INTERNATIONAL BESTSELLER

Over 150,000 copies sold in hardcover

Secrets & Lies

Digital Security in a Networked World

With new information about post-9/11 security

Bruce Schneier

Praise for *Secrets and Lies*

"Successful companies embrace risk, and Schneier shows how to bring that thinking to the Internet."

–Mary Meeker, Managing Director and Internet Analyst, Morgan Stanley, Dean Witter

"Bruce shows that concern for security should not rest in the IT department alone, but also in the business office . . . *Secrets and Lies* is the breakthrough text we've been waiting for to tell both sides of the story."

–Steve Hunt, Vice President of Research, Giga Information Group

"Good security is good business. And security is not (just) a technical issue; it's a people issue! Security expert Bruce Schneier tells you why and how. If you want to be successful, you should read this book before the competition does."

–Esther Dyson, Chairman, EDventure Holdings

"Setting himself apart, Schneier navigates rough terrain without being overly technical or sensational—two common pitfalls of writers who take on cybercrime and security. All this helps to explain Schneier's long-standing cult-hero status, even—indeed especially—among his esteemed hacker adversaries."

–*Industry Standard*

"All in all, as a broad and readable security guide, *Secrets and Lies* should be near the top of the IT required-reading list."

–*eWeek*

"*Secrets and Lies* should begin to dispel the fog of deception and special pleading around security, and it's fun."

–*New Scientist*

"This book should be, and can be, read by any business executive, no specialty in security required . . . At Walker Digital, we spent millions of dollars to understand what Bruce Schneier has deftly explained here."

–Jay S. Walker, Founder of Priceline.com

"Just as *Applied Cryptography* was the bible for cryptographers in the 90's, so *Secrets and Lies* will be the official bible for INFOSEC in the new millennium. I didn't think it was possible that a book on business security could make me laugh and smile, but Schneier has made this subject very enjoyable."

–Jim Wallner, National Security Agency

"The news media offer examples of our chronic computer security woes on a near-daily basis, but until now there hasn't been a clear, comprehensive guide that puts the wide range of digital threats in context. The ultimate knowledgeable insider, Schneier not only provides definitions, explanations, stories, and strategies, but a measure of hope that we can get through it all."

–Steven Levy, author of *Hackers* and *Crypto*

"In his newest book, *Secrets and Lies: Digital Security in a Networked World*, Schneier emphasizes the limitations of technology and offers managed security monitoring as the solution of the future."

–*Forbes Magazine*

Secrets and Lies

Secrets and Lies

DIGITAL SECURITY IN A NETWORKED WORLD

Bruce Schneier

Wiley Publishing, Inc.

Publisher: Robert Ipsen
Editor: Carol Long
Managing Editor: Micheline Frederick
Associate New Media Editor: Brian Snapp
Text Design & Composition: North Market Street Graphics

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Schneier, Bruce, 1963–
Secrets and lies : digital security in a networked world / Bruce Schneier.
p. cm.
"Wiley Computer Publishing."
ISBN 0-471-25311-1 (cloth : alk. paper) ISBN 0-471-45380-3 (paper : alk. paper)
1. Computer security. 2. Computer networks—Security measures. I. Title.
QA76.9.A25 S352 2000
005.8—dc21 00-042252

Printed in the United States of America.

10 9 8 7 6 5 4 3 2

To Karen: DMASC

Contents

Introduction to the Paperback Edition

It's been over three years since the first edition of *Secrets and Lies* was published. Reading through it again after all this time, the most amazing thing is how little things have changed. Today, two years after 9/11 and in the middle of the worst spate of computer worms and viruses the world has ever seen, the book is just as relevant as it was when I wrote it.

The attackers and attacks are the same. The targets and the risks are the same. The security tools to defend ourselves are the same, and they're just as ineffective as they were three years ago. If anything, the problems have gotten worse. It's the hacking tools that are more effective and more efficient. It's the ever-more-virulent worms and viruses that are infecting more computers faster. Fraud is more common. Identity theft is an epidemic. Wholesale information theft—of credit card numbers and worse—is happening more often. Financial losses are on the rise. The only good news is that cyberterrorism, the post-9/11 bugaboo that's scaring far too many people, is no closer to reality than it was three years ago.

The reasons haven't changed. In Chapter 23, I discuss the problems of complexity. Simply put, complexity is the worst enemy of security. As systems get more complex, they necessarily get less secure. Today's computer and network systems are far more complex than they were when I wrote the first edition of this book, and they'll be more complex still in another three years. This means that today's computers and networks are less secure than they were earlier, and they will be even less

secure in the future. Security technologies and products may be improving, but they're not improving quickly enough. We're forced to run the Red Queen's race, where it takes all the running you can do just to stay in one place.

