

 **WILEY**

Security Engineering

Ross Anderson

SECOND EDITION

A Guide to Building Dependable
Distributed Systems

Anderson

Security Engineering

SECOND
EDITION



WILEY

"Security engineering is different from any other kind of programming. . . . if you're even *thinking* of doing any security engineering, you need to read this book."

— Bruce Schneier

"This is the best book on computer security. Buy it, but more importantly, read it and apply it in your work."

— Gary McGraw

This book created the discipline of security engineering

The world has changed radically since the first edition was published in 2001. Spammers, virus writers, phishermen, money launderers, and spies now trade busily with each other in a lively online criminal economy — and as they specialize, they get better. New applications, from search to social networks to electronic voting machines, provide new targets. And terrorism has changed the world. In this indispensable, fully updated guide, Ross Anderson reveals how to build systems that stay dependable whether faced with error or malice.

Ross Anderson is Professor of Security Engineering at Cambridge University and a pioneer of security economics. Widely recognized as one of the world's foremost authorities on security, he has published many studies of how real security systems fail and made trailblazing contributions to numerous technologies from peer-to-peer systems and API analysis through hardware security.

Here's straight talk about

- **Technical engineering basics** — cryptography, protocols, access controls, and distributed systems
- **Types of attack** — phishing, Web exploits, card fraud, hardware hacks, and electronic warfare
- **Specialized protection mechanisms** — what biometrics, seals, smartcards, alarms, and DRM do, and how they fail
- **Security economics** — why companies build insecure systems, why it's tough to manage security projects, and how to cope
- **Security psychology** — the privacy dilemma, what makes security too hard to use, and why deception will keep increasing
- **Policy** — why governments waste money on security, why societies are vulnerable to terrorism, and what to do about it

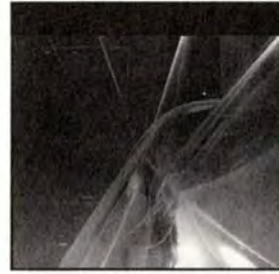
Visit the author's Web site at www.ross-anderson.com

Visit Wiley's Web site at www.wiley.com/compbooks

 **WILEY**
wiley.com

COMPUTERS/Security





Security Engineering

A Guide to Building Dependable Distributed Systems Second Edition

Ross J. Anderson



WILEY

Wiley Publishing, Inc.

**Security Engineering: A Guide to Building Dependable Distributed Systems,
Second Edition**

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256

Copyright © 2008 by Ross J. Anderson. All Rights Reserved.

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-06852-6

Manufactured in the United States of America

10 9 8 7 6 5 4 3

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (800) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Library of Congress Cataloging-in-Publication Data
Anderson, Ross, 1956-

Security engineering : a guide to building dependable distributed systems / Ross J Anderson. — 2nd ed.
p. cm.

Includes bibliographical references and index.

ISBN 978-0-470-06852-6 (cloth)

1. Computer security. 2. Electronic data processing—Distributed processing. I. Title.

QA76.9.A25A54 2008

005.1—dc22

2008006392

Trademarks: Wiley, the Wiley logo, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

To Shireen



Credits

Executive Editor

Carol Long

**Senior Development
Editor**

Tom Dinse

Production Editor

Tim Tate

Editorial Manager

Mary Beth Wakefield

Production Manager

Tim Tate

**Vice President
and Executive Group
Publisher**

Richard Swadley

**Vice President
and Executive
Publisher**

Joseph B. Wikert

**Project Coordinator,
Cover**

Lynsey Stanford

Proofreader

Nancy Bell

Indexer

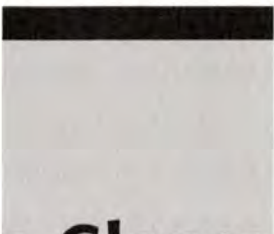
Jack Lewis

Cover Image

© Digital Vision/Getty Images

Cover Design

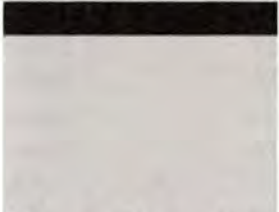
Michael E. Trent



Contents at a Glance

Preface to the Second Edition	xxv
Foreword by Bruce Schneier	xxvii
Preface	xxix
Acknowledgments	xxxv
Part I	
Chapter 1 What Is Security Engineering?	3
Chapter 2 Usability and Psychology	17
Chapter 3 Protocols	63
Chapter 4 Access Control	93
Chapter 5 Cryptography	129
Chapter 6 Distributed Systems	185
Chapter 7 Economics	215
Part II	
Chapter 8 Multilevel Security	239
Chapter 9 Multilateral Security	275
Chapter 10 Banking and Bookkeeping	313
Chapter 11 Physical Protection	365
Chapter 12 Monitoring and Metering	389
Chapter 13 Nuclear Command and Control	415

Chapter 14	Security Printing and Seals	433
Chapter 15	Biometrics	457
Chapter 16	Physical Tamper Resistance	483
Chapter 17	Emission Security	523
Chapter 18	API Attacks	547
Chapter 19	Electronic and Information Warfare	559
Chapter 20	Telecom System Security	595
Chapter 21	Network Attack and Defense	633
Chapter 22	Copyright and DRM	679
Chapter 23	The Bleeding Edge	727
Part III		
Chapter 24	Terror, Justice and Freedom	769
Chapter 25	Managing the Development of Secure Systems	815
Chapter 26	System Evaluation and Assurance	857
Chapter 27	Conclusions	889
Bibliography		893
Index		997



Contents

Preface to the Second Edition	xxv
Foreword by Bruce Schneier	xxvii
Preface	xxix
Acknowledgments	xxxv
Part I	
Chapter 1 What Is Security Engineering?	3
Introduction	3
A Framework	4
Example 1—A Bank	6
Example 2—A Military Base	7
Example 3—A Hospital	9
Example 4—The Home	10
Definitions	11
Summary	15
Chapter 2 Usability and Psychology	17
Introduction	17
Attacks Based on Psychology	18
Pretexting	19
Phishing	21
Insights from Psychology Research	22
What the Brain Does Worse Than the Computer	23
Perceptual Bias and Behavioural Economics	24
Different Aspects of Mental Processing	26
Differences Between People	27
Social Psychology	28
What the Brain Does Better Than Computer	30

Passwords	31
Difficulties with Reliable Password Entry	32
Difficulties with Remembering the Password	33
Naive Password Choice	34
User Abilities and Training	35
Design Errors	37
Operational Issues	39
Social-Engineering Attacks	40
Trusted Path	42
Phishing Countermeasures	43
Password Manglers	43
Client Certs or Specialist Apps	44
Using the Browser's Password Database	44
Soft Keyboards	45
Customer Education	45
Microsoft Passport	46
Phishing Alert Toolbars	47
Two-Factor Authentication	47
Trusted Computing	48
Fortified Password Protocols	49
Two-Channel Authentication	49
The Future of Phishing	50
System Issues	52
Can You Deny Service?	53
Protecting Oneself or Others?	53
Attacks on Password Entry	54
Interface Design	54
Eavesdropping	55
Technical Defeats of Password Retry Counters	55
Attacks on Password Storage	56
One-Way Encryption	56
Password Cracking	57
Absolute Limits	57
CAPTCHAs	59
Summary	60
Research Problems	61
Further Reading	61
Chapter 3 Protocols	63
Introduction	63
Password Eavesdropping Risks	65
Who Goes There? — Simple Authentication	66
Challenge and Response	70
The MIG-in-the-Middle Attack	73
Reflection Attacks	76
Manipulating the Message	78
Changing the Environment	79

Chosen Protocol Attacks	80
Managing Encryption Keys	82
Basic Key Management	83
The Needham-Schroeder Protocol	84
Kerberos	85
Practical Key Management	86
Getting Formal	87
A Typical Smartcard Banking Protocol	87
The BAN Logic	88
Verifying the Payment Protocol	89
Limitations of Formal Verification	90
Summary	91
Research Problems	92
Further Reading	92
Chapter 4 Access Control	93
Introduction	93
Operating System Access Controls	96
Groups and Roles	98
Access Control Lists	99
Unix Operating System Security	100
Apple's OS/X	101
Windows — Basic Architecture	102
Capabilities	103
Windows — Added Features	104
Middleware	107
Database Access Controls	107
General Middleware Issues	108
ORBs and Policy Languages	109
Sandboxing and Proof-Carrying Code	110
Virtualization	111
Trusted Computing	111
Hardware Protection	113
Intel Processors, and 'Trusted Computing'	114
ARM Processors	116
Security Processors	116
What Goes Wrong	117
Smashing the Stack	118
Other Technical Attacks	119
User Interface Failures	121
Why So Many Things Go Wrong	122
Remedies	124
Environmental Creep	125
Summary	126
Research Problems	127
Further Reading	127

Chapter 5	Cryptography	129
	Introduction	129
	Historical Background	130
	An Early Stream Cipher — The Vigenère	131
	The One-Time Pad	132
	An Early Block Cipher — Playfair	134
	One-Way Functions	136
	Asymmetric Primitives	138
	The Random Oracle Model	138
	Random Functions — Hash Functions	140
	Properties	141
	The Birthday Theorem	142
	Random Generators — Stream Ciphers	143
	Random Permutations — Block Ciphers	144
	Public Key Encryption and Trapdoor One-Way Permutations	146
	Digital Signatures	147
	Symmetric Crypto Primitives	149
	SP-Networks	149
	Block Size	150
	Number of Rounds	150
	Choice of S-Boxes	151
	Linear Cryptanalysis	151
	Differential Cryptanalysis	152
	Serpent	153
	The Advanced Encryption Standard (AES)	153
	Feistel Ciphers	155
	The Luby-Rackoff Result	157
	DES	157
	Modes of Operation	160
	Electronic Code Book	160
	Cipher Block Chaining	161
	Output Feedback	161
	Counter Encryption	162
	Cipher Feedback	163
	Message Authentication Code	163
	Composite Modes of Operation	164
	Hash Functions	165
	Extra Requirements on the Underlying Cipher	166
	Common Hash Functions and Applications	167
	Asymmetric Crypto Primitives	170
	Cryptography Based on Factoring	170
	Cryptography Based on Discrete Logarithms	173
	Public Key Encryption — Diffie Hellman and ElGamal	174
	Key Establishment	175
	Digital Signature	176
	Special Purpose Primitives	178

	Elliptic Curve Cryptography	179
	Certification	179
	The Strength of Asymmetric Cryptographic Primitives	181
	Summary	182
	Research Problems	183
	Further Reading	183
Chapter 6	Distributed Systems	185
	Introduction	185
	Concurrency	186
	Using Old Data Versus Paying to Propagate State	186
	Locking to Prevent Inconsistent Updates	188
	The Order of Updates	188
	Deadlock	189
	Non-Convergent State	190
	Secure Time	191
	Fault Tolerance and Failure Recovery	192
	Failure Models	193
	Byzantine Failure	193
	Interaction with Fault Tolerance	194
	What Is Resilience For?	195
	At What Level Is the Redundancy?	197
	Service-Denial Attacks	198
	Naming	200
	The Distributed Systems View of Naming	200
	What Else Goes Wrong	204
	Naming and Identity	204
	Cultural Assumptions	206
	Semantic Content of Names	207
	Uniqueness of Names	207
	Stability of Names and Addresses	208
	Adding Social Context to Naming	209
	Restrictions on the Use of Names	210
	Types of Name	211
	Summary	211
	Research Problems	212
	Further Reading	213
Chapter 7	Economics	215
	Introduction	215
	Classical Economics	216
	Monopoly	217
	Public Goods	219
	Information Economics	220
	The Price of Information	220
	The Value of Lock-In	221
	Asymmetric Information	223

Game Theory	223
The Prisoners' Dilemma	225
Evolutionary Games	226
The Economics of Security and Dependability	228
Weakest Link, or Sum of Efforts?	229
Managing the Patching Cycle	229
Why Is Windows So Insecure?	230
Economics of Privacy	232
Economics of DRM	233
Summary	234
Research Problems	235
Further Reading	235

Part II

Chapter 8	Multilevel Security	239
	Introduction	239
	What Is a Security Policy Model?	240
	The Bell-LaPadula Security Policy Model	242
	Classifications and Clearances	243
	Information Flow Control	245
	The Standard Criticisms of Bell-LaPadula	246
	Alternative Formulations	248
	The Biba Model and Vista	250
	Historical Examples of MLS Systems	252
	SCOMP	252
	Blacker	253
	MLS Unix and Compartmented Mode Workstations	253
	The NRL Pump	254
	Logistics Systems	255
	Sybard Suite	256
	Wiretap Systems	256
	Future MLS Systems	257
	Vista	257
	Linux	258
	Virtualization	260
	Embedded Systems	261
	What Goes Wrong	261
	Composability	261
	The Cascade Problem	262
	Covert Channels	263
	The Threat from Viruses	265
	Polyinstantiation	266
	Other Practical Problems	267
	Broader Implications of MLS	269

Summary	272
Research Problems	272
Further Reading	272
Chapter 9 Multilateral Security	275
Introduction	275
Compartmentation, the Chinese Wall and the BMA Model	277
Compartmentation and the Lattice Model	277
The Chinese Wall	281
The BMA Model	282
The Threat Model	284
The Security Policy	287
Pilot Implementations	289
Current Privacy Issues	290
Inference Control	293
Basic Problems of Inference Control in Medicine	293
Other Applications of Inference Control	296
The Theory of Inference Control	297
Query Set Size Control	298
Trackers	298
More Sophisticated Query Controls	298
Cell Suppression	299
Maximum Order Control and the Lattice Model	300
Audit Based Control	300
Randomization	301
Limitations of Generic Approaches	302
Active Attacks	304
The Value of Imperfect Protection	305
The Residual Problem	306
Summary	309
Research Problems	310
Further Reading	310
Chapter 10 Banking and Bookkeeping	313
Introduction	313
The Origins of Bookkeeping	315
Double-Entry Bookkeeping	316
A Telegraphic History of E-commerce	316
How Bank Computer Systems Work	317
The Clark-Wilson Security Policy Model	319
Designing Internal Controls	320
What Goes Wrong	324
Wholesale Payment Systems	328
SWIFT	329
What Goes Wrong	331
Automatic Teller Machines	333
ATM Basics	334

What Goes Wrong	337
Incentives and Injustices	341
Credit Cards	343
Fraud	344
Forgery	345
Automatic Fraud Detection	346
The Economics of Fraud	347
Online Credit Card Fraud — the Hype and the Reality	348
Smartcard-Based Banking	350
EMV	351
Static Data Authentication	352
Dynamic Data Authentication	356
Combined Data Authentication	356
RFID	357
Home Banking and Money Laundering	358
Summary	361
Research Problems	362
Further Reading	363
Chapter 11 Physical Protection	365
Introduction	365
Threats and Barriers	366
Threat Model	367
Deterrence	368
Walls and Barriers	370
Mechanical Locks	372
Electronic Locks	376
Alarms	378
How not to Protect a Painting	379
Sensor Defeats	380
Feature Interactions	382
Attacks on Communications	383
Lessons Learned	386
Summary	387
Research Problems	388
Further Reading	388
Chapter 12 Monitoring and Metering	389
Introduction	389
Prepayment Meters	390
Utility Metering	392
How the System Works	393
What Goes Wrong	395
Taxi Meters, Tachographs and Truck Speed Limiters	397
The Tachograph	398
What Goes Wrong	399
How Most Tachograph Manipulation Is Done	400

Tampering with the Supply	401
Tampering with the Instrument	401
High-Tech Attacks	402
The Digital Tachograph Project	403
System Level Problems	404
Other Problems	405
The Resurrecting Duckling	407
Postage Meters	408
Summary	412
Research Problems	413
Further Reading	414
Chapter 13 Nuclear Command and Control	415
Introduction	415
The Evolution of Command and Control	417
The Kennedy Memorandum	418
Authorization, Environment, Intent	419
Unconditionally Secure Authentication	420
Shared Control Schemes	422
Tamper Resistance and PALs	424
Treaty Verification	426
What Goes Wrong	427
Secrecy or Openness?	429
Summary	430
Research Problems	430
Further Reading	430
Chapter 14 Security Printing and Seals	433
Introduction	433
History	434
Security Printing	435
Threat Model	436
Security Printing Techniques	437
Packaging and Seals	443
Substrate Properties	443
The Problems of Glue	444
PIN Mailers	445
Systemic Vulnerabilities	446
Peculiarities of the Threat Model	447
Anti-Gundecking Measures	448
The Effect of Random Failure	449
Materials Control	450
Not Protecting the Right Things	451
The Cost and Nature of Inspection	451
Evaluation Methodology	453
Summary	454
Research Problems	454
Further Reading	455

Chapter 15 Biometrics	457
Introduction	457
Handwritten Signatures	458
Face Recognition	461
Bertillonage	464
Fingerprints	464
Verifying Positive or Negative Identity Claims	466
Crime Scene Forensics	469
Iris Codes	472
Voice Recognition	475
Other Systems	476
What Goes Wrong	477
Summary	481
Research Problems	482
Further Reading	482
 Chapter 16 Physical Tamper Resistance	 483
Introduction	483
History	485
High-End Physically Secure Processors	486
Evaluation	492
Medium Security Processors	494
The iButton	494
The Dallas 5000 Series	495
FPGA Security, and the Clipper Chip	496
Smartcards and Microcontrollers	499
History	500
Architecture	501
Security Evolution	501
The State of the Art	512
Defense in Depth	513
Stop Loss	513
What Goes Wrong	514
The Trusted Interface Problem	514
Conflicts	515
The Lemons Market, Risk Dumping and Evaluation	516
Security-By-Obscurity	517
Interaction with Policy	517
Function Creep	518
So What Should One Protect?	518
Summary	520
Research Problems	520
Further Reading	520
 Chapter 17 Emission Security	 523
Introduction	523
History	524

Technical Surveillance and Countermeasures	526
Passive Attacks	530
Leakage Through Power and Signal Cables	530
Red/Black Separation	530
Timing Analysis	531
Power Analysis	531
Leakage Through RF Signals	534
Active Attacks	538
Tempest Viruses	538
Nonstop	539
Glitching	540
Differential Fault Analysis	540
Combination Attacks	540
Commercial Exploitation	541
Defenses	541
Optical, Acoustic and Thermal Side Channels	542
How Serious are Emsec Attacks?	544
Governments	544
Businesses	545
Summary	546
Research Problems	546
Further Reading	546
Chapter 18 API Attacks	547
Introduction	547
API Attacks on Security Modules	548
The XOR-To-Null-Key Attack	549
The Attack on the 4758	551
Multiparty Computation, and Differential Protocol Attacks	552
The EMV Attack	553
API Attacks on Operating Systems	554
Summary	555
Research Problems	557
Further Reading	557
Chapter 19 Electronic and Information Warfare	559
Introduction	559
Basics	560
Communications Systems	561
Signals Intelligence Techniques	563
Attacks on Communications	565
Protection Techniques	567
Frequency Hopping	568
DSSS	569
Burst Communications	570
Combining Coverttness and Jam Resistance	571
Interaction Between Civil and Military Uses	572

Surveillance and Target Acquisition	574
Types of Radar	574
Jamming Techniques	575
Advanced Radars and Countermeasures	577
Other Sensors and Multisensor Issues	578
IFF Systems	579
Improvised Explosive Devices	582
Directed Energy Weapons	584
Information Warfare	586
Definitions	587
Doctrine	588
Potentially Useful Lessons from Electronic Warfare	589
Differences Between E-war and I-war	591
Summary	592
Research Problems	592
Further Reading	593
Chapter 20 Telecom System Security	595
Introduction	595
Phone Phreaking	596
Attacks on Metering	596
Attacks on Signaling	599
Attacks on Switching and Configuration	601
Insecure End Systems	603
Feature Interaction	605
Mobile Phones	606
Mobile Phone Cloning	607
GSM Security Mechanisms	608
Third Generation Mobiles — 3gpp	617
Platform Security	619
So Was Mobile Security a Success or a Failure?	621
VOIP	623
Security Economics of Telecomms	624
Frauds by Phone Companies	625
Billing Mechanisms	627
Summary	630
Research Problems	631
Further Reading	632
Chapter 21 Network Attack and Defense	633
Introduction	633
Vulnerabilities in Network Protocols	635
Attacks on Local Networks	636
Attacks Using Internet Protocols and Mechanisms	638
SYN Flooding	638
Smurfing	639
Distributed Denial of Service Attacks	640

Spam	642
DNS Security and Pharming	643
Trojans, Viruses, Worms and Rootkits	644
Early History of Malicious Code	644
The Internet Worm	645
How Viruses and Worms Work	646
The History of Malware	647
Countermeasures	650
Defense Against Network Attack	652
Configuration Management and Operational Security	652
Filtering: Firewalls, Spam Filters, Censorware and Wiretaps	654
Packet Filtering	654
Circuit Gateways	655
Application Relays	655
Ingress Versus Egress Filtering	657
Architecture	657
Intrusion Detection	660
Types of Intrusion Detection	661
General Limitations of Intrusion Detection	662
Specific Problems Detecting Network Attacks	664
Encryption	665
SSH	665
WiFi	666
Bluetooth	668
HomePlug	668
IPsec	669
TLS	670
PKI	672
Topology	675
Summary	676
Research Problems	677
Further Reading	678
Chapter 22 Copyright and DRM	679
Introduction	679
Copyright	680
Software	681
Books	688
Audio	689
Video and Pay-TV	690
Typical System Architecture	690
Video Scrambling Techniques	691
Attacks on Hybrid Scrambling Systems	693
DVB	697
DVD	698
HD-DVD and Blu-ray	701
AACs — Broadcast Encryption and Traitor Tracing	701

Blu-ray and SPDC	703
General Platforms	704
Windows Media Rights Management	705
Other Online Rights-Management Systems	706
Peer-to-Peer Systems	707
Rights Management of Semiconductor IP	709
Information Hiding	710
Watermarks and Copy Generation Management	711
General Information Hiding Techniques	712
Attacks on Copyright Marking Schemes	714
Applications of Copyright Marking Schemes	718
Policy	718
The IP Lobby	720
Who Benefits?	722
Accessory Control	723
Summary	725
Research Problems	725
Further Reading	726
Chapter 23 The Bleeding Edge	727
Introduction	727
Computer Games	728
Types of Cheating	730
Aimbots and Other Unauthorized Software	732
Virtual Worlds, Virtual Economies	733
Web Applications	734
eBay	735
Google	736
Social Networking Sites	739
Privacy Technology	745
Anonymous Email — The Dining Cryptographers and Mixes	747
Anonymous Web Browsing — Tor	749
Confidential and Anonymous Phone Calls	751
Email Encryption	753
Steganography and Forensics Countermeasures	755
Putting It All Together	757
Elections	759
Summary	764
Research Problems	764
Further Reading	765
Part III	
Chapter 24 Terror, Justice and Freedom	769
Introduction	769
Terrorism	771
Causes of Political Violence	772

The Psychology of Political Violence	772
The Role of Political Institutions	774
The Role of the Press	775
The Democratic Response	775
Surveillance	776
The History of Government Wiretapping	776
The Growing Controversy about Traffic Analysis	779
Unlawful Surveillance	781
Access to Search Terms and Location Data	782
Data Mining	783
Surveillance via ISPs — Carnivore and its Offspring	784
Communications Intelligence on Foreign Targets	785
Intelligence Strengths and Weaknesses	787
The Crypto Wars	789
The Back Story to Crypto Policy	790
DES and Crypto Research	792
The Clipper Chip	793
Did the Crypto Wars Matter?	794
Export Control	796
Censorship	797
Censorship by Authoritarian Regimes	798
Network Neutrality	800
Peer-to-Peer, Hate Speech and Child Porn	801
Forensics and Rules of Evidence	803
Forensics	803
Admissibility of Evidence	806
Privacy and Data Protection	808
European Data Protection	809
Differences between Europe and the USA	810
Summary	812
Research Problems	813
Further Reading	813
Chapter 25 Managing the Development of Secure Systems	815
Introduction	815
Managing a Security Project	816
A Tale of Three Supermarkets	816
Risk Management	818
Organizational Issues	819
The Complacency Cycle and the Risk Thermostat	820
Interaction with Reliability	821
Solving the Wrong Problem	822
Incompetent and Inexperienced Security Managers	823
Moral Hazard	823
Methodology	824
Top-Down Design	826
Iterative Design	827

Lessons from Safety-Critical Systems	829
Security Requirements Engineering	834
Managing Requirements Evolution	835
Bug Fixing	836
Control Tuning and Corporate Governance	838
Evolving Environments and the Tragedy of the Commons	839
Organizational Change	841
Managing Project Requirements	842
Parallelizing the Process	844
Risk Management	846
Managing the Team	848
Summary	852
Research Problems	853
Further Reading	854
Chapter 26 System Evaluation and Assurance	857
Introduction	857
Assurance	858
Perverse Economic Incentives	858
Project Assurance	860
Security Testing	861
Formal Methods	862
Quis Custodiet?	862
Process Assurance	863
Assurance Growth	866
Evolution and Security Assurance	868
Evaluation	869
Evaluations by the Relying Party	870
The Common Criteria	873
What the Common Criteria Don't Do	876
Corruption, Manipulation and Inertia	878
Ways Forward	881
Hostile Review	882
Free and Open-Source Software	882
Semi-Open Design	884
Penetrate-and-Patch, CERTs, and Bugtraq	885
Education	886
Summary	887
Research Problems	887
Further Reading	887
Chapter 27 Conclusions	889
Bibliography	893
Index	997

Telecom System Security

I rarely had to resort to a technical attack. Companies can spend millions of dollars toward technological protections and that's wasted if somebody can basically call someone on the telephone and either convince them to do something on the computer that lowers the computer's defenses or reveals the information they were seeking.

