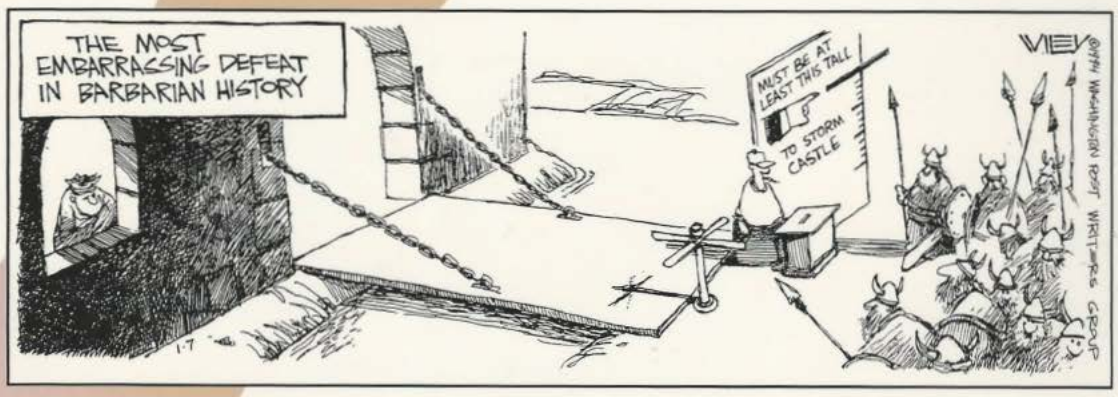
Firewalls and Internet Security

Repelling the Wily Hacker

William R. Cheswick
Steven M. Bellovin



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

RISC/os is a trademark of MIPS Computer Corporation. SunOS is a trademark of Sun Microsystems. RSAREF is a trademark of RSA Data Security, Inc. SPARCstation is a trademark of SPARC International, Inc. PostScript is a registered trademark of Adobe Systems, Inc. Ethernet is a trademark of Xerox Corporation. The X Window System and Kerberos are trademarks of the Massachusetts Institute of Technology. Datakit VCS is a registered trademark of AT&T. UNIX is a technology trademark of X/Open Company, Ltd. S/Key is a trademark of Bellcore.

The publisher offers discounts on this book when ordered in quantity for special sales. For more information please contact:

Corporate & Professional Publishing Group
Addison-Wesley Publishing Company
One Jacob Way
Reading, Massachusetts 01867

Library of Congress Cataloging-in-Publication Data
Cheswick, William R.
Firewalls and Internet Security : repelling the wily hacker / William R. Cheswick, Steven M. Bellovin.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-63357-4
1. Internet (Computer networks) 2. Computer networks--Security measures. I. Bellovin, Steven M. II. Title.
TK5105.875.I57C44 1994
005.8--dc20 94-10747
CIP

Copyright © 1994 by AT&T Bell Laboratories, Inc.

All rights reserved. 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, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

This book was typeset by the authors in 10-point Times-Roman, Michael Urban's Tengwar font, and Joel Hoffman's Hclassic Hebrew font, using LaTeX, a fair amount of hacking, and a plethora of .sty files snarfed from the internet.

ISBN 0-201-63357-4

Text printed on recycled paper.
3 4 5 6 7 8 9 10-CRW-97969594
Third Printing, July 1994

[Bellovin, 1989]. In fact, TCP's three-way handshake at connection establishment time provides more protection than do some other protocols.

2.1.4 UDP

The *User Datagram Protocol* (*UDP*) [Postel, 1980] extends to application programs the same level of service used by IP. Delivery is on a best-effort basis; there is no error correction, retransmission, or lost, duplicated, or re-ordered packet detection. Even error detection is optional with UDP.

To compensate for these disadvantages, there is much less overhead. In particular, there is no connection setup. This makes UDP well suited to query/response applications, where the number of messages exchanged is small compared to the connection setup/teardown costs incurred by TCP.

When UDP is used for large transmissions it tends to behave badly on a network. The protocol itself lacks flow control features, so it can swamp hosts and routers and cause extensive packet loss.

UDP uses the same port number and server conventions as does TCP, but in a separate address space. Similarly, servers usually (but not always) inhabit low-numbered ports. There is no notion of a circuit. All packets destined for a given port number are sent to the same process, regardless of the source address or port number.

