripter in a relatively short amount of time to support a large number of security features.

X-Chat (Figure 16.2) supports SSL connections, as recommended by this book, without any extra third-party libraries. The official build of this client on the Windows platform is shareware, expiring after 30 days of use. However, if you are feeling adventurous, you can build it yourself from source (www.xchat.org) or you can download other builds for free (www.silverex.org). As this project is under GPL (GNU Public License), the shareware title on the Windows platform is simply to recoup some of the losses (in time and effort) of making a Windows binary and installer for each build.

Figure 16.2 X-Chat Client



Opera IRC Client

The Opera IRC client is part of the latest versions of the Opera Web browser (www.opera.com). Essentially an add-on, this client offers DCC (Direct Cable Connection) transfers, but no support for scripting. This client is a commercial product, which requires payment to use (or use of a freeware version of opera that is supported by advertisements), and does not yet support SSL.

ChatZilla

The ChatZilla IRC client is part of the Mozilla suite (also a Web browser, available from www.mozilla.org). Also an add-on, this client offers DCC support, but no support for scripting, similar to most add-on clients. This client supports SSL connections natively, and is free under the terms of the MPL (Mozilla Public License).

WinBot

WinBot (www.winbot.co.uk) is an IRC bot designed to secure IRC channels. It offers very little in the way of actual chatting capabilities, but it provides security features that other authors only bundle as add-on scripts or expect you to create yourself. This client supports scripting, either in its own proprietary language, or in Perl, plus it supports plugins.

This client is freeware (without source code) and does not yet support native SSL connections. SSL support is planned in a later version of the program.

Visual IRC (vIRC)

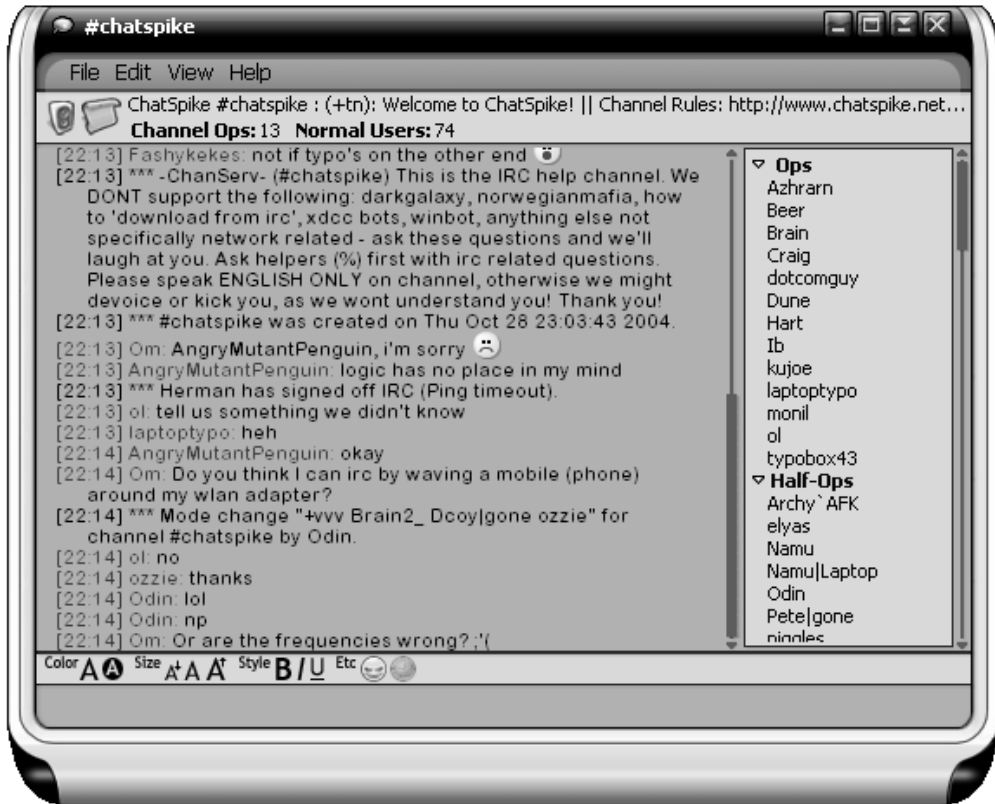
Visual IRC (www.visualirc.net) is a freeware IRC client with support for DCC. It does not yet support SSL, but has support for scripting. As this client's user base is slightly smaller than that of the others, it is less of a target for malicious users.

Trillian

Primarily an Instant Messaging (IM) client (see the first section of this book, which is dedicated to instant messaging programs), Trillian (Figure 16.3) also supports IRC. It supports not only DCC, but also *encrypted* DCC (which is, unfortunately, proprietary and only compatible with other Trillian users). It has very minimal and somewhat unstable built-in scripting support, which is officially deprecated. However, its support for SSL and SSL-encrypted DCC sessions are somewhat redeeming. Please note that in the past, this client has had security vulnerabilities, so please keep an eye on the vendor site (www.trillian.cc) for any patches that may arise.

This client comes in two versions, a free version and a full version. Both are identical in terms of IRC, the extra features for paying users being in the Instant Messaging sections of the program.

Figure 16.3 Trillian



UNIX IRC Clients

Because IRC originated on the UNIX platform, there are many IRC clients available for it. Most of these are text-based terminal programs, such as the simplistic ircII, while others are advanced GUI (graphical user interface) clients such as KVirc. There are many other IRC clients that fill the gap in between such as IRSSI and BitchX.

X-Chat

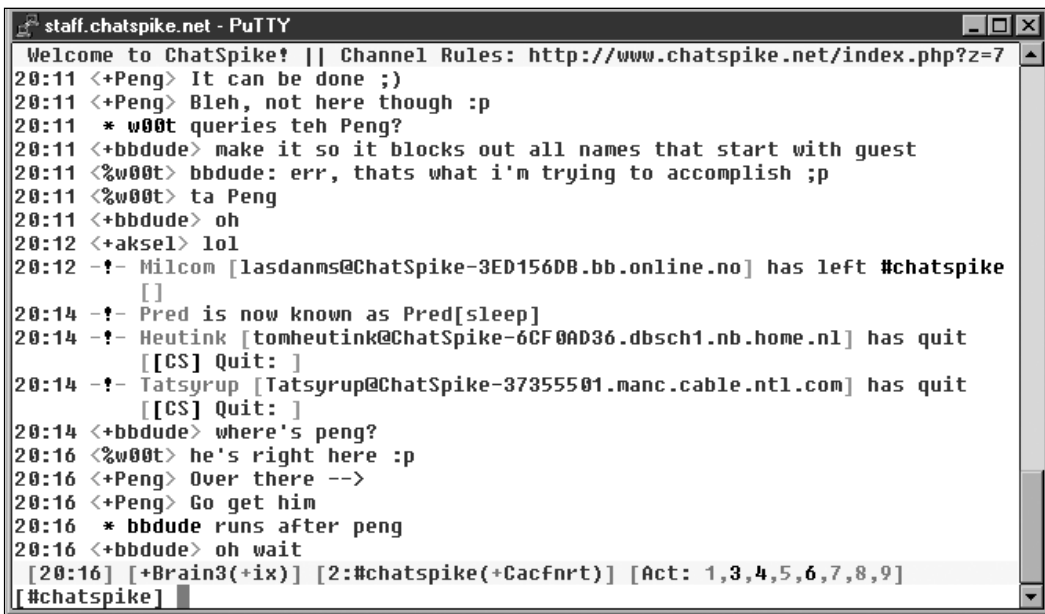
Similar in operation to the Windows counterpart, X-Chat on UNIX has native SSL support *if your system has OpenSSL libraries*. It also has DCC send support, and Perl

scripting. To script in Perl, the Perl language must be installed on your system. Unlike the Windows version, X-Chat for UNIX is free under the terms of the GNU General Public License and is available from www.xchat.org.