As a result, today computer security is at a crossroads. It's failing, regularly, and with increasingly serious results. CEOs are starting to notice. When they finally get fed up, they'll demand improvements. (Either that or they'll abandon the Internet, but I don't believe that is a likely possibility.) And they'll get the improvements they demand; corporate America can be an enormously powerful motivator once it gets going.

For this reason, I believe computer security will improve eventually. I don't think the improvements will come in the short term, and I think they will be met with considerable resistance. This is because the engine of improvement will be fueled by corporate boardrooms and not computer-science laboratories, and as such won't have anything to do with technology. Real security improvement will only come through liability: holding software manufacturers accountable for the security and, more generally, the quality of their products. This is an enormous change, and one the computer industry is not going to accept without a fight.

But I'm getting ahead of myself here. Let me explain why I think the concept of liability can solve the problem.

It's clear to me that computer security is not a problem that technology can solve. Security solutions have a technological component, but security is fundamentally a people problem. Businesses approach security as they do any other business uncertainty: in terms of risk management. Organizations optimize their activities to minimize their cost–risk product, and understanding those motivations is key to understanding computer security today. It makes no sense to spend more on security than the original cost of the problem, just as it makes no sense to pay liability compensation for damage done when spending money on security is cheaper. Businesses look for financial sweet spots—adequate security for a reasonable cost, for example—and if a security solution doesn't make business sense, a company won't do it.

This way of thinking about security explains some otherwise puzzling security realities. For example, historically most organizations haven't spent a lot of money on network security. Why? Because the costs have

been significant: time, expense, reduced functionality, frustrated end-users. (Increasing security regularly frustrates end-users.) On the other hand, the costs of ignoring security and getting hacked have been, in the scheme of things, relatively small. We in the computer security field like to think they're enormous, but they haven't really affected a company's bottom line. From the CEO's perspective, the risks include the possibility of bad press and angry customers and network downtime—none of which is permanent. And there's some regulatory pressure, from audits or lawsuits, which adds additional costs. The result: a smart organization does what everyone else does, and no more. Things are changing; slowly, but they're changing. The risks are increasing, and as a result spending is increasing.

This same kind of economic reasoning explains why software vendors spend so little effort securing their own products. We in computer security think the vendors are all a bunch of idiots, but they're behaving completely rationally from their own point of view. The costs of adding good security to software products are essentially the same ones incurred in increasing network security—large expenses, reduced functionality, delayed product releases, annoyed users—while the costs of ignoring security are minor: occasional bad press, and maybe some users switching to competitors' products. The financial losses to industry worldwide due to vulnerabilities in the Microsoft Windows operating system are not borne by Microsoft, so Microsoft doesn't have the financial incentive to fix them. If the CEO of a major software company told his board of directors that he would be cutting the company's earnings per share by a third because he was going to really—no more pretending—take security seriously, the board would fire him. If I were on the board, *I* would fire him. Any smart software vendor will talk big about security, but do as little as possible, because that's what makes the most economic sense.

Think about why firewalls succeeded in the marketplace. It's not because they're effective; most firewalls are configured so poorly that they're barely effective, and there are many more effective security products that have never seen widespread deployment (such as e-mail encryption). Firewalls are ubiquitous because corporate auditors started demanding them. This changed the cost equation for businesses. The cost of adding a firewall was expense and user annoyance, but the cost of not having a firewall was failing an audit. And even worse, a company

without a firewall could be accused of not following industry best practices in a lawsuit. The result: everyone has firewalls all over their network, whether they do any actual good or not.

As scientists, we are awash in security technologies. We know how to build much more secure operating systems. We know how to build much more secure access control systems. We know how to build much more secure networks. To be sure, there are still technological problems, and research continues. But in the real world, network security is a business problem. The only way to fix it is to concentrate on the business motivations. We need to change the economic costs and benefits of security. We need to make the organizations in the best position to fix the problem *want* to fix the problem.

To do that, I have a three-step program. None of the steps has anything to do with technology; they all have to do with businesses, economics, and people.

STEP ONE: ENFORCE LIABILITIES

This is essential. Remember that I said the costs of bad security are not borne by the software vendors that produce the bad security. In economics this is known as an externality: a cost of a decision that is borne by people other than those making the decision. Today there are no real consequences for having bad security, or having low-quality software of any kind. Even worse, the marketplace often rewards low quality. More precisely, it rewards additional features and timely release dates, even if they come at the expense of quality. If we expect software vendors to reduce features, lengthen development cycles, and invest in secure software development processes, they must be liable for security vulnerabilities in their products. If we expect CEOs to spend significant resources on their own network security—especially the security of their customers—they must be liable for mishandling their customers' data. Basically, we have to tweak the risk equation so the CEO cares about actually fixing the problem. And putting pressure on his balance sheet is the best way to do that.