3 It is much easier to spoof UDP packets than TCP packets, since there are no handshakes or sequence numbers. Extreme caution is therefore indicated when using the source address from any such packet. Applications that care *must* make their own arrangements for authentication.

2.1.5 ICMP

The *Internet Control Message Protocol* (*ICMP*) [Postel, 1981a] is the low-level mechanism used to influence the behavior of TCP and UDP connections. It can be used to inform hosts of a better route to a destination, to report trouble with a route, or to terminate a connection because of network problems. It also supports the single most important low-level monitoring tool for system and network administrators: the *ping* program [Stevens, 1990].

4 Many ICMP messages received on a given host are specific to a particular connection or are triggered by a packet sent by that machine. In such cases, the IP header and the first 64 bits of the transport header are included in the ICMP message. The intent is to limit the scope of any changes dictated by ICMP. Thus, a `Redirect` message or a `Destination Unreachable` message should be connection-specific. Unfortunately, older ICMP implementations do not use this extra information. When such a message arrives, *all* connections between some pair of hosts will be affected. If you receive a `Destination Unreachable` saying that some packet could not reach host FOO.COM, all connections to FOO.COM will be torn down. This is true even if the original message was triggered by a port-specific firewall filter; it is therefore considered polite for firewalls to refrain from generating ICMP messages that might tear down legitimate calls originating from the same machine. We should also note that some parts of the

hacker community are fond of abusing ICMP to tear down connections; programs to exploit this vulnerability have been captured.

5 Worse things can be done with `Redirect` messages. As explained in the following section, anyone who can tamper with your knowledge of the proper route to a destination can probably penetrate your machine. The `Redirect` messages should be obeyed only by hosts, not routers, and only when the message comes from a router on a directly attached network. However, not all routers (or, in some cases, their administrators) are that careful; it is sometimes possible to abuse ICMP to create new paths to a destination. If that happens, you are in serious trouble indeed.

2.2 Routers and Routing Protocols

> "Rōō′•ting" is what fans do at a football game, what pigs do for truffles under oak trees in the Vaucluse, and what nursery workers intent on propagation do to cuttings from plants. "Rou′•ting" is how one creates a beveled edge on a tabletop or sends a corps of infantrymen into full-scale, disorganized retreat. Either pronunciation is correct for *routing*, which refers to the process of discovering, selecting, and employing paths from one place to another (or to many others) in a network.
>
> *Open Systems Networking: TCP/IP and OSI*
> —DAVID M. PISCITELLO AND A. LYMAN CHAPIN

Routing protocols are mechanisms for the dynamic discovery of the proper paths through the Internet. They are fundamental to the operation of TCP/IP. Routing information establishes two paths: from the calling machine to the destination and back. (The second path is usually the reverse of the first. When they aren't, it is called an *asymmetric route* and is generally not a good thing.) From a security perspective, it is the return path that is often more important. When a target machine is attacked, what path do the reverse-flowing packets take to the attacking host? If the enemy can somehow subvert the routing mechanisms, then the target can be fooled into believing that the enemy's machine is really a trusted machine. If that happens, authentication mechanisms that rely on source address verification will fail.

6 There are a number of ways to attack the standard routing facilities. The easiest is to employ the IP *loose source route* option. With it, the person initiating a TCP connection can specify an explicit path to the destination, overriding the usual route selection process. According to RFC 1122 [Braden, 1989b], the destination machine must use the inverse of that path as the return route, whether or not it makes any sense, which in turn means that an attacker can impersonate any machine that the target trusts.

The easiest way to defend against source routing problems is to reject packets containing the option. Many routers provide this facility. Source routing is rarely used for legitimate reasons, although those do exist. For example, it can be used for debugging certain network problems. You will do yourself little harm by disabling it. Alternatively, some versions of *rlogind* and *rshd*

Here the remote site, A.SOME.EDU, is transferring mail to the local machine, INET.ATT.COM. It is a simple protocol. Postmasters and hackers learn these commands and occasionally type them by hand.

10 Notice that the caller specified a return address in the "MAIL FROM" command. At this level there is no reliable way for the local machine to verify the return address. *You do not know for sure who sent you mail based on SMTP.* You must use some higher level mechanism if you need trust or privacy.

An organization needs at least one mail guru. It helps to concentrate the mailer expertise at a gateway, even if the inside networks are fully connected to the Internet. Then administrators on the inside need only get their mail to the gateway mailer. The gateway can ensure that outgoing mail headers conform to standards. The organization becomes a better network citizen when there is a single, knowledgeable contact for reporting mailer problems.