IRSSI

IRSSI (www.irssi.org) is a console UNIX client that has Perl scripting and DCC support. Rock-solid and stable, this client (Figure 16.4) is lightweight and fast (due to the fact it has no graphical interface). It supports SSL natively if your system has the OpenSSL libraries.

Figure 16.4 IRSSI



```

staff.chatspike.net - PuTTY
Welcome to ChatSpike! || Channel Rules: http://www.chatspike.net/index.php?z=7
20:11 <+Peng> It can be done ;)
20:11 <+Peng> Bleh, not here though :p
20:11 * w00t queries teh Peng?
20:11 <+bbdude> make it so it blocks out all names that start with guest
20:11 <%w00t> bbdude: err, thats what i'm trying to accomplish ;p
20:11 <%w00t> ta Peng
20:11 <+bbdude> oh
20:12 <+aksel> lol
20:12 -!- Milcom [lasdanms@ChatSpike-3ED156DB.bb.online.no] has left #chatspike
20:14 -!- Pred is now known as Pred[sleep]
20:14 -!- Heutink [tomheutink@ChatSpike-6CF0AD36.dbsch1.nb.home.nl] has quit
20:14 -!- Tatsyrup [Tatsyrup@ChatSpike-37355501.manc.cable.nt1.com] has quit
20:14 <+bbdude> where's peng?
20:16 <%w00t> he's right here :p
20:16 <+Peng> Over there -->
20:16 <+Peng> Go get him
20:16 * bbdude runs after peng
20:16 <+bbdude> oh wait
[20:16] [+Brain3(+ix)] [2:#chatspike(+Cacfnrt)] [Act: 1,3,4,5,6,7,8,9]
[#chatspike]

```

BitchX

BitchX (www.bitchx.org) is an established console UNIX client. It supports DCC and scripting. It is relatively stable, but, the phrase “user beware” applies to this IRC client. Packed full of *kiddie* features and heavier than a sizeable elephant, this client could very easily turn around and bite you in the nether regions security-wise. It has many settings and options that are all enabled by default, such as flood detection, exploit detection designed to detect exploits of other clients, and a variety of takeover commands for messing up a channel. While some users swear by this client,

you should make sure you know what you are doing before you use it. In the past, hackers have exploited it by changing the configuration scripts so that a backdoor is installed along with copies of the program. There are a few modified copies of this program circulating that contain backdoors, so only ever use the official site for this client even though it is GPL. At the last check, this client does not support SSL.

KVIrc

KVIrc is a graphical UNIX client utilizing the Qt toolkit. It supports DCC, SSL, and scripting (Perl and its own proprietary language), and requires the OpenSSL libraries installed on your system to utilize SSL. This client also features SSL-encrypted DCC, but it is not compatible with Trillian's EDCC (encrypted DCC) protocol due to differences in the handshake used to initiate the chat.

sirc

The sirc client (www.iagora.com/~espel/sirc.html) is programmed and extendable in Perl, and is a console client. It supports DCC, but at present does not support SSL, and is of course scriptable, being written in a scripting language itself.

ircII

The ircII client is possibly the oldest usable client available today. It is a console UNIX client with a very basic feature set, supporting DCC and scripting. This client has been vulnerable to exploits in the past, but current versions are stable.

Apple Macintosh IRC Clients

The Apple Macintosh is possibly the least-used operating system covered in this chapter. Because of this there are fewer IRC clients available for this platform. However, the ones that do exist are of good quality and are generally stable and well designed with many features not found elsewhere. Examples of this are the proprietary extensions of ChatNet and Snak.

ChatNet

ChatNet is a commercial IRC client offering among other things, chat over a LAN (local area network) using AppleTalk (this is separate to its IRC functionality and not directly compatible with IRC). The full version of this client is \$25 and oriented toward channel moderation. This client supports DCC (either in its 'IRC' mode and its 'AppleTalk' mode), however it does not support scripting or SSL.

Snak

The Snak IRC client (available from www.snak.com/Snak.html) is a graphical client that is released under a shareware style license. It functions for 30 days then requires a payment of \$20 for continued use. It is scriptable using AppleScript and the same language as the UNIX ircII client, and supports DCC. It has the ability to download MP3 files from channels that list them.

Homer

Homer is a shareware IRC client. Its website shut down long ago, but if you can find a copy, it will still work on some IRC servers (not Undernet). From a security standpoint it is not recommended, as it is no longer maintained and could therefore contain unfixed exploitable bugs.

Ircle

Ircle bears the logo “Probably the most popular IRC client for Mac OS.” It is a shareware client with a 30-day evaluation period. This client supports DCC, but does not natively support SSL—to use SSL with Ircle, a third party program called ‘stunnel’ is required, available from <http://www.stunnel.org/>. It supports scripting (via AppleScript, which is similar in look and feel to BASIC) and is actively maintained.

MacIRC

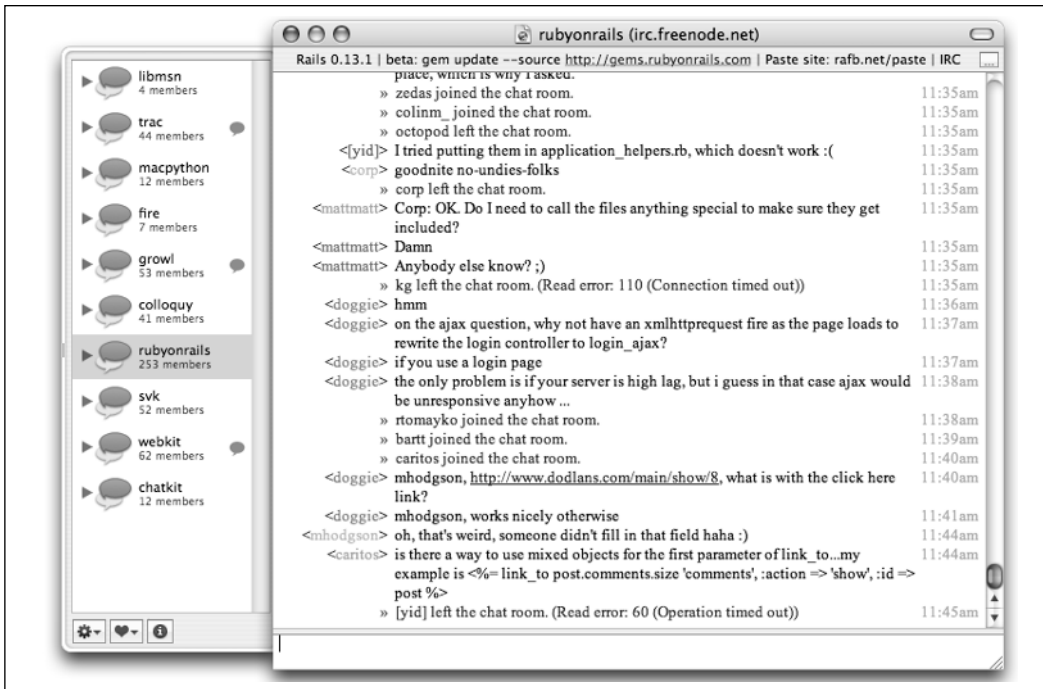
MacIRC is no longer maintained, and its website is no more, replaced by a squatter’s search engine. From a security standpoint we do not recommend this client, as it may contain unresolved bugs. A past copy of their page (obtained via archive.org) states, “As may have become clear from this Web site, MacIRC has not been in active development for several years. This situation was a result of work-related commitments of Chris Bergmann, its author; MacIRC was never a full-time job for him. For this reason, we will no longer be providing support for MacIRC, nor accepting shareware payments. Feel free to use MacIRC 0.9.7 and earlier for as long as you wish.”