This could happen in several different ways. Legislatures could impose liability on the computer industry by forcing software manufacturers to live with the same product liability laws that affect other indus-

tries. If software manufacturers produced a defective product, they would be liable for damages. Even without this, courts could start imposing liability-like penalties on software manufacturers and users. This is starting to happen. A U.S. judge forced the Department of Interior to take its network offline, because it couldn't guarantee the safety of American Indian data it was entrusted with. Several cases have resulted in penalties against companies that used customer data in violation of their privacy promises, or collected that data using misrepresentation or fraud. And judges have issued restraining orders against companies with insecure networks that are used as conduits for attacks against others. Alternatively, the industry could get together and define its own liability standards.

Clearly this isn't all or nothing. There are many parties involved in a typical software attack. There's the company that sold the software with the vulnerability in the first place. There's the person who wrote the attack tool. There's the attacker himself, who used the tool to break into a network. There's the owner of the network, who was entrusted with defending that network. One hundred percent of the liability shouldn't fall on the shoulders of the software vendor, just as 100 percent shouldn't fall on the attacker or the network owner. But today 100 percent of the cost falls on the network owner, and that just has to stop.

However it happens, liability changes everything. Currently, there is no reason for a software company not to offer more features, more complexity, more versions. Liability forces software companies to think twice before changing something. Liability forces companies to protect the data they're entrusted with.

STEP TWO: ALLOW PARTIES TO TRANSFER LIABILITIES

This will happen automatically, because CEOs turn to insurance companies to help them manage risk, and liability transfer is what insurance companies do. From the CEO's perspective, insurance turns variable-cost risks into fixed-cost expenses, and CEOs like fixed-cost expenses because they can be budgeted. Once CEOs start caring about security—and it will take liability enforcement to make them really care—they're going to look to the insurance industry to help them out. Insurance companies are not stupid; they're going to move into cyberinsurance in a big

way. And when they do, they're going to drive the computer security industry...just as they drive the security industry in the brick-and-mortar world.

A CEO doesn't buy security for his company's warehouse—strong locks, window bars, or an alarm system—because it makes him feel safe. He buys that security because the insurance rates go down. The same thing will hold true for computer security. Once enough policies are being written, insurance companies will start charging different premiums for different levels of security. Even without legislated liability, the CEO will start noticing how his insurance rates change. And once the CEO starts buying security products based on his insurance premiums, the insurance industry will wield enormous power in the marketplace. They will determine which security products are ubiquitous, and which are ignored. And since the insurance companies pay for the actual losses, they have a great incentive to be rational about risk analysis and the effectiveness of security products. This is different from a bunch of auditors deciding that firewalls are important; these are companies with a financial incentive to get it right. They're not going to be swayed by press releases and PR campaigns; they're going to demand real results.

And software companies will take notice, and will strive to increase the security in the products they sell, in order to make them competitive in this new "cost plus insurance cost" world.

STEP THREE: PROVIDE MECHANISMS TO REDUCE RISK

This will also happen automatically. Once insurance companies start demanding real security in products, it will result in a sea change in the computer industry. Insurance companies will reward companies that provide real security, and punish companies that don't—and this will be entirely market driven. Security will improve because the insurance industry will push for improvements, just as they have in fire safety, electrical safety, automobile safety, bank security, and other industries.

Moreover, insurance companies will want it done in standard models that they can build policies around. A network that changes every month or a product that is updated every few months will be much harder to

insure than a product that never changes. But the computer field naturally changes quickly, and this makes it different, to some extent, from other insurance-driven industries. Insurance companies will look to security processes that they can rely on: processes of secure software development before systems are released, and the processes of protection, detection, and response that I talk about in Chapter 24. And more and more, they're going to look toward outsourced services.

For over four years I have been CTO of a company called Counterpane Internet Security, Inc. We provide outsourced security monitoring for organizations. This isn't just firewall monitoring or IDS monitoring but full network monitoring. We defend our customers from insiders, outside hackers, and the latest worm or virus epidemic in the news. We do it affordably, and we do it well. The goal here isn't 100 percent perfect security, but rather adequate security at a reasonable cost. This is the kind of thing insurance companies love, and something I believe will become as common as fire-suppression systems in the coming years.

The insurance industry prefers security outsourcing, because they can write policies around those services. It's much easier to design insurance around a standard set of security services delivered by an outside vendor than it is to customize a policy for each individual network. Today, network security insurance is a rarity—very few of our customers have such policies—but eventually it will be commonplace. And if an organization has Counterpane—or some other company—monitoring its network, or providing any of a bunch of other outsourced services that will be popping up to satisfy this market need, it'll easily be insurable.