The mail gateway is also an excellent place for corporate mail aliases for every person in a company. (When appropriate, such lists must be guarded carefully: they are tempting targets for industrial espionage.)

From a security standpoint, the basic SMTP by itself is fairly innocuous. It could, however, be the source of a *denial-of-service* attack, an attack that's aimed at preventing legitimate use of the machine. Suppose I arrange to have 50 machines each mail you 1000 1-MB mail messages. Can your systems handle it? Can they handle the load? Is the spool directory large enough?

The mail aliases can provide the hacker with some useful information. Commands such as

```
VRFY <postmaster>
VRFY <root>
```

often translate the mail alias to the actual login name. This can give clues about who the system administrator is and which accounts might be most profitable if successfully attacked. It's a matter of policy whether this information is sensitive or not. The *finger* service, discussed later, can provide much more information.

The EXPN subcommand expands a mailing list alias; this is problematic because it can lead to a loss of confidentiality. A useful technique is to have the alias on the well-known machine point to an inside machine, not reachable from the outside so that the expansion can be done there without risk.

11 The most common implementation of SMTP is contained in *sendmail* [Costales, 1993]. This program is included in most UNIX software distributions, but you get less than you pay for. *Sendmail* is a security nightmare. It consists of tens of thousands of lines of C and often runs as *root*. It is not surprising that this violation of the principle of *minimal trust* has a long and infamous history of intentional and unintended security holes. It contained one of the holes used by the Internet Worm [Spafford, 1989a, 1989b; Eichin and Rochlis, 1989; Rochlis and Eichin, 1989] and was mentioned in a *New York Times* article [Markoff, 1989]. Privileged programs should be as small and modular as possible. An SMTP daemon does not need to run as *root*.

For most gatekeepers, the big problem is configuration. The *sendmail* configuration rules are infamously obtuse, spawning a number of useful how-to books such as [Costales, 1993] and [Avolio and Vixie, 1994]. And even when a mailer's rewrite rules are relatively easy, as in the

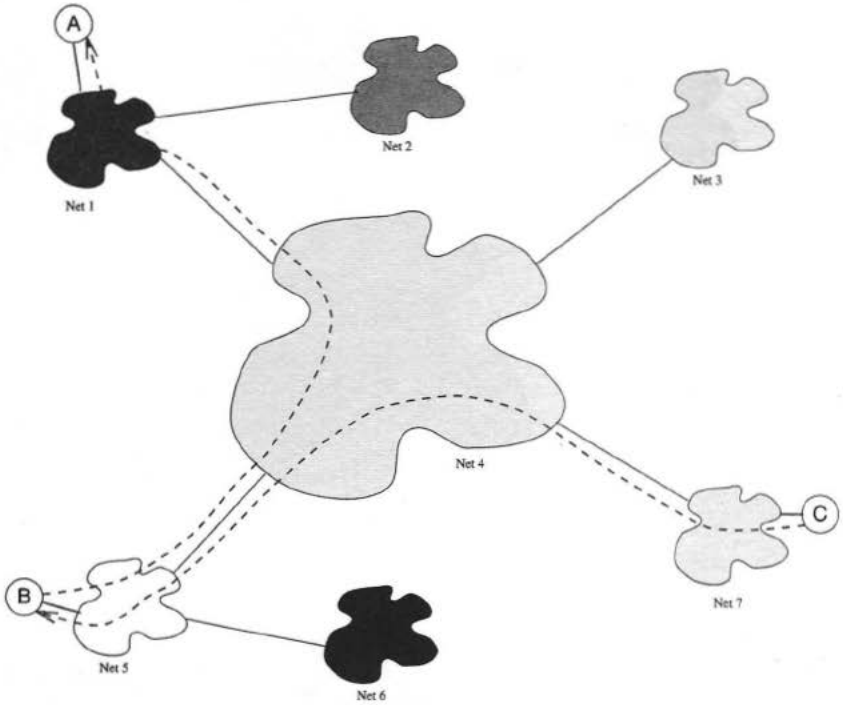


Figure 3.3: An example of transitive trust.

users should pass through a security gateway when crossing the firewall; otherwise, if their home machines, which live outside of the firewall, are compromised, the sensitive equipment on the inside could be next. The firewall controls the access and the trust in a carefully predictable way.