This client supports scripting using AppleScript (if you can find a working trustworthy download for this client—no downloads or documentation for this client are on any form of *official* website any more) and it also supports DCC, however it does not support SSL.

Colloquy

Colloquy (Figure 16.5) is actively maintained, and supports both the IRC and SILC protocols (see below for a description of SILC). As with most Apple IRC clients, Colloquy is scriptable using AppleScript. Its IRC features are based upon the UNIX IRSSI client, supporting DCC file transfer and SSL IRC connections. Testimonials on its website state that it is “the best Apple-Scriptable IRC client for the Mac.” It is an open source client, released under the GNU General Public License. This client will only run on OS 10.3 or higher, as it takes advantage of features not available in older versions of Apple’s operating systems.

Figure 16.5 Colloquy



Other IRC Clients

Whenever an attempt is made to place things into categories, computer programs or otherwise, there is always a small group of programs that do not properly fit into any category. These clients are listed here. In this section, we will cover the miscellanies of Java IRC clients, CGI IRC clients, and the security-minded IRC-look-alike, SILC.

PJIRC

PJIRC (www.pjirc.com) is an open source java client. Its main advantage is that you do not need to install an application to make use of IRC using this applet. Many IRC networks provide an install of this applet for public use. It does not support SSL, but it does support DCC (not many java clients do). It is not scriptable, but the fact that it is open source (unlike many similar java clients) makes it extendable.

J-Pilot

J-Pilot is a shareware java IRC client that is provided without source (cabins/jars only). Its registration cost is \$49.99 and it is commonly found on IRC network sites (most of which have not paid this registration fee). It does not support SSL or DCC and is not scriptable. However, it can be extended using a JavaScript function embedded into a Web page. It is available from www.jpilot.com/products/jirc/index.html.

CGI:IRC

CGI:IRC (<http://cgiirc.sourceforge.net>) is a CGI IRC client written in Perl. It creates the connection to IRC itself by proxy, so if you host it on an SSL-enabled Web server, it can indirectly support SSL (to make it secure, it should connect to *localhost*). It does not and will not support DCC or scripting due to its nature, but is usable from practically any location with Internet access.

SILC

SILC is a collaborative effort (mainly by the GAIM authors) to create a separate, secure group messaging protocol. Although SILC is not IRC by definition (the protocol is in fact completely different) it deserves mention due to the fact it is designed to offer secure communications. It is released under the GNU General Public License and is available from www.silcnet.org. The concepts of SILC networks are very similar to IRC in that it has channels with operators, owners, and various things you will find similar to an IRC client, which is why this program is listed. It is not compatible directly with SSL-enabled IRC servers, as it has many differences from IRC, which make it unable to connect.

Summary

As we have seen in this chapter, there are many IRC clients to choose from, no matter what your operating system. From initially one IRC client on UNIX, many have sprung up for all platforms, supporting many differing features, but all are generally compatible in the sense that they will connect to IRC networks. The choice of IRC client is very much up to the individual user. From a security standpoint, avoid features you will not use. These will only serve as a possible vector for exploitation in the future, so if you aren't going to use a feature, don't load it, disable it, or choose a client without the feature.

Table 16.1 is a summary of the features of each client.

Table 16.1 IRC Clients and Features

Client Name	Operating System	Has SSL	Has DCC	Has Scripting
mIRC	Windows	Yes	Yes	Yes
X-Chat	Windows/UNIX	Yes	Yes	Yes
Opera IRC Client	Windows	No	Yes	No
ChatZilla	Windows	No	Yes	No
WinBot	Windows	No	Yes	Yes
Visual IRC	Windows	No	Yes	Yes
Trillian	Windows	No	Yes	No
IRSSI	UNIX	Yes	Yes	Yes
BitchX	UNIX	No	Yes	Yes
KVIrc	UNIX	Yes	Yes	Yes
Sirc	UNIX	No	Yes	Yes
IrcII	UNIX	No	Yes	Yes
ChatNet	Apple Mac	No	Yes	Yes
Snak	Apple Mac	No	Yes	Yes
Homer	Apple Mac	Unknown	Unknown	Unknown
Ircle	Apple Mac	No	Yes	Yes
MacIRC	Apple Mac	No	Yes	Yes
Colloquy	Apple Mac	Yes	Yes	Yes
PJIRC	Any	No	Yes	No

Continued

Table 16.1 continued IRC Clients and Features

Client Name	Operating System	Has SSL	Has DCC	Has Scripting
J-Pilot	Any	No	No	No
CGI:IRC	Any	No	No	No
SILC	Windows/UNIX	Yes	Yes	No

Solutions Fast Track

Windows IRC Clients

- ☑ mIRC is the most popular client, but this popularity comes at a steep price. It is often the most abused by malicious attackers looking for an avenue into a computer.
- ☑ Some IM clients, such as Trillian, include features for IRC connectivity even though they themselves are not primarily IRC clients.
- ☑ Most noticeable on the Windows platform, viruses and worms may be written purely in the scripting languages of IRC clients, and can be devastatingly effective.

UNIX IRC Clients

- ☑ A UNIX IRC client is just as likely to contain a backdoor in its package as a Windows IRC client. The IRC client BitchX is a constant target of such attackers, and packages for IRC clients should be fetched only from the official site.
- ☑ UNIX IRC clients, as with the entire UNIX operating system, are much more dependent upon external tools and libraries to function correctly. The Windows philosophy is to package such libraries with an application, whereas the UNIX philosophy is to make the user fetch such libraries separately. Always check these third-party libraries for exploits on a regular basis, especially in the case of SSL libraries.

- ☑ Many UNIX IRC clients run under terminal emulation rather than as graphical programs. This can be an advantage where a graphical desktop is unavailable.

Apple Macintosh Clients

- ☑ The majority of Apple Mac IRC clients are shareware. Be prepared to hunt around and check that there is no free client that does what you want before paying for software.
- ☑ Many Apple Mac IRC clients are no longer maintained (not enough time to develop the programs, advancement of technology, etcetera). However, use of an IRC client that is no longer maintained is not recommended under any circumstances, as from a security standpoint you are opening yourself up to attack via possibly exploitable code.
- ☑ The open source Apple Mac IRC clients receive large amounts of praise. Open source alternatives are usually a better choice from a security standpoint, as the ability to audit and fix the program yourself provides you with more insight into how it functions and ensures that patches and updates are released more often.