Actually, this isn't a three-step program. It's a one-step program with two inevitable consequences. Enforce liability, and everything else will flow from it. It has to. There's no other alternative.

Much of Internet security is a common: an area used by a community as a whole. Like all commons, keeping it working benefits everyone, but any individual can benefit from exploiting it. (Think of the criminal justice system in the real world.) In our society we protect our commons—environment, working conditions, food and drug practices, streets, accounting practices—by legislating those areas and by making companies liable for taking undue advantage of those commons. This kind of thinking is what gives us bridges that don't collapse, clean air and water, and sanitary restaurants. We don't live in a "buyer beware" society; we hold companies liable when they take advantage of buyers.

There's no reason to treat software any differently from other products. Today Firestone can produce a tire with a single systemic flaw and they're liable, but Microsoft can produce an operating system with multiple systemic flaws discovered per week and not be liable. Today if a home builder sells you a house with hidden flaws that make it easier for burglars to break in, you can sue the home builder; if a software company sells you a software system with the same problem, you're stuck with the damages. This makes no sense, and it's the primary reason computer security is so bad today. I have a lot of faith in the marketplace and in the ingenuity of people. Give the companies in the best position to fix the problem a financial incentive to fix the problem, and fix it they will.

ADDITIONAL BOOKS

I've written two books since *Secrets and Lies* that may be of interest to readers of this book:

Beyond Fear: Thinking Sensibly About Security in an Uncertain World is a book about security in general. In it I cover the entire spectrum of security, from the personal issues we face at home and in the office to the broad public policies implemented as part of the worldwide war on terrorism. With examples and anecdotes from history, sports, natural science, movies, and the evening news, I explain to a general audience how security really works, and demonstrate how we all can make ourselves safer by thinking of security not in absolutes, but in terms of trade-offs—the inevitable cash outlays, taxes, inconveniences, and diminished freedoms we accept (or have forced on us) in the name of enhanced security. Only after we accept the inevitability of trade-offs and learn to negotiate accordingly will we have a truly realistic sense of how to deal with risks and threats.

http://www.schneier.com/bf.html

Practical Cryptography (written with Niels Ferguson) is about cryptography as it is used in real-world systems: about cryptography as an engineering discipline rather than cryptography as a mathematical science. Building real-world cryptographic systems is vastly different from the abstract world depicted in most books on cryptography, which assumes a pure mathematical ideal that magically solves your security problems.

Designers and implementers live in a very different world, where nothing is perfect and where experience shows that most cryptographic systems are broken due to problems that have nothing to do with mathematics. This book is about how to apply the cryptographic functions in a real-world setting in such a way that you actually get a secure system.

http://www.schneier.com/book-practical.html

FURTHER READING

There's always more to say about security. Every month there are new ideas, new disasters, and new news stories that completely miss the point. For almost six years now I've written *Crypto-Gram*, a free monthly e-mail newsletter that tries to be a voice of sanity and sense in an industry filled with fear, uncertainty, and doubt. With more than 100,000 readers, *Crypto-Gram* is widely cited as the industry's most influential publication. There's no fluff. There's no advertising. Just honest and impartial summaries, analyses, insights, and commentaries about the security stories in the news.

To subscribe, visit:

http://www.schneier.com/crypto-gram.html

Or send a blank message to:

crypto-gram-subscribe@chaparraltree.com

You can read back issues on the Web site, too. Some specific articles that may be of interest are:

Risks of cyberterrorism:
http://www.schneier.com/crypto-gram-0306.html#1

Militaries and cyberwar:
http://www.schneier.com/crypto-gram-0301.html#1

The "Security Patch Treadmill":
http://www.schneier.com/crypto-gram-0103.html#1

Full disclosure and security:
http://www.schneier.com/crypto-gram-0111.html#1

How to think about security:
http://www.schneier.com/crypto-gram-0204.html#1

What military history can teach computer security (parts 1 and 2):
http://www.schneier.com/crypto-gram-0104.html#1
http://www.schneier.com/crypto-gram-0105.html#1

Thank you for taking the time to read *Secrets and Lies.* I hope you enjoy it, and I hope you find it useful.

Bruce Schneier
January 2004

Preface

I have written this book partly to correct a mistake.

Seven years ago I wrote another book: *Applied Cryptography.* In it I described a mathematical utopia: algorithms that would keep your deepest secrets safe for millennia, protocols that could perform the most fantastical electronic interactions—unregulated gambling, undetectable authentication, anonymous cash—safely and securely. In my vision cryptography was the great technological equalizer; anyone with a cheap (and getting cheaper every year) computer could have the same security as the largest government. In the second edition of the same book, written two years later, I went so far as to write: "It is insufficient to protect ourselves with laws; we need to protect ourselves with mathematics."