Transitive trust may also be an issue. In Figure 3.3 suppose that machine A, in full accord with local security policy, decides to extend trust to machine B. Similarly, machine B decides to trust machine C, again in accordance with its own policies. The result, though, is that machine A is now trusting machine C, whether it wants to or not, and whether it knows it or not. A firewall will prevent this. The diodes from Figure 3.2 would prevent machine A from trusting machine B, or machine B from trusting machine C. Again, trust can be controlled through a firewall. But machine C could, if it wished, trust machine B.

3.3 Packet-Filtering Gateways

Packet filters can provide a cheap and useful level of gateway security. Used by themselves, they are cheap: the filtering abilities come with the router software. Since you probably need a router

to connect to the Internet in the first place, there is no extra charge. Even if the router belongs to your network service provider, you'll probably find that they'll install any filters you wish.

Packet filters work by dropping packets based on their source or destination addresses or ports. In general, no context is kept; decisions are made only from the contents of the current packet. Depending on the type of router, filtering may be done at input time, at output time, or both. The administrator makes a list of the acceptable machines and services and a stoplist of unacceptable machines or services. It is easy to permit or deny access at the host or network level with a packet filter. For example, one can permit any IP access between host A and B, or deny any access to B from any machine but A.

Most security policies require finer control than this: they need to define access to specific services for hosts that are otherwise untrusted. For example, one might want to allow any host to connect to machine A, but only to send or receive mail. Other services may or may not be permitted. Packet filtering allows some control at this level, but it is a dangerous and error-prone process. To do it right, one needs intimate knowledge of TCP and UDP port utilization on a number of operating systems.

> *This is one of the reasons we do not like packet filters very much: if you get these tables wrong you may inadvertently let in the Bad Guys [Chapman, 1992].*

Even with a perfectly implemented filter, some compromises can be dangerous. We discuss these later.

Configuring a packet filter is a three-step process. First, of course, one must know what should and should not be permitted. That is, one must have a security policy, as explained in Section 1.2. Next, the allowable types of packets must be specified formally, in terms of logical expressions on packet fields. Finally—and this can be remarkably difficult—the expressions must be rewritten in whatever syntax your vendor supports.

An example is helpful. Suppose that one part of your security policy was to allow inbound mail (SMTP, port 25), but only to your gateway machine. However, mail from some particular site SPIGOT is to be blocked, because of their penchant for trying to mail several gigabytes of data at a time. A filter that implemented such a ruleset might look like this.

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	*we don't trust these people*
allow	OUR-GW	25	*	*	*connection to our SMTP port*

The rules are applied in order from top to bottom. Packets not explicitly allowed by a filter rule are rejected. That is, every ruleset is followed by an implicit rule reading like this.

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	*default*

This fits with our general philosophy: all that is not expressly permitted is prohibited.

Note carefully the distinction between the first ruleset, and the one following, which is intended to implement the policy "any inside host can send mail to the outside."

from an outgoing call. But UDP has no such indicator: we are forced to rely on the source port number, which is subject to forgery.

An example will illustrate the problem. Suppose an internal host wishes to query the UDP *echo* server on some outside machine. The originating packet would carry the address

$$\langle \textit{localhost, localport, remotehost, 7} \rangle,$$

where *localport* is in the high-numbered range. But the reply would be

$$\langle \textit{remotehost, 7, localhost, localport} \rangle,$$

and the router would have no idea that *localport* was really a safe destination. An incoming packet

$$\langle \textit{remotehost, 7, localhost, 2049} \rangle,$$

is probably an attempt to subvert our NFS server; and, while we could list the known dangerous destinations, we do not know what new targets will be added next week by a system administrator in the remote corners of our network. Worse yet, the RPC-based services use dynamic port numbers, sometimes in the high-numbered range. As with TCP, indirectly named services are not amenable to protection by packet filters.

A conservative stance therefore dictates that we ban virtually all *outgoing* UDP calls. It is not that the requests themselves are dangerous; rather, it is that we cannot trust the responses. The only exceptions are those protocols where there is a peer-to-peer relationship. A good example is NTP, the Network Time Protocol. In normal operation, messages are both from and to port 123. It is thus easy to admit replies, because they are to a fixed port number, rather than to an anonymous high-numbered port. But one use of NTP—setting the clock when rebooting—will not work, because the client program will not use port 123. (Of course, a booting computer probably shouldn't ask an outsider for the time.)