Other IRC Clients

- ☑ Although you can place java applet IRC clients onto secure websites (SSL websites), this does not make the applet's traffic secure, as the applet will most likely connect to IRC using an unsecured connection. Be aware of information leaks via unsecured connections in this manner.
- ☑ There are other collaborative efforts to produce secure alternatives to IRC, such as SILC. Note, however, that these alternatives are, in terms of the low-level protocol, nothing like IRC and only behave like IRC on the surface; you cannot use an IRC client to connect to a SILC server or any other similar project.
- ☑ It is possible to make a totally secure Web-based solution simply by placing a CGI:IRC client on a HTTPS (secured) Web server.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form.

- Q.** Does the fact that some IRC clients cost money make them better than free or open source alternatives?
- A.** No. You must judge the merits of each client yourself. If this chapter helps you with that task, it has achieved its goal.
- Q.** Are all possible IRC clients listed in this section?
- A.** No. To list all IRC clients and their derivative projects would take an entire book in itself. Such a book would also be out of date long before it even made it to print. The IRC clients listed here are the ones known to the authors with security-related merits and pitfalls.
- Q.** Is the SILC protocol likely to replace the IRC protocol as the de facto standard in the near future?
- A.** This is unlikely. There are many thousands of IRC networks based on the original IRC protocol (some with extensions and modifications of their own such as extra commands) and the original protocol is likely to remain the standard for some time to come.

Index

A

- A&M Records Inc. v. Napster Inc., 234
- ABC (Yet Another BitTorrent Client), 290
- AES (Advanced Encryption Standard), 182
- AIM (AOL Instant Messenger)
 - AIMDES worm, 39–42
 - DoS vulnerability, 44–46
 - features described, 31–37
 - file transfers, sharing, 35–36, 50
 - and ICQ, 136
 - interoperability of, 96
 - introduction, architecture, 26–30
 - monitoring with EtherBoss Monitor, 15
 - Oscarbot/Opanki worm, 43–44
 - protocol described, 30–31, 48
 - restricting ports, 50
 - security threats (table), 38–39
 - Velkbot Trojan, 43–44
 - versions, and security, 49
- AIM-Canbot Trojan, 16
- AIMDES worm, 39–42
- AimEncrypt encryption, 165
- AIM Express, 27
- Altnet Secureinstall, 232
- AltNet software, 328–329
- America Online. *See* AOL
- antivirus programs
 - and FastTrack, 344
 - Trillian's options, 167–168
- AOL (America Online)
 - Instant Messenger. *See* AIM
 - Buddy List, 26
 - and ICQ, 134
 - and Nullsoft, 241
 - protocols, 26–27
- Apple iTunes Music store, 236
- Apple Macintosh IRC clients, 428–430, 434
- application sharing
 - described, 9, 18
 - and MSN Messenger feature, 107, 108–110, 129
- architecture
 - AIM, 26–30, 45–46
 - BitTorrent network, 291–296
 - FastTrack network, 332–336, 354–355
 - Gnutella, 243–246, 264
 - MSN Messenger, 96–98, 128–129
 - peer-to-peer networks, 5, 10
 - Skype, 180–183
 - Yahoo! Messenger, 52–57, 90
- audio chat, 9, 20, 34–35
- audio file compression and file distribution, 222
- Audio Setup Wizard, Yahoo!
 - Messenger vulnerability, 88
- authentication

- and interoperability between clients, 23
- in MSN Messenger, 100, 111
- Avalanche vs. BitTorrent, 317
- Azureus client, 288–289, 293

B

backdoors

- and client security, 16
- and MSN Messenger Remote Assistance, 110

- Velkbot Trojan, 43–44

- W32.Chod.B@mm worm, 80

- WORM_CHOD.B worm, 149

bahamut IRC software, 369–370

bahamutircu, 370

bandwidth

- BitTorrent issues, mitigation, 310–312

- FastTrack issues, mitigation, 347–352, 355–356

Basic OSCAR Service (BOS) servers, 29

BBC (British Broadcasting Corporation), distribution of content, 237

Bearshare client, 242–243

Bencode format, 296

benign bots, 401

Betamax, Sony, 233–234

BitchX IRC client, 427–428

BitComet client, 289–290

BitTornado, 288

BitTorrent network

- architecture and data flow, 291–296, 313

- bandwidth issues, mitigation, 310–314

- described, 228, 230, 237

- features, security risks, 308–310

- history, clients for, 286–291, 314

- protocol analysis, 296–308

- torrent files, 297–299

black listing, 211

Blizzard Entertainment, 286–287

blocking

- AIM ports, traffic, 50

- MSN Messenger use, 105

- ports and services, Yahoo! Messenger, 87–88

- Skype, 211–212

- supernode-related packets, 356

- Yahoo! Messenger on network, 93

Blowfish encryption, 164

BOPM (blitzed open proxy monitor), 393

botnets, IRC

- controlling, 401–403

- DDOS, turned bot-armies, 400–401, 416

- described, types, 385, 388–393, 395–396

- reprisals, 404

bouncers, deploying, 379–380

Brilliant Digital Entertainment, 232, 328–329

buddy lists, 12, 26

buffer overflows, Skype vulnerability, 190–192

C

- CacheLogic, 268
- Cain and Able network monitor utility, 12, 171
- Callto Handling Buffer Overflow, 190–192
- cameras, Web. *See* Web camera
- cellular phones and IM clients, 22
- certificates, and AIM encryption, 33
- Cerulean Studios' Trillian, 160–161
- CGI:IRC IRC client, 431
- chanfix system, 380
- channel botnets, 390–392, 401
- channels, in IRC network, 364
- ChanServ service, 382–384
- chat rooms, Yahoo! Messenger, 64
- chat systems
 - See also specific client*
 - conferences. *See* conferences
 - Internet Relay Chat. *See* Internet Relay Chat (IRC)
 - text, group, video, audio, 9
- ChatNet IRC client, 428
- ChatZilla IRC client, 425
- clients
 - See also specific IM client*
 - for BitTorrent network, 286–291
 - choked by peer, 304
 - common P2P, and networks (table), 230
 - connecting to FastTrack, 337–339
 - eDonkey network, 268–274
 - for FastTrack network, 320–327
 - FastTrack supernode, 348
 - filtering, 419
 - Gnutella, 240–243
 - IP addresses revealed through usage, 14–15
 - IRC (table), 432–433
 - major instant messaging, 6–7
 - and malicious infections, 45
 - preventing interoperability between, 23
 - security, 16–17
 - services and networks, 6–7
 - spyware bundling and FastTrack, 328–332
 - third-party instant messaging, 10–11
 - UNIX IRC, 426–428
 - Web-based. *See* Web-based clients
 - which to choose, 22
 - Windows IRC, 422–426
- codes
 - malicious. *See* malicious code/software
 - MSN Messenger connection status (table), 101
- Cohen, Bram, 286, 287, 290, 313
- Colloquy IRC client, 430
- compressed files, decoding, 221
- conferences
 - described, 62
 - Skype, 186–189
 - Yahoo! Messenger, 63–64
- Connection Preferences option, AIM, 28–30

connections

- Goggle Talk, 168–170
- ICQ settings (fig.), 135
- NAT multiplexing, 193
- Skype settings (fig.), 183
- tracker, BitTorrent, 299–302
- Trillian properties (fig.), 161
- UDP vs. TCP (fig.), 199
- Yahoo! Messenger details (table), 55
- Consumer Empowerment company, 320–321, 323
- cookies and Yahoo! Messenger sign in, 53
- copying equipment, legal concerns, 233–234
- copyright infringement
 - and BitTorrent, 308–310
 - and eDonkey, eMule, 274–275
 - and FastTrack, 343
 - and IRC, 408–411, 417
 - and P2P networks, 224–227, 281
 - and peer-to-peer networks, 257
- Core Security Technologies, ICQ client vulnerabilities, 149–151