It's just not true. Cryptography can't do any of that.

It's not that cryptography has gotten weaker since 1994, or that the things I described in that book are no longer true; it's that cryptography doesn't exist in a vacuum.

Cryptography is a branch of mathematics. And like all mathematics, it involves numbers, equations, and logic. Security, palpable security that you or I might find useful in our lives, involves people: things people know, relationships between people, people and how they relate to machines. Digital security involves computers: complex, unstable, buggy computers.

Mathematics is perfect; reality is subjective. Mathematics is defined;

computers are ornery. Mathematics is logical; people are erratic, capricious, and barely comprehensible.

The error of *Applied Cryptography* is that I didn't talk at all about the context. I talked about cryptography as if it were The Answer™. I was pretty naïve.

The result wasn't pretty. Readers believed that cryptography was a kind of magic security dust that they could sprinkle over their software and make it secure. That they could invoke magic spells like "128-bit key" and "public-key infrastructure." A colleague once told me that the world was full of bad security systems designed by people who read *Applied Cryptography.*

Since writing the book, I have made a living as a cryptography consultant: designing and analyzing security systems. To my initial surprise, I found that the weak points had nothing to do with the mathematics. They were in the hardware, the software, the networks, and the people. Beautiful pieces of mathematics were made irrelevant through bad programming, a lousy operating system, or someone's bad password choice. I learned to look beyond the cryptography, at the entire system, to find weaknesses. I started repeating a couple of sentiments you'll find throughout this book: "Security is a chain; it's only as secure as the weakest link." "Security is a process, not a product."

Any real-world system is a complicated series of interconnections. Security must permeate the system: its components and connections. And in this book I argue that modern systems have so many components and connections—some of them not even known by the systems' designers, implementers, or users—that insecurities always remain. No system is perfect; no technology is The Answer™.

This is obvious to anyone involved in real-world security. In the real world, security involves processes. It involves preventative technologies, but also detection and reaction processes, and an entire forensics system to hunt down and prosecute the guilty. Security is not a product; it itself is a process. And if we're ever going to make our digital systems secure, we're going to have to start building processes.

A few years ago I heard a quotation, and I am going to modify it here: If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.

This book is about those security problems, the limitations of technology, and the solutions.

HOW TO READ THIS BOOK

Read this book in order, from beginning to end.

No, really. Many technical books are meant to skim, bounce around in, and use as a reference. This book isn't. This book has a plot; it tells a story. And like any good story, it makes less sense telling it out of order. The chapters build on each other, and you won't buy the ending if you haven't come along on the journey.

Actually, I want you to read the book through once, and then read it through a second time. This book argues that in order to understand the security of a system, you need to look at the entire system—and not at any particular technologies. Security itself is an interconnected system, and it helps to have cursory knowledge of everything before learning more about anything. But two readings is probably too much to ask; forget I mentioned it.

This book has three parts. Part 1 is "The Landscape," and gives context to the rest of the book: who the attackers are, what they want, and what we need to deal with the threats. Part 2 is "Technologies," basically a bunch of chapters describing different security technologies and their limitations. Part 3 is "Strategies": Given the requirements of the landscape and the limitations of the technologies, what do we do now?

I think digital security is about the coolest thing you can work on today, and this book reflects that feeling. It's serious, but fun, too. Enjoy the read.

Secrets and Lies

1

Introduction

During March 2000, I kept a log of security events from various sources. Here are the news highlights:

Someone broke into the business-to-business Web site for SalesGate.com and stole about 3,000 customer records, including credit card numbers and other personal information. He posted some of them on the Internet.

For years, personal information has "leaked" from Web sites (such as Intuit) to advertisers (such as DoubleClick). When visitors used various financial calculators on the Intuit site, a design glitch in the Web site's programming allowed information they entered to be sent to DoubleClick. This happened without the users' knowledge or consent, and (more surprising) without Intuit's knowledge or consent.

Convicted criminal hacker Kevin Mitnick testified before Congress. He told them that social engineering is a major security vulnerability: He can often get passwords and other secrets just by pretending to be someone else and asking.

A Gallup poll showed that a third of online consumers said that they might be less likely to make a purchase from a Web site, in light of recent computer-security events.

Personal data from customers who ordered the PlayStation 2 from the Sony Web site were accidentally leaked to some other customers. (This is actually a rampant problem on all sorts of sites. People try to

check out, only to be presented with the information of another random Web customer.)

Amazon.com pays commissions to third-party Web sites for referrals. Someone found a way to subvert the program that manages this, enabling anyone to channel information to whomever. It is unclear whether Amazon considers this a problem.