— Kevin Mitnick

There are two kinds of fools. One says, "This is old, therefore it is good." The other one says, "This is new, therefore it is better".

— Dean William Inge

20.1 Introduction

The protection of telecommunications systems is an important case study for a number of reasons. First, many distributed systems rely on the fixed or mobile phone network in ways that are often not obvious, and the dependability of these networks is declining. For example, POTS — the 'plain old telephone system' — typically required exchanges to have backup generators with enough diesel to survive a six-week outage in the electricity supply, while cellular systems typically use batteries that will last at most 48 hours. What's worse, the electricity companies rely on mobile phones to direct their engineers when repairing faults. When people realised that this could cause serious problems where outages lasted more than two days, the electricity companies started buying satellite phones as a backup.

Second, the history of telecomms security failures is very instructive. Early attacks were carried out on phone companies by enthusiasts ('phone phreaks') to get free calls; then the phone system's vulnerabilities started to be

exploited by crooks to evade police wiretapping; then premium rate calls were introduced, which created the motive for large-scale fraud; then when telecomms markets were liberalized, some phone companies started conducting attacks on each other's customers; and some phone companies have even attacked each other. At each stage the defensive measures undertaken were not only very expensive but also tended to be inadequate for various reasons. The same pattern is repeating with the Internet — only with history much speeded up. A number of the policy issues that arose with wireline phones, such as wiretapping, have played themselves out again on the Internet.

Finally, the latest developments in telecomms, from VOIP at the consumer level to the adoption of IP networks as the underlying technology by telecomms providers, create further interactions. Skype's two-day failure in August 2007 following Microsoft's Patch Tuesday is just one case in point. Systems are becoming much more complex and interdependent, and are generally not being engineered to the old standards.

20.2 Phone Phreaking

The abuse of communication services goes back centuries. Before Sir Rowland Hill invented the postage stamp, postage was paid by the recipient. Unsolicited mail became a huge problem — especially for famous people — so recipients were allowed to inspect a letter and reject it rather than paying for it. People soon worked out schemes to send short messages on the covers of letters which their correspondents rejected. Regulations were brought in to stop this, but were never really effective [979].

A second set of abuses developed with the telegraph. The early optical telegraphs, which worked using semaphores or heliographs, were abused by people to place foreknowledge bets on races; if you could learn which horse had won before the bookmaker did, you were well away. People would bribe operators, or 'hack the local loop' by observing the last heliograph station through a telescope. Here too, attempts to legislate the problem away were a failure [1215]. The problems got even more widespread when the electric telegraph brought costs down; the greater volumes of communication, and the greater flexibility that got built into and on top of the service, led to greater complexity and volume of abuse.

The telephone was to be no different.

20.2.1 Attacks on Metering

Early metering systems were wide open to abuse.

- In the 1950's, the operator in some systems had to listen for the sound of coins dropping on a metal plate to tell that a callbox customer had paid,

so some people acquired the knack of hitting the coinbox with a piece of metal that struck the right note.

- Initially, the operator had no way of knowing which phone a call had come from, so she had to ask the caller his number. He could give the number of someone else — who would be charged. This was risky to do from your own phone, so people did it from call boxes. Operators started calling back to verify the number for international calls, so people worked out social engineering attacks ('This is IBM here, we'd like to book a call to San Francisco and because of the time difference can our Managing Director take it at home tonight? His number's xxx-yyyy'). So call box lines had a feature added to alert the operator. But in the UK implementation, there was a bug: a customer who had called the operator from a callbox could depress the rest for a quarter second or so, whereupon he'd be disconnected and reconnected (often to a different operator), with no signal this time that the call was from a callbox. He could then place a call to anywhere and bill it to any local number.
- Early systems also signalled the entry of a coin by one or more pulses, each of which consisted of the insertion of a resistance in the line followed by a brief open circuit. At a number of colleges, enterprising students installed 'magic buttons' which could simulate this in a callbox in the student union so people could phone for free. (The bill in this case went to the student union, for which the magic button was not quite so amusing.)

Attacks on metering mechanisms continue. Many countries have moved their payphones to chip cards in order to cut the costs of coin collection and vandalism. Some of the implementations have been poor (as I remarked in the chapter on tamper resistance) and villains have manufactured large quantities of bogus phone cards. Other attacks involve what's called *clip-on*: physically attaching a phone to someone else's line to steal their service.

In the 1970's, when international phone calls were very expensive, foreign students would clip their own phone on to a residential line in order to call home; an unsuspecting home owner could get a huge bill. Despite the fact that in most countries the cable was the phone company's legal responsibility up to the service socket in the house, phone companies were mostly adamant that householders should pay and could threaten to blacklist them if they didn't. Now that long distance calls are cheap, the financial incentive for clip-on fraud has largely disappeared. But it's still enough of a problem that the Norwegian phone company designed a system whereby a challenge and response are exchanged between a wall-socket mounted authentication device and the exchange software before a dial tone is given [673].

Clip-on fraud had a catastrophic effect on a family in Cramlington, a town in the North East of England. The first sign they had of trouble was hearing

a conversation on their line. The next was a visit from the police who said there'd been complaints of nuisance phone calls. The complainants were three ladies, all of whom had a number one digit different from a number to which this family had supposedly made a huge number of calls. When the family's bill was examined, there were also calls to clusters of numbers that turned out to be payphones; these had started quite suddenly at the same time as the nuisance calls. When the family had complained later to the phone company about a fault, their connection was rerouted and this had solved the problem.

But the phone company denied the possibility of a tap, despite the report from their maintenance person which noted that the family's line had been tampered with at the distribution cabinet. (The phone company later claimed this report was in error.) It turned out that a drug dealer had lived close by, and it seemed a reasonable inference that he'd tapped their line in order to call his couriers at the payphones. By using an innocent family's phone line instead of his own, he not only saved on the phone bill, but also had a better chance of evading police surveillance. But both the police and the local phone company refused to go into the house where the dealer had lived, claiming it was too dangerous — even though the dealer had by now got six years in jail. The Norwegian phone company declined an invitation to testify about clip-on for the defence. The upshot was that the subscriber was convicted of making harassing phone calls, in a case widely believed to have been a miscarriage of justice. There was discussion at the time about whether the closing of ranks between the phone company and the police was a bureaucratic reflex — or something more sinister. Since 9/11, it's emerged that many phone companies have been giving the police easy access to systems for years, often without warrants, in return for favours. The logical consequence was a policy of covering up anything that could stray into this territory — even if the coverup caused collateral damage. I'll discuss all this later in the third part of this book.

Stealing dial tone from cordless phones is another variant on the theme. In the 1990s, this became so widespread in Paris that France Telecom broke with phone company tradition and announced that it was happening, claiming that the victims were using illegally imported cordless phones which were easy to spoof [745]. Yet to this day I am unaware of any cordless phones — authorised or not — with decent air link authentication. The new digital cordless phones use the DECT standard which allows for challenge-response mechanisms [1283] but the equipment sold so far seems to simply send a handset serial number to the base station.

Social engineering is also widespread. A crook calls you pretending to be from AT&T security and asks whether you made a large number of calls to Peru on your calling card. When you deny this, he says that they were obviously fake and, in order to reverse out the charges, can he confirm that your card number is 123-456-7890-6543? No, you say (if you're not really alert),

it's 123-456-7890-5678. Now 123-456-7890 is your phone number and 5678 your password, so you've just given that caller the ability to bill calls to you.

The growth of premium rate phone services during the 1990s also led to scamsters developing all sorts of tricks to get people to call them: pager messages, job ads, fake emergency messages about relatives, 'low cost' calling cards with 0900 access numbers, you name it. (In fact, the whole business of tricking people into calling expensive premium numbers enabled crooks to develop a lot of the techniques we now see used in email as part of phishing attacks.) The 809 area code for the Caribbean used to be a favourite cover for crooks targeting U.S. subscribers; many people weren't aware that 'domestic' numbers (numbers within the USA's +1 international direct dialling code) extend outside the relatively cheap USA (and Canada). Even though many people have now learned that +1 809 is 'foreign' and more expensive, the introduction of still more Caribbean area codes, such as +1 345 for the Cayman Islands, has made it even harder to spot premium rate numbers.

Phone companies advised their customers 'Do not return calls to unfamiliar telephone numbers' and 'Beware of faxes, e-mail, voice mail and pages requesting a return call to an unfamiliar number' [22] — but how practical is that? Just as banks now train their customers to click on links in marketing emails and thus make them vulnerable to phishing attacks, so I've had junk marketing calls from my phone company — even though I'm on the do-not-call list. And as for governments, they have tended to set up weak regulators to oversee phone system abuses at home, and avoid anything that might get them involved in trying to regulate premium rate scams overseas. For example, they let phone companies harass their customers into paying bills for overseas services even when they knew that the overseas traffic was fraudulent.

Indeed, by no means all premium-rate scams involved obviously dodgy companies running sex lines; as I write in 2007, the British press are full of stores about how TV companies rip off their customers by getting them to call premium lines in order to compete, and vote, in all sorts of shows. It's turned out that many of these are recorded, so the calls are totally futile; and even the live ones are managed so that people who live in the wrong part of the country or speak with the wrong accent have no chance. The authorities tried to leave this to 'self-regulation' and on-air apologies from TV bosses, until a public outcry (whipped up by their competitors) led to the file being sent to the police in October 2007. It's a recurring pattern that the biggest scams are often run by 'respectable' companies rather than by Russian gangsters.

20.2.2 Attacks on Signaling

The term 'phone phreaking' refers to attacks on signaling as well as pure toll fraud. Until the 1980s, phone companies used signalling systems that worked *in-band* by sending tone pulses in the same circuit that carried the speech. The

first attack I've heard of dates back to 1952, and by the mid-to-late 1960s many enthusiasts in both America and Britain had worked out ways of rerouting calls. One of the pioneers, Joe Engresia, had perfect pitch and discovered as a child that he could make free phone calls by whistling a tone he'd heard in the background of a long-distance call. His less gifted colleagues typically used home-made tone generators, of which the most common were called *blue boxes*. The trick was to call an 0800 number and then send a 2600Hz tone that would *clear down* the line at the far end — that is, disconnect the called party while leaving the caller with a trunk line connected to the exchange. The caller could now enter the number he really wanted and be connected without paying. Phone phreaking was one of the roots of the computer hacker culture that took root in the Bay Area and was formative in the development and evolution of personal computers [835]. For example, Steve Jobs and Steve Wozniak first built blue boxes before they diversified into computers [502].

Phone phreaking started out with a strong ideological element. In those days most phone companies had monopolies. They were large, faceless and unresponsive. In America, AT&T was such an abusive monopoly that the courts eventually broke it up; most phone companies in Europe were government departments. People whose domestic phone lines had been involved in a service theft found they were stuck with the charges. If the young man who had courted your daughter was (unknown to you) a phone phreak who hadn't paid for the calls he made to her, you would suddenly find the company trying to extort either the young man's name or a payment. Phone companies were also aligned with state security. Phone phreaks in many countries discovered signalling codes or switch features that would enable the police or the spooks to tap your phone from the comfort of their desks, without having to send out a lineman to install a wiretap. Back in the days of Vietnam and student protests, this was inflammatory stuff. Phone phreaks were counterculture heroes, while phone companies were hand-in-hand with the forces of darkness.

As there was no way to stop blue-box type attacks so long as telephone signalling was carried in-band, the phone companies spent years and many billions of dollars upgrading exchanges so that the signaling was moved out-of-band, in separate channels to which the subscribers had no easy access. Gradually, region by region, the world was closed off to blue box attacks. There are still a few places left. For example, the first time that USAF operations were disrupted by an 'information warfare' attack by noncombatants was in 1994 when two British hackers broke into the Rome Air Force Base via an analog link through an ancient phone system in Argentina which they used to hold up investigators [1202]. There's also an interesting legacy vulnerability in wiretapping systems: common phone-tapping equipment was designed to be backwards compatible with in-band signalling, with the result that you can evade surveillance by using a blue box to convince the police equipment that

you've hung up. The telephone exchange ignores this signal, so you remain on the phone but with the police recording stopped [1151].

But to defeat a modern telephone network — as opposed to its law-enforcement add-ons — different techniques are needed.

20.2.3 Attacks on Switching and Configuration

The second wave of attacks targeted the computers that did the switching. Typically these were Unix machines on a LAN in the exchange, which also had machines with administrative functions such as scheduling maintenance. By hacking one of these less well guarded machines, a phreak could go across the LAN and break into the switching equipment — or in to other secondary systems such as subscriber databases. For a survey of PacBell's experience of this, see [271]; for Bellcore's, see [722].

Using these techniques, unlisted phone numbers could be found, calls could be forwarded without a subscriber's knowledge, and all sorts of mischief became possible. A Californian phone phreak called Kevin Poulsen got root access to many of PacBel's switches and other systems in 1985–88: this apparently involved burglary as much as hacking (he was eventually convicted of conspiring to possess fifteen or more counterfeit, unauthorized and stolen access devices.) He did petty things like obtaining unlisted phone numbers for celebrities and winning a Porsche from Los Angeles radio station KIIS-FM. Each week KIIS would give a Porsche to the 102nd caller, so Poulsen and his accomplices blocked out all calls to the radio station's 25 phone lines save their own, made the 102nd call and collected the Porsche. He was also accused of unlawful wiretapping and espionage; these charges were dismissed. In fact, the FBI came down on him so heavily that there were allegations of an improper relationship between the agency and the phone companies, along the lines of 'you scratch our backs with wiretaps when needed, and we'll investigate your hacker problems' [472].

Although the unauthorized wiretapping charges against Poulsen were dismissed, the FBI's sensitivity does highlight the possibility that attacks on phone company computers can be used by foreign intelligence agencies to conduct remote wiretaps. Some of the attacks mentioned in [271] were from overseas, and the possibility that such tricks might be used to crash the whole phone system in the context of an information warfare attack has for some years worried the NSA [495, 754]. Countries that import their telephone exchanges rather than building their own are in an even worse position; a prudent nations will assume that its telephone switchgear has vulnerabilities known to the government of the country from which they bought it. (It was notable that during the invasion of Afghanistan in 2001, Kabul had two exchanges: an old electromechanical one and a new electronic one. The USAF bombed only the first of these.)

But although high-tech attacks do happen, and newspaper articles on phone phreaking tend to play up the 'evil hacker' aspects, most real attacks are much simpler. Many involve insiders, who deliberately misconfigure systems to provide free calls from (or through) favored numbers. This didn't matter all that much when the phone company's marginal cost of servicing an extra phone call was near zero, but with the modern proliferation of value-added services, people with access to the systems can be tempted to place (or forge) large numbers of calls to accomplices' sex lines. Deregulation, and the advent of mobile phones, have also made fraud serious as they give rise to cash payments between phone companies [317]. Insiders also get up to mischief with services that depend on the security of the phone network. In a hack reminiscent of Poulsen, two staff at British Telecom were dismissed after they each won ten tickets for Concorde from a phone-in offer at which only one randomly selected call in a thousand was supposed to get through [1266].

As for outsiders, the other 'arch-hacker' apart from Poulsen was Kevin Mitnick, who got arrested and convicted following a series of break-ins, many of which involved phone systems and which made him the target of an FBI manhunt. They initially thought that he was a foreign agent who was abusing the U.S. phone system in order to wiretap sensitive U.S. targets. As I mentioned in Chapter 2, he testified after his release from prison that almost all of his exploits had involved social engineering. He came out with the quote at the head of this chapter: 'Companies can spend millions of dollars toward technological protections and that's wasted if somebody can basically call someone on the telephone and either convince them to do something on the computer that lowers the computer's defenses or reveals the information they were seeking' [895]. So phone company systems are vulnerable to careless insiders as well as malicious insiders — just like hospital systems and many others we've discussed.

A worrying recent development is the emergence of switching exploits by organisations. The protocols used between phone companies to switch calls — notably 5ESS — aren't particularly secure, as the move from in-band to out-of-band signaling was supposed to restrict access to trusted parties. But once again, changing environments undermine security assumptions. Now that there are many entrepreneurial phone companies rather than a handful of large ones, all sorts of people have access to the switching. An example is location service. This is provided for a fee by mobile networks; you can register your child's mobile, or your employees' mobiles, and trace them through a website. One entrepreneur undercut this service in the UK by using the switching interface exported by a local telco. While such issues can generally be resolved by contracts, litigation and regulation, there remains a lingering worry that attackers might bring down a telco by exploiting access to its switching and network management. This worry increases as telcos migrate

their networks to IP, and they start to converge with VOIP services that give users access to the IP layer. I'll return to VOIP later.

20.2.4 Insecure End Systems

After direct attacks on the systems kept on phone company premises, the next major vulnerabilities of modern phone systems are insecure terminal equipment and feature interaction.

There have been a number of cases where villains exploited people's answering machines. The same technique can be used for at least two different purposes: the relatively innocuous one of tricking someone into dialling a premium rate number, or the somewhat more sinister one of using their answering machine as a covert remailer for a voicemail message. The problem arises from phone company switches that give you dial tone twelve seconds after the other party hangs up. So a terrorist who wants to send an untraceable instruction to a colleague can record on your answering machine thirteen blank seconds, followed by the tones needed to dial his colleague's number and the secret message. He then calls again, gets the machine to play back its messages, and hangs up on it.

But the really big frauds using insecure end systems are directed against companies and government departments. Attacks on corporate *private branch exchange* systems (PBXes) had become big business by the mid-1990's and cost business billions of dollars a year [322]. PBXes are usually supplied with facilities for *refiling* calls, also known as *direct inward system access* (DISA). The typical application is that the company's sales force can call in to an 0800 number, enter a PIN or password, and then call out again taking advantage of the low rates a large company can get for long distance calls. As you'd expect, these PINs become known and get traded by villains [911]. The result is known as *dial-through* fraud.

In many cases, the PINs are set to a default by the manufacturer, and never changed by the customer. In other cases, PINs are captured by crooks who monitor telephone traffic in hotels anyway on order to steal credit card numbers; phone card numbers and PBX PINs are a useful sideline. Many PBX designs have fixed engineering passwords that allow remote maintenance access, and prudent people reckon that any PBX will have at least one back door installed by the manufacturer to give easy access to law enforcement and intelligence agencies (it's said, as a condition of export licensing). Of course such features get discovered and abused. In one case, the PBX at Scotland Yard was compromised and used by villains to refile calls, costing the Yard a million pounds, for which they sued their telephone installer. The crooks were never caught [1244]. This was particularly poignant, as one of the criminals' motivations in such cases is to get access to communications that will not be tapped. Businesses who're the victims of such crimes nevertheless find

the police reluctant to investigate, and one reason for this is that the phone companies aren't particularly helpful — presumably as they don't like having their bills disputed [1088].

In another case, Chinese gangsters involved in labor market racketeering — smuggling illegal immigrants from Fujian, China, into Britain where they were put to work in sweatshops, on farms and so on — hacked the PBX of an English district council and used it to refile over a million pounds' worth of calls to China. The gang was tackled by the police after a number of its labourers died; they were picking shellfish in Morecambe Bay when the tide came in and drowned them. The council had by now discovered the discrepancy in its phone bills and sued the phone company for its money back. The phone company argued that it wasn't to blame, even although it had supplied the insecure PBX. Here, too, the gangsters were interested not just in saving money but in evading surveillance. (Indeed, they routed their calls to China via a compromised PBX in Albania, so that the cross-border segment of the call, which is most likely to be monitored by the agencies, was between whitelisted numbers; the same trick seems to have been used in the Scotland Yard case, where the crooks made their calls via the USA.)

Such cases apart, dial-through fraud is mostly driven by premium rate services: the main culprits are crooks who are in cahoots with premium line owners. Most companies don't understand the need to guard their 'dial tone' and don't know how to even if they wanted to. PBXes are typically run by company telecomms managers who know little about security, while the security manager often knows little about phones. This is changing, albeit slowly, as VOIP technologies take over and the company phone network merges with the data network.

Exploits of insecure end-systems sometimes affect domestic subscribers too. A notorious case was the Moldova scam. In 1997, customers of a porn site were told to download a 'viewer' program that dropped their phone line and connected them to a phone number in Moldova (having turned off their modem speakers so they wouldn't notice). The new connection stayed up until they turned off their computers; thousands of subscribers incurred hundreds of thousands of dollars in international long distance charges at over \$2 per minute. Their phone companies tried to collect this money but there was an outcry. Eventually the subscribers got their money back, and the Federal Trade Commission enjoined and prosecuted the perpetrators [456]. Since then there have been a number of copycat scams [870]; but as more and more people move to cable modems or broadband, and their PCs are no longer able to dial out on the plain old telephone system, this kind of abuse is getting less common. The latest twist is premium-rate mobile malware: in 2006, for example, the Red Browser worm cashed out by sending \$5 SMSs to Russia [633].

Premium rate scams and anonymous calling are not the only motives. Now that phones are used more and more for tasks such as voting, securing entry into apartment buildings, checking that offenders are observing their parole terms, and authenticating financial transactions, more motives are created for ever more creative kinds of mischief, and especially for hacks that defeat caller line ID. For example, caller-line ID hacks make middleperson attacks on payment systems easier; SMS spoofing and attacks on the SS7 signaling in the underlying network can have similar effects [897].

And sometimes attacks are conducted by upstanding citizens for perfectly honorable motives. A neat example, due to Udi Manber, is as follows. Suppose you have bought something which breaks, and the manufacturer's helpline only has an answering machine. To get service, you have to take the answering machine out of service. This can often be done by recording its message, and playing it back so that it appears as the customer message. With luck the machine's owner will think it's broken and it'll be sent off for maintenance.

20.2.5 Feature Interaction

More and more cases of telephone manipulation involve feature interaction.

- Inmates at the Clallam Bay Correctional Center in Washington state, who were only allowed to make collect calls, found an interesting exploit of a system which the phone company ('Fone America') introduced to handle collect calls automatically. The system would call the dialled number and a synthesised voice would say: 'If you will accept a collect call from ... (name of caller) ... please press the number 3 on your telephone twice'. Prisoners were supposed to state their name for the machine to record and insert. The system had, as an additional feature, the ability to have the greeting delivered in Spanish. Inmates did so, and when asked to identify themselves, said 'If you want to hear this message in English, press 33'. This worked often enough that they could get through to corporate PBXes and talk the operator into giving them an outside line. The University of Washington was hit several times by this scam [476].
- A number of directory-enquiry services will connect you to the number they've just given you, as a service to motorists who can't dial while driving. But this can often be used to defeat mechanisms that depend on endpoint identification. Adulterers use it to prevent their spouses seeing lovers' numbers on the family phone bill, and naughty children use it to call sex lines despite call barring [977].

- Call forwarding is a source of many scams. In the old days, it was used for pranks, such as kids social-engineering a phone company operator to forward calls for someone they didn't like to a sex line. Nowadays, it's quite often both professional and nasty. For example, a fraudster may tell a victim to confirm her phone number with the bank by dialing a sequence of digits — which forwards her incoming calls to a number controlled by the attacker. So the bank's callback mechanisms are defeated when the customer isn't aware that a certain sequence of dialed numbers can alter the behavior of her phone.
- British Telecom launched a feature called 'Ringback'. If you dial an engaged number, you can then enter a short code and as soon as the called number is free, both your phone and theirs will ring. The resulting call is billed to you. However, when you used ringback used from a pay phone, it was the phone's owner who ended up with the bill. People with private pay phones, such as pub landlords and shopkeepers, lost a lot of money, which the phone company was eventually obliged to refund [652].
- Conference calls also cause a lot of trouble. For example, football hooligans in some countries are placed under a curfew that requires them to be at home during a match, and to prove this by calling the probation service, which verifies their number using caller ID. The trick is to get one of your kids to set up a conference call with the probation service and the mobile you've taken to the match. If the probation officer asks about the crowd noise, you tell him it's the TV and you can't turn it down or your mates will kill you. (And if he wants to call you back, you get your kids to forward the call.)

This brings us to the many problems with mobile phones.

20.3 Mobile Phones

Since their beginnings as an expensive luxury in the early 1980s, mobile phones have become one of the big technological success stories. By 2007, we now have over a billion subscribers; it's said that over a billion phones will be sold this year and the total subscriber base may rise to two billion. In developed countries, most people have at least one mobile, and many new electronic services are being built on top of them. Scandinavia has led here: you get a ferry ticket in Helsinki by sending a text message to the vending machine, and you get a can of Coke the same way. You can also scan a bar code at a bus stop with your phone camera, and get sent a text message 90 seconds before the next bus arrives; that way you don't have to stand out in the snow.

Growth is rapid in developing countries too, where the wireline network is often dilapidated and people used to wait years for phone service to be installed. In some places it's the arrival of mobile phone service that's connected villages to the world. Criminals also make heavy use of mobiles, and not just for communications: and in large tracts of the third world, mobile phone units have become a *de facto* currency. If you get kidnapped in Katanga, the kidnappers will tell your relatives in Kinshasa to buy mobile phone units and text them the magic numbers. In developed countries, the criminal interest is largely in communications, and most police wiretaps are now on mobile numbers.

So mobile phones are very important to the security engineer, both as part of the underlying infrastructure and as a channel for service delivery. They can also teach us a lot about fraud techniques and countermeasures.