D

- DALnet network, 372, 381
- data hashing and FastTrack, 344–345
- data sharing and BitTorrent, 310–312
- DCC file transfers, 386, 414, 418

- DDOS (distributed denial-of-service) attacks, 392, 416
- decentralized networks, 333–334
- decoding compressed files, 221
- decrypting passwords on the fly, 12
- denial of service (DoS) attacks
 - and AIM, 44
 - BitTorrent and, 287
 - bot programs and, 402
 - and FastTrack, 347
- descriptor packets, Gnutella, 247–252
- DHT connections, and BitTorrent, 307–308
- dictionaries and Bencode format, 296
- Diffie-Hellman-based key exchange protocol, 163
- Digital Millennium Copyright Act (DMCA), 309, 330
- Digital Rights Management (DRM), 237
- Direct Revenue, 328
- directories, sharing, 256
- distributed denial-of-service (DDoS) attacks, 392, 416
- distributed hash table (DHT) technology, 288
- DMCA (Digital Millennium Copyright Act), 309, 330
- DRM (Digital Rights Management), 237
- dsniff packet capture utility, 12, 21, 27
- dynamic content and IM, 220

E

eDonkey network
 copyright issues, 235
 described, 227–228, 230, 280–283
 and eMule networks, 271–274
 features, security risks, 274–279
 history, installation, 268–271, 280
 vulnerabilities, 278–279, 283
 EFnet network, 372, 379–380, 394, 418
 eMule and eDonkey, 271–274, 280–283
 encryption
 Advanced Encryption Standard (AES), 182
 AIM, 32–33, 48–49
 AimEncrypt, 165
 Blowfish, 164
 described, 9, 18
 Exclusive-OR (XOR), 27
 Goggle Talk, 171
 ICQ, 137
 instant messaging's lack of, 15
 MSN Messenger, 106
 Skype, 182, 184
 Trillian, 161–165
 Web-based clients, 173
 Yahoo! Messenger, 53, 61
 EtherBoss Monitor, 15
 Ethereum traffic analyzer, 336–337
 Exclusive-OR (XOR) protocol, 27
 Extensible Messaging and Presence Protocol (XMPP), 168–169, 174

F

fake files
 and eDonkey, eMule, 276
 and FastTrack, 344, 357
 Fanning, Shawn, 223
 FastTrack network
 architecture, 332–336, 354–355
 bandwidth issues, mitigation, 347–352, 355–356
 features, security risks, 230, 343–347, 353
 and Gnutella network, 243
 history, clients for, 320–327, 354
 protocol analysis, 336–342, 355
 and ultrapeer, 226–227
 file extensions, double, 413
 file sharing
 AIM, 35–36, 50
 bots, 396
 copyright issues, 275
 described, 9, 18
 eDonkey, misconfigured sharing, 277, 282
 FastTrack network, 346
 Gnutella, 256
 “leeching” files and, 272, 294–295
 Skype, 188
 Yahoo! Messenger, 66
 Yahoo! Messenger connection, 56
 File Transfer Protocol. *See* FTP
 file transfers
 AIM feature, 35–36, 48
 described, 9, 19, 20, 65
 on FastTrack network, 339–341

- FTP and, 220
- and Gnutella networks, 244, 252–254
- hostile request for (fig.), 13
- ICQ feature, 140–141
- and instant messaging, 5
- IRC, 386–387, 395
- malicious files, and IRC, 411–413
- and malicious infections, 12–13
- MSN Messenger function, 112–113, 129
- peer-to-peer network vs. IM, 10
- revealing IP addresses, 257
- worms and, 13–14
- and Yahoo! Messenger, 56, 65, 93
- files
 - fake, 276, 344, 357
 - poisoned, 276, 281, 344
 - torrent, 297–299
- filtering server or client, 419
- firewalls
 - FastTrack rules, 348–349
 - Gnutella rules, 259–260
 - and IRC networks, 414–415, 417–418
 - and NAT, 192–197
- FLAP protocol, 27, 30–31, 48
- Ford, Bryan, 199–200
- Frankel, Justin, 240
- Free Peers, Inc., 242
- Freenet P2P network, 229
- FreeNode network, 373
- friend-to-friend (F2F) networks, 229

- Friis, Janus, 180, 320
- FrostWire, 241
- fserve bots, 388–389, 396
- FTP (File Transfer Protocol)
 - and AIM protocol, 33
 - and file distribution, 220, 223
 - and file transfers, 11–12, 65, 112
 - and ICQ file transfer, 140–141

G

- G3 Torrent client, 291
- Gaim client, 11
- GameSurge network, 372–373
- gateways
 - evading antivirus scanning, 12–13
 - security, protecting e-mail, 16
- Gator Advertising Information Network (GAIN), 328
- Global Index (Skype), 181
- GMail account and Goggle Talk, 160
- Gnucleus client, 243
- Gnutella
 - architecture, 243–246
 - clients and network, 240–243
 - features, security risks, 230, 254–257, 264–265
 - firewall port configuration, 261–262
 - more information about, 265
 - peer-to-peer network, 225–226, 240, 263–264
 - protocol, 246–254

technical countermeasures for,
257–262

Goggle Talk

- introduction, features, 160,
168–175
- security issues, 176

Grokster Ltd. v. MGM Studios Inc.,
234–235, 326–327

group chat

- AIM, 33–34
- described, 8–9, 20
- ICQ (Multi User Chat), 137–138
- Skype, 186–187

Group Policy, Skype settings,
202–205

H

handshake, BitTorrent, 302–303

hashes, and fake files, 344–345

HBO and poison peers, 309–310

Homer IRC client, 429

Hotmail e-mail service, 96, 102,
120–122

HTTP headers and network control
evasion, 87

HTTPS (HTTP over Secure
Sockets Layer)

- AIM-supported proxy, 29
- and Microsoft Passport
identification, 100–101
- and Skype, 182–183

hybrid networks, 332

Hypertext Transfer Protocol
(HTTP), 87

HTTP over Secure Sockets Layer
(HTTPS), 29, 100–101

I

ICQ

- client security, 149–154
- features, 135–141
- introduction to, history of, 8,
134–135
- malicious code, vulnerabilities,
141–154

ident, in IRC network, 364, 380

identity theft described, 12

IDSs (Intrusion Detection Systems)

- and Gnutella network, 259–260
- and IRC networks, 414–415,
417–418

Snort, 262

- utilities warning of file changes,
415

IM. *See* instant messaging

IM2 client described, 11

iMesh client, 327

IMlogic

- AIM protections, 50
- ICQ client security threats (table),
143
- Threat Center, 37

iMP distribution of BBC content,
237

infected files and peer-to-peer
networks, 231–233

installing P2P securely, 231–233

instant messaging (IM)