The CIA director denied that the United States engages in economic espionage, but did not go on to deny the existence of the massive intelligence-gathering system called ECHELON.

Pierre-Guy Lavoie, 22, was convicted in Quebec of breaking into several Canadian and U.S. government computers. He will serve 12 months in prison.

Japan's Defense Agency delayed deployment of a new defense computer system after it discovered that the software had been developed by the members of the Aum Shinrikyo cult.

A new e-mail worm, called Pretty Park, spread across the Internet. It's a minor modification of one that appeared last year. It spreads automatically, by sending itself to all the addresses listed in a user's Outlook Express program.

Novell and Microsoft continued to exchange barbs about an alleged security bug with Windows 2000's Active Directory. Whether or not this is a real problem depends on what kind of security properties you expect from your directory. (I believe it's a design flaw in Windows, and not a bug.)

Two people in Sicily (Giuseppe Russo and his wife, Sandra Elazar) were arrested after stealing about 1,000 U.S. credit card numbers on the Internet and using them to purchase luxury goods and lottery tickets.

A hacker (actually a bored teenager) known as "Coolio" denied launching massive denial-of-service attacks in February 2000. He admitted to hacking into about 100 sites in the past, including cryptography company RSA Security and a site belonging to the U.S. State Department.

Attackers launched a denial-of-service attack against Microsoft's Israeli Web site.

Jonathan Bosanac, a.k.a. "The Gatsby," was sentenced to 18 months in prison for hacking into three telephone company sites.

The military of Taiwan announced that it discovered more than 7,000 attempts by Chinese hackers to enter the country's security systems. This tantalizing statistic was not elaborated on.

Here are some software vulnerabilities reported during March 2000:

A vulnerability was reported in Microsoft Internet Explorer 5.0 (in Windows 95, 98, NT 4.0, and 2000) that allows an attacker to set up a Web page giving him the ability to execute any program on a visitor's machine.

By modifying the URL, an attacker can completely bypass the authentication mechanisms protecting the remote-management screens of the Axis StarPoint CD-ROM servers.

If an attacker sends the Netscape Enterprise Server 3.6 a certain type of long message, a buffer overflow crashes a particular process. The attacker can then execute arbitrary code remotely on the server.

It is possible to launch some attacks (one denial-of-service attack, and another attack against a CGI script) that Internet Security Systems's RealSecure Network Intrusion Detection software does not detect.

By sending a certain URL to a server running Allaire's ColdFusion product, an attacker can receive an error message giving information about the physical paths to various files.

Omniback is a Hewlett-Packard product that performs system backup routines. An attacker can manipulate the product to cause a denial-of-service attack.

There is a vulnerability in the configuration of Dosemu, the DOS emulator shipped with Corel Linux 1.0, that allows users to execute commands with root privileges.

By manipulating the contents of certain variables, an attacker can exploit a vulnerability in DNSTools 1.0.8 to execute arbitrary code.

SGI has a package called InfoSearch that automatically converts text documentation to HTML Web content. A bug in the CGI script allows attackers to execute commands on the server at the Web server privilege level.

Several vulnerabilities were discovered in the e-mail client The Bat!, allowing an attacker to steal files from users' computers.

Microsoft's Clip Art Gallery lets users download clip art files from the Web. Under certain conditions, a malformed clip art file can let arbitrary code execute on the user's computer.

If you send a long login name and password (even an incorrect one) to BisonWare's FTP Server 3.5, it will crash.

An intruder can crash Windows 95 and 98 computers using specially coded URLs.

Here is a list of the 65 Web sites known to be defaced during the month, as listed at the attrition.org Web site. In this context, "defaced" means that someone broke into the Web site and modified the home page:

Tee Plus; Suede Records; Masan City Hall; The Gallup Organization; Wired Connection; Vanier College; Name Our Child; Mashal Books; Laboratório de Matemática Aplicada da Universidade Federal do Rio de Janeiro; Elite Calendar; Centro de Processamento de Dados do Rio de Janeiro; Parliament of India; United Network for Organ Sharing; UK Jobs; Tennessee State University; St. Louis Metropolitan Sewer District; College of the Siskiyous; Russian Scientific Center for Legal Information, Ministry of Justice; RomTec Plc; Race Lesotho; Monmouth College; University of St. Thomas Library; Int Idea Sweden; Goddard College; Association of EDI Users; Bitstop, Inc; Custom Systems; Classic Amiga; 98 Skate; CU Naked; Korea National University of Education; PlayStation 2; Association for Windows NT System Professionals; K.Net Telecomunicações Ltda.; CyberCT Malaysia; Birmingham Windows NT User Group; Bloem S.A.; Aware, Inc.; Ahmedabad Telephone Online Directory, Ahmedabad Telecom District; Fly Pakistan; Quality Business Solutions; Out; Internet Exposure; Belgium Province de Hainaut; Glen Cove School District; Germantown Academy; Federatie van Wervings en Selectiebureaus; Engineering Export Promotion Council, Ministry of Commerce, India; AntiOnline's AntiCode; Pigman; Lasani; What Online; Weston High School; Vasco Boutique; True Systems; Siemens Italy; Progress Korea; Phase Devices Ltd.; National Treasury Employees Union; National Postal Mail Handlers Union; Metricks; Massachusetts Higher Education Network; The London Institute; Fort Campbell School System; and MaxiDATA Tecnologia e Informatica Ltda.