3.3.9 Filtering Other Protocols

Other protocols are layered on top of IP as well; depending on your environment, these may need to be filtered also. Of particular import is ICMP, the Internet Control Message Protocol. There have been instances of hackers abusing it for denial-of-service attacks. On the other hand, filtering out ICMP denies one useful information. At the very least, internal management hosts should be allowed to receive such messages so that they can perform network diagnostic functions. For example, *traceroute* relies on the receipt of `Time Exceeded` and `Port Invalid` packets.

Some routers can distinguish between "safe" and "unsafe" ICMP messages, or permit the filter to specify the message types explicitly. This lets more of your machines send and respond to things like *ping* requests. On the other hand, it lets an outsider map your network.

Most of the other higher level protocols are not important in most environments. Still, if you do use others, the same care needs to be taken as for TCP and UDP. One such protocol of growing importance is the IP-over-IP protocol used by the MBone; if you are not careful, it can be used to bypass your firewall. Router filter lists should also be configured to reject all unneeded protocols.

3.3.14 Summary

Many advanced gateway designs rely in part on packet filtering. They are likely to work well, but pure packet filters leave us feeling uncomfortable. Some of these designs become ineffective if a vendor software problem compromises the packet filter. We have heard of at least two such software problems to date,[1] although the vendors are very careful about such things. Worse yet, it takes either a conservative stance or a great deal of knowledge about the Internet to design an effective packet filter. Also, there are highly desirable services that cannot be implemented in a pure packet-filtering environment.

We are inclined to place our trust in a simpler design. Packet filters are a useful tool, but they do not leave us with confidence in their correctness and hence their safety.

3.4 Application-Level Gateways

An application-level gateway represents the opposite extreme in firewall design. Rather than using a general-purpose mechanism to allow many different kinds of traffic to flow, special-purpose code can be used for each desired application. Although this seems wasteful, it is likely to be far more secure than any of the alternatives. One need not worry about interactions among different sets of filter rules, nor about holes in thousands of hosts offering nominally secure services to the outside. Only a chosen few programs need be scrutinized.

Application gateways have another advantage that in some environments is quite critical: it is easy to log and control *all* incoming and outgoing traffic. The SEAL package [Ranum, 1992] from Digital Equipment Corporation takes advantage of this. Outbound FTP traffic is restricted to authorized individuals, and the effective bandwidth is limited. The intent is to prevent theft of valuable company programs and data. While of limited utility against insiders, who could easily dump the desired files to tapes or floppies, it is a powerful weapon against electronic intruders who lack physical access.

Electronic mail is often passed through an application-level gateway, regardless of what technology is chosen for the rest of the firewall. Indeed, mail gateways are valuable for their other properties, even without a firewall. Users can keep the same address, regardless of which machine they are using at the time. This book, for example, was composed on no fewer than six different computers, but mail sent from any of them would bear a return address of RESEARCH.ATT.COM. The gateway machines also worry about mail header formats and logging (mail logging is a postmaster's friend) and provide a centralized point for monitoring the behavior of the electronic mail system.

It is equally valuable to route incoming mail through a gateway. One person can be aware of all internal connectivity problems, rather than leaving it to hundreds of random system administrators around the Internet. Reasonably constant mail addresses—*Firstname.Lastname*@ORG.DOMAIN is popular—can be accepted and processed. Different technologies, such as *uucp*, can be used to deliver mail internally. Indeed, the need for incoming mail gateways is so obvious that the DNS

[1] See CERT Advisory CA-92:20, December 10, 1992, and CERT Advisory CA-93:07, April 22, 1993.

9.6 Information Leakage

Most protocols give away some information. Often, that is the intent of the person using those services: to gather such information. Welcome to the world of computer spying. The information itself could be the target of commercial espionage agents or it could be desired as an aid to a break-in. The *finger* protocol is one obvious example, of course; apart from its value to a password-guesser, the information can be used for social engineering. ("Hey, Robin—the battery on my hand-held authenticator died out here in East Podunk; I had to borrow an account to send this note. Could you send me the keying information for it?" "Sure, no problem; I knew you were traveling. Thanks for posting your schedule.")

Even such mundane information as phone and office numbers can be helpful. Woodward and Bernstein used a *Committee to Re-Elect the President* phone book to deduce its organizational structure [Woodward and Bernstein, 1974]. If you're in doubt about what information can be released, check with your corporate security office; they're in the business of saying "no."