- See also specific programs*
- common security issues, 11–17
 - described, 3–5, 19
 - and encryption, 15
 - free software, 23
 - ICQ function, 136–137
 - major services, 6–7
 - and peer-to-peer networks, 10
 - popularity of, 7–8
 - security issues, 11–16, 21–22
 - third-party clients, 10–11
 - Web-based clients, 172–173
- Internet Relay Chat (IRC)
- Apple Macintosh clients, 428–430
 - botnets, types and uses, 388–393, 395–397
 - client summary (table), 432–433
 - copyright and other
 - infringements, 408–411
 - described, history, jargon, 5, 362–368, 374
 - and information leakage, 407, 416–417
 - networks and servers, 378–386
 - preventing malicious code, files, 411–415
 - server software packages, 368–375
 - and Trillian, 160
 - UNIX clients, 426–428
 - Windows clients, 422–426
 - worms and, 38
- Internet Security System's X-Force Database, 44
- Internet, the
- black and white listing, 211, 414
 - blocking with packet filters, 212
 - interoperability of instant messaging, 52, 96
 - intrusion-detection systems. *See* IDSs
 - IP (Internet Protocol)
 - and ICQ file transfer, 140–141
 - and peer-to-peer networks, 4
 - IP addresses
 - detection during usage, 14
 - file transfers revealing, 257
 - and Gnutella network, 243
 - and NAT, 192
 - ipbote botnet example, 404–407
 - iptables, string match module, FastTrack, 349–350
 - IRC. *See* Internet Relay Chat
 - ircd 2.11.x, 369
 - ircd-hybrid, 369
 - Ircle IRC client, 429
 - ircII IRC client, 428
 - IRCnet network, 372, 385, 395, 418
 - IRSSI IRC client, 427
 - iTunes, 236
- ## J
- J-Pilot IRC client, 431
 - Jabber client, 11, 168
- ## K
- K-Lite client, 332
 - Kademlia network, 269, 274

Kademlia overlay network protocol,
307–308

KaZaA, 226, 320, 323

Kazaa Lite client, 329–330

Kazaa Lite Resurrection (KLR),
331–332

Kazaa loaders, 330–331

Kazaa P2P service, 180, 232,
321–325, 356

keyboard loggers and client security,
16

Kopete IM client, 11

KVIrc IRC client, 428

L

leechers and seeders, 272, 294–295

legal concerns

- FastTrack network, 346
- file sharing, 275, 396–397
- Napster, copyright issues, 224–225
- peer-to-peer networks, 233–236

LimeWire file sharing program

- described, 240, 265–266
- firewall port configuration,
261–262
- Gnutella client, 241–242

loader utilities, Kazaa loaders,
330–331

logging

- message, 15, 106
- Skype chat history, 184–185
- Trillian's activity history, 165–166
- Yahoo! Messenger message
archiving, 61–62

LST command (MSN Messenger),
102–103

M

MacIRC client, 429

malicious code/software

- See also* Trojans, viruses, worms
- and AIM client security, 37–46
- and eDonkey, eMule, 275–276
- and FastTrack, 343–344

ICQ, 141–149

- and MSN Messenger, 114–115,
130
- and peer-to-peer networks,
231–233

Skype vulnerability to, 188–189

and Trillian client security,
166–168

transfer, and IRC, 411–413

Yahoo! Messenger, and client
security, 67–87

malicious users and ICQ instant
messaging, 136

message archiving

- ICQ, 138–139
- MSN Messenger, 106, 129
- Skype, 184–185
- Yahoo! Messenger, 61–62, 90

message logging, 15, 106

Messenger Key and password
recovery, 136

MetaMachine's eDonkey, 268–270,
282

- MFTP (Multisource File Transfer Protocol), 271
- MGM Studios Inc. v. Grokster Ltd., 234–236
- Microsoft
 - Avalanche, 317
 - MSN Messenger. *See* MSN Messenger
 - NetMeeting, 108
 - Passport identification, 99, 100
 - Windows. *See* Windows
- Mirabilis
 - and AOL, 134, 135
 - ICQ. *See* ICQ
- Miranda encryption plug-in, 165
- Miranda Instant Messenger, 11
- mIRC IRC client, 422–423
- mobile messaging, 10, 18
- Mobile Status Notification Protocol (MSNP) and MSN Messenger, 98–99
- Morpheus client, 243, 325–326
- MP3s
 - and AIM security, 33
 - Napster and, 223–224
- MSN
 - Messenger. *See* MSN Messenger
 - popularity of, 8
 - protocol, 103
 - vulnerability, 18
 - worm (fig.), 14
- MSN Messenger
 - application sharing, 108–110
 - architecture, 96–98
 - client security, 126–127
 - encryption, message archiving, 106
 - features summary, 128
 - file transfer functions, 112–113
 - instant messaging function, 104–105
 - introduction, architecture, protocol, 96–104
 - malicious code, security threats, 115–120
 - remote assistance function, 110–111
 - servers and functions (table), 97
 - W32.Kelvir.R worm, 120–122
 - W32.Picrate.C@mm worm, 122–126
 - Web camera settings, 114
 - whiteboard feature, 107–108
 - WORM_CHOD.B worm, 147–149
- MSN Track Monitor, 105
- MSNP (Mobile Status Notification Protocol), 98–99
- multi-network capabilities
 - described, 10, 20
- Multi User Chat (ICQ), 137–138
- multiplexing, 193
- Multisource File Transfer Protocol (MFTP), 271
- MusicCity, 326
- Mute friend-to-friend network, 229

N

Napster

- and centralized network design, 332–333

- file sharing, 223–225

- network and clients (table), 230

- v. A&M Records Inc., 234

NAT (network address translation)

- and firewalls, 192–197

- and Skype client, 195–197

NAT check tool, 199–200

.net Messenger, 96

- .NET Passport, MSN Messenger's use of, 99

NetMeeting, 108

NetSplit, 378

Netstat utility, 257

networks

- BitTorrent P2P, 228

- blocking Yahoo! Messenger on, 93

- common P2P, and clients (table), 230

- FastTrack and supernodes (fig.), 335

- friend-to-friend (F2F), 229

- Gnutella. *See* Gnutella

- hybrid, semicentralized, decentralized, 332–334

- IRC, 5, 371, 378–386, 394–395

- peer-to-peer. *See* peer-to-peer networks

- nick, in IRC network, 364–367

- NickServ service, 381–382

- node ping, pong payloads, 337–339

- Nullsoft, 225, 240

- Nutella, 241

O

- Oikarinen, Jarkko, 362

open source

- eMule, 270, 282

- Ethereal, 336–337

- and FreeNode network, 373

- Gnutella, 225

- Miranda Instant Messenger, 11

- vs. commercial IRC clients, 435

- OpenNap network, 224, 325

- OpenProjects, 373

- Opera IRC client, 425

- OSCAR protocol, 26–27, 134–135, 163

- Oscarbot/Opanki worm, 42–43

- OverNet network, 227, 230, 269, 272

P

- P2P. *See* peer-to-peer networks

- P2PWall tools, 350–352

packet capture utilities

- Cain and Able utility, 12, 171

- Ethereal, 105

- and identity theft, 12

- recording chats with, 137

packet filters

- blocking Internet with, 212

- and proxy servers, 193
- Passport identification, MSN
 - Messenger's use of, 100–102
- passwords
 - and AIM protocol, 27
 - decrypting, 12
 - and MSN Messenger sign-in, 100
 - and peer-to-peer networks, 4
 - recovering, 136–137
 - W32.Chod.B@mm worm theft of, 81
 - and Yahoo! Messenger sign in, 53–54
- Peer Directed Projects Centre (PDPC), 373
- peer-to-peer (P2P) networks
 - See also specific network*
 - concerns, vulnerabilities, 231–233
 - copyright issues, 223–225, 245–255, 281
 - and filesharing programs, 240
 - future of, 236–237
 - and instant messaging, 10
 - introduction to, 4–5, 220–223
 - leechers and seeders, 294–295
 - legal concerns, 233–236
 - poisoned files and, 276
 - swarming approach, 227–231
- Peer Wire Protocol (PWP)
 - messages, 302, 305–307
- peers
 - described, 223, 245
 - and leechers, 295
 - poison, 309–310
 - states, and BitTorrent, 304
- Pepper, Tom, 240
- phones and SkypeOut, SkypeIn, 180
- pings and pongs, 245
- PJIRC IRC client, 431
- platforms for IM clients, 22
- poison peers, 309–310
- poisoned files
 - and FastTrack, 344
 - and peer-to-peer networks, 276, 281
- pongs, pings, 245
- pornographic material, blocking, 15–16
- port scans, and IRC networks, 415
- ports
 - listening to with trackers, 311
 - and peer-to-peer networks, 231
- preventing
 - AIM worm attacks, 44–46, 49–50
 - copyright infringement, IRC networks, 408–411
 - information leakage from IRC networks, 407
 - malicious file transfers, 413–414
 - P2P network infections, 282
 - SPIM (instant messaging SPAM), 15–16, 92
 - Yahoo! Messenger worms, 68, 92
- Process Explorer, 145, 146
- processes, viewing, 145

protection bots, 390–391

protocols

See also specific protocol

AIM, 26–31

BitTorrent, 296–308, 314

Diffie-Hellman-based key
exchange, 163

FastTrack network, 335–342, 355

Gnutella, 246–254, 264

ICQ, 134

IRC file transfer, 386–387

Jabber, 168

MSN Messenger's use of, 98–99,
103

peer-to-peer networks, 228–229

Remote Desktop Protocol
(RDP), 110

Yahoo! Messenger, 57–59, 90

YMSG protocol, 53

proxies

AIM-supported, 29

MSN Messenger's settings (fig.),
98

Yahoo! Messenger's use of, 54–55

proxy botnets, 392

proxy servers

BitTorrent and, 317

and FastTrack, 357

and Gnutella clients, 258–259

and MSN Messenger, 105

packet-filtering features of, 193
queried compromised or open,
393

Skype support, configuration,
205–211

psybnc, 379

Q

Qnext IM client, 11

Quakenet network, 371–372

Query descriptor packets, Gnutella,
249–250

QueryHits descriptor packets,
Gnutella, 250–252

R

Recording Industry Association of
America. *See* RIAA

remote assistance, MSN Messenger
function, 110–111

Remote Desktop Protocol (RDP)
and MSN Messenger remote
assistance, 110

remote workstations

AIM user browsing for files on,
34–35

and MSN Messenger Remote
Assistance, 110

removing botnets, 402–404

RIAA (Recording Industry
Association of America)

network monitoring by, 275

and LimeWire, 241

v. FastTrack clients, 323

v. Napster, 234

v. The People, 235–236

rizon network, 372

routers, NAT, 192–197

RSA encryption, 182

S

samhain utility, 415

Sarbanes-Oxley IM regulations, 18

Secunia's security database, 126

SecureIM settings, Trillian (fig.), 163

security

AIM client, 37–46

BitTorrent risks, 308–310

black and white listing, 211

client, 16–17

common instant messaging issues,
11–16, 21–22

eDonkey network, risks, 274–279,
281–283

FastTrack risks, 343–347, 355

Gnutella, risks, 254–257, 264–266

Goggle Talk, 176

ICQ client threats (table), 143

Internet Security System's X-
Force Database of threats, 44

MSN Messenger client, 115–120,
126–127, 130

Skype issues, 183–184, 190, 216

Trillian, 176

Web-based clients, 172–172

WHOIS fields, 411

Yahoo! Messenger client, 67–68,
87–88

seeders and leechers, 294–295

semicentralized networks, 332

SerarchIRC, 378

servants on Gnutella network, 243

servers

IRC. *See* Internet Relay Chat

proxy. *See* proxy servers

for Yahoo! Messenger, 56–57

services

filtering, 419

major instant messaging, 6–7

networks, and clients, 6

Shareaza client, 291

shares, open, and peer-to-peer
networks, 231

sharing

applications. *See* application
sharing

data, and BitTorrent, 310

files. *See* file sharing

Sharman Networks, 322, 324

Short Message Service (SMS)
messages, 136

SILC IRC client, 431, 435

sirc IRC client, 428

Skype

architecture, 180–183

blocking in the enterprise,
211–212

checking connections, 206–211

configuring network devices,
197–200

controlling ports, 201–202

features, security information, 7,
180, 183–189, 213–215

home and business
implementations, 195–197

malicious code vulnerabilities,
189–190

NAT and firewalls, 192–197

ports required for, 200–202

- preventing file sharing, 216
 - running traffic through Windows Firewall, 202–205
 - security issues, 22, 188–190, 216
 - using proxy servers and, 205–211
 - SNAC data and AIM protocol, 31
 - Snak IRC client, 429
 - Snort IDS
 - BitTorrent rules, 312
 - FastTrack rules, 352
 - Gnutella rules, 262
 - social engineering and instant messaging, 12, 18, 45, 136–137, 170
 - SOCKS 4, 5
 - AIM-supported proxy, 29
 - and Skype, 183, 205
 - Sony Corp. v. Universal City Studios, 233–234
 - spam control in ICQ (fig.), 142
 - SPIM (instant messaging SPAM), preventing, 15–16, 92
 - spimming (IM spam) routine, 43
 - spyware
 - bundling, and FastTrack alternative clients, 328–332, 354
 - in P2P software, 232–233
 - StreamCast networks, 326
 - string match module
 - Gnutella, 260
 - iptables, and FastTrack, 349–350
 - super node workstation, 5
 - Super Yahoo Messenger Archive Decoder, 61
 - supernodes
 - and FastTrack, 321, 334–336, 348
 - preventing computer from becoming, 356
 - swarming, 227, 271
 - SYN command (MSN Messenger), 102
 - SysInternals' Process Explorer, 146
- ## T
- Task Manager (Windows), viewing processes with, 145
 - TCP/IP (Transmission Control Protocol/Internet Protocol)
 - and eDonkey, eMule, 273
 - and FLAP, 30
 - and ICQ, 135
 - and MSN Messenger whiteboard, 107
 - and NAT, 193
 - and Skype, 182
 - and Snort blocking, 262
 - and UDP, 198–200
 - text chat described, 8–9, 20
 - text messaging, MSN Messenger's, 107
 - third-party clients described, 21
 - time stamps, in IRC network, 365–368
 - TOC protocol, 27
 - trackers, BitTorrent, 288, 292–293, 299–302, 311, 314
 - traffic analyzer Ethereal, 336–337

transaction ID, Hotmail e-mail service, 103

Transmission Control Protocol/Internet Protocol. *See* TCP/IP

Trillian
 connecting to multiple services, 7
 and encryption, 22
 introduction, features, 8, 160–166, 174–176
 IRC client, 425–426

Trojans
 See also malicious code/software
 AIM-Canbot Trojan, 16
 and Gnutella network, 255–256
 risk with file transfer, 12–13
 Velkbot, 43–44

trusted peers in friend-to-friend (F2F) networks, 229

U

UDP (User Datagram Protocol) and Skype, 182, 198–200

UltraPeers, and Gnutella, 226–227, 245–246

Undernet network, 372, 384–385, 394

Universal City Studios v. Sony Corp, 233–234

UNIX
 and IRC, 362
 IRC clients, 426–428, 433–434

UnrealIRCd IRC daemon, 370–371

URLs (Uniform Resource Locators)
 stopping, 92
 worms and, 13, 37–38, 49, 130

User Datagram Protocol. *See* UDP

uTorrent, 290–291

V

Velkbot Trojan, 43–44

video chat described, 9, 20

video messaging, MSN Messenger's, 107

viruses
 and FastTrack, 343–344
 and Gnutella network, 255–256
 preventing in P2P networks, 282
 risk with file transfer, 12–13
 Trillian's antivirus options, 167–168

Visual IRC (vIRC) client, 425

voice chat
 Goggle Talk, 171–172
 in ICQ, 139–8
 MSN Messenger function, 111–112
 Skype, 185–186
 Yahoo! Messenger, 56–57, 63–64

voice messaging, MSN Messenger's, 107

VoIP (voice over IP) services

- described, 9, 20
- Goggle Talk, 160, 169–170
- Skype, 185–186
- vulnerabilities of peer-to-peer networks, 231–233

W

- W32/Worm.Aimdes.A, 38
- W32.Chod.B@mm worm, 69–81
- W32.Kelvir.R worm, 120–122
- W32.Picrate.C@mm worm, 81–87, 122–126
- Web-based clients benefits, features, 7, 172–175, 177
- Web camera
 - ICQ settings, 141
 - MSN Messenger settings, 114–115
 - Yahoo! Messenger settings, 66–67
- Web services integration, 10, 21
- Web sites
 - AIM security issues, 45
 - Goggle Talk, 169
 - IM client, 19
 - Jabber Foundation, 169
 - McAfee's AVERT group, 42
 - Skype security, 190
- Webcam, Yahoo! Messenger connection, 56
- white listing, 211, 414
- whiteboard features, 10, 18, 107–108
- WHOIS fields, security, 411

- WinBot IRC client, 425
- Windows
 - and ICQ, 134
 - IRC clients, 422–426, 433
 - privilege escalation vulnerability, 88
 - Task Manager, viewing processes with, 145
- Windows Firewall and Skype, 202–205
- Windows Messenger, 96
- Windows XP, Service Pack 2, and Skype, 202
- WinMX Peer Network Protocol (WPNP), 228–230
- wireless networks and Skype, 206
- workstations
 - and message logging, 15
 - MSN Messenger Remote Assistance, 110
 - remote. *See* remote workstations
 - super node, 5
 - and Web-based client vulnerability, 173
- WORM_CHOD.B worm, 147–149
- worms
 - See also specific worm*
 - and FastTrack, 343–344
 - and file transfers, 13–14
 - growth of IM, 17
 - ICQ-infecting, 143–149
 - preventing AIM, 49–50
 - preventing in P2P networks, 282

WORM_VAMPIRE.A worm,
143–146
WORM_WOOTBOT.GX, 37
WPNP (WinMX Peer Network
Protocol), 228–230

X

X-Chat IRC client, 424, 426–427
XDCC protocol, 387
XML documents and MSN
Messenger's message archiving,
106
XMPP (Extensible Messaging and
Presence Protocol), 168–169,
174
XOR encryption, 27

Y

Yahoo! Messenger
blocking on network, 93

features, security information,
59–67, 90–91
introduction, architecture, 52–57,
90
malicious code and client security,
67–91
privilege escalation vulnerability,
88
protocol, 57–59, 90
Web camera settings, 66–67
and Yahoo!, 92
Yahoo! Pager, 52
YMSG protocol, 53, 57–58

Z

Z-Kazaa Tags, 341–343
Zennström, Niklas, 180, 320–321,
326
Zhou, Hongzhen, 126

Syngress: *The Definition of a Serious Security Library*

Syn•gress (sin-gres): *noun, sing.* Freedom from risk or danger; safety. See *security*.



AVAILABLE DEC 2005
order @
www.syngress.com

Skype Me! From Single User to Small Enterprise and Beyond Michael Gough

This first-ever book on Skype takes you from the basics of getting Skype up and running on all platforms, through advanced features included in SkypeIn, SkypeOut, and Skype for Business. *Skype Me!* teaches you everything from installing a headset to configuring a firewall to setting up Skype as telephone Base to developing your own customized applications using the Skype Application Programming Interface.

ISBN: 159749-032-6

Price: \$34.95 US \$48.95 CAN

Google Hacking for Penetration Testers

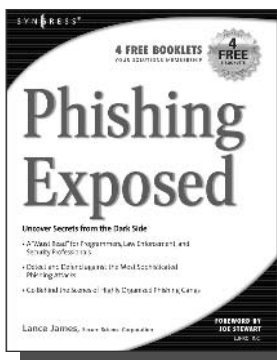
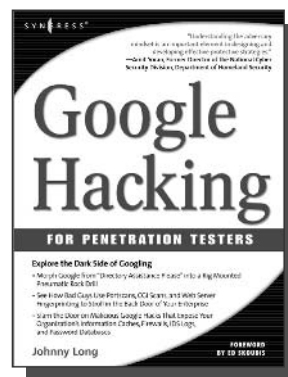
Johnny Long, Foreword by Ed Skoudis

AVAILABLE NOW
order @
www.syngress.com

What many users don't realize is that the deceptively simple components that make Google so easy to use are the same features that generously unlock security flaws for the malicious hacker. Vulnerabilities in website security can be discovered through Google hacking, techniques applied to the search engine by computer criminals, identity thieves, and even terrorists to uncover secure information. This book beats Google hackers to the punch, equipping web administrators with penetration testing applications to ensure their site is invulnerable to a hacker's search.

ISBN: 1-93183-636-1

Price: \$44.95 US \$65.95 CAN



AVAILABLE NOW
order @
www.syngress.com

Phishing Exposed

Lance James, Secure Science Corporation,
Foreword by Joe Stewart

If you have ever received a phish, become a victim of a phish, or manage the security of a major e-commerce or financial site, then you need to read this book. The author of this book delivers the unconcealed techniques of phishers including their evolving patterns, and how to gain the upper hand against the ever-accelerating attacks they deploy. Filled with elaborate and unprecedented forensics, *Phishing Exposed* details techniques that system administrators, law enforcement, and fraud investigators can exercise and learn more about their attacker and their specific attack methods, enabling risk mitigation in many cases before the attack occurs.

ISBN: 1-59749-030-X

Price: \$49.95 U.S. \$69.95 CAN

SYNGRESS®