And finally, attacks against a home computer, attached to the Internet via a cable modem, belonging to a random friend of mine:

- Twenty-six scans, looking for vulnerabilities to exploit.
- Four particularly determined attempts at breaking into the computer, including basic vulnerability scans and piles of other crafty hacker tricks.

Actually, the lists only run through March 7, 2000. I got tired of keeping records after that.

Looking over this list, what strikes me is the wide array of problems, vulnerabilities, and attacks. Some of these vulnerabilities are in supposedly secure software products; one is even in a security product. Some of them are in e-commerce systems that were probably designed with security in mind. Some of them are in new products, others are in products that have been sold for years. Sometimes the vendor doesn't even agree that there is a problem.

The first seven days of March 2000 were not exceptional. Other weeks would have similar logs; some would have much worse. In fact, data portend that things are getting worse: The number of security vulnerabilities, breaches, and disasters is increasing over time. Even as we learn more about security—how to design cryptographic algorithms, how to build secure operating systems—we build things with less security. Why this is so, and what can be done about it, is the subject of this book.

SYSTEMS

The notion of a "system" is relatively new to science. Eastern philosophers have long seen the world as a single system with various components, but Westerners have segmented the world into separate things that interact in different ways.

Machines have only recently become systems. A pulley is a machine; an elevator is a complex system with many different machines. Systems interact: An elevator interacts with the building's electrical system, its fire-control system, and probably even its environmental control system. Computers interact to form networks, and networks interact to form even larger networks, and . . . you get the idea.

Admiral Grace Hopper said: "Life was simple before World War II. After that, we had systems." This is an insightful comment.

Once you start conceptualizing systems, it's possible to design and build on a more complex scale. It's the difference between a building and a skyscraper, a cannon and a Patriot missile, a landing strip and an airport. Anyone can build a traffic light, but it takes a different mindset to conceive of a citywide traffic control system.

The Internet is probably the most complex system ever developed. It contains millions of computers, connected in an inconceivably complex physical network. Each computer has hundreds of software programs running on it; some of these programs interact with other programs on the computer, some of them interact with other programs on other computers across the network. The system accepts user input from millions of people, sometimes all at the same time.

As the man said: "Sir, it is like a dog standing upon his hind legs, you are not surprised to see it not done well, you are surprised to see it done at all."

Systems have several interesting properties relevant to this book.

First, they are complex. Machines are simple: a hammer, a door hinge, a steak knife. Systems are much more complicated; they have components, feedback loops, mean times between failure, infrastructure. Digital systems are daedal; even a simple computer program has hundreds of thousands of lines of computer code doing all sorts of different things. A complex computer program has thousands of components, each of which has to work by itself and in interaction with all the other components. This is why object-oriented programming was developed: to deal with the complexity of digital systems.

Second, systems interact with each other, forming even larger systems. This can happen on purpose—programmers use objects to deliberately break large systems down into smaller systems, engineers break large mechanical systems into smaller subsystems, and so on—and it can happen naturally. The invention of the automobile led to the development of the modern system of roads and highways, and this in turn interacted with other systems in our daily lives to produce the suburb. The air-traffic control system interacts with the navigation systems on aircrafts, and the weather prediction system. The human body interacts with other human bodies and with the other systems on the planet. The Internet has intertwined itself with almost every major system in our society.

Third, systems have emergent properties. In other words, they do things that are not anticipated by the users or designers. The telephone system, for example, changed the way people interact. (Alexander Graham Bell had no idea that a telephone was a personal communications device; he thought you could use it to call ahead to warn that a telegram was coming.) Automobiles changed the way people meet, date, and fall in love. Environmental-control systems in buildings have effects on people's health, which affects the health care system. Word processing systems have changed the way people write. The Internet is full of emergent properties; think about eBay, virtual sex, collaborative authoring.

And fourth, systems have bugs. A bug is a particular kind of failure. It's an emergent property of a system, one that is not desirable. It's different from a malfunction. When something malfunctions, it no longer works properly. When something has a bug, it misbehaves in a particular way, possibly unrepeatable, and possibly unexplainable. Bugs are unique to systems. Machines can break, or fail, or not work, but only a system can have a bug.