In a similar vein, some sites offer access to an online phone book. Such things are convenient, of course, but in the corporate world, they're often considered sensitive. Headhunters love such things; they find them useful when trying to recruit people with particular skills. Nor is such information entirely benign at universities; privacy considerations (and often legal strictures) dictate some care about what information can be released.

Another fruitful source of data is the DNS. We have already described the wealth of data that can be gathered from it, ranging from organizational details to target lists. But controlling the outflow is hard; often, the only solution is to limit the externally visible DNS to list gateway machines only.

Sophisticated hackers know this, of course, and don't take you at your word about what machines exist. They do address space and port number scans, looking for hidden hosts and interesting services. The best defense here is a good firewall; if they can't send packets to a machine, it's much less likely to be penetrated.

9.7 Denial-of-Service

Some people like to slash car tires or deface walls. Others like to crash other people's computer systems. Vandalism—wanton destructive behavior, for its own sake—has been with us for millennia (were those cave paintings really Neanderthal graffiti?); the emergence of new technologies has simply provided new venues for its devotees. Computer networks are no exception. Thus, there are individuals who use them only to annoy.

Such behavior takes many forms. The crudest and easiest form is to try to fill up someone's disks, by mailing or using FTP to send a few hundred megabytes. It's hard to set an absolute upper bound on resource consumption. Apart from the needs of legitimate power users, it's just too easy to send 1 MB a few hundred times instead. Besides, that creates lots of receiving processes on your machine, tying it up still further. The best you can do is to provide sufficient resources to handle just about anything (disk space costs much less than a dollar a megabyte these days), in the right spots (i.e., separate areas for mail, FTP, and especially precious log data), and to make

provisions for graceful failure. A mailer that cannot accept and queue an entire incoming mail job should indicate that to the sender. It should not give an "all clear" response until it knows that the message is safely squirreled away.

Other forms of computer vandalism are more subtle. Some folks delight in sending bogus ICMP packets to a site, to disrupt its communications. Sometimes these are `Destination Unreachable` messages; sometimes they are the more confusing—and more deadly—messages that reset the host's subnet mask. (And why, pray tell, do hosts listen to such messages when they've sent no such inquiry?) Other hackers play games with routing protocols, not to penetrate a machine, but to deny it the ability to communicate with its peers.

Aggressive filtering can do a lot to protect you, but there are no absolute guarantees; it can be very hard to tell the difference between genuine messages, ordinary failures, and enemy action.

Networking/Security

Firewalls and Internet Security

"I wish that Cheswick and Bellovin had written this six years ago. I could have nailed down my system and avoided chasing a spy around the world."
—Cliff Stoll, author of *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*

As a user of the Internet, you are fortunate to be tied into the world's greatest communication and information exchange — but not without a price. As a result of this connection, your computer, your organization's network, and everywhere that network reaches are all vulnerable to potentially disastrous infiltration by hackers.

Written by the AT&T Bell Labs researchers who tracked the infamous "Berferd" hacker and also built the firewall gateway at Bell Labs, *Firewalls and Internet Security* gives you invaluable advice and practical tools for protecting your organization's computers from the very real threat of a hacker attack through the Internet. You will learn how to plan and execute a security strategy that will thwart the most determined and sophisticated of hackers — while still allowing you easy access to Internet services.

In particular, the authors show you a step-by-step plan for setting up a "firewall" gateway — a dedicated computer equipped with safeguards that acts as a single, more easily defended, Internet connection. They even include a description of their most recent gateway, the tools they used to build it, and the hacker attacks they devised to test it.

You wil
mented
ed havo
informat
the legal

With thi
zation w

* Quantum Books *
Phone: 617-494-5042
4 Cambridge Center * Cambridge, MA 02142
e mail: quanbook@world.std.com
FIREWALLS INTERNET SECURITY 202
CHESWICK INTERNET ADDISO
$26.95 Special* $24.95 *** NC
0201633574

t docu-
s creat-
nd vital
ers, and

organi-

William R. Cheswick and **Steven M. Bellovin** are both senior researchers at AT&T Bell Laboratories, where they have designed and maintain AT&T's Internet gateway. They have published numerous papers in the field, which have established them as experts on the subject.

Cover illustration by Wiley Miller
♻ Text printed on recycled paper
Corporate & Professional Publishing Group
Addison-Wesley Publishing Company

90000
9 780201 633573
ISBN 0-201-63357-4