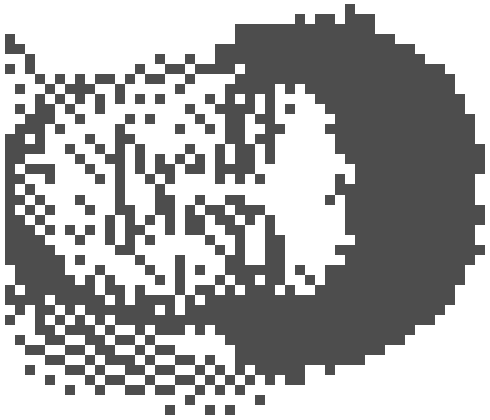
ches Fri Apr 20 07:46:11 EDT 1990



The Design of a Secure Internet Gateway

Bill Cheswick
ches@research.att.com

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The Internet supports a vast and growing community of computers users around the world. Unfortunately, this network can provide anonymous access to this community by the unscrupulous, careless, or dangerous. On any given Internet there is a certain percentage of poorly-maintained systems. AT&T has a large internal Internet that we wish to protect from outside attacks, while providing useful services between the two.

This paper describes our Internet gateway. It is an application-level gateway that passes mail and many of the common Internet services between our internal machines and the Internet. This is accomplished without IP connectivity using a pair of machines: a trusted internal machine and an untrusted