SYSTEMS AND SECURITY

These properties all have profound effects on the security of systems. Finessing the precise definition of *secure* for now, the reason that it is so hard to secure a complex system like the Internet is, basically, because it's a complex system. Systems are hard to secure, and complex systems are that much more operose.

For computerized systems, the usual coping mechanism is to ignore the system and concentrate on the individual machines . . . the technologies. This is why we have lots of work on security technologies like cryptography, firewalls, public-key infrastructures, and tamper-resistance. These technologies are much easier to understand and to discuss, and much easier to secure. The conceit is that these technologies can mystically imbue the systems with the property of `<reverence type = "hushed"> Security </reverence>`.

This doesn't work, and the results can be seen in my security log from seven days of March 2000. Most of the security events can be traced to one or more of the four system properties previously listed.

Complex. The security problem with Windows 2000's Active Directory can be directly traced to the complexity of any computer-based directory system. This is why I believe it is a design flaw; Microsoft made a design decision that facilitated usability, but hurt security.

Interactive. An interaction between the software on Intuit's Web site and the software that DoubleClick uses to display ads to Web users resulted in information leaking from one to the other.

Emergent. According to the news story, Sony programmers had no idea why credit card information leaked from one user to another. It just happened.

Bug Ridden. The vulnerability in Netscape Enterprise Server 3.6 was caused by a programming bug. An attacker could exploit the bug to cause a security problem.

Many pages of this book (Part 3 in particular) are devoted to explaining in detail why security has to be thought of as a system within larger systems, but I'd like you to keep two things in mind from the beginning.

The first is the relationship between security theory and security practice. There has been a lot of work on security theory: the theory of cryptography, the theory of firewalls and intrusion detection, the theory of biometrics. Lots of systems are fielded with great theory, but fail in practice.

Yogi Berra once said, "In theory there is no difference between theory and practice. In practice there is."

Theory works best in ideal conditions and laboratory settings. A common joke from my college physics class was to "assume a spherical cow of uniform density." We could only make calculations on idealized systems; the real world was much too complicated for the theory. Digital system security is the same way: We can design idealized operating systems that are provably secure, but we can't actually build them to work securely in the real world. The real world involves design trade-offs, unseen variables, and imperfect implementations.

Real-world systems don't lend themselves to theoretical solutions; thinking they do is old-school reductionist. It only works if the spherical cow has the same emergent properties as the real Holstein. It often doesn't, and that's why scientists are not engineers.

The second thing to keep in mind is the relationship between prevention, detection, and reaction. Good security encompasses all three: a

vault to protect the lucre, alarms to detect the burglars trying to open the vault, and police that respond to the alarms and arrest the burglars. Digital security tends to rely wholly on prevention: cryptography, firewalls, and so forth. There's generally no detection, and there's almost never any response or auditing. A prevention-only strategy only works if the prevention mechanisms are perfect; otherwise, someone will figure out how to get around them. Most of the attacks and vulnerabilities listed in this chapter were the result of bypassing the prevention mechanisms. Given this reality, detection and response are essential.

PART 1

THE LANDSCAPE

Computer security is often advertised in the abstract: "This system is secure." A product vendor might say: "This product makes your network secure." Or: "We secure e-commerce." Inevitably, these claims are naïve and simplistic. They look at the security of the product, rather than the security of the system. The first questions to ask are: "Secure from whom?" and "Secure against what?"

They're real questions. Imagine a vendor selling a secure operating system. Is it secure against a hand grenade dropped on top of the CPU? Against someone who positions a video camera directly behind the keyboard and screen? Against someone who infiltrates the company? Probably not; not because the operating system is faulty, but because someone made conscious or unconscious design decisions about what kinds of attacks the operating system was going to prevent (and could possibly prevent) and what kinds of attacks it was going to ignore.

Problems arise when these decisions are made without consideration. And it's not always as palpable as the preceding example. Is a secure telephone secure against a casual listener, a well-funded eavesdropper, or a national intelligence agency? Is a secure banking system secure against consumer fraud, merchant fraud, teller fraud, or bank manager fraud? Does that other product, when used, increase or decrease the security of whatever needs to be secured? Exactly what a particular security technology does, and exactly what it does not do, is just too abstruse for many people.

Security is never black and white, and context matters more than technology. Just because a secure operating system won't protect against hand grenades doesn't mean that it is useless; it just means that we can't throw away our walls and door locks and window bars. Different security technologies have important places in an overall security solution. A system might be secure against the average criminal, or a certain type of industrial spy, or a national intelligence agency with a certain skill set. A system might be secure as long as certain mathematical advances don't