20.3.1 Mobile Phone Cloning

The first generation of mobile phones used analog signals with no real authentication. The handset simply sent its serial numbers in clear over the air link. (In the U.S. system, there were two of them: one for the equipment, and one for the subscriber.) So villains built devices to capture these numbers from calls in the neighborhood. (I've even seen a phone that a student had reprogrammed to do this by a simple software hack.) One of the main customers was the *call-sell operation* that would steal phone service and resell it cheaply, often to immigrants or students who wanted to call home. The call-sell operators would hang out at known pitches with cloned mobiles, and their customers would queue up to phone home for a few dollars.

So a black market developed in phone serial numbers. The call-sell market was complemented by the market for anonymous communications for criminals: enterprising engineers built mobile phones which used a different identity for each call. Known as *tumblers*, these were particularly hard for the police to track [636]. The demand for serial numbers grew rapidly and satisfying it was increasingly difficult, even by snooping at places like airports where lots of mobiles were turned on. So prices rose, and as well as passive listening, active methods started to get used.

Modern mobile phones are cellular, in that the operator divides the service area up into cells, each covered by a base station. The mobile uses whichever base station has the strongest signal, and there are protocols for handing off calls from one cell to another as the customer roams. (For a survey of mobile phone technology, see [1061].) The active attack consists of a fake base station, typically at a place with a lot of passing traffic such as a freeway bridge. As phones pass by, they hear a stronger base station signal and attempt to register by sending their serial numbers.

A number of mechanisms were tried to cut the volume of fraud. Most operators developed or bought intrusion detection systems, which watch out

for suspicious patterns of activity. A number of heuristics were developed. For example, genuine mobiles which roam and call home regularly, but then stop calling home, have usually been stolen; other indicators include too-rapid movement (such as calls being made from New York and LA within an hour of each other) and even just a rapid increase in call volume or duration.

In the chapter on electronic warfare, I mentioned RF fingerprinting — a formerly classified military technology in which signal characteristics that arise from manufacturing variability in the handset's radio transmitter are used to identify individual devices and tie them to the claimed serial numbers [534]. Although this technique works — it was used by Vodafone in the UK to almost eliminate cloning fraud from analogue mobiles — it is expensive as it involves modifying the base stations. (Vodafone also used an intrusion detection system that tracked customer call patterns and mobility, described in [1283]; their competitor Cellnet simply blocked international calls from analogue mobiles, which helped move its high value customers to its more modern digital network.) Another proposed solution was to adopt a cryptographic authentication protocol, but there are limits on how much can be done without changing the whole network. For example, one can use a challenge-response protocol to modify the serial number [485]. But many of the mechanisms people proposed to fortify the security of analog cellular phones have turned out to be weak [1305].

Eventually the industry decided to upgrade to a new digital system. Revenue protection was an issue, but far from the only one; digital systems offered more efficient use of bandwidth, and a whole host of new features — including easier international roaming (important in Europe with lots of small countries jammed close together), and the ability to send and receive short text messages. (Text messages were almost an afterthought; the designers didn't realise they'd be hugely popular.) From the operators' viewpoint, the move to standard digital equipment cut costs and enabled rapid, wide-scale deployment.

20.3.2 GSM Security Mechanisms

The second generation of mobile phones adopted digital technology. Most handsets worldwide use the *Global System for Mobile Communications*, or GSM, which was designed from the start to facilitate international roaming; it was founded when 15 companies signed up to the GSM Association in 1987, and service was launched in 1992. As of 2007, the GSM system extends to over two billion handsets in over 200 countries; a typical developed country has more handsets in service than it has people [133]. The USA, Japan, Korea and Israel had different second-generation digital standards (although the USA has GSM service too). Since about 2001, most countries also have a third-generation service, which I'll describe in the next section.

The designers of GSM set out to secure the system against cloning and other attacks: their goal was that GSM should be at least as secure as the wireline system. What they did, how they succeeded and where they failed, make an interesting case history.

The authentication protocols are described in a number of places, such as [232] (which also describes the mechanisms in an incompatible U.S. system). The industry initially tried to keep secret the cryptographic and other protection mechanisms which form the core of the GSM protocols. This didn't work: some eventually leaked and the rest were discovered by reverse engineering. I'll describe them briefly here.

Each network has two databases, a *home location register* (HLR) that contains the location of its own mobiles, and a *visitor location register* (VLR) for the location of mobiles which have roamed in from other networks. These databases enable incoming calls to be forwarded to the correct cell.

The handsets are commodity items. They are personalised using a *subscriber identity module* (SIM) — a smartcard you get when you sign up for a network service, and which you load into your handset. The SIM can be thought of as containing three numbers:

1. there may be a personal identification number that you use to unlock the card. In theory, this stops stolen mobiles being used. In practice, many networks set an initial PIN of 0000, and most users never change it or even use it;
2. there's an *international mobile subscriber identification* (IMSI), a unique number that maps on to your mobile phone number;
3. finally there is a *subscriber authentication key* K_i , a 128-bit number that serves to authenticate that IMSI and is known to your home network.

Unlike the banks, which used master keys to generate PINs, the phone companies decided that master keys were too dangerous. So instead of diversifying a master key KM to manufacture the authentication keys as $K_i = \{IMSI\}_{KM}$, the keys are generated randomly and kept in an authentication database attached to the HLR.

The protocol used to authenticate the handset to the network runs as follows (see Figure 20.1). On power-up, the SIM may request the customer's PIN; if this isn't configured, or once it's entered correctly, the SIM emits the IMSI, which the handset sends to the nearest base station. It's relayed to the subscriber's HLR, which generates five *triplets*. Each triplet consists of:

- RAND, a random challenge;
- SRES, a response; and
- K_c , a ciphering key.

The relationship between these values is that RAND, encrypted under the SIM's authentication key K_i , gives an output which is SRES concatenated with K_c :

$$\{RAND\}_{K_i} = (SRES|K_c)$$

The standard way to do this encryption is using a one-way function called Comp128, or A3/A8. (A3 refers to the SRES output and A8 to the K_c output). Comp128 is a hash function with 40 rounds, described in detail in [226], and like most proprietary algorithms that were designed in the shadows and fielded quietly in the 1980s and 90s, it turns out to be vulnerable to cryptanalysis. The basic design of the function is much like in Figure 5.9 — each round consists of table lookups followed by mixing. There are five tables, with 512, 256, 128, 64 and 32 byte entries each, and the hash function uses them successively in each block of five rounds; there are eight of these blocks. This may seem very complex, but once its design became public, a vulnerability was soon noticed. Four of the bytes at the output of the second round depend only on the value of the same bytes of the input. This four-byte to four-byte channel is called a *narrow pipe* and it's possible to probe it by tweaking input bytes until you detect a collision. Once all the details have been worked out, it turns out that you need about 150,000 suitably chosen challenges to extract the key [1306, 1307]. The effect is that given access to a SIM issued by a network that uses Comp128, the authentication key can be extracted in several hours using software that is now freely available.

This attack is yet another example of the dangers of using a secret crypto primitive that has been evaluated by only a few friends; the cryptanalytic techniques necessary to find the flaw were well known [1287] and if Comp128 had been open to hostile public scrutiny, the flaw would most probably have been found. Thankfully, a phone company can replace Comp128 with a proper hash function such as SHA-256 without affecting anyone else; the hash function is present only in the SIM cards it issues to its customers and the software at its HLR. In any case, there don't seem to be any industrial-scale attacks based on a vulnerable hash function; the normal user doesn't have any incentive to crack his K_i out from his SIM as it doesn't let him bill calls to (or steal calls from) anyone else.

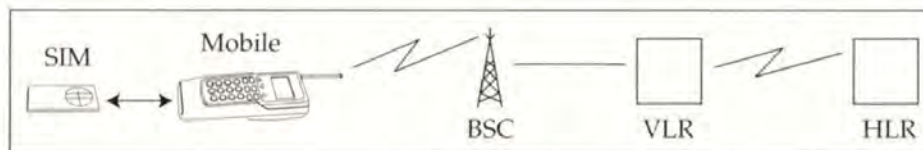


Figure 20.1: GSM authentication system components

Anyway, the triplets are sent to the base station, which now presents the first RAND to the mobile. It passes this to the SIM, which computes SRES. The mobile returns this to the base station and if it's correct the mobile and the base station can now communicate using the ciphering key K_c . So the whole authentication protocol runs as in Figure 20.2.

SIM → HLR	IMSI
HLR → BSC	(RAND, SRES, K_c), ...
BSC → SIM	RAND
SIM → BSC	SRES
BSC → mobile	{traffic} $_{K_c}$

Figure 20.2: GSM authentication protocol

There are several vulnerabilities in this protocol. In most countries the communications between base stations and the VLR pass unencrypted on microwave links¹. So an attacker could send out an IMSI of his choice and then intercept the triplet on the microwave link back to the local base station. A German mobile operator, which offered a reward of 100,000 Deutschmarks to anyone who could bill a call to a mobile number whose SIM card was held in their lawyer's office, backed down when asked for the IMSI [44].

Second, triples can be replayed. An unscrupulous foreign network can get five triples while you are roaming on it and then keep on reusing them to allow you to phone as much as you want. (Home networks could stop this by contract but they don't.) So the visited network doesn't have to refer back to your home network for further authorisation — and even if they do, it doesn't protect you as the visited network might not bill you for a week or more. So your home network can't reliably shut you down while you roam and it may still be liable to pay the roamed network the money. Dishonest networks also defraud roaming customers by *cramming* — by creating false billing records, a practice I'll describe in more detail later. So even if you thought you'd limited your liability by using a pre-paid SIM, you might still end up with your network trying to collect money from you. This is why, to enable roaming with a pre-paid SIM, you're normally asked for a credit card number. You can end up being billed for more than you expected.

¹The equipment can encrypt traffic, but the average phone company has no incentive to switch the cryptography on. Indeed, as intelligence agencies often monitor the backhaul near major switching nodes as an efficient means of getting warrantless access to traffic, a phone company that did switch on the crypto might find that persons unknown started jamming the link to make them stop.

The introduction of GSM caused significant shifts in patterns of crime generally. The authentication mechanisms made phone cloning difficult, so the villains switched their modus operandi to buying phones using stolen credit cards, using stolen identities or bribing insiders [1352]. Robbery was another issue. We've had a spate of media stories in Britain about kids being mugged for their phones. Mobile phone crime did indeed increase 190% between 1995 and 2002, but to keep this in context, the number of subscribers went up 600% in the same period [583]. Some of the theft is bullying — kids taking smaller kids' phones; some is insurance fraud by subscribers who've dropped their phones in the toilet and report them as stolen as their insurance doesn't cover accidental damage; but there is a hard core of theft where muggers take phones and sell them to fences. Many of the fences either work at mobile phone shops that have authorised access to tools for reprogramming the International Mobile Equipment Identifier (IMEI), the serial number in the handset, or else have links to organised criminals who ship the handsets abroad. Things are worse in Brazil, where kidnapping is endemic: there are now fake kidnappings in which a child's phone is snatched by a criminal who phones its parents to demand a ransom².

From about 1997, prepaid mobile phones were introduced. This kicked off a period of rapid growth in the industry as the technology became available to people without credit ratings. For example, prepaids make up 90% of the market in Mexico but 15% in the USA. Worldwide, they're over half, and growing. They also made anonymous communication much more practical, and many criminals have started using them. The issues include not just evading police wiretapping but stalking, extortion, bullying and other kinds of harassment. Prepaids also facilitate simple frauds; if your identity isn't checked when you buy a phone, there's little risk to you if you recharge it with a stolen credit card [343].

It must be said though that most people who use prepaid phones for crime are only getting lightweight anonymity, and remain undetected only because the police don't put serious effort into catching petty crooks. If a really serious crime is committed, traffic analysis will be used, and most criminals don't have any clue of the level of operational discipline needed to stop this. As I already remarked, the alleged 9/11 mastermind Khalid Shaikh Mohammed was caught when he used a prepaid SIM from the same batch as one that had been used by another Al-Qaida member; and after the failed 21/7 London bombings, the would-be bomber Husein Osman fled to Rome, where he was promptly caught. He had changed the SIM in his mobile phone en route; but

²There are also completely fake kidnappings in which the bad guys just lie: they say they've snatched the child, and if they call its phone it will be killed. The perpetrators of these crimes are often in prison and have little to lose. 20% of parents still pay up rather than take the risk.

call records show not just the IMSI from the SIM, but also the IMEI from the handset. If you've got all the world's police after you, just changing the SIM isn't anything like enough. Operational security requires a detailed technical understanding of how networks operate, and levels of training and discipline that are most unusual outside national intelligence agencies.

Finally, prepaid mobiles were a gift to crooked call-sell operators. As the billing infrastructure is only invoked when a phone goes on-hook, it's possible for a bad guy to make a call last all day: the numbers called by each of his clients are simply added to a long conference call one after the other. At the end of the day, the phone goes on-hook, a bill for thousands of dollars is generated, the alarm goes off, and the crook tosses the phone in the river. The following day he buys another. That's why network operators typically tear down any call that lasts more than a few hours.

In addition to authentication, the GSM system is supposed to provide two further kinds of protection — location security and call content confidentiality.

The location security mechanism is that once a mobile is registered to a network, it is issued with a *temporary mobile subscriber identification* (TMSI), which acts as its address as it roams through the network. The attack on this mechanism uses a device called an *IMSI-catcher*, which is sold to police forces [488]. The IMSI-catcher, which is typically operated in a police car tailing a suspect, pretends to be a GSM base station. Being closer than the genuine article, its signal is stronger and the mobile tries to register with it. The IMSI catcher claims not to understand the TMSI, so the handset helpfully sends it the cleartext IMSI. This feature is needed if mobiles are to be able to roam from one network to another without the call being dropped, and to recover from failures at the VLR [1283]. The police can now get a warrant to intercept the traffic to that mobile or — if they're in a hurry — just do a middleperson attack in which they pretend to be the network to the mobile and the mobile to the network.

The GSM system is supposed to provide call content confidentiality by encrypting the traffic between the handset and the base station once the authentication and registration are completed. The speech is digitized, compressed and chopped into packets; each packet is encrypted by xor-ing it with a pseudorandom sequence generated from the ciphering key K_c and the packet number. The algorithm commonly used in Europe is A5/1.

A5/1, like Comp128, was originally secret; like Comp128, it was leaked and attacks were quickly found on it. The algorithm is shown in Figure 20.3. There are three linear feedback shift registers of lengths 19, 22 and 23 and their outputs are combined using exclusive-or to form the output keystream. The nonlinearity in this generator comes from a majority-clocking arrangement whereby the middle bits c_i of the three shift registers are compared and the two or three shift registers whose middle bits agree are clocked.

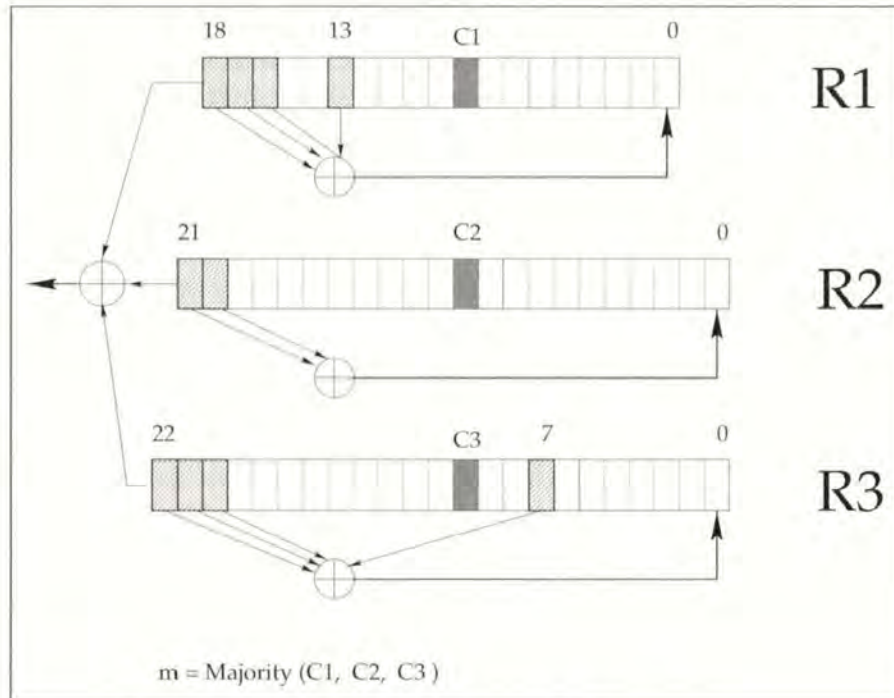


Figure 20.3: A5 (courtesy of Alex Biryukov and Adi Shamir)

The obvious attack on this arrangement is to guess the two shorter registers and then work out the value of the third. As there are 41 bits to guess, one might think that about 2^{40} computations would be needed on average. It's slightly more complex than this, as the generator loses state; many states have more than one possible precursor, so more guesses are needed. This led to an attack using a lot of FPGAs that is sold for about \$1m³. Then Biryukov and Shamir found some optimizations and tradeoffs that let A5 be broken with much less effort. Their basic idea is to compute a large file of special points to which the state of the algorithm converges, and then look for a match with the observed traffic. Given this precomputed file, the attack can use several seconds of traffic and several minutes' work on a PC, or several minutes of traffic and several seconds' work [173]. Reverse engineering actual systems also showed that the keying of A5 was deliberately weakened. Although in theory A5 has a 64-bit key (the initial loads for the shift registers) the actual implementations set the ten least significant key bits to zero. Anyway, the response of the GSM vendors to these disclosures was to introduce a third cipher, A5/3, which is based on a strong block cipher known as Kasumi that's also used in third-generation mobile phones.

³A system of 15 boxes, each with 20 cards, each with 18 ICs, each with 32 cores, each running at 150MHz, checking one key per Hz, takes 7.5 sec per key and burns 15KW.

However, this attempt to fix the content-confidentiality problem has been undermined by an attack that exploits the underlying authentication protocol and that is now coming into widespread use by police and surveillance folks. It relies on the fact that phone companies in many countries use a weakened version of A5/1, called A5/2. Mobile phones are generally able to use either algorithm, so that they can roam freely. The attack was invented by Elad Barkan, Eli Biham and Nathan Keller [117], and runs as follows. If you're following a suspect who uses his mobile, you record the call, including the initial protocol exchange of challenge and response. Once he's finished, you switch on your IMSI-catcher and cause him to register with your bogus base station. The IMSI-catcher tells his phone to use A5/2 rather than A5/1, and a key is duly set up — with the IMSI-catcher sending the challenge that was used before. So the mobile phone generates the same key K_c as before. As this is now being used in a weak cipher, it can be cracked quickly, giving access to the target conversation recorded previously. In fact it doesn't matter whether that was protected using the medium-security A5/1 or the high-security A5/3. The facts that there's a low-security algorithm, that keys are determined by one of the two principals, and that keys are shared between algorithms, together make the whole system weak.

The A5/1 vulnerability was introduced following pressure from Europe's intelligence agencies, and the ostensible justification was that they didn't want even halfways decent security exported to potentially hostile countries, as it would be harder to spy on them. The conspiracy theorists had a field day with all this — for example, there was a political row in Australia when it turned out that A5/2 was being used there, as it was seen as implying that European governments saw Australia as an intelligence target rather than an ally.

The truth is, as always, more subtle. Weak ciphers can definitely help in some tactical situations. Consider the case I mentioned in the chapter on electronic warfare, where the New Zealand navy sent a frigate to monitor a coup in Fiji. If the Fijian phone company had been allowed to use A5/1 rather than A5/2, this would not have frustrated the mission: the sigint officers could snatch the triplets off the microwave links, hack the location register, and if all else fails, brute force a key. But being able to break traffic quickly is very convenient.

On the other hand, imposing weak cipher security on less developed countries can also have operational costs. In 2007, the Taliban started tapping mobile phone calls made by British soldiers to friends and relatives back home, whom they then called. The wife of one RAF officer was told: 'You'll never see your husband alive — we have just killed him'. It was some hours before she could confirm that he was still safe and well [603]. British troops have now been banned from using mobile phones — which doesn't exactly help morale.

In domestic or friendly-country operations, it's even more complex. The agencies can get lawful access to cleartext from phone companies, but sometimes the warrant procedures are (or are felt to be) too cumbersome. Hence

the sweetheart deals with some phone companies to get access without warrants; this has led to a number of scandals recently. For example, it emerged that AT&T had been giving the US authorities access to itemised billing data for years. Sweetheart deals were common with companies that used to be monopolies, but are becoming less common as most countries have now deregulated telecomms and have hundreds of phone companies, not just one. Some phone companies are distrusted by the authorities because they're owned by foreigners or even by gangsters. Others are tiny, have never been tapped before, the police just don't know who to talk to. Other companies are respectable and willing to comply with warrants, but unable to do so. One of Britain's most closely guarded law-enforcement and intelligence secrets for several years was that one of the mobile phone networks couldn't be conveniently tapped. Engineers tried to hook up their complex, messy systems to the intelligence community's complex, messy systems but the project went off the rails and they just could not get the thing to work. The police were really scared that crooks might find this out, and migrate en masse to that network.

How are we to make sense of this mess? It's as well to admit that there are many reasons, some honourable and some less so, why government employees may want to get access to traffic by technical means rather than by using the lawful interfaces provided for the purpose. But that doesn't mean that such access should be allowed. The huge improvements in liberty, prosperity and quality of life that the West has enjoyed since the Enlightenment and the Industrial Revolution are due in large part to our replacing the divine right of kings with freedom under the law in a democracy. We elect legislators to make the rules, and if we decide through them that the police can't torture suspects any more, then they'd better not try. Similarly, if our legislators decide that police and intelligence agencies should get warrants to wiretap, then the agencies must obey the law like anyone else. The GSM security story provides a good example of how the deliberate insertion of vulnerabilities can have wide-ranging and unforeseen engineering consequences. It must also be said that the facilities that phone companies and ISPs are now being compelled to provide for properly warranted law-enforcement access are often so poorly engineered that they can be abused [1151]. In 2004–5, persons unknown tapped the mobile phones of the Greek Prime Minister and about a hundred of that country's political, law enforcement and military elite, by subverting the wiretapping facilities built into Vodafone's Greek network. Both Vodafone, and their equipment supplier Ericsson, were heavily fined [1042]. Colleagues and I warned about this problem years ago [4] and I expect it to get worse. I'll discuss it at greater length in Part III.

There are further policy issues with location privacy. There was a storm in Switzerland in 1997 when the press found that the phone company was routinely giving location data to the police [1030], while in the USA, the FCC ordered mobile phone companies to be able to locate people 'so that 911 calls

could be dispatched to the right place'. This was imposed on every user of mobile phone service, rather than letting users decide whether to buy mobile location services or not. Privacy activists were not happy with this.

Anyway, the net effect is that the initial GSM security mechanisms provided slightly better protection than the wireline network in countries allowed to use A5/1, and slightly worse protection elsewhere, until the protocol attacks were discovered and exploited. Now privacy is slightly worse everywhere, as people with the right equipment can get fairly straightforward access to traffic. But it's probably not a big deal. Relatively few people ever get followed around by an investigator with professional snooping equipment — and if you're such a person, then ways to detect and prevent semi-active attacks on your mobile phone are just a small part of the tradecraft you need to know. If you're an average subscriber, the privacy threat comes from possible abuse of data collected by the phone company, such as itemized billing data. From that viewpoint, the vulnerabilities in the communications security mechanisms neither expose you to additional wiretapping, nor prevent the frauds that are likely to cause you the most grief.

20.3.3 Third Generation Mobiles – 3gpp

The third generation of digital mobile phones was initially known as the *Universal Mobile Telecommunications System* (UMTS) and now as the *Third Generation Partnership Project* (3gpp, or just 3g). These systems are now available almost everywhere, the exception being in China which is working on its own proprietary variant. Elsewhere, the security is much the same as GSM, but upgraded to deal with a number of GSM's known vulnerabilities. Third generation systems entered service in 2003–2004; the main advantage of 3g over GSM is higher data rates; instead of the 9.6kb/s of GSM and the tens of kilobits per second of GPRS, third-generation data rates are in the hundreds of thousands to millions of bits per second. The vision is that 3g will enable all sorts of mobile services, from mobile TV to laptops that just go online anywhere.

The overall security strategy is described in [1310], and the security architecture is at [1298]. The crypto algorithms A5/1, A5/2 and Comp128 are replaced by various modes of operation of a block cipher called Kasumi [696]. Kasumi is public and is based on a design by Mitsuru Matsui called Misty, which was properly peer-reviewed and has now withstood public scrutiny for a decade [844]. All keys are now 128 bits. Cryptography is used to protect the integrity and confidentiality of both message content and signalling data, rather than just content confidentiality, and the protection at least runs from the handset to a fairly central node, rather than simply to the local base station. This means the picking up the triples, or the plaintext, from the microwave backhaul is no longer an attack. The authentication is now two-way rather

than one-way, ending the vulnerability to rogue base stations; so IMSI-catchers don't work against third generation mobiles. Instead, there is a properly engineered interface for lawful interception [1299]. This can supply key material as well as plaintext, so that if the police follow a suspect, record a call and identify the mobile in the process, they can decrypt that call later, rather than being limited to the plaintext of calls recorded by the phone company after they get their warrant approved.

The protocol mechanics work as follows (see Figure 20.4). In the basic 3gpp protocol, the authentication is pushed back from the base station controller to the visitor location register. The home location register is now known as the *home environment* (HE) and the SIM as the *UMTS SIM* (USIM). The home environment chooses a random challenge RAND as before and enciphers it with the USIM authentication key K to generate a response RES, a confidentiality key CK , and integrity key IK , and an anonymity key AK .

$$\{RAND\}_K = (RES|CK|IK|AK)$$

There is also a sequence number SEQ known to the HE and the USIM. A MAC is computed on RAND and SEQ, and then the sequence number is masked by exclusive-or'ing it with the anonymity key. The challenge, the expected response, the confidentiality key, the integrity key, and the masked sequence number made up into an *authentication vector* AV which is sent from the HE to the VLR. The VLR then sends the USIM the challenge, the masked sequence number and the MAC; the USIM computes the response and the keys, unmaskes the sequence number, verifies the MAC, and if it's correct returns the response to the VLR.

USIM → HE	IMSI (this can optionally be encrypted)
HE → VLR	RAND, XRES, CK, IK, SEQ ⊕ AK, MAC
VLR → USIM	RAND, SEQ ⊕ AK, MAC
USIM → VLR	RES

Figure 20.4: 3gpp authentication protocol

The UMTS standards set out are many other features, including details of sequence number generation, identity and location privacy mechanisms, backwards compatibility with GSM, mechanisms for public-key encryption of authentication vectors in transit from HEs to VLRs, and negotiation of various optional cryptographic mechanisms.

The net effect is that confidentiality will be improved over GSM: eavesdropping on the air link is prevented by higher-quality mechanisms, and the current attacks on the backbone network, or by bogus base stations, are excluded. Police wiretaps are done at the VLR. In a number of countries, third-generation mobiles were hard for the police to tap in the first few years,

as they had to learn to operate through formal channels and also to integrate their systems with those of the network operators. In the second phase, it's proposed to have end-to-end encryption, so that the call content and some of the associated signaling will be protected from one handset to another. This led to government demands for a *key escrow protocol* — a protocol to make keys available to police and intelligence services on demand. The catch is that if a mobile phone call takes place from a British phone company's subscriber using a U.S. handset, roaming in France, to a German company's subscriber roaming in Switzerland using a Finnish handset, and the call goes via a long distance service based in Canada and using Swedish exchange equipment, then which of these countries' intelligence agencies will have access to the keys? [1299] (Traditionally, most of them would have had access to the call content one way or another.)

One solution pushed by the agencies in Britain and France is the so-called Royal Holloway protocol [663], designed largely by Vodafone, which gives access to the countries where the subscribers are based (so in this case, Britain and Germany). This is achieved by using a variant of Diffie-Hellman key exchange in which the users' private keys are obtained by encrypting their names under a super-secret master key known to the local phone company and/or intelligence agency. Although this protocol has been adopted in the British civil service and the French health service, it is at odds with the phone company security philosophy that master keys are a bad thing. The protocol is also clunky and inefficient [76].

So 3gpp won't provide a revolution in confidentiality, merely a modest improvement. As with GSM, its design goal is that security should be comparable with that of the wired network [621] and this looks like being achieved.

20.3.4 Platform Security

The final point I need to make here is that as mobile phones become more widespread and more programmable, they may suffer from the malware problems that have plagued the PC. They have followed the pattern predicted by security economics. At first, the platform vendors — the firms selling operating systems, such as Symbian, Microsoft and Linux — didn't incorporate much security, as it would have got in application developers' way and appealing to complementers is vital when building share in a new market with network externalities. Then, as one platform pulled ahead of the others, the malware writers targeted it. In 2007, viruses and worms are being detected at the rate of about 300 per annum for Symbian phones, and one or two a year for the others. Symbian has started a program of hardening their platform, with progressively more sophisticated access controls, code signing, and so on.

Mobile phone platforms have also acquired DRM mechanisms. The Open Mobile Alliance DRM version 1 supports the download of ringtones, while version 2 supports music and video download with more general mechanisms. Version 1 is widely used, but version 2 has been held up by an interesting dispute. The main OMA promoter, Nokia, would like to become the distribution channel of choice for mobile music, just as Apple has become for PCs with iTunes. However the network operators are extremely reluctant to let Nokia have a business relationship with their customers directly, and this has held up deployment.

In general, security is made more difficult by the long and complex supply chain that delivers mobile phone service to customers. IP companies like ARM own the chip designs; foundries such as Infineon make the chips; handset designers like Samsung manufacture the actual mobiles; Symbian provides the operating system; Nokia may provide some more software for a download interface; a network operator such as Vodafone provides the national infrastructure; and a local operating company then bills the customer. There has been a tendency for everyone in this chain to see security as a problem to be tossed over the fence to the companies on either side. On top of this there might be apps provided by games vendors, and corporate apps such as fleet management that are provided by third-party software houses. Add the next generation of location-based services and offerings from the likes of Google and eBay, and the whole thing becomes complex beyond belief.

One final aspect of mobile phone platforms is locking. In many countries, handsets are subsidised out of future call revenue: you get a cheap phone in return for a year's contract with a network. The downside is that your phone is locked to the network. Even some prepaid phones are locked, and carry an implicit subsidy from expected future token sales. However, in some countries, this business model isn't permitted. There has thus arisen a brisk trade in unlocking tools and services, some of which are devised by finding exploits in the phone software, and others using the kind of hardware reverse-engineering techniques I described in Chapter 16. Legal skirmishing between the phone companies and the unlocking services came to a head after Apple launched the iPhone, which had a twist on this business model: the networks for which you could buy it were those that had paid Apple the most for the privilege. This annoyed iPhone purchasers as the U.S. network of choice, AT&T, was fairly slow. The iPhone was duly hacked, and AT&T sent its lawyers after the unlockers. It also shipped a software upgrade that disabled unlocked phones.

Laws on locking and unlocking vary widely. In the USA, the Digital Millennium Copyright Act (DMCA), which prohibits interference with any technical mechanism that enforces copyright, has a specific exemption for cellphone unlocking, in order that copyright law shouldn't be abused to stifle competition in the mobile phone market. It's now being argued that

this exemption covers only people who unlock their own phones, but doesn't extend to the sale of unlocking tools or services [821]. So in America, unlocking is legal if you're a customer (though the provider may brick your phone), and may be open to challenge if you do it commercially. At the other end of the scale, courts in France and Germany have held mobile phone locking to be illegal: Apple will have to offer unlocked models for sale there if it offers the product at all. Such variations in law and business practice have led to the development of a thriving grey market whereby phones are shipped from one country to another; this in turn has driven secondary markets for unlocking tools and even for assorted frauds.

20.3.5 So Was Mobile Security a Success or a Failure?

Whether mobile-phone security has been a success or a failure depends on whom you ask.

From the point of view of cryptography, it was a failure. Both the Comp128 hash function and the A5 encryption algorithm were broken once they became public. In fact, GSM is often cited as an object lesson in Kerckhoffs' Principle — that cryptographic security should reside in the choice of the key, rather than in the obscurity of the mechanism. The mechanism will leak sooner or later and it's better to subject it to public review before, rather than after, a hundred million units have been manufactured. (GSM security wasn't a disaster for most cryptographers, of course, as it provided plenty opportunities to write research papers.)

From the phone companies' point of view, GSM was a success. The shareholders of GSM operators such as Vodafone have made vast amounts of money, and a (small) part of this is due to the challenge-response mechanism in GSM stopping cloning. The crypto weaknesses were irrelevant as they were never exploited (at least not in ways that did significant harm to call revenue). There are one or two frauds that persist, such as the long conference call trick; but on balance the GSM design has been good to the phone companies.

From the criminals' point of view, GSM was also fine. It did not stop them stealing phone service: the *modus operandi* merely changed, with the cost falling on credit card companies or on individual victims of 'identity theft' or street robbery. It did not stop calls from anonymous phones; the rise of the prepaid phone industry made them even easier. (The phone companies were happy with both of these changes.) And of course GSM did nothing about dial-through fraud.

From the point of view of the large-country intelligence agencies, GSM was fine. They have access to local and international traffic in the clear anyway, and the weakened version of A5 facilitates tactical signint against developing countries. And the second wave of GSM equipment is bringing some juicy features, such as remote control of handsets by the operator [1061]. If you can

subvert (or masquerade as) the operator, then there seems to be nothing to stop you quietly turning on a target's mobile phone without his knowledge and listening to the conversation in the room.

From the point of view of the police and low-resource intelligence agencies, things are not quite so bright. The problem isn't the added technical complexity of GSM networks: court-ordered wiretaps can be left to the phone company (although finding the number to tap can be a hassle if the suspect is mobile). The problem is the introduction of prepaid mobile phones. This not only decreases the signal to noise ratio of traffic analysis algorithms and makes it harder to target wiretaps, but also encourages crimes such as extortion and stalking.

From the customer's point of view, GSM was originally sold as being completely secure. Was this accurate? The encryption of the air link certainly did stop casual eavesdropping, which was an occasional nuisance with analog phones. (There had been some high-profile cases of celebrities being embarrassed, including one case in Britain where Prince Charles was overheard talking to his mistress Camilla Parker-Bowles before his divorce from Princess Diana, and one in the USA involving Newt Gingrich.) But almost all the phone tapping in the world is done by large intelligence agencies, to whom the encryption doesn't make much difference.

Things are even less positive for the subscriber when we look at billing. Cryptographic authentication of handsets can't stop the many frauds perpetrated by premium rate operators and phone companies. If anything it makes it harder to wriggle out of bogus charges, as the phone company can say in court that your smartcard and your PIN must have been used in the handset that made the call. The same will apply to 3rd generation phones. The one minor compensation is that GSM facilitated the spread of prepaid phones, which can limit the exposure.

So the security features designed into GSM don't help the subscriber much. They were designed to provide 'security' from the phone company's point of view: they dump much of the toll fraud risk, while not interrupting the flow of premium rate business — whether genuine or fraudulent.

In the medium term, the one ray of comfort for the poor subscriber is that the increasing complexity of both handsets and services may create regulatory pressure for transparent mechanisms that enable the customer to control the amount she's billed. There are a number of factors pushing for this, such as the growing vulnerability of platforms to malware; and a number of factors pushing in the other direction, such as the phone companies' desire to keep pricing opaque so that they can rip customers off. I mean this not just in the strict sense that phone companies often defraud their customers, but also in the colloquial sense that confusion pricing is a mainstay of phone company economics. I'll go into this in more detail in the next section, where I'll look at the economics of telecomms and how they relate to fraud and abuse.

20.3.6 VOIP

The latest development in telephone is voice over IP (VOIP), in which voice traffic is digitised, compressed and routed over the Internet. This had experimental beginnings in the 1970s; products started appearing in the 1990s; but decent call quality requires bandwidth of about 80 kbit/sec, more than can be got from a dial-up modem. As a result, VOIP only really took off once a critical mass of households had broadband connections. Since about 2005, it has become big business, with eBay purchasing Skype in 2006 for \$2.6bn. In fact, most normal phone calls are digitized and sent over IP networks belonging to the phone companies, so in a technical sense almost all phone calls are 'VOIP', but in line with common usage I'll use the term only for those calls made by customers over an IP service that they access directly, via their ISP.

The VOIP market is still fragmented, and suffers many of the problems you'd expect. Most products were shipped quickly in a race to market, with security an afterthought at best. There was the usual crop of stack overflows and other vulnerabilities at the implementation level. The most popular VOIP protocol, the Session Initiation Protocol (SIP), turns out to have vulnerabilities that enable third parties to wrongly bill calls to subscribers [1375]. These could give rise to difficult disputes between VOIP services and their customers.

There are many issues with VOIP as the market shakes down. The leading system, Skype, is a closed system using proprietary protocols while its many competitors are fragmented. One business opportunity is 'click-to-call' whereby an advertisement could contain a VOIP link enabling a prospective customer to click to speak to a representative; early efforts have run up against compatibility problems. Technical problems range from differing protocols through how to deal with the jitter (time delay variation) caused by the variable quality of service on the Internet, and how to deal with network address translation are firewalls.

The interaction with security is complex. Corporate security policies can result in firewalls refusing to pass VOIP traffic. Phone calls can be more secure if made over VOIP, as encryption is easy to add and with some services (such as Skype) it comes turned on by default. For some time at least, it may be more difficult for police and intelligence services to get access to call contents and to records of who called whom; many services have not yet got round to engineering a law-enforcement interface, and in the case of Skype, its peer-to-peer nature might make that difficult. The FBI is currently pushing for the CALEA regulations to be applied rigorously to VOIP providers, while the industry and civil liberties groups are resisting. (I'll have more to say about this in Part III.) Wiretaps are not the only issue; click-to-call clearly will have other security, privacy and safety issues in the context of social networking sites, especially those used by minors.

A more high-profile regulatory issue is that governments want emergency calls made through VOIP services to work reliably, and provide information about the location of the caller. This is hard; an IP packet stream can be coming from anywhere, and no-one owns enough of the Internet to guarantee quality of service. At a deeper level than that, the dispute over dependability is just the latest tussle in the forty years' war between computer companies and phone companies, which I've alluded to from time to time. Computer companies are innovative and entrepreneurial, racing to market with products that are just about good enough; phone companies are slow-moving and heavily regulated, with a fifteen-year product cycle and services engineered for 99.999% availability. Ever since people started using modems on a large scale in the 1960s, there has been one battle after another — which the computer companies pretty well always win. A glimpse into the future may have been provided by a two-day Skype outage in August 2007, caused by poorly-engineered protocols. After a large number of computers rebooted following Microsoft's 'patch Tuesday' security update, tens of millions of Skype clients simultaneously tried to contact the network. The congestion caused the network to fail, yet the clients kept on trying. We can probably expect a lot more failures. Although a VOIP handset looks like a phone and works like a phone, it's more like email in terms of reliability. If the power goes off, so does your service.

The main problems beyond that have to do with the incompatibility of the phone companies' business model and the nature of the Internet. Phone companies make their money by charging you vastly different rates for different types of content: their typical rates work out at a hundredth of a cent per megabyte for cable TV, eight cents per megabyte for wireline phone, three dollars a megabyte for mobile phone, and a whopping three thousand dollars a megabyte for text messages. The opportunity exploited by VOIP is to arbitrage these, and it's hardly surprising that the phone companies do what they can to get in the way. ISPs who are also phone companies deliberately cause problems for VOIP to stop it competing with their phone service; this is particularly pronounced with mobile IP service. For these reasons, we'd better look at the economics of phone companies and the interaction with security.

20.4 Security Economics of Telecomms

Phone companies are classic examples of a business with extremely high fixed costs and very low marginal costs. Building a nationwide network costs billions and yet the cost of handling an additional phone call is essentially zero. As I discussed in Chapter 7 on Economics, this has a couple of implications.

First, there's a tendency towards dominant-firm markets in which the winner takes all. Indeed for many years telephone service was considered in

most countries to be a 'natural monopoly' and operated by the government; the main exception was the USA where the old AT&T system was heavily regulated. After the breakup of AT&T following an antitrust case, and Margaret Thatcher's privatisation of BT, the world moved to a different model, of regulated competition. The details vary from one country to another but, in general, some sectors (such as mobile phones) had a fixed number of allowed competitors; others (such as long-distance provision) were free for companies to compete in; and others (such as local loop provision) remained de facto monopolies but were regulated.

Second, because the marginal cost of service provision is zero, the competitive sectors (such as long-distance calling) saw prices drop quickly to a very low level — in many cases below the level needed to recoup the investment. (The large investments made during the dotcom bubble end up being treated as sunk costs.)

In such a market, firms will try to maintain price discrimination by whatever means they can. In many telecomms markets, the outcome is *confusion pricing* — products are continually churned, with new offerings giving generous introductory discounts to compete with the low-cost providers, but with rates sneakily raised afterwards. The effect is to discriminate between 'people who will spend money to save time, and people who will spend time to save money'. If you can be bothered to continually check prices, you can get really good deals, but often at the cost of indifferent service. If you don't have the time to keep scrutinising your phone bills, and the latest emails you get from your ISP advising you of the latest service changes, you can find that the call home you made from your mobile while on business in France just ate up all that month's savings. In the end, you can end up paying about the same amount per month that you did in the past. Andrew Odlyzko, a scholar of phone-company economics, suggests the eventual way forward will be fixed-price contracts: for a certain amount per month you'll get a home phone, ISP service, a couple of mobiles and a decent allowance of air minutes and texts [982]. In the meantime, telecomms pricing remains murky, contentious and far from transparent. This leads directly to abuse.

20.4.1 Frauds by Phone Companies

One of the steadily growing scams is the unscrupulous phone company that bills lots of small sums to unwitting users. It collects phone numbers in various ways. (For example, if you call an 800 number, then your own number will be passed to the far end regardless of whether you tried to block caller line ID.) The wicked phone company then bills you a few dollars. Your own phone company passes on this charge and you find there's no effective way to dispute it. Sometimes the scam uses a legal loophole: if you call an 800 number in the USA, the company may say 'Can we call you right back?' and if you agree

then you're deemed to have accepted the charges, which are likely to be at a high premium rate. The same can happen if you respond to voice prompts as the call progresses. These practices are known as *cramming*.

I was myself the victim on an attempt at cramming. On holiday in Barcelona, my wife's bag was snatched, so we called up and cancelled the phone that she'd had in it. Several months later, we got a demand from our mobile provider to pay a few tens of dollars roaming charges recently incurred by that SIM card in Spain. In all probability, the Spanish phone company was simply putting through a few charges to a number that they'd seen previously, in the knowledge that they'd usually get away with it. My mobile service provider initially insisted that even though I'd cancelled the number, I was still liable for calls billed to it months afterwards and had to pay up. I got out of the charges only because I'd met the company's CEO at an academic seminar and was able to get his private office to fix the problem. Customers without such access usually get the short end of the stick. Indeed, UK phone companies' response to complaints has been to offer its customers 'insurance' against fraudulent charges. That they can get away with this is a clear regulatory (and indeed policing) failure.

Another problem is *slamming* — the unauthorized change of a subscriber's long distance telephone service provider without their consent. The slammers tell your local phone company that you have opted for their service; your phone company routes your long distance calls through them; they hope you don't notice the change and dispute the bill; and the telephone charges can then be jacked up. Some local phone companies, such as Bell Atlantic, allow their customers to freeze their chosen long distance carrier [22].

It would be a mistake to assume that cramming and slamming are just done by small fly-by-night operators. AT&T is one of the worst offenders, having been fined \$300,000 not only for slamming, but for actually using forged signatures of subscribers to make it look as if they had agreed to switch to their service. They got caught when they forged a signature of the deceased spouse of a subscriber in Texas [390]. As for the UK, slamming wasn't even made illegal until 2005.

Another problem is the fly-by-night phone company. As anyone in the USA is legally entitled to set up a phone company, it is straightforward to set one up, collect some cash from subscribers, and then vanish once the invoices for interconnect fees come in. Companies also advertise sex lines with normal phone numbers to trap the unwary, then send huge bills to the subscriber at his residential addresses and try to intimidate him into paying. The regulation of premium-rate providers and their business practices is a widespread problem.

And it's not just the small operators that indulge in sharp practice. An example that affects even some large phone companies is the short termination of international calls.

Although premium-rate numbers are used for a number of more or less legitimate purposes such as software support, many of them exploit minors or people with compulsive behaviour disorders. So regulators have forced phone companies in many countries to offer premium-rate number blocking to subscribers. Phone companies get round this by disguising premium rate numbers as international ones. I mentioned scams with Caribbean numbers in section 20.2.1 above. Now many other phone companies from small countries with lax regulators have got into the act, and offer sex line operators a range of numbers on which they share the revenue.

Often a call made to a small-country phone company doesn't go anywhere near its ostensible destination. One of the hacks used to do this is called *short termination*, and here's an example that surfaced a few years ago. Normally calls for the small Pacific country of Tuvalu went via Telstra in Perth, Australia, where they were forwarded by satellite. However, the sex line numbers were marked as invalid in Telstra's system, so they were automatically sent via the second-choice operator — a company in New Zealand. (The girls — or to be more precise, the elderly retired hookers who pretend to be young girls — were actually in Manchester, England.) Technically, this is an interesting case of a fallback mechanism being used as an attack vehicle. Legally, it is hard to challenge as there is an international agreement (the Nairobi Convention) that stops phone companies selectively blocking international destinations. So if you want to stop your kids phoning the sex line in Tuvalu, you have to block all international calls, which makes it harder for you to phone that important client in Germany.

Problems like these are ultimately regulatory failures, and they are increasingly common. (For example, in the Moldova scam I mentioned earlier, the calls didn't go to Moldova but to Canada [251].) They are continuing to get worse as technology makes new, complex services possible and the regulators fail to keep up.

20.4.2 Billing Mechanisms

Billing mechanisms are a growing source of problems, as the economic forces discussed above lead to ever-more-complex rate cards. Even the phone companies themselves sometimes fall foul of the growing complexity. Sometimes their rates get so complex that people can arbitrage against them; there has been more than one case in which an entrepreneur found he could set up an international premium-rate service and be paid more per minute for calling it, than it cost him to call it using the best discount rate. Phone companies have tried to recover such trading profits through the courts, claiming fraud — with mixed success.

The security of the billing mechanisms covers a much wider range of issues. Present arrangements are inadequate for a number of reasons.

- A *call detail record* (CDR) is only generated once the calling phone goes on-hook. This was a long-established feature of wireline networks, but once the environment changed to mobile it became a serious problem. As I mentioned above, someone running a call-sell operation can set up a long conference call using a stolen or prepaid mobile, which clients join and leave one after the other. The phone stays off-hook continuously for hours. As soon as it goes on-hook, a CDR for several thousand dollars is generated, and the alarm goes off. The operator throws the phone in the river and starts using the next one. By 1996, this had become so serious that Vodafone introduced a six hour limit on all mobile calls. But it won't be acceptable to just drop all 3gpp calls after six hours. Many users are expected to have always-on internet connections (such as from their laptops) with relatively light packet traffic most of the time.
- More seriously, the back-end accounting system was designed in the days when phone companies were sleepy government departments or national monopolies, and there were no premium-rate services through which real money could be extracted from the system. So it has little in the way of audit and non-repudiation. In effect, phone company A tells phone company B, 'please debit your customer no. X the sum of \$Y and send me the money' — and it does. Even when these debits are mistaken or fraudulent, phone company B has no incentive to quibble, as it gets a cut. The result, as we saw with the cramming cases above, is that fraud slowly rises as insiders abuse the system. This is no longer fit for purpose in a world with many phone companies, quite a few of which are unscrupulous. The regulators aren't effective, and the only real backward pressure comes from the growing number of prepay customers.
- The phone companies also want to be able to charge for relatively high-value product and service delivery, extending the current premium services through location-based services ('give me a map showing me how to drive to the nearest McDonalds') to music and video downloads and extra services such as the Finnish ferry tickets, cans of coke from vending machines, and (most recently) parking meters in London. The accounting system will have to become a lot more robust, and dispute resolution mechanisms fairer and more transparent, if this potential is to be realised.
- All this interacts with platform security. If malware becomes widespread on mobile phones, then the botnet herders who control subverted phones will be able to pay for all sorts of goods and services by getting infected machines to send text messages. Recent history suggests that any exploits that can be industrialised to make money on a large scale, will be.

So how can phone payment systems be improved?

One proposed way of implementing this is to incorporate a micropayment mechanism [92]. The idea is that the phone will send regular *tick payments* to each of the networks or service providers which are due to be paid for the call. The tick payments can be thought of as electronic coins and are cryptographically protected against forgery. At the start of the call, the handset will compute a number of phone ticks by repeated hashing: $t_1 = h(t_0)$, $t_2 = h(t_1)$, and so on, with t_k (for some credit limit k , typically 2^{10} units) being signed by the phone company. The phone will then release ticks regularly in order to pay for the services as the call progresses. It starts by releasing t_k , then t_{k-1} , then t_{k-2} and so on. If a charge is subsequently disputed — whether by a subscriber or a network operator — the party claiming an amount of (say) j ticks must exhibit the ticks t_{k-j} and t_k , the latter with a certificate. As the hash function h is one-way, this should be hard to do unless the handset actually released that many ticks. The tick t_{k-j} can now be checked by applying the hash function to it j times and verifying that the result is t_k ⁴. One advantage of tick payments is that as well as protecting the phone companies from conference call frauds, it could protect the subscriber from many more. It could enable phone application designers to empower users in new ways: for example, you might by default decide to accept calls, or play games, or do other functions, only so long as they cost less than a dollar, and require user intervention for anything more expensive. At present, that kind of functionality just isn't available, except via the rather clunky mechanism of only loading so much airtime into your phone.

The industry's proposed solution is to redesign the call data record to contain a lot more information. In addition to time, duration and called number, it will have fields for data quantity, location and quality-of-service. This is not just to support possible future differential charging for quality of service, but also to help with emergency call tracking, and to comply with a 2002 European directive on telecomms requires all mobile operators retain location information on mobile phones for at least a year, for law enforcement purposes. There was a proposal for an online cost-control mechanism to limit the charges incurred for each user [901], but this appears to have stalled. The cost-control mechanisms are not being standardized but can involve forwarding charging data from either the local network or the gateway to the home environment which will be able to have the call terminated if the available credit is exhausted (as with a prepaid SIM card) or if the use appears to be fraudulent. It's tempting to draw a parallel with NATO's failure to make IFF work properly across different countries' armed forces, which I discussed in Chapter 19. The public good (in this case, transparent and dependable cost

⁴This protocol is an example of multiple simultaneous discovery, having been invented by our group at Cambridge, by Pedersen, and by Rivest and Shamir, independently in 1995 [40, 1013, 1077].

control) isn't high on the agendas of the stakeholders who'd have to work together to make it happen. Indeed, it's even worse. It would be hard to find a general anywhere in North America or Europe who didn't agree that decent IFF would be a Good Thing; but many of the major stakeholders depend for their existence on confusion pricing and would see a decent charging system as a serious threat.

The extreme difficulty of engineering a global solution has left the market open to a variety of local ones. In the USA, there is a move to integrate RFID-based credit cards with NFC-compatible mobile phones. In the UK, a scheme called PayForIt was launched in 2007 by the main mobile operators that aims to replace premium SMS services with a WAP-based protocol that 'provides a uniform payment experience' and requires 'clear pricing, access to terms and conditions and merchant contact details before a purchase is confirmed'. O2, one of the big networks, required all its users to switch from June 2007. They said there would be 'reduced customer care issues'.

Personally, I am sceptical, because of the lack of bankable guarantees for the customer; the terms and conditions state that 'any queries or complaints regarding Goods and Services must be referred to the Supplier,' while firms advertising PayForIt transaction acquisition quote fixed prices and seem ready to accept all comers, respectable or otherwise. Perhaps the scheme will simply hold up the growth of phone-based payments by making them more fiddly and restricting them to more capable handsets, while not resolving the underlying trust problem. A law that enforced customer rights would be better for all. In its absence, the phone companies appear to be setting up a banking system but without the regulations and controls that long and bitter experience has shown necessary for bank stability and trustworthiness. In civilised countries, mafiosi are not allowed to run banks. But gangsters have a basic human right (in America at least) to own a phone company. So phone companies should not run banks.

20.5 Summary

Phone fraud is a fascinating case study. People have been cheating phone companies for decades, and recently the phone companies have been vigorously returning the compliment. To start off with, systems were not really protected at all, and it was easy to evade charges and redirect calls. The mechanism adopted to prevent this — out-of-band signalling — proved inadequate as the rapidly growing complexity of the system opened up many more vulnerabilities. These range from social engineering attacks on users through poor design and management of terminal equipment such as PBXes to the exploitation of various hard-to-predict feature interactions.

On the mobile front, the attempts to secure GSM and its third generation successor make an interesting case study. Their engineers concentrated on communications security threats rather than computer security threats, and they concentrated on the phone companies' interests at the expense of the customers'. Their efforts were not entirely in vain but did not give a definitive solution.

Overall, the security problems in telecomms are the result of environmental changes, including deregulation, which brought in many new phone companies. But the main change was the introduction of premium rate numbers. While previously phone companies sold a service with a negligible marginal cost of provision, suddenly real money was involved; and while previously about the only serious benefit to be had from manipulating the system was free calls, or calls that were hard for the police to tap, suddenly serious money could be earned. The existing protection mechanisms were unable to cope with this evolution. However, the major phone companies are so threatened by price competition that their business models are now predicated on confusion pricing. So the incentives for an overhaul of the billing mechanisms are just not there.

Ultimately, I suspect, the regulator will have to step in. The best solution in the USA could be the extension of Regulation E, which governs electronic banking, to phone companies — as they have become *de facto* banks. When you can use your mobile phone to buy ferry tickets and songs, and feed parking meters, it's performing all the functions of an electronic purse, which if issued by a traditional payment service provider such as VISA or Mastercard would fall squarely under Reg E. Also, I believe that either the FCC or the Federal Reserve should have the right to ban known criminals from owning or managing regulated phone companies. Europe is even further behind, but there is some action in the regulatory pipeline: a draft Directive that will impose a Europe-wide duty on companies to disclose security breaches in telecomms to affected customers, similar to the breach notification laws in force in over 30 U.S states in 2007. This at least will be a start.

Research Problems

Relatively little research is done outside phone company and intelligence agency labs on issues related specifically to phone fraud and wiretapping. There is growing interest in traffic analysis, which I'll discuss later; and in the likely effects of next-generation value added services, which are bound to introduce new feature interactions and other vulnerabilities. The interaction between all communications (especially mobile), platform security, and the mechanisms used to protect distributed systems security, also looks like fertile ground for both interesting research and expensive engineering errors. Society

expects greater resilience and availability from the phone system than from the Internet — for example, to get through to emergency services — and as the two systems converge there will be some interesting assurance problems.

Further Reading

There are a lot of scattered articles about phone fraud, but nothing I know of that brings everything together. The underlying technologies are described in a number of reference books, such as [1061] on GSM, and more can be found on websites such as [1190]. An overview of UMTS can be found in [642], and a number of relevant research papers at [92]. To keep up with phone fraud, a useful resource is the Discount Long Distance Digest [390]. NIST has a guide on how to evaluate your PBX for vulnerabilities [943]. Finally, there's a survey of threats to mobile payment systems by the Mobile Payment Forum that gives a summary of the state of play as of 2002 [897].

Network Attack and Defense

Whoever thinks his problem can be solved using cryptography, doesn't understand his problem and doesn't understand cryptography.

— Attributed by Roger Needham and Butler Lampson to Each Other

*If you spend more on coffee than on IT security, then you will be hacked.
What's more, you deserve to be hacked.*

— Richard Clarke, Former U.S. Cybersecurity Tsar

21.1 Introduction

So far we've seen a large number of attacks against individual computers and other devices. But attacks increasingly depend on connectivity. Consider the following examples.

1. An office worker clicks on an attachment in email. This infects her PC with malware that compromises other machines in her office by snooping passwords that travel across the LAN.
2. The reason she clicked on the attachment is that the email came from her mother. The malware had infected her mother's machine and then sent out a copy of a recent email, with itself attached, to everyone in mum's address book.
3. Her mother in turn got infected by an old friend who chose a common password for his ISP account. When there are many machines on a network, the bad guys don't have to be choosy; rather than trying to guess the password for a particular account, they just try one password over and over for millions of accounts. Given a webmail account, they can send out bad email to the whole contact list.

4. Another attack technique that makes sense only in a network context is *Google hacking*. Here, the bad guys use search engines to find web servers that are running vulnerable applications.
5. The malware writers infect a whole lot of PCs more or less at random using a set of tricks like these. They then look for choice pickings, such as machines in companies from which large numbers of credit card numbers can be stolen, or web servers that can be used to host phishing web pages as well. These may be auctioned off to specialists to exploit. Finally they sell on the residual infected machines for under a dollar a time to a *botnet herder* — who operates a large network of compromised machines that he rents out to spammers, phishermen and extortionists.
6. One of the applications is *fast-flux*. This changes the IP address of a web site perhaps once every 20 minutes, so that it's much more difficult to take down. A different machine in the botnet acts as the host (or as a proxy to the real host) with each change of IP address, so blocking such an address has at most a temporary effect. Fast-flux hosting is used by the better phishing gangs for their bogus bank websites.

There are many attacks, and defenses, that emerge once we have large numbers of machines networked together. These depend on a number of factors, the most important of which are the protocols the network uses. A second set of factors relate to the *topology of the network*: is every machine able to contact every other machine, or does it only have direct access to a handful of others? In our example above, a virus spreads itself via a social network — from one friend to another, just like the flu virus.

I've touched on network aspects of attack and defense before, notably in the chapters on telecomms and electronic warfare. However in this chapter I'm going to try to draw together the network aspects of security in a coherent framework. First I'm going to discuss networking protocols, then malware; then defensive technologies, from filtering and intrusion detection to the widely-used crypto protocols TLS, SSH, IPsec and wireless LAN encryption. Finally I'll discuss network topology. The most immediate application of this bundle of technologies is the defence of networks of PCs against malware; however as other devices go online the lessons will apply there too. In addition, many network security techniques can be used for multiple purposes. If you invent a better firewall, then — like it or not — you've also invented a better machine for online censorship and a better police wiretap device as well. Conversely, if mobility and virtual private networks make life tough for the firewall designer, they can give the censor and the police wiretap department a hard time, too.

21.2 Vulnerabilities in Network Protocols

This book isn't an appropriate place to explain basic network protocols. The telegraphic summary is as follows. The *Internet Protocol* (IP) is a stateless protocol that transfers packet data from one machine to another; IP version 4 uses 32-bit *IP addresses*, often written as four decimal numbers in the range 0–255, such as 172.16.8.93. People have started to migrate to IP version 6, as the 4 billion possible IPv4 addresses will have been allocated sometime between 2010 and 2015; IPv6 uses 128-bit addresses. Most modern kit is ready to use IPv6 but the changeover, which companies will probably do one LAN at a time, will no doubt throw up some interesting problems. The *Domain Name System* (DNS) allows mnemonic names such as `www.ross-anderson.com` to be mapped to IP addresses of either kind; there's a hierarchy of DNS servers that do this, ranging from thirteen top-level servers down through machines at ISPs and on local networks, which cache DNS records for performance and reliability.

The core routing protocol of the Internet is the *Border Gateway Protocol* (BGP). The Internet consists of a large number of *Autonomous Systems* (ASs) such as ISPs, telcos and large companies, each of which controls a range of IP addresses. The routers — the specialized computers that ship packets on the Internet — use BGP to exchange information about what routes are available to get to particular blocks of IP addresses, and to maintain routing tables so they can select efficient routes.

Most Internet services use a protocol called *transmission control protocol* (TCP) that is layered on top of IP and provides virtual circuits. It does this by splitting up the data stream into IP packets and reassembling it at the far end, automatically retransmitting any packets whose receipt is not acknowledged. IP addresses are translated into the familiar Internet host addresses using the *domain name system* (DNS), a worldwide distributed service in which higher-level name servers point to local name servers for particular domains. Local networks mostly use ethernet, in which devices have unique ethernet addresses (also called MAC addresses) that are mapped to IP addresses using the *address resolution protocol* (ARP). Because of the growing shortage of IP addresses, most organisations and ISPs now use the *Dynamic Host Configuration Protocol* (DHCP) to allocate IP addresses to machines as needed and to ensure that each IP address is unique. So if you want to track down a machine that has done something wicked, you will often have to get the logs that map MAC addresses to IP addresses.

There are many other components in the protocol suite for managing communications and providing higher-level services. Most of them were developed in the good old days when the net had only trusted hosts and security wasn't a concern. So there is little authentication built in. This is

a particular problem with DNS and with BGP. For example, if a small ISP mistakenly advertises to a large neighbour that it has good routes to a large part of the Internet, it may be swamped by the traffic. Here at least there are sound economic incentives, in that ISPs either swap, or pay for, routes with their peers; BGP security is in effect bodged up using manual intervention. DNS is trickier, in that disruptions are often malicious rather than mistaken. A DNS server may be fed wrong information to drive clients to a wicked website. This can be done either wholesale, by an attack on the DNS servers of a large ISP, or at the local level. For example, many homes have a wireless router attached to a broadband connection, and the router contains the address of the DNS server the customer uses. In an attack called *Drive-By Pharming*, the villain lures you to view a web page containing javascript code that sets your router's DNS server to one under his control [1213]. The effect is that next time you try to go to www.citibank.com, you may be directed to a phishing site that emulates it. For this reason it's a really good idea to change the default password on your home router.

21.2.1 Attacks on Local Networks

Suppose the attacker controls one of your PCs. Perhaps one of your employees was careless; or maybe he's gone bad, and wants to take over an account in someone else's name to defraud you, or to do some other bad thing such as downloading child porn in the hope of framing someone. There are several possibilities open to him.

1. He can install packet sniffer software to harvest passwords, get the root password, and thus take over a suitable account. Password-sniffing attacks can be blocked if you use challenge-response password generators, or a protocol such as Kerberos or ssh to ensure that clear text passwords don't go over the LAN. I described Kerberos in Chapter 3, and I'll describe SSH later.
2. Another approach is to masquerade as a machine where the target user — say the sysadmin — has already logged on. It is often possible for the attacker simply to set his MAC address and IP address to those of the target. In theory, the target machine should send 'reset' packets when it sees traffic to its IP address that's not in response to its own packets; but many machines nowadays have personal firewalls, which throw away 'suspicious' packets. As a result, the alarm doesn't get raised [300].
3. There's a whole host of technical address-hijacking attacks that work fine against old-fashioned LANs. An example I gave in the first edition of my book was that the attacker gives wrong answers to ARP messages, claiming to be the target, and may stop the target machine noticing and

raising the alarm by sending it a false subnet mask. Another possibility is to send bogus DHCP messages. Attacks like this may or may not work against a modern switched ethernet, depending on how it's configured.

4. A further set of attacks target particular platforms. For example, if the target company uses Linux or Unix servers, they are likely to use Sun's *Network File System* (NFS) for file sharing. This allows workstations to use a network disk drive as if it were a local disk, and has a number of well-known vulnerabilities to attackers on the same LAN. When a volume is first mounted, the client gets a *root filehandle* from the server. This is in effect an access ticket that refers to the root directory of the mounted filesystem, but that doesn't depend on the time, or the server generation number, and can't be revoked. There is no mechanism for per-user authentication: the server must trust a client completely or not at all. Also, NFS servers often reply to requests from a different network interface to the one on which the request arrived. So it's possible to wait until an administrator is using a file server and then masquerade as him to overwrite the password file. Filehandles can also be intercepted by network sniffing, though again, switched ethernet makes this harder. Kerberos can be used to authenticate clients and servers, but many firms don't use it; getting it to work in a heterogeneous environment can be difficult.

So the ease with which a bad machine on your network can take over other machines depends on how tightly you have the network locked down, and the damage that a bad machine can do will depend on the size of the local network. There are limits to how far a sysadmin can go; your firm might need to run a complex mixture of legacy systems for which Kerberos just can't be got to work. Also, a security-conscious system administrator can impose real costs. At our lab we argued with our sysadmins for years, trying to get access to the Internet for visiting guests, while they resisted on both technical protection and policy grounds (our academic network shouldn't be made available to commercial users). In the end, we solved the problem by setting up a separate guest network that is connected to a commercial ISP rather than to the University's backbone.

This raises a wider problem: where's the network boundary? In the old days, many companies had a single internal network, connected to the Internet via a firewall of some kind. But compartmentation often makes sense, as I discussed in Chapter 9: separate networks for each department can limit the damage that a compromised machine can do. There may be particularly strong arguments for this if some of your departments may have high protection requirements, while others need great flexibility. In our university, for example, we don't want the students on the same LAN that the payroll folks use; in fact we separate student, staff and administrative networks, and the first two of these

are also separated by college and department. Recently, mobility and virtual networks have made definition of clear network boundaries even harder. This debate goes by the buzz-word of *deperimeterisation* and I'll return to it later.

One final attack is worth mentioning under the heading of attacks on local networks, and that's the *rogue access point*. Occasionally one finds WiFi access points in public areas, such as airports, that have been deployed maliciously. The operator might sit in the airport lounge with a laptop that accesses the Internet via a paid WiFi service and advertises a free one; if you use it, he'll be able to sniff any plaintext passwords you enter, for example to your webmail or Amazon account, and if you tried to do online banking he might conceivably send you to a malicious site. So the effects can be somewhat like drive-by pharming, although more reliable and less scalable. In addition, rogue access points may also be devices that employees have installed for their own convenience in defiance of corporate policy, or even official nodes that have been misconfigured so that they don't encrypt the traffic. Whether unencrypted WiFi traffic is a big deal will depend on the circumstances; I'll discuss this in more detail later when we come to encryption.

21.2.2 Attacks Using Internet Protocols and Mechanisms

Moving up now to the Internet protocol suite, the basic problem is similar: there is no real authenticity protection in the default mechanisms. This is particularly manifest at the lower level TCP/IP protocols, and has given rise to many attacks.

Consider for example the 3-way handshake used by Alice to initiate a TCP connection to Bob and set up sequence numbers.

This protocol can be exploited in a surprising number of different ways. Now that service denial is becoming really important, let's start off with the simplest service denial attack: the *SYN flood*.

21.2.2.1 SYN Flooding

The attack is quite simply to send a large number of SYN packets and never acknowledge any of the replies. This leads the recipient (Bob, in Figure 21.1) to accumulate more records of SYN packets than his software can handle. This attack had been known to be theoretically possible since the 1980s but came to public attention when it was used to bring down Panix, a New York ISP, for several days in 1996.

A technical fix has been incorporated in Linux and some other systems. This is the so-called 'SYNcookie'. Rather than keeping a copy of the incoming SYN packet, *B* simply sends out as *Y* an encrypted version of *X*. That way, it's

```

A → B: SYN; my number is X
B → A: ACK; now X+1
      SYN; my number is Y
A → B: ACK; now Y+1
      (start talking)

```

Figure 21.1: TCP/IP handshake

not necessary to retain state about sessions which are half-open. Despite this, SYN floods are still a big deal, accounting for the largest number of reported attacks (27%) in 2006, although they were only the third-largest in terms of traffic volume (18%, behind UDP floods and application-layer attacks) [86].

There is an important general principle here: when you're designing a protocol that anyone can invoke, don't make it easy for malicious users to make honest ones consume resources. Don't let anyone in the world force your software to allocate memory, or do a lot of computation. In the online world, that's just asking for trouble.

21.2.2.2 Smurfing

A common way of bringing down a host in the 90s was *smurfing*. This exploited the *Internet control message protocol* (ICMP), which enables users to send an echo packet to a remote host to check whether it's alive. The problem was with broadcast addresses that are shared by a number of hosts. Some implementations of the Internet protocols responded to pings to both the broadcast address as well as the local address — so you could test a LAN to see what was alive. A collection of such hosts at a broadcast address is called a *smurf amplifier*. Bad guys would construct a packet with the source address forged to be that of the victim, and send it to a number of smurf amplifiers. These would then send a flurry of packets to the target, which could swamp it. Smurfing was typically used by kids to take over an *Internet relay chat* (IRC) server, so they could assume control of the chatroom. For a while this was a big deal, and the protocol standards were changed in August 1999 so that ping packets sent to a broadcast address are no longer answered [1144]. Another part of the fix was socio-economic: vigilante sites produced lists of smurf amplifiers. Diligent administrators spotted their networks on there and fixed them; the lazy ones then found that the bad guys used more and more of their bandwidth, and thus got pressured into fixing the problem too. By now (2007), smurfing is more or less fixed; it's no longer an attack that many people use.

But there's a useful moral: *don't create amplifiers*. When you design a network protocol, be extremely careful to ensure that no-one who puts one packet in can get two packets out. It's also important to avoid feedback and loops. A classic example was *source routing*. A feature of early IP that enabled the sender

of a packet to specify not just its destination but the route that it should take. This made attacks too easy: you'd just send a packet from A to B to C to B to C and so on, before going to its final destination. Most ISPs now throw away all packets with source routing set. (There was an alarm in early 2007 when it turned out that source routing had found its way back into the specification for IPv6, but that's now been fixed [417].)

21.2.2.3 *Distributed Denial of Service Attacks*

As the clever ways of creating service-denial attacks have been closed off one by one, the bad guys have turned increasingly to brute force, for example by sending floods of UDP packets from infected machines. The *distributed denial of service* (DDoS) attack made its appearance in October 1999 with the attack already mentioned on a New York ISP, Panix. In DDoS, the attacker subverts a large number of machines over a period of time and, or on a given signal, these machines all start to bombard the target with traffic [391]. Curiously, most of the machines in the first botnets around 1999–2000 were U.S. medical sites. The FDA insisted that medical Unix machines which were certified for certain clinical uses have a known configuration. Once bugs were found in this, there was a guaranteed supply of vulnerable machines; an object lesson in the dangers of monoculture.

Nowadays, botnets are assembled using all sorts of vulnerabilities, and a market has arisen whereby people who specialise in hacking machines can sell their product to people who specialise in herding them and extracting value. Compromised machines typically pass down a kind of value chain; they are first used for targeted attacks, then for sending spam, then (once they get known to spam filters) for applications like fast flux, and then finally (once they're on all the blacklists) for DDoS.

DDoS attacks have been launched at a number of high-profile web sites, including Amazon and Yahoo, but nowadays the major sites have so much bandwidth that they're very hard to dent. The next development was extortionists taking out online horserace-betting sites just before popular race meetings that would have generated a lot of business, and demanding ransoms not to do it again. Some bookmakers moved their operations to high-bandwidth hosting services such as Akamai that are highly distributed and can cope with large packet volumes, and others to specialist ISPs with packet-washing equipment that filters out bad packets at high speed. However the real fix for extortion wasn't technical. First, the bookmakers got together, compared notes, and resolved that in future none of them would pay any ransom. Second, the Russian government was leant on to deal with the main gang; three men were arrested in 2004 and sent to prison for eight years in 2006 [791].

For a while, there was a technical arms race. Attackers started to spoof source IP addresses, and to reflecting packets off innocuous hosts [1011]. One

countermeasure was traceback: the idea was that whenever a router forwards a packet, it would also send an ICMP packet to the destination, with a probability of about 1 in 20,000, containing details of the previous hop, the next hop, and as much of the packet as will fit. Large-scale flooding attacks could thus be traced back to the responsible machines, even despite forged source IP addresses [154]. However, this arms race has largely fizzled out, and for a number of reasons. First, Microsoft changed their network stack to make it much harder for an infected machine to send a packet with a spoofed IP address; you now need to hack the operating system, not just any old application. Second, more and more equipment at the edges of the network won't accept spoofed packets, and now about half of broadband ISPs filter them out [86].

However, the main reasons for the arms race stopping had to do with economics. Tracing back packets didn't work across different autonomous systems; if you were a large telco, for example, you would not give network access to your competitors' technical staff. So the bad guys found that in practice nobody came after them, and stopped using spoofing. Also, once markets emerged round about 2004 for botnet machines to be bought and sold (along with credit card numbers, spam contracts and other criminal services), the price of compromised machines fell so low, and botnets started to become so large, that the whole game changed. Instead of using a handful of compromised machines to send out clever attacks via amplifiers using spoofed source addresses, the bad guys simply burn thousands of end-of-life botnet machines to send the bad packets directly. The rapier has been replaced with the Kalashnikov.

Most recently, in 2005–7, there have been attempts to target core services such as DNS and thus take down the whole Internet. DNS has now been expanded to thirteen servers (the maximum the protocol will support), and many of them use *anycast* — a protocol whereby when you ask to resolve the domain name of a host into an IP address, the result that you get depends on where you are. The net effect is that DNS has become a massively distributed system using a lot of very fast machines connected to very high-capacity networks. In the end, the brute force of the modern DDoS attack was simply answered by even more brute force. If a hundred peasants with Kalashnikovs are going to shoot at you, you'd better buy a tank with good enough armor to absorb the fire.

Large-scale DDoS attacks on critical services seem quiescent at present. There are a few residual worries, though.

- There's a rising tide of DDoS attacks that happen by accident rather than as a result of malice. For example, in 2003 the University of Wisconsin-Madison found itself receiving hundreds of thousands of packets per second requesting the time. It turned out that Netgear had sold some 700,000 routers that were hard-coded to ask their time server what time

it was, and to ask again a second later if no response was received. Netgear ended up paying the university to maintain a high-bandwidth time server for them. There have been dozens of similar incidents [303].

- There's a steady stream of DDoS attacks by spammers and phishermen on the websites of organisations that try to hinder their activities, such as Artists Against 419 and Spamhaus.
- There are continuing worries that DDoS attacks might come back on an even larger scale. As of September 2007, there are several botnets with over half a million machines [742]. The operators of such networks can send out packet floods that will disable all but the biggest sites. These worries have been amplified in some quarters by the 2007 attacks on Estonia (even although that attack would not have harmed a large commercial target like Microsoft or Google). The highest attack rate seen in 2006 was 24 Gbit/sec, compared with 10 Gbit/sec in 2004 and 1.2 Gbit/sec in 2002 [86]. A further order-of-magnitude increase could put all but the most distributed targets at risk.
- Even if we never see the Internet taken down by a monster botnet, attacks can still be carried out on smaller targets. Prior to the Estonia incident, there had been a DDoS attack on the servers of an opposition party in Kyrgyzstan, and these followed the site when it was relocated to North America [1081]. Certainly, DDoS puts a weapon in the hands of gangsters that can be rented out to various unsavoury people.
- That said, one mustn't forget online activism. If a hundred thousand people send email to the White House protesting against some policy or other, is this a DDoS attack? Protesters should not be treated as felons; but drawing legislative distinctions can be hard.

21.2.2.4 Spam

Spam is in some respects similar to a DDoS attack: floods of generally unwanted traffic sent out for the most part by botnets, and often with clear criminal intent. The technical aspects are related, in that both email and the web protocols (`smtp` and `http`) assume wrongly that the lower levels are secure. Just as DDoS bots may forge IP addresses, spam bots may forge the sender's email address.

Spam differs in a various ways, though, from packet-level DDoS. First, it's enough of an annoyance and a cost for there to be real pressure on ISPs to send less of it. If you're a medium-sized ISP you will typically peer with other ISPs as much as you can, and buy routes from large telcos only where you can't trade for them; if other ISPs see you as a source of spam they may drop your peering arrangements, which costs real money. As a result, some ISPs are starting to do *egress filtering*: they monitor spam coming out of their own networks and then quarantine the infected PCs into a 'walled garden' from which they have access to antivirus software but little else [300].

Second, unlike DDoS, spam does not seem to be tailing off. It does appear that spammers are consolidating, in that most spam comes from several dozen large gangs. This is apparent from the 'lumpiness' of spam statistics: if there were hundreds of thousands of mom-and-pop spam operations, you'd expect to see spam volumes pretty constant, but this is no longer what we see [305]. So rather than spending more money on spam filters, it might be cheaper to get the police to arrest the gangs. Trends do change over time, though. Between 2006 and 2007, we've seen a drop in *backscatter* — in messages sent to the people whose email addresses were forged by spammers. Quite a lot of this came from anti-virus products, and it was pointed out that the vendors were often breaking the antispam laws by sending messages saying 'Product X found worm Y in the message you sent'. If worm Y was known to use address forgery, the only conceivable purpose of sending such a message to the party who hadn't sent the offending message was to advertise product X. At the same time, there's been a huge increase in mule recruitment spam.

21.2.2.5 DNS Security and Pharming

I've given two examples so far of attacks in which a user is duped by being directed to a malicious DNS server, with the result that when he tries to go to his bank website he ends up entering his password into a fake one instead. This is generally referred to as *pharming*. I mentioned drive-by pharming, in which people's home routers are reconfigured by malicious javascript in web pages they download, and rogue access points in which the attacker offers a WiFi service to the victim and then has complete control over all his unprotected traffic.

There are a number of other variants on this theme, including feeding false DNS records to genuine servers. Older DNS servers would accept additional records without checking; if they asked your server where X was, you could volunteer an IP address for Y as well. This has been fixed but there are still older servers in use that are vulnerable. Such attacks are often referred to as *DNS cache poisoning* as they basically affect users who trust the information about the target that's cached by the attacked machine. They've been used not just for pharming but also for simple vandalism, such as replacing the web site of a target company with something offensive. They can also be used for censorship; China has used DNS spoofing against dissident websites for years, and by 2007 was also using it to make Voice of America news unavailable [1297].

A number of researchers have worked on a proposed upgrade to the security of DNS, but they have turned out to be hard to deploy for economic reasons; most of the things that secure DNS would do can be done by TLS without the need for new infrastructure, and individual network operators don't get enough benefit from DNS security until enough other operators have adopted them first [997].

21.3 Trojans, Viruses, Worms and Rootkits

Computer security experts have long been aware of the threat from malicious code. The first such programs were *Trojan Horses*, named after the horse the Greeks left supposedly as a gift for the Trojans but which contained soldiers who opened the gates of Troy to the Greek army. The use of the term for malicious code goes back many years (see the discussion in [774] p 7). There are also *viruses* and *worms*, which are self-propagating malicious programs, and to which I've referred in earlier chapters. There is debate about their precise definitions; the common usage is that a Trojan is a program that does something malicious (such as capturing passwords) when run by an unsuspecting user, while a worm is something that replicates and a virus is a worm which replicates by attaching itself to other programs.

Finally, the most rapidly-growing problem is the *rootkit* — a piece of software that once installed on a machine surreptitiously places it under remote control. Rootkits can be used for targeted attacks (law enforcement agencies use them to turn suspects' laptops into listening devices) or for financial fraud (they may come with keyloggers that capture passwords). One of the most salient features of rootkits nowadays is stealth; they try to hide from the operating system so that they can't be located and removed using standard tools. But sooner or later rootkits are identified and tools to remove them are written. On the other side of the coin, most PCs infected in this way end up in botnets, so another way of framing the problem is how botnets are set up, maintained and used. The rootkit vendors now do after-sales-service, supplying their customers the botnet herders with the tools to upgrade rootkits for which removal tools are becoming available.

21.3.1 Early History of Malicious Code

Malicious code, or *malware*, seems likely to appear whenever a large enough number of users share a computing platform. It goes back at least to the early 1960's, when machines were slow and their CPU cycles were carefully rationed between different groups of users — with students often at the tail of the queue. Students invented tricks such as writing computer games with a Trojan inside to check if the program is running as root, and if so to create an extra privileged account with a known password. By the 1970s, large time-sharing systems at universities were the target of more and more pranks involving Trojans. All sorts of tricks were developed. In 1978, John Shoch and Jon Hupp of Xerox PARC wrote a program they called a *worm*, which replicated itself across a network looking for idle processors so it could assign them tasks. They discussed this in a paper in 1982 [1164].

In 1984, Ken Thompson wrote a classic paper 'On Trusting Trust', in which he showed that even if the source code for a system were carefully

inspected and known to be free of vulnerabilities, a trapdoor could still be inserted [1247]. Thompson's trick was to build the trapdoor into the compiler. If this recognized that it was compiling the login program, it would insert a trapdoor such as a master password that would work on any account. (This developed an idea first floated by Paul Karger and Robert Schell during the Multics evaluation in 1974 [693].) Of course, someone might try to stop this by examining the source code for the compiler, and then compiling it again from scratch. So the next step is to see to it that, if the compiler recognizes that it's compiling itself, it inserts the vulnerability even if it's not present in the source. So even if you can buy a system with verifiably secure software for the operating system, applications and tools, the compiler binary can still contain a Trojan. The moral is that vulnerabilities can be inserted at any point in the tool chain, so you can't trust a system you didn't build completely yourself.

1984 was also the year when computer viruses appeared in public following the thesis work of Fred Cohen. He performed a series of experiments with different operating systems in which he showed how code could propagate itself from one machine to another, and (as I mentioned in Chapter 8) from one compartment of a multilevel system to another. This caused alarm and consternation, and within about three years we started to see the first real live viruses in the wild¹. Almost all of them were PC viruses as DOS was the predominant operating system. They spread from one user to another when users shared programs on diskettes or via bulletin boards.

One early innovation was the 'Christma' virus, which spread round IBM mainframes in December 1987. It was a program written in the mainframe command language REXX that had a header saying 'Don't read me, EXEC me' and code that, if executed, drew a Christmas tree on the screen — then sent itself to everyone in the user's contacts file. It was written as a prank, rather than out of malice; and by using the network (IBM's BITNET) to spread, it was ahead of its time.

The next year came the Internet worm, which alerted the press and the general public to the problem.

21.3.2 The Internet Worm

The first famous case of a service denial-attack was the Internet worm of November 1988 [421]. This was a program written by Robert Morris Jr that exploited a number of vulnerabilities to spread from one machine to another. Some of these were general (e.g. 432 common passwords were used in a guessing attack, and opportunistic use was made of `.rhosts` files), and others

¹That's when I first came across them, as a security guy working in a bank; we now learn that the first ever computer virus in the wild was written for the Apple II by a 9th-grader in 1981 [1101].

were system specific (problems with `sendmail`, and the `fingerd` bug mentioned in section 4.4.1). The worm took steps to camouflage itself; it was called `sh` and it encrypted its data strings (albeit with a Caesar cipher).

Its author claimed that this code was not a deliberate attack on the Internet — merely an experiment to see whether code could replicate from one machine to another. It was successful. It also had a bug. It should have recognised already infected machines, and not infected them again, but this feature didn't work. The result was a huge volume of communications traffic that completely clogged up the Internet.

Given that the Internet (or more accurately, its predecessor the Arpanet) had been designed to provide a very high degree of resilience against attacks — up to and including a strategic nuclear strike — it was remarkable that a program written by a student could disable it completely.

What's less often remarked on is that the mess was cleaned up and normal service restored within a day or two; that it only affected Berkeley Unix and its derivatives (which may say something about the dangers of the Microsoft monoculture today); and that sites that kept their nerve and didn't pull their network connection recovered more quickly as they could find out what was happening and get the fixes.

21.3.3 How Viruses and Worms Work

A virus or worm typically has two components — a replication mechanism and a payload. A worm simply makes a copy of itself somewhere else when it's run, perhaps by breaking into another system (as the Internet worm did) or mailing itself as an attachment to the addresses on the infected system's address list (as many recent worms have done). In the days of DOS viruses, the commonest way for a virus to replicate was to append itself to an executable file and patch itself in, so that the execution path jumps to the virus code and then back to the original program.

Given a specific platform, there are usually additional tricks available to the virus writer. For example, if the target system was a DOS PC with a file called `ACCOUNTS.EXE`, one could introduce a file called `ACCOUNTS.COM`, which DOS will execute in preference. DOS viruses could also attack the boot sector or the partition table, and there are even printable viruses — viruses all of whose opcodes are printable ASCII characters, so that they can even propagate on paper. A number of DOS viruses are examined in detail in [817].

The second component of a virus is the payload. This will usually be activated by a trigger, such as a date, and may then do one or more of a number of bad things:

- make selective or random changes to the machine's protection state (this is what we worried about with multilevel secure systems);

- make changes to user data (some early viruses would trash your hard disk while some recent ones encrypt your disk and ask you to pay a ransom for the decryption key);
- lock the network (e.g., start replicating at maximum speed);
- perform some nefarious task (e.g. use the CPU for DES keysearch);
- get your modem to phone a premium-rate number in order to transfer money from you to a telephone scamster;
- install spyware or adware in your machine. This might just tell marketers what you do online — but it might steal your bank passwords and extract money from your account;
- install a *rootkit* — software that hides in your machine having taken it over. This is typically used to recruit your machine into a botnet, so that it can be used later for spam, phishing and distributed denial of service attacks at the botnet herder's pleasure.

The history of malware, and of countermeasures, has some interesting twists and turns.

21.3.4 The History of Malware

By the late 1980s and early 1990s, PC viruses had become such a problem that they gave rise to a whole industry of anti-virus software writers and consultants. Many people thought that this couldn't last, and that the move from DOS to 'proper' operating systems such as Windows would solve the problem. Some of the anti-virus pioneers even sold their companies; one of them tells his story in [1198].

However, the move to 32-bit operating systems gave only temporary respite. Soon, the spread of interpreted languages provided fertile soil for mischief. Bad Java applets flourished in the late 1990s as people found ways of penetrating Java implementations in browsers [859]. By the start of the 21st century, the main vector was the macro languages in products such as Word, and the main transmission mechanism had become the Internet [95, 209]; by 2000, macro viruses accounted for almost all incidents of mobile malicious code. Indeed, an insider says that the net 'saved' the antivirus industry [669]. A more cynical view is that the industry was never really under threat, as people will always want to share code and data, and in the absence of trustworthy computing platforms one can expect malware to exploit whichever sharing mechanisms they use. Another view is that Microsoft is responsible as they were reckless in incorporating such powerful scripting capabilities in all sorts of products. As they say, your mileage may vary.

In passing, it's worth noting that malicious data can also be a problem. An interesting example is related by David Mazières and Frans Kaashoek who

operated an anonymous remailer at MIT. This device decrypted incoming messages from anywhere on the net, uncompressed them and acted on them. Someone sent them a series of 25 Mbyte messages consisting of a single line of text repeated over and over; these compressed very well and so were only small ciphertexts when input, but when uncompressed they quickly filled up the spool file and crashed the system [849]. There are similar attacks on other programs that do decompression such as MPEG decoders. However, the most egregious cases involve not malicious data but malicious code.

Anyway, the next phase of malware evolution may have been the 'Love Bug' virus in 2000. This was actually a self-propagating worm; it propagated by sending itself to everyone in the victim's address book, and the subject line 'I love you' was calculated to get people to open it. In theory, companies can defend themselves against such things by filtering out Microsoft executables; in practice, life isn't so simple. A large Canadian company with 85,000 staff did just this, but many of their staff had personal accounts at web-based email services, and so the Love Bug virus got into the company without going through the mail filter at the firewall. The company had configured its employees' mail clients so that each of them had the entire corporate directory in her personal address book. The result was meltdown as 85,000 mail clients each tried to send an email to each of 85,000 addresses. The Love Bug was followed by a number of similar worms, which persuaded people to click on them by offering pictures of celebs such as Anna Kournikova, Britney Spears and Paris Hilton. There were also 'flash worms' that propagated by scanning the whole Internet for machines that were vulnerable to some exploit or other, and taking them over; worms of this type, such as Code Red and Slammer, infected all vulnerable machines within hours or even minutes, and caused some alarm about what sort of defences might possibly react in time [1220].

At about the same time, in the early 2000s, we saw a significant rise in the amount of *spyware* and *adware*. Spyware is technology that collects and forwards information about computer use without the owner's authorization, or with at best a popup box that asks users to agree to perform some obscure function, so that even those who don't just reflexively click it away will not really know what they're agreeing to. This doesn't pass muster as 'consent' under European data-protection and unfair-contracts laws, but enforcement is weak. Adware may bombard the user with advertising popups and can be bundled with spyware. The vendors of this tiresome crud have even sued antivirus companies who blacklisted their wares. This all complicates everything.

A large change came about in 2004 or so. Until then, we saw a huge range of different viruses and payloads. Most virus writers did so for fun, for bragging rights, to impress their girlfriends — basically, they were amateurs. Since then, the emergence of an organised criminal economy in information goods has made the whole business much more professional. The goal of the malware

writers is to recruit machines that can be sold on for cash to botnet herders and for other exploits.

Most viruses in the 1980s and 1990s were very flaky; although tens of thousands of different viruses were reported, very few actually spread in the wild. It's actually rather difficult to write a virus that spreads properly; if it's not infectious enough it won't spread, while if you make it too infectious then it gets noticed quickly and dealt with. However, those viruses that did spread often spread very widely and infected millions of machines.

A widespread self-replicating worm may bring a huge ego boost to a teenage programmer, but for the Mafia it's not optimal. Such a worm becomes headline news and within a few hours the world's anti-virus vendors are upgrading their products to detect and remove it. Even the mass media get in on the act, telling the public not to click on any link in such-and-such an email. Now that the writers are focussed on money rather than bragging rights, they release more attacks but limited ones. Furthermore, rather than using self-replicating worms — which attract attention by clogging up the Internet — the modern trend is towards manually-controlled exploit campaigns.

In September 2007 the largest botnet was perhaps the Storm network, with around a million machines. Its herders are constantly recruiting more machines to it, using one theme after another. For example, following the start of the National Football League season on September 6th, they sent out spam on September 9th saying simply 'Football ... Need we say more? Know all the games, what time, what channel and all the stats. Never be in the dark again with this online game tracker', following by a link to a URL from which the gullible download a file called tracker.exe that installs a rootkit in their machine. Using techniques like this — essentially, professional online marketing — they constantly grow their network. And although the media refer to Storm as a 'worm', it isn't really: it's a Trojan and a rootkit. Victims have to click away several warnings before they install it; Windows warns them that it isn't signed and asks them if they really want to install it. However, Windows pops up so many annoying dialog boxes that most people are well trained to click them away. In the case of Storm, it was targeted by Microsoft's malicious software removal tool on September 11th, and Redmond reported that over a quarter of a million machines had been cleaned; they also estimated that Storm had half a million active machines, with perhaps a few hundred thousand that were not being actively used. The network — the most powerful supercomputer on the planet — earned its living by being rented out to pump-and-dump operators and pharmacy scammers [742]. Two other networks were also identified as having over half a million bots; Gozi and Nugache use the same peer-to-peer architecture as Storm, and by the end of 2007 these networks were getting increasingly sophisticated and exploring new criminal business models [1134].

So the malware business now operates on an industrial scale, with the top botnet herders controlling roughly the same number of machines as Google. Big business has been built on the fact that users have been trained to click on stuff. As malware goes industrial, Trojans are becoming more common than viruses; when the latter email themselves out from an infected machine, they draw attention to themselves and the machine's more likely to get cleaned up, while with Trojans the botnet herder sends the infectious traffic directly, which also given him better control [1239]. And once you install something, there's no telling whether it's a rootkit, or malicious spyware that will use a keystroke logger to steal your banking passwords, or a 'normal' piece of spyware that will simply collect your personal data for sale to the highest bidder. Truth to tell, the categories are hard to separate cleanly.

21.3.5 Countermeasures

Within a few months of the first PC viruses appearing in the wild in 1987, companies had set up to sell antivirus software. This led to an arms race in which each tried to outwit the other. Early software came in basically two flavours — *scanners* and *checksummers*.

Scanners are programs that search executable files for a string of bytes known to be from an identified virus. Virus writers responded in various ways, such as specific counterattacks on popular antivirus programs; the most general technique is *polymorphism*. The idea here is to change the code each time the virus or worm replicates, to make it harder to write effective scanners. The usual technique is to encrypt the code using a simple cipher, and have a small header that contains decryption code. With each replication, the virus re-encrypts itself under a different key, and tweaks the decryption code by substituting equivalent sequences of instructions.

Checksummers keep a list of all the authorised executables on the system, together with checksums of the original versions, typically computed using a hash function. The main countermeasure is *stealth*, which in this context means that the virus watches out for operating system calls of the kind used by the checksummer and hides itself whenever a check is being done.

Researchers have also looked into the theory of malware replication. In order for a virus infestation to be self-sustaining, it needs to pass an *epidemic threshold* — at which its rate of replication exceeds the rate at which it's removed [711]. This depends not just on the infectivity of the virus itself but on the number (and proportion) of connected machines that are vulnerable. Epidemic models from medicine go over to some extent, though they are limited by the different topology of software intercourse (sharing of software is highly localised) and so predict higher infection rates than are actually observed. (I'll return to topology later.) People have also tried to use immune-system models to develop distributed strategies for malware detection [482].

One medical lesson which does seem to apply is that the most effective organisational countermeasure is centralised reporting and response using selective vaccination [712].

In the practical world, antivirus software and managerial discipline are to a certain extent substitutes, but to be really effective, you have to combine tools, incentives and management. In the old days of DOS-based file viruses, this came down to providing a central reporting point for all incidents, and controlling all software loaded on the organisation's machines. The main risks were files coming in via PCs used at home both for work and for other things (such as kids playing games), and files coming in from other organisations. But how do you get staff to sweep all incoming email and diskettes for viruses? One effective strategy, adopted at a London law firm, was to reward whoever found a virus with a box of chocolates — which would then be invoiced to the company that had sent the infected file.

Now that malware arrives mostly in email attachments or in web pages, things are often more technical, with automatic screening and central reporting. A company may filter executables out at the firewall, and see to it that users have prudent default settings on their systems — such as disabling active content on browsers and macros in word processing documents. Of course, this creates a clash with usability. People will also create all sorts of unauthorized communications channels, so you have to assume that screening can't be perfect; staff must still be trained not to open suspicious email attachments, and in recovery procedures so they can deal with infected backups. In short, the issues are more complex and diffuse. But as with the organic kind of disease, prevention is better than cure; and software hygiene can be integrated with controls on illegal software copying and unauthorised private use of equipment.

Recently, antivirus software seems to be getting steadily less effective. The commercialisation of botnets and of machine exploitation has meant that malware writers have decent tools and training. Almost all Trojans and other exploits are undetectable by the current antivirus products when first launched — as their writers test them properly — and many of them run their course (by recruiting their target number of machines) without coming to the attention of the antivirus industry. The net effect is that while antivirus software might have detected almost all of the exploits in circulation in the early 2000s, by 2007 the typical product might detect only a third of them.

And as for the rootkits that the exploits leave behind, they are also much better written than a few years ago, and rarely cause trouble for the owner of the machine on which they're installed. Some rootkits even install up-to-date antivirus software to stop any competing botnet from taking the machine over. They also use all sorts of stealth techniques to hide from detectors. What's more, the specialists who sell the rootkits provide after-sales service; if a removal kit is shipped, the rootkit vendor will rapidly ship countermeasures.

It's not at all clear that technical defences are keeping up with malware. On the global scale, police action against the large gangs is needed, and although it's starting to ramp up, there's a long way to go. Well-run firms can use managerial discipline to contain the threat, but for private users of Windows machines, the outlook isn't particularly rosy. One survey suggested that 8% of sales of new PCs are to people who've simply given up on machines that have become so infested with adware and other crud as to become unusable [325]; and there is a growing threat from *keyloggers* that capture everything the user does at his machine. Some of these are simply spyware that sells information to marketers; others look out for bank passwords and other key data that can be used to commit fraud directly.

21.4 Defense Against Network Attack

In defending against network attack, there are broadly speaking four sets of available tools.

1. First is management — keeping your systems up-to-date and configured in ways that will minimise the attack surface;
2. Next is filtering — the use of firewalls to stop bad things like Trojans and network exploits, and to detect signs of attack and compromise if anything gets through;
3. Next is intrusion detection — having programs monitoring your networks and machines for signs of malicious behaviour;
4. Finally there's encryption — protocols such as TLS and SSH that enable you to protect specific parts of the network against particular attacks.

Let's work through these in turn.

21.4.1 Configuration Management and Operational Security

The great majority of technical attacks on systems in the period 2000–07 exploited already known vulnerabilities. The typical cycle is that Microsoft announces a set of security patches once a month; as soon as they come out, the attackers start reverse engineering them; within a few days, the vulnerabilities that they fixed are understood and exploits appear. A well-run firm will test its operational systems quickly on Patch Tuesday and apply the patches, provided they don't break anything important. If they do break something, that will be fixed as quickly as reasonably possible.

Tight configuration management is not just about patches, though. Many software products ship with unsafe defaults, such as well-known default

passwords. It's a good idea to have someone whose job it is to understand and deal with such problems. It's also common to remove unnecessary services from machines; there is usually no reason for every workstation in your company to be running a mail server, and ftp server and DNS, and stripping things down can greatly reduce the attack surface. Frequent reinstallation is another powerful tool: when this was first tried at MIT during Project Athena, a policy of overnight reinstallation of all software greatly cut the number of sysadmins needed to look after student machines. Operations like call centres often do the same; that way if anyone wants to install unauthorised software they have to do it again every shift, and are more likely to get caught. There are also network configuration issues: you want to know your network's topology, and have some means of hunting down things like rogue access points. If all this is done competently, then you can deal with most of the common technical attacks. (You'll need separate procedures to deal with bugs that arise in your own code, but as most software is bought rather than written these days, configuration management is most of the battle.)

There are many tools to help the sysadmin in these tasks. Some enable you to do centralized version control so that patches can be applied overnight and everything kept in synch; others look for vulnerabilities in your network. When the first such tool came out (Satan [503]) there was quite a lot of controversy; this has led to some countries passing laws against 'hacking tools'. Now there are dozens of such tools, but they have to be used with care.

However, a strategy of having your system administrators stop all vulnerabilities at source is harder than it looks; even diligent organisations may find it's just too expensive to fix all the security holes at once. Patches may break critical applications, and it seems to be a general rule that an organisation's most critical systems run on the least secure machines, as administrators have not dared to apply upgrades and patches for fear of losing service.

This leads us to operational security, and the use of filtering tools such as firewalls.

Operational security, as mentioned in Chapter 2 and Chapter 8, is about training staff to not expose systems by foolish actions. There we were largely interested in social engineering attacks involving the telephone; the main way of getting unauthorised access to information is still to phone up and pretend to be someone who's entitled to know.

Now the main way machines get compromised in 2007 is because people click on links in email that cause them to download and install rootkits. Of course you must train your staff to not click on links in mail, but don't expect that this alone will fix the problem; many banks and other businesses expect their customers to click on links, and many of your staff will have to do some clicking to get their work done. You can shield low-grade staff by not giving them administrator access to their machines, and you can shield your creative

staff by letting them buy Macs; but there is still going to be a residual risk. One common way of dealing with it is to strip out all executables at your firewall.

21.4.2 Filtering: Firewalls, Spam Filters, Censorware and Wiretaps

The most widely sold solution to the 'problems of Internet security' is the *firewall*. This is a machine which stands between a local system and the Internet and filters out traffic that might be harmful. The idea of a 'solution in a box' has great appeal to many organisations, and is now so widely accepted that it's seen as an essential part of corporate due diligence. (This in itself creates a risk — many firms prefer expensive firewalls to good ones.)

Firewalls are just one example of systems that examine streams of packets and perform filtering operations. Bad packets may be thrown away, or modified in such a way as to make them harmless. They may also be copied to a log or audit trail. Very similar systems are also used for Internet censorship and for law-enforcement wiretapping; almost everything I'll discuss in this section goes across to those applications too. Developments in any of these fields potentially affect the others; and actual systems may have overlapping functions. For example, many corporate firewalls or mail filters screen out pornography, and some even block bad language, while ISP systems that censor child pornography or dissenting political speech may report the perpetrators automatically to the authorities.

Filters come in basically three flavours, depending on whether they operate at the IP packet level, at the TCP session level or at the application level.

21.4.2.1 Packet Filtering

The simplest kind of filter merely inspects packet addresses and port numbers. This functionality is also available in routers, in Linux and indeed in Windows. A firewall can block IP spoofing by ensuring that only 'local' packets leave a network, and only 'foreign' ones enter. It can also stop denial-of-service attacks in which malformed packets are sent to a host. It's also easy to block traffic to or from 'known bad' IP addresses. For example, IP filtering is a major component of the censorship mechanisms in the Great Firewall of China; a list of bad IP addresses can be kept in router hardware, which enables packet filtering to be done at great speed.

Basic packet filtering is also available as standard on most machines and can be used for more mundane firewalling tasks. For example, packet filters can be configured to block all traffic except that arriving on specific port numbers. The configuration might be initially to allow the ports used by common services such as email and web traffic, and then open up ports as the protected machine or subnet uses them.

However, packet filters can be defeated by a number of tricks. For example, a packet can be fragmented in such a way that the initial fragment (which passes the firewall's inspection) is overwritten by a subsequent fragment, thereby replacing the source address with one that violates the firewall's security policy. Another limitation is that maintaining a blacklist is difficult, and especially so when it's not the IP address specifically you want to block, but something that resolves into an IP address, especially on a transient basis. For example, the phishermen are starting to use tricks like fast-flux in which a site's IP address changes several times an hour.

21.4.2.2 *Circuit Gateways*

The next step up is a more complex arrangement, a *circuit gateway*, that operates at level 4, typically by reassembling and examining all the packets in each TCP session. This is more expensive than simple packet filtering; its main advantage is that it can also provide the added functionality of a *virtual private network* whereby corporate traffic passed over the Internet is encrypted from firewall to firewall. I'll discuss the IPSEC protocol that's used for this in the last section of this chapter.

TCP-level filtering can be used to do a few more things, such as DNS filtering. However, it can't screen out bad things at the application level, such as malicious code, image spam and unlawful images of child abuse. Thus it may often be programmed to direct certain types of traffic to specific application filters. An example is British Telecom's CleanFeed system, which tries to prevent its customers getting access to child pornography. As some bad sites are hosted on public web services and blocking all the web pages at the service would be excessive, TCP/IP filtering is used to redirect traffic with such sites to a proxy that can examine it in detail.

21.4.2.3 *Application Relays*

The third type of firewall is the *application relay*, which acts as a proxy for one or more services. Examples are mail filters that try to weed out spam, and web proxies that block or remove undesirable content. The classic example is a corporate rule about stripping out code, be it straightforward executables, active content in web pages, macros from incoming Word documents. Over the period 2000–07, this has been a constant arms race between the firewall vendors, the spammers, and people trying to circumvent controls to get their work done.

The flood of Word macro viruses around 2000 led many firms to strip out Word macros (or even all Word documents) from email. Workers got round this by zipping documents first. Firewalls started unzipping them to inspect them, whereupon people started encrypting them using zip's password feature, and

putting the password in the email plaintext. Once firewalls started to cope with this, the spammers started putting zip passwords in images attached to the mail along with the zip file. Eventually, many companies started adopting a policy of not sending out Word documents, but Pdf documents instead; this not only made it easier to get past firewalls, but also stopped people carelessly sending out documents containing the last few dozen edits. Needless to say, the spammers now send out Pdf attachments — and their botnets have the power to make all the attachments different, for example by combining text, and image, and a number of random color blocks for background. Rootkit executables are now often distributed as web links; August 2007 saw floods of messages telling people they'd got a card, while in September it was links to a bogus NFL site. For complete protection, you have to filter executables in your web proxy too (but this would really get in the way of users who wish to run the latest applications). There is no sign of this arms race abating.

An application relay can also turn out to be a serious bottleneck. This applies not just to the corporate application, but in censorship. An example is the Great Firewall of China, which tries to block mail and web content that refers to banned subjects. Although the firewall can block 'known bad' sites by simple IP filtering, finding forbidden words involves deep packet inspection — which needs much more horsepower. An investigation by Richard Clayton, Steven Murdoch and Robert Watson showed bad content wasn't in fact blocked; machines in China simply sent 'reset' packets to both ends of a connection on which a bad word had appeared. This was almost certainly because they needed a number of extra machines for the filtering, rather than doing it in the router; one side-effect was that you could defeat the firewall by ignoring these reset packets [308]. (Of course, someone within China who did that might eventually get a visit from the authorities.)

At the application level in particular, the pace of innovation leaves the firewall vendors (and the censors and the wiretappers) trailing behind. A good example is the move to edge-based computing. Google's word processor — Google Documents — is used by many people to edit documents online, simply to save them the cost of buying Microsoft Word. As a side-effect, its users can instantly share documents with each other, creating a new communications channel of which classical filters are unaware. So the service might be used to smuggle confidential documents out of a company, to defeat political censors, or to communicate covertly. (It even blurs the distinction between traffic and content, which is central to the legal regulation of wiretapping in most countries.) Even more esoteric communications channels are readily available — conspirators could join an online multi-user game, and pass their messages via the silver dragon in the sixth dungeon.

Another problem is that application-level filtering can be very expensive, especially of high-bandwidth web content. That's why a number of web filtering systems are hybrids, such as the CleanFeed mechanism mentioned above where only those domains that contain at least some objectionable

content are sent to full http proxy filtering. Application proxies can also interact with other protection mechanisms. Not only can spammers (and others) use encryption to defeat content inspection; but some corporate web proxies are set up to break encryption by doing middleperson attacks on TLS. Even if you think you're giving an encrypted credit card number to Amazon, your encrypted session may just be with your employer's web proxy, while it runs another encrypted session with Amazon's web server. I'll discuss TLS later in this chapter.

21.4.2.4 *Ingress Versus Egress Filtering*

At present, most firewalls look outwards and try to keep bad things out, but a growing number look inwards and try to stop bad things leaving. The pioneers were military mail systems that monitor outgoing traffic to ensure that nothing classified goes out in the clear; around 2005 some ISPs started looking at outgoing mail traffic to try to detect spam. The reason is that ISPs which host lots of infected machines and thus pump out lots of spam damage their peering relationships with other ISPs, which costs real money; so various systems have been developed to help them spot infected machines, that can then be restricted to a 'walled garden' from which they can access anti-virus software but not much else [300].

If companies whose machines get used in service denial attacks start getting sued, as has been proposed in [1285], then egress filtering can at least in principle be used to detect and stop such attacks. However, at present the incentives just aren't there, and so although people care about spam floods, almost nobody at the ISP level bothers about packet floods. This might of course change as attacks get worse or if the regulatory environment changes.

Another possible development is egress filtering for privacy, given the rising tide of spyware. Software that 'phones home', whether for copyright enforcement and marketing purposes, can disclose highly sensitive material such as local hard disk directories. Prudent organizations will increasingly wish to monitor and control this kind of traffic. In the long term we expect that 'pervasive computing' will fill our homes with all sorts of gadgets that communicate, so I wouldn't be surprised to see home firewalls that enable the householder to control which of them 'phone home', and for what purpose.

21.4.2.5 *Architecture*

Many firms just buy a firewall because it's on the tick-list of due-diligence things their auditors want to see. In that case, the sensible choice is a simple filtering router, which won't need much maintenance and won't get in the way. Where security's taken seriously, one possible approach is to invest in a really serious firewall system, which might consist of a packet filter connecting the outside world to a screened subnet, also known as a *demilitarized zone*

(DMZ), which in turn contains a number of application servers or proxies to filter mail, web and other services. The DMZ may then be connected to the internal network via a further filter that does network address translation.

In [323], there is a case study of how a firewall was deployed at Hanscom Air Force Base. The work involved surveying the user community to find what network services were needed; devising a network security policy; using network monitors to discover unexpected services that were in use; and lab testing prior to installation. Once it was up and running, the problems included ongoing maintenance (due to personnel turnover), the presence of (unmonitored) communications to other military bases, and the presence of modem pools. Few non-military organizations are likely to take this much care.

An alternative approach is to have more networks, but smaller ones. At our university, we have firewalls to separate departments, although we've got a shared network backbone and there are some shared central services. There's no reason why the students and the finance department should be on the same network, and a computer science department has got quite different requirements (and users) from a department of theology — so the network security policies should be different too. In any case keeping each network small limits the scope of any compromise.

You may even find both a big corporate firewall and departmental boundaries. At defense contractors, you may expect to find not just a fancy firewall at the perimeter, but also pumps separating networks operating at different clearance levels, with filters to ensure that classified information doesn't escape either outwards or downwards (Figure 21.2).

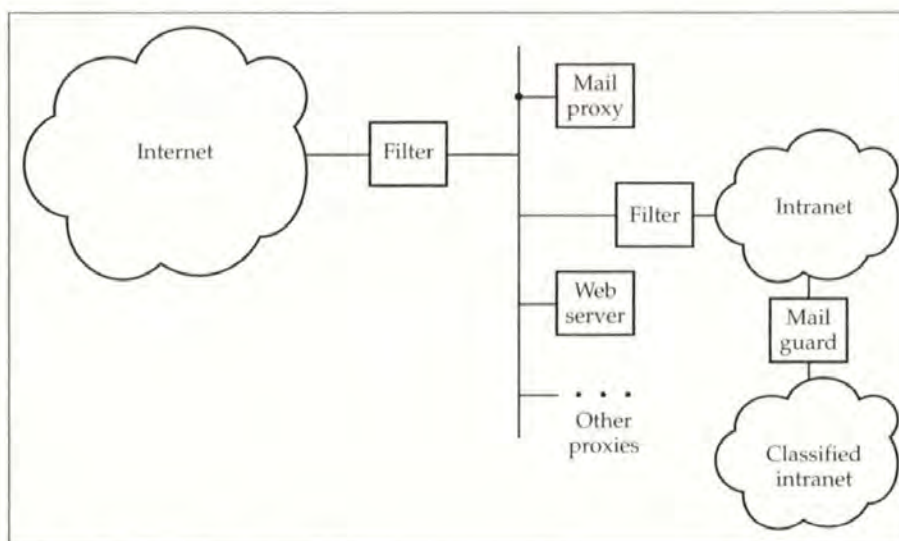


Figure 21.2: Multiple firewalls

Which is better? Well, it depends. Factors to consider when designing a network security architecture are simplicity, usability, maintainability, deperimeterisation, underblocking versus overblocking, and incentives.

First, since firewalls do only a small number of things, it's possible to make them very simple and remove many of the complex components from the underlying operating system, removing a lot of sources of vulnerability and error. If your organization has a large heterogeneous population of machines, then loading as much of the security management task as possible on a small number of simple boxes makes sense. On the other hand, if you're running something like a call centre, with a thousand identically-configured PCs, it makes sense to put your effort into keeping this configuration tight.

Second, elaborate central installations not only impose greater operational costs, but can get in the way so much that people install back doors, such as cable modems that bypass your firewall, to get their work done. In the 1990s, UK diplomats got fed up waiting for a 'secure' email system from government suppliers; they needed access to email so they could talk to people who preferred to use it. Some of them simply bought PCs from local stores and got accounts on AOL, thus exposing sensitive data to anyone who tapped the network or indeed guessed an AOL password. In fact, the diplomats of over a hundred countries had webmail accounts compromised when they were foolish enough to rely on Tor for message confidentiality, and got attacked by a malicious Tor exit node (an incident I'll discuss in section 23.4.2.) So a prudent system administrator will ensure that he knows the actual network configuration rather than just the one stated by 'policy'.

Third, firewalls (like other filtering products) tend only to work for a while until people find ways round them. Early firewalls tended to let only mail and web traffic through; so writers of applications from computer games to anonymity proxies redesigned their protocols to make the client-server traffic look as much like normal web traffic as possible. Now, of course, in the world of Web 2.0, more and more applications are actually web-based; so we can expect the same games to be played out again in the web proxy. There are particular issues with software products that insist on calling home. For example, the first time you use Windows Media Player, it tells you you need a 'security upgrade'. What's actually happening is that it 'individualizes' itself by generating a public-private keypair and sending the public key to Microsoft. If your firewall doesn't allow this, then WMP won't play protected content. Microsoft suggests you use their ISA firewall product, which will pass WMP traffic automatically. Quite a few issues of trust, transparency and competition may be raised by this!

Next, there's *deperimeterization* — the latest buzzword. Progress is making it steadily harder to put all the protection at the perimeter. The very technical ability to maintain a perimeter is undermined by the proliferation of memory sticks, of laptops, of PDAs being used for functions that used to be done on

desktop computers, and by changing business methods that involve more outsourcing of functions — whether formally to subcontractors or informally to advertising-supported web apps. If some parts of your organisation can't be controlled well (e.g. the sales force and the R&D lab) while others must be (the finance office) then separate networks are needed. The crumbling of the perimeter will be made even worse by mobility, and by the proliferation of web applications. This is complemented by a blunting of the incentive to do things at the perimeter, as useful things become harder to do. The difference between code and data is steadily eroded by new scripting languages; a determination to not allow javascript in the firm is quickly eroded by popular web sites that require it; and so on.

And then there's our old friend the Receiver Operating Characteristic or ROC curve. No filtering mechanism has complete precision, so there's inevitably a trade-off between underblocking and overblocking. If you're running a censorship system to stop kids accessing pornography in public libraries, do you underblock, and annoy parents and churches when some pictures get through, or do you overblock and get sued for infringing free-speech rights? Things are made worse by the fact that the firewall systems used to filter web content for sex, violence and bad language also tend to block free-speech sites (as many of these criticise the firewall vendors — and some offer technical advice on how to circumvent blocking.)

Finally, security depends at least as much on incentives as on technology. A sysadmin who's looking after a departmental network used by a hundred people he knows, and who will personally have to clear up any mess caused by an intrusion or a configuration error, is much more motivated than someone who's merely one member of a large team looking after thousands of machines.

21.4.3 Intrusion Detection

It's a good idea to assume that attacks will happen, and it's often cheaper to prevent some attacks and detect the rest than it is to try to prevent everything. The systems used to detect bad things happening are referred to generically as *intrusion detection systems*. The antivirus software products I discussed earlier are one example; but the term is most usually applied to boxes that sit on your network and look for signs of an attack in progress or a compromised machine [1100]. Examples include:

- spam coming from a machine in your network;
- packets with forged source addresses — such as packets that claim to be from outside a subnet coming from it, or packets that claim to be from inside arriving at it;
- a machine trying to contact a 'known bad' service such as an IRC channel that's being used to control a botnet.

In cases like this, the IDS essentially tells the sysadmin that a particular machine needs to be scrubbed and have its software reinstalled.

Other examples of intrusion detection, that we've seen in earlier chapters, are the mechanisms for detecting mobile phone cloning and fraud by bank tellers. There are also bank systems that look at customer complaints of credit card fraud to try to figure out which merchants have been leaking card data, and stock market systems that try to detect insider trading by looking for increases in trading volume prior to a price-sensitive announcement and other suspicious patterns of activity. And there are 'suspect' lists kept by airport screeners; if your name is down there, you'll be selected 'at random' for extra screening. Although these intrusion detection systems are all performing very similar tasks, their developers don't talk to each other much. One sees the same old wheels being re-invented again and again. But it's starting slowly to become a more coherent discipline, as the U.S. government has thrown hundreds of millions at the problem.

The research program actually started in the mid-1990s and was prompted by the realisation that many systems make no effective use of log and audit data. In the case of Sun's operating system Solaris, for example, we found in 1996 that the audit formats were not documented and tools to read them were not available. The audit facility seemed to have been installed to satisfy the formal checklist requirements of government systems buyers rather than to perform any useful function. There was at least the hope that improving this would help system administrators detect attacks, whether after the fact or even when they were still in progress. Since 9/11, of course, there has been a great switch of emphasis to doing data mining on large corpora of both government and commercial data, looking for conspiracies.

21.4.3.1 Types of Intrusion Detection

The simplest intrusion detection method is to sound an alarm when a threshold is passed. Three or more failed logons, a credit card expenditure of more than twice the moving average of the last three months, or a mobile phone call lasting more than six hours, might all flag the account in question for attention. More sophisticated systems generally fall into two categories.

Misuse detection systems operate using a model of the likely behaviour of an intruder. A banking system may alarm if a user draws the maximum permitted amount from a cash machine on three successive days; and a Unix intrusion detection system may look for user account takeover by alarming if a previously naive user suddenly started to use sophisticated tools like compilers. Indeed, most misuse detection systems, like antivirus scanners, look for a *signature* — a known characteristic of a particular attack.

Anomaly detection systems attempt the much harder job of looking for anomalous patterns of behaviour in the absence of a clear model of the

attacker's *modus operandi*. The hope is to detect attacks that have not been previously recognized and cataloged. Systems of this type often use AI techniques — neural networks have been fashionable from time to time.

The dividing line between misuse and anomaly detection is somewhat blurred. A good borderline case is Benford's law, which describes the distribution of digits in random numbers. One might expect that numbers beginning with the digits '1', '2', ... '9' would be equally common. But in fact with numbers that come from random natural sources, so that their distribution is independent of the number system in which they're expressed, the distribution is logarithmic: about 30% of decimal numbers start with '1'. Crooked clerks who think up numbers to cook the books, or even use random number generators without knowing Benford's law, are often caught using it [846]. Another borderline case is the *honey trap* — something enticing left to attract attention. I mentioned, for example, that some hospitals have dummy records with celebrities' names in order to entrap staff who don't respect medical confidentiality.

21.4.3.2 General Limitations of Intrusion Detection

Some intrusions are really obvious. If what you're worried about is a script kiddie vandalizing your corporate web site, then the obvious defence is to have a machine somewhere that fetches the page regularly, inspects it, and rings a really loud alarm when it changes. (Make sure you do this via an outside proxy, and don't forget that it's not just your own systems at risk. The kiddie could replace your advertisers' pictures with porn, for example, and then you'd want to pull the links to them pretty fast.)

But in the general case, intrusion detection is hard. Cohen proved that detecting viruses (in the sense of deciding whether a program is going to do something bad) is as hard as the halting problem, so we can't ever expect a complete solution [311].

Another fundamental limitation comes from the fact that there are basically two different types of security failure — those which cause an error (which we defined in 6.3 to be an incorrect state) and those which don't. An example of the former is a theft from a bank which leaves traces on the audit trail. An example of the latter is an undetected confidentiality failure caused by a radio microphone placed by a foreign intelligence service in your room. The former can be detected (at least in principle, and forgetting for now about the halting problem) by suitable processing of the data available to you. But the latter can't be. It's a good idea to design systems so that as many failures as possible fall into the former category, but it's not always practicable [289].

There's also the matter of definitions. Some intrusion detection systems are configured to block any instances of suspicious behaviour and in extreme cases to take down the affected systems. Quite apart from opening the door to

service denial attacks, this turns the intrusion detection system into firewall or an access control mechanism; and as we've already seen, access control is in general a hard problem and incorporates all sorts of issues of security policy which people often disagree on or simply get wrong.

I prefer to define an intrusion detection system as one that monitors the logs and draws the attention of authority to suspicious occurrences. This is closer to the way mobile phone operators work. It's also critical in financial investigations; see [1095] for a discussion by a special agent with the U.S. Internal Revenue Service, of what he looks for when trying to trace hidden assets and income streams. A lot hangs on educated suspicion based on long experience. For example, a \$25 utility bill may lead to a \$250,000 second house hidden behind a nominee. Building an effective system means having the people, and the machines, each do the part of the job they're best at; and this means getting the machine to do the preliminary filtering.

Then there's the cost of false alarms. For example, I used to go to San Francisco every May, and I got used to the fact that after I'd used my UK debit card in an ATM five days in a row, it would stop working. This not only upsets the customer, but the villains quickly learn to exploit it. (So do the customers — I just started making sure I got enough dollars out in the first five days to last me the whole trip.) As in so many security engineering problems, the trade-off between the fraud rate and the insult rate is the critical one — and, as discussed in section 15.9, you can't expect to improve this trade-off simply by looking at lots of different indicators. In general, you must expect that an opponent will always get past the threshold if he's patient enough and either does the attack very slowly, or does a large number of small attacks.

A difficult policy problem with commercial intrusion detection systems is *redlining*. When insurance companies used claim statistics on postcodes to decide the level of premiums to charge, it was found that many poor and black areas suffered high premiums or were excluded altogether from cover. In a number of jurisdictions this is now illegal. In general, if you build an intrusion detection system based on data mining techniques, you are at serious risk of discriminating. If you use neural network techniques, you'll have no way of explaining to a court what the rules underlying your decisions are, so defending yourself could be hard. Opaque rules can also contravene European data protection law, which entitles citizens to know the algorithms used to process their personal data.

Already in 1997, systems introduced to profile U.S. airline passengers for terrorism risk, so they could be subjected to more stringent screening, were denounced by the American-Arab Anti-Discrimination Committee [823]. Since 9/11 such problems have become much worse. How do we judge the balance point beyond which we just radicalize people and breed more attacks? I'll come back to this in Part III.

21.4.4 Specific Problems Detecting Network Attacks

Turning now to the specific problem of detecting network intrusion, the problem is much harder than (say) detecting mobile phone cloning for a number of reasons. Network intrusion detection products still don't work very well, with both high missed alarm and false alarm rates. It's common not to detect actual intrusions until afterwards — although once one is detected by other means, the traces can be found on the logs.

The reasons for the poor performance include the following, in no particular order.

- The Internet is a very noisy environment — not just at the level of content but also at the packet level. A large amount of random crud arrives at any substantial site, and enough of it can be interpreted as hostile to provide a significant false alarm rate. A survey by Bellovin [149] reports that many bad packets result from software bugs; others are the fault of out-of-date or corrupt DNS data; and some are local packets that escaped, travelled the world and returned.
- There are 'too few attacks'. If there are ten real attacks per million sessions — which is almost certainly an overestimate — then even if the system has a false alarm rate as low as 0.1%, the ratio of false to real alarms will be 100. We talked about similar problems with burglar alarms; it's also a well known problem for medics running screening programs for diseases like HIV where the test error exceeds the organism's prevalence. In general, where the signal is far below the noise, the guards get tired and even the genuine alarms get missed.
- Many network attacks are specific to particular versions of software, so a general misuse detection tool must have a large and constantly changing library of attack signatures.
- In many cases, commercial organisations appear to buy intrusion detection systems simply in order to tick a 'due diligence' box to satisfy insurers or consultants. That means the products aren't always kept up to date.
- Encrypted traffic can't easily be subjected to content analysis any more than it can be filtered for malicious code.
- The issues we discussed in the context of firewalls largely apply to intrusion detection too. You can filter at the packet layer, which is fast but can be defeated by packet fragmentation; or you can reconstruct each session, which takes more computation and so is not really suitable for network backbones; or you can examine application data, which is more expensive still — and needs to be constantly updated to cope with the arrival of new applications and attacks.

- You may have to do intrusion detection both locally and globally. The antivirus side of things may have to be done on local machines, especially if the malware arrives on encrypted web sessions; on the other hand, some attacks are *stealthy* — the opponent sends 1–2 packets per day to each of maybe 100,000 hosts. Such attacks are unlikely to be found by local monitoring; you need a central monitor that keeps histograms are kept of packets by source and destination address and by port.

So it appears unlikely that a single-product solution will do the trick. Future intrusion detection systems are likely to involve the coordination of a number of monitoring mechanisms at different levels both in the network (backbone, LAN, individual machine) and in the protocol stack (packet, session and application).

21.4.5 Encryption

In the context of preventing network attacks, many people have been conditioned to think of encryption. Encryption usually does a lot less than you might hope, as the quote from Butler Lampson and Roger Needham at the head of this chapter suggests. But it can sometimes be useful. Here I'm going to describe briefly the four most relevant network encryption scenarios: SSH; the local link protection offered by WiFi, Bluetooth and HomePlug; IPSec; and TLS. Finally I'll briefly discuss public key infrastructures (PKI), which are used to support the last two of these.

21.4.5.1 SSH

When I use my laptop to read email on my desktop machine, or do anything with any other machine in our lab for that matter, I use a protocol called *secure shell* (SSH) which provides encrypted links between Unix and Windows hosts [1369, 1, 988]. So when I come in from home over the net, my traffic is protected, and when I log on from the PC at my desk to another machine in the lab, the password I use doesn't go across the LAN in the clear.

SSH was initially written in 1995 by Tatu Ylönen, a researcher at Helsinki University of Technology in Finland, following a password-sniffing attack there. It not only sets up encrypted connections between machines, so that logon passwords don't travel across the network in the clear; it also supports other useful features, such as forwarding X sessions, which led to its rapid adoption. (In fact it's a classic case study in how to get a security product accepted in the marketplace; see [1083] for an analysis. Normally people don't want to use encryption products until a lot of other people are using them too, because of network effects; so the trick is to bundle some real other benefits with the product.)

There is a proprietary SSH product from Ylönen's company, and a number of open implementations such as OpenSSH and Putty; there's also an associated file transfer protocol SCP ('secure copy'). There are various configuration options, but in the most straightforward one, each machine has a public-private keypair. The private key is protected by a passphrase that the user types at the keyboard. To connect from (say) my laptop to a server at the lab, I ensure that my public key is loaded on, and trusted by, the server. Manual key installation is at once a strength and a weakness; it's strong in that management is intuitive, and weak as it doesn't scale particularly well. In any case, when I wish to log on to the server I'm prompted for my passphrase; a key is then set up; and the traffic is both encrypted and authenticated. Fresh keys are set up after an hour, or after a Gigabyte of traffic has been exchanged.

Possible problems with the use of SSH include the fact that the earliest version, SSH 1.0, is vulnerable to middleperson attacks because of a poor key-exchange protocol; and that if you're typing at the keyboard one character at a time, then each character gets sent in its own packet. The packet inter-arrival times can leak a surprising amount of information about what you're typing [1203]. However, the worst is probably that most SSH keys are stored in the clear, without being protected by a password at all. The consequence is that if a machine is compromised, the same can happen to every other machine that trusts an SSH key installed on it.

21.4.5.2 WiFi

WiFi is a technology used for wireless local area networks, and is very widely used: people use it at home to connect PCs to a home router, and businesses use it too, connecting devices such as tills and payment terminals as well as PCs. Games consoles and even mobile phones make increasing use of wireless LANs.

Wifi has come with a series of encryption protocols since its launch in 1997. The first widely-used one, WEP (for *wired equivalent privacy*), was shown to be fairly easily broken, even when configured correctly. Standardised with IEEE 802.11 in 1999, WEP uses the RC4 stream cipher to encrypt data with only a cyclic redundancy check for integrity. Nikita Borisov, Ian Goldberg and David Wagner showed that this led to attacks in depth [210]. Known plaintext allows keystream to be stripped off and reused; in addition, the initial values used in encryption were only 24 bits, which enabled IV collisions to be found leading to further depth attacks. False messages could be encrypted and injected into a wireless LAN, opening it to other attacks. What's more, the key was only 40 bits long in early implementations, because of U.S. export rules; so keys could be brute-forced.

That merely whetted cryptanalysts' appetite. Shortly afterwards, Scott Fluhrer, Itzhak Mantin and Adi Shamir found a really devastating attack.

It is an initialisation weakness in the RC4 stream cipher that interacts with the way in which WEP set up its initialization vectors [479]; in short order, Adam Stubblefield, John Ioannidis and Avi Rubin turned this into a working attack on WEP [1230]. Vendors bodged up their products so they would not use the specific weak keys exploited in the initial attack programs; later programs used a wider range of weak keys, and the attacks steadily improved. The history of the attack evolution is told in [993]; the latest attack, by Erik Tews, Ralf-Philipp Weinmann and Andrei Pyshkin, recovers 95% of all keys within 85,000 packets [1245]. Now there are publicly-available tools that will extract WEP keys after observing a few minutes' traffic.

Stronger encryption systems, known as Wi-Fi Protected Access (WPA), aim to solve this problem and are available on most new products. WPA shipped in 2003, and was an intermediate solution that still uses RC4. WPA2 shipped in 2004; it is also called the Robust Security Network (RSN) and uses the AES block cipher in counter mode with CBC-MAC. Within a few years, as older machines become obsolete, WPA2 should solve the cipher security problem.

So what are we to make of WiFi security? There has been a lot of noise in the press about how people should set passwords on their home routers, in case a bad man stops outside your house and uses your network to download child porn. However, a straw poll of security experts at WEIS 2006 showed that most did not bother to encrypt their home networks; drive-by downloads are a fairly remote threat. For most people in the UK or America, it's just convenient to have an open network for your guests to use, and so that you and your neighbours can use each others' networks as backups. Things are different in countries where you pay for download bandwidth; there, home router passwords are mostly set.

Things are different for businesses because of the possibility of targeted attacks. If you use a Windows machine with Windows shares open, then someone on your LAN can probably use that to infect you with malware. A random home owner may not be at much risk — with botnets trading at about a dollar a machine, it's not worth someone's while to drive around town infecting machines by hand. But if you're a high-value target, then the odds change significantly. In March 2007, retail chain TJ Maxx reported that some 45.7 million credit card numbers had been stolen from its systems; these card numbers largely related to sales in 2003 and 2004, and had been stolen from 2005 but discovered only in December 2006. The Wall Street Journal reported that an insecure WiFi connection in St Paul, Mn., was to blame [1014]; the company's SEC filing about the incident is at [1252], and the Canadian Privacy Commissioner concluded that 'The company collected too much personal information, kept it too long and relied on weak encryption technology to protect it — putting the privacy of millions of its customers at risk' [1047]. Banks sued the company, with VISA claiming fraud losses of over \$68m from the compromise of 65 million accounts; the banks eventually

settled for damages of \$41m [544]. Since June 2007, the banking industry's Payment Card Industry Data Security Standard (PCI DSS) requires companies processing credit card data to meet certain data security standards, and VISA or Mastercard can fine member banks whose merchants don't comply with it. (However, enforcement has historically been weak: it turned out that VISA had known about TJX's compliance problems, and had allowed them an extension until 2009 [1349].)

The latest implementations of WiFi are coming with mechanisms that encourage users to set up WPA encryption, and usability is a big deal for the other local connectivity protocols too.

21.4.5.3 Bluetooth

Bluetooth is another protocol used for short-range wireless communication. It's aimed at *personal area networks*, such as linking a headset to a mobile phone, or linking a mobile phone in your pocket to a hands-free phone interface in your car. It's also used to connect cameras and phones to laptops, keyboards to PCs and so on. Like WiFi, the initially deployed security protocol turned out to have flaws. In the original version, devices discover each other, and the users confirm that they wish two devices to pair by entering the same PIN at their keyboards. An attacker who's present during this pairing process can observe the traffic and then brute-force the PIN. Worse, Ollie Whitehouse, Yaniv Shaked and Avishai Wool figured out how to force two devices to rerun the pairing protocol, so that PIN-cracking attacks could be performed even on devices that were already paired [1341, 1156]. Denis K  gler also showed how to manipulate the frequency hopping so as to do a man-in-the-middle attack [747]. It's possible to mitigate these vulnerabilities by only doing pairing in a secure place and refusing requests to rekey.

Now, from version 2.1 (released in 2007), Bluetooth supports Secure Simple Pairing, an improved protocol [802]. This uses elliptic curve Diffie-Hellmann key exchange to thwart passive eavesdropping attacks, but man-in-the-middle attacks are harder; they are dealt with by generating a six digit number for numerical comparison, with a view to reducing the chance of an attack succeeding to one in a million. However, because one or both of the devices might lack a keyboard or screen (or both), it's also possible for the six-digit number to be generated at one device and entered as a passkey at another; and there's a 'just works' mode that's fully vulnerable to a middleperson attack. Finally, there's a capability to load keys out of band, such as from some other protocol that the devices use.

21.4.5.4 HomePlug

HomePlug is a protocol used for communication over the mains power line. An early version had a low bitrate, but HomePlug AV, available from 2007,

supports broadband. (Declaration of interest: I was one of the protocol's designers.) It aims to allow TVs, set-top boxes, personal video recorders, DSL and cable modems and other such devices to communicate in the home without additional cabling. We were faced with the same design constraints as the Bluetooth team: not all devices have keyboards or screens, and we needed to keep costs low. After much thought we decided to offer only two modes of operation: secure mode, in which the user manually enters into her network controller a unique AES key that's printed on the label of every device, and robust or 'simple connect' mode in which the keys are exchanged without authentication. In fact, the keys aren't even encrypted in this mode; its purpose is not to provide security but to prevent wrong associations, such as when your speakers wrongly get their audio signal from the apartment next door.

We considered offering a public-key exchange protocol, as with Bluetooth, but came to the conclusion that it didn't achieve much. If there's a middleperson attack going on where the attacker knocks out your set-top box using a jammer and connects a bogus box of the same type to your mains, then the chances are that you'll go to your network controller (some software on your PC) and see a message 'Set-top box type Philips 123 seeks admission to network. Certificate hash = 12345678. Admit/deny?' In such a circumstance, most people will press 'admit' and allow the attacker in. The only way to prevent them is to get them to read the certificate hash from the device label and type it in — and if they're going to do that, they might as well type in the key directly [967]. In short, our design was driven by usability, and we weren't convinced that public-key crypto actually bought us anything.

Time will tell which approach was best. And if we turn out to have been wrong, HomePlug (like Bluetooth and the latest versions of WiFi) lets keys be set up from other protocols by out-of-band mechanisms. So all devices in the office or home could end up with their keys managed by a single mechanism or device; and this could be convenient, or a source of vulnerabilities, depending on how future security engineers build it.

21.4.5.5 IPsec

Another approach is to do encryption and/or authentication at the IP layer using a protocol suite known as IPsec. IPsec defines a *security association* as the combination of keys, algorithms and parameters used to protect a particular packet stream. Protected packets are either encrypted or authenticated (or both); in the latter case, an authentication header is added that protects data integrity using HMAC-SHA1, while in the former the packet is encrypted and encapsulated in other packets. (The use of encryption without authentication is discouraged as it's insecure [151].) There's also an *Internet Key Exchange* (IKE) protocol to set up keys and negotiate parameters. IKE has been through a number of versions (some of the bugs that were fixed are discussed in [465]).

IPsec is widely used by firewall vendors who offer a *virtual private network* facility with their products; that is, by installing one of their boxes in each branch between the local LAN and the router, all the internal traffic can pass encrypted over the Internet. Individual PCs, such as workers' laptops and home PCs, can in theory join a VPN given a firewall that supports IPsec, but this is harder than it looks. Compatibility has been a major problem with different manufacturers' offerings just not working with each other; although firewall-to-firewall compatibility has improved recently, getting random PCs to work with a given VPN is still very much a hit-or-miss affair.

IPsec has the potential to stop some network attacks, and be a useful component in designing robust distributed systems. But it isn't a panacea. Indeed, virtual private networks exacerbate the 'deperimeterization' problem already discussed. If you have thousands of machines sitting in your employee's homes that are both in the network (as they connect via a VPN) and connected to the Internet (as their browser talks to the Internet directly via the home's cable modem) then they become a potential weak point. (Indeed, the U.S. Department of Justice ruled in 2007 that employees can't use their own PCs or PDAs for work purposes; all mobile devices used for departmental business must be centrally managed [108].)

21.4.5.6 TLS

Recall that when discussing public key encryption, I remarked that a server could publish a public key KS and any web browser could then send a message M containing a credit card number to it encrypted using KS : $\{M\}_{KS}$. This is in essence what the TLS protocol (formerly known as SSL) does, although in practice it is more complicated. It was developed to support encryption and authentication in both directions, so that both `http` requests and responses can be protected against both eavesdropping and manipulation. It's the protocol that's activated when you see the padlock on your browser toolbar.

Here is a simplified description of the version as used to protect web pages that solicit credit card numbers:

1. the client sends the server a *client hello* message that contains its name C , a transaction serial number $C\#$, and a random nonce N_C ;
2. the server replies with a *server hello* message that contains its name S , a transaction serial number $S\#$, a random nonce N_S , and a certificate CS containing its public key KS . The client now checks the certificate CS back to a root certificate issued by a company such as Verisign and stored in the browser;
3. the client sends a *key exchange* message containing a *pre-master-secret* key, K_0 , encrypted under the server public key KS . It also sends a *finished* message with a message authentication code (MAC) computed on all the

messages to date. The key for this MAC is the *master-secret*, K_1 . This key is computed by hashing the pre-master-secret key with the nonces sent by the client and server: $K_1 = h(K_{CS}, N_C, N_S)$. From this point onward, all the traffic is encrypted; we'll write this as $\{\dots\}_{KCS}$ in the client-server direction and $\{\dots\}_{KSC}$ from the server to the client. These keys are generated in turn by hashing the nonces with K_1 .

4. The server also sends a *finished* message with a MAC computed on all the messages to date. It then finally starts sending the data.

$C \rightarrow S : C, C\#, N_C$

$S \rightarrow C : S, S\#, N_S, CS$

$C \rightarrow S : \{K_0\}_{KS}$

$C \rightarrow S : \{\text{finished}, \text{MAC}(K_1, \text{everything to date})\}_{KCS}$

$S \rightarrow C : \{\text{finished}, \text{MAC}(K_1, \text{everything to date})\}_{KSC}, \{\text{data}\}_{KSC}$

The design goals included minimising the load on the browser, and then minimising the load on the server. Thus the public key encryption operation is done by the client, and the decryption by the server; the standard encryption method (*ciphersuite*) uses RSA for which encryption can be arranged to be very much faster than decryption. (This was a wrong design decision as browsers generally have a lot more compute cycles to spare than servers; it has created a brisk aftermarket for crypto accelerator boards for web servers.) Also, once a client and server have established a pre-master-secret, no more public key operations are needed as further master secrets can be obtained by hashing it with new nonces.

The full protocol is more complex than this, and has gone through a number of versions. It supports a number of different ciphersuites, so that export versions of browsers for example can be limited to 40 bit keys — a condition of export licensing that was imposed for many years by the U.S. government. Other ciphersuites support signed Diffie-Hellman key exchanges for transient keys, to provide forward and backward secrecy. TLS also has options for bidirectional authentication so that if the client also has a certificate, this can be checked by the server. In addition, the working keys KCS and KSC can contain separate subkeys for encryption and authentication. For example, the most commonly used ciphersuite uses the stream cipher RC4 for the former and HMAC for the latter, and these need separate keys.

Although early versions of SSL had a number of bugs [1308], version 3 and later (called TLS since version 3.1) appear to be sound (but they have to be implemented carefully [189]). They are being used for much more than electronic commerce — an example being medical privacy [280]. In our local teaching hospital, clinical personnel were issued with smartcards containing TLS certificates enabling them to log on to systems containing patient records. This meant, for example, that researchers could access clinical data from home during an emergency, or from their university offices if doing research. TLS

has also been available as an authentication option in Windows from Windows 2000 onwards; you can use it instead of Kerberos if you wish.

Another application is in mail, where more and more mail servers now use TLS opportunistically when exchanging emails with another mail server that's also prepared to use it. This stops passive eavesdropping, although it leaves open the possibility of middleperson attacks. To stop them too, you need some means of authenticating the public keys you use, and that brings us to the topic of public-key certificates.

21.4.5.7 PKI

During the dotcom boom, a number of companies achieved astronomical valuations by cornering the market in public-key certificates. The leading European certificate provider, Baltimore, achieved an eleven-figure market cap before crashing and burning in 2001. Investors believed that every device would need a public-key certificate in order to connect to other devices; you'd need to pay Baltimore (or Thawte, or Verisign) ten bucks every two years to renew the certificate on your toaster, or it wouldn't talk to your fridge.

As I discussed above, the keys in devices like fridges and toasters are best set up by local mechanisms such as the Bluetooth and HomePlug pairing mechanisms. But public key infrastructures are still used in a number of applications. First, there are the certificates that web sites use with TLS and that activate the security icon in your browser. Second, there are private infrastructures, such as those used by banks to set up keys for SWIFT, by mobile phone companies to exchange messages between Home Location Registers, and by companies that use TLS to authenticate users of their networks.

There is frequent semantic confusion between 'public (key infrastructure)' and '(public key) infrastructure'. In the first, the infrastructure can be used by whatever new applications come along; I'll call this an *open PKI*. In the second, it can't; I'll call this a *closed PKI*.

PKI has a number of intrinsic limitations, many of which have to do with the first interpretation — namely that the infrastructure is provided as a public service that anyone can use. I discussed many of the underlying problems in Chapter 7. Naming is difficult, and a certificate saying 'Ross Anderson has the right to administer the machine foo.com' means little in a world with dozens of people (and machines) of that name. Also, there's Kent's law: the more applications rely on a certificate, the shorter its useful life will be.

This is the 'one key or many' debate. As the world goes digital, should I expect to have a single digital key to replace each of the metal keys, credit cards, swipe access cards and other tokens that I currently carry around? Or should each of them be replaced by a different digital key? Multiple keys protect the customer: I don't want to have to use a key with which I can remortgage my house to make calls from a payphone. It's just too easy to

dupe people into signing a message by having the equipment display another, innocuous, one².

However, the killer turned out to be business needs. Multiple keys are more convenient for business, as sharing access tokens can lead to greater administrative costs and liability issues. There were many attempts to share keys; the smartcard industry tried to market 'multifunction smartcards' through the 1990s that could work as bank cards, electricity meter cards and even building access cards. Singapore even implemented such a scheme, in which even military ID doubled as bank cards. However, such schemes have pretty well died out. In one that I worked on — to reuse bank cards in electricity meters — the issues were control of the customer base and of the process of developing, upgrading and reissuing cards. In other cases, projects foundered because no-one could agree which company's logo would go on the smartcard.

Now the standard PKI machinery (the X.509 protocol suite) was largely developed to provide an electronic replacement for the telephone book, so it tends to assume that everyone will have a unique name and a unique key in an open PKI architecture. Governments hoped for a 'one key fits all' model of the world, so they could license and control the keys. But, in most applications, the natural solution is for each business to run its own closed PKI, which might be thought of at the system level as giving customers a unique account number which isn't shared with anyone else. Since then, the CA market has fractured; whereas in the late 1990s, Internet Explorer shipped with only a handful of CA keys (giving huge if temporary fortunes to the firms that controlled them), now the version in Windows XP contains hundreds.

This in turn leads to issues of trust. You don't really know who controls a key whose signature is accepted by a typical browser. Further issues include

- If you remove one of the 200-plus root certificates from Windows XP Service Pack 2, then Windows silently replaces it — unless you've got the skill to dissect out the software that does this [613]. Vista comes with fewer root certificates — but you can't delete them at all. This could be bad news for a company that doesn't want to trust a competitor, or a government that doesn't want to trust foreigners. For example, the large CA Verisign also does wiretap work for the U.S. government; so if I were running China, I wouldn't want any Chinese PC to trust their certificates (as Verisign could not just sign bad web pages — they could also sign code that Chinese machines would install and run).
- Usability is dreadful, as many sites use out-of-date certs, or certs that correspond to the wrong company. As a result, users are well trained to ignore security warnings. For example, when a New Zealand bank

²I just don't know how to be confident of a digital signature I make even on my own PC — and I've worked in security for over fifteen years. Checking all the software in the critical path between the display and the signature software is way beyond my patience.

messed up its certificate with the result that users got warned it didn't correspond to the bank, only one user out of 300 stopped — the rest just went ahead with their business [569].

- It's bad enough that the users don't care whether certificates work; yet the CAs don't seem to care, either. The UK certifier Tscheme was set up by industry as a self-regulatory scheme under the Electronic Communications Act as *'a source of independent assurance for all types of e-business and e-government transactions — especially for those transactions that depend on reliable, secure online identities'*. It was noticed in July 2006 that <https://www.tscheme.org/> had its certification path misconfigured: there was a certificate missing in the middle of the chain, so verification failed unless you manually added the missing cert. By December 2007, it still wasn't properly fixed. According to the documentation, the 'HMG Root CA' should certify everything, yet it doesn't certify the Tscheme 'Trustis FPS Root CA', and neither is included in the standard Firefox distribution. In the CA world, it seems, everyone wants to be root, and no-one wants anyone else's signature on their keys, as then they'd have no reason to exist. So stuff still doesn't work.
- Many users disable security features on their browsers, even if these weren't disabled by default when the software shipped. Recall that the third step of the TLS protocol was for the client browser to check the cert against its stored root certificates. If the check fails, the browser may ask the client for permission to proceed; but many browsers are configured to just proceed anyway.
- Certs bind a company name to a DNS name, but their vendors are usually not authorities on either; they hand out certificates after cursory due diligence, and their 'certification practice statements' they go out of their way to deny all liability.
- There are still technical shortcomings. For example, the dominant certificate format (X.509) does not have the kind of flexible and scalable 'hot card' system which the credit card industry has evolved, but rather assumes that anyone relying on a cert can download a *certificate revocation list* from the issuing authority. Also, certs are designed to certify names, when for most purposes one wants to certify an authorization.

Behind all this mess lies, as usual, security economics. During the dotcom boom in the 1990s, the SSL protocol (as TLS then was) won out over a more complex and heavyweight protocol called SET, because it placed less of a burden on developers [72]. This is exactly the same reason that operating systems such as Windows and Symbian were initially developed with too little security — they were competing for pole position in a two-sided market.

The downside in this case was that the costs of compliance were dumped on the users — who are unable to cope [357].

In short, while public key infrastructures can be useful in some applications, they are not the universal solution to security problems that their advocates claimed in the late 1990s. It's a shame some governments still think they can use PKI as a mechanism for empire-building and social control.

21.5 Topology

The topology of a network is the pattern in which its nodes are connected. The Internet classically is thought of as a cloud to which all machines are attached, so in effect every machine is (potentially) in contact with every other one. So from the viewpoint of a flash worm that propagates from one machine to another directly, without human intervention, and by choosing the next machine to attack at random, the network can be thought of as a fully connected graph. However, in many networks each node communicates with only a limited number of others. This may result from physical connectivity, as with PCs on an isolated LAN, or with a camera and a laptop that are communicating via Bluetooth; or it may come from logical connectivity. For example, when a virus spreads by mailing itself to everyone in an infected machine's address book, the network that it's infecting is one whose nodes are users of the vulnerable email client and whose edges are their presence in each others' address books.

We can bring other ideas and tools to bear when the network is in effect a *social network* like this. In recent years, physicists and sociologists have collaborated in applying thermodynamic models to the analysis of complex networks of social interactions; the emerging discipline of network analysis is reviewed in [965], and has been applied to disciplines from criminology to the study of how new technologies diffuse.

Network topology turns out to be important in service denial attacks. Rulers have known for centuries that when crushing dissidents, it's best to focus on the ringleaders; and when music industry enforcers try to close down peer-to-peer file-sharing networks they similarly go after the most prominent nodes. There's now a solid scientific basis for this. It turns out that social networks can be modelled by a type of graph in which there's a power-law distribution of vertex order; in other words, a small number of nodes have a large number of edges ending at them. These well-connected nodes help make the network resilient against random failure, and easy to navigate. Yet Reka Albert, Hawoong Jeong and Albert-László Barabási showed that they also make such networks vulnerable to targeted attack. Remove the well-connected nodes, and the network is easily disconnected [20].

This has prompted further work on how topology interacts with conflict. For example, Shishir Nagaraja and I extended Albert, Jeong and Barabási's work to the dynamic case in which, at each round, the attacker gets to destroy some nodes according to an attack strategy, and the defender then gets to replace a similar number of nodes using a defense strategy. We played attack and defense strategies off against each other; against a decapitation attack, the best defense we found was a cell structure. This helps explain why peer-to-peer systems with ring architectures turned out to be rather fragile — and why revolutionaries have tended to organise themselves in cells [924]. George Danezis and Bettina Wittneben applied these network analysis ideas to privacy, and found that by doing traffic analysis against just a few well-connected organisers, a police force can identify a surprising number of members of a dissident organisation. The reason is that if you monitor everyone who calls, or is called by, the main organisers of a typical social network, you get most of the members — unless effort was expended in organising it in a cell structure in the first place [345].

These techniques may well become even more relevant to network attack and defence for a number of reasons. First, early social-network techniques have produced real results; the capture of Saddam Hussein used a layered social network analysis [623]. Second, as people try to attack (and defend) local networks organised on an ad-hoc basis using technologies like WiFi and Bluetooth, topology will matter more. Third, social networking sites — and more conventional services like Google mail that use introductions to acquire new customers — have a lot of social network information that can be used to track people; if a phisherman uses Google mail, the police can look for the people who introduced him, and then for everyone else they introduced, when searching for contacts. Fourth, as social structure starts to be used against wrongdoers (and against citizens by repressive regimes) people will invest in cell-structured organisations and in other stealth techniques to defeat it. Finally, there are all sorts of useful things that can potentially be done with topological information. For example, people may be more able to take a view on whether devices that are 'nearby' are trustworthy, and you may not need to filter traffic so assiduously if it can come from only a few sources rather than the whole Internet. This isn't entirely straightforward, as systems change over time in ways that undermine their builders' trust assumptions, but it can still be worth thinking about.

21.6 Summary

Preventing and detecting attacks that are launched over networks, and particularly over the Internet, is probably the most newsworthy aspect of security engineering. The problem is unlikely to be solved any time soon, as many

different kinds of vulnerability contribute to the attacker's toolkit. Ideally, people would run carefully written code on trustworthy platforms. In real life, this won't happen always, or even often. In the corporate world, there are grounds for hope that firewalls can keep out the worst of the attacks, careful configuration management can block most of the rest, and intrusion detection can catch most of the residue that make it through. Home users are less well placed, and most of the machines being recruited to the vast botnets we see in action today are home machines attached to DSL or cable modems.

Hacking techniques depend partly on the opportunistic exploitation of vulnerabilities introduced accidentally by the major vendors, and partly on techniques to social-engineer people into running untrustworthy code. Most of the bad things that result are just the same bad things that happened a generation ago, but moved online, on a larger scale, and with a speed, level of automation and global dispersion that leaves law enforcement wrong-footed.

Despite all this, the Internet is not a disaster. It's always possible for a security engineer, when contemplating the problems we've discussed in this chapter, to sink into doom and gloom. Yet the Internet has brought huge benefits to billions of people, and levels of online crime are well below the levels of real-world crime. I'll discuss this in more detail when we get to policy in Part III; for now, note that the \$200 m–\$1 bn lost in the USA to phishing in 2006 was way below ordinary fraud involving things like checks, not to mention the drug trade or even the trade in stolen cars.

Herd effects matter. A useful analogy for the millions of insecure computers is given by the herds of millions of gnu that once roamed the plains of Africa. The lions could make life hard for any one gnu, but most of them survived for years by taking shelter in numbers. The Internet's much the same. There are analogues of the White Hunter, who'll carefully stalk a prime trophy animal; so you need to take special care if anyone might see you in these terms. And if you think that the alarms in the press about 'Evil Hackers Bringing Down the Internet' are somehow equivalent to a hungry African peasant poaching the game, then do bear in mind the much greater destruction done by colonial ranching companies who had the capital to fence off the veld in 100,000-acre lots. Economics matters, and we are still feeling our way towards the kinds of regulation that will give us a reasonably stable equilibrium. In fact, colleagues and I wrote a report for the European Network and Information Security Agency on the sort of policy incentives that might help [62]. I'll return to policy in Part III.

Research Problems

Seven years ago, the centre of gravity in network security research was technical: we were busy looking for new attacks on protocols and applications

as the potential for denial-of-service attacks started to become clear. Now, in 2007, there are more threads of research. Getting protocols right still matters and it's unfortunate (though understandable in business terms) that many firms still ship products quickly and get them right later. This has led to calls for vendor liability, for example from a committee of the UK Parliament [625]. On the security-economics front, there is much interesting work to be done on decent metrics: on measuring the actual wickedness that goes on, and feeding this not just into the policy debate but also into law enforcement.

Systems people do a lot of work on measuring the Internet to understand how it's evolving as more and more devices, people and applications join in. And at the level of theory, more and more computer scientists are looking at ways in which network protocols could be aligned with stakeholder interests, so that participants have less incentive to cheat [971].

Further Reading

The early classic on Internet security was written by Steve Bellovin and Bill Cheswick [157]; other solid books are by Simson Garfinkel and Eugene Spafford on Unix and Internet security [517], and by Terry Escamilla on intrusion detection [439]. These give good introductions to network attacks (though like any print work in this fast-moving field, they are all slightly dated). The seminal work on viruses is by Fred Cohen [311], while Java security is discussed by Gary McGraw and Ed Felten [859] as well as by LiGong (who designed it) [539]. Eric Rescorla has a book on the details of TLS [1070]; another useful description — shorter than Eric's but longer than the one I gave above — is by Larry Paulson [1010]. Our policy paper for ENISA can be found at [62].

It's important to know a bit about the history of attacks — as they recur — and to keep up to date with what's going on. A survey of security incidents on the Internet in the late 1990s can be found in John Howard's thesis [626]. Advisories from CERT [321] and bugtraq [239] are one way of keeping up with events, and hacker sites such as www.phrack.com bear watching. However, as malware becomes commercial, I'd suggest you also keep up with the people who measure botnets, spam and phishing. As of 2007, I'd recommend Team Cymru at <http://www.cymru.com/>, the Anti-Phishing Working Group at <http://www.antiphishing.org/>, the Shadowserver Foundation at <http://www.shadowserver.org/>, Symantec's half-yearly threat report at www.symantec.com/threatreport/, and our blog at www.lightbluetouchpaper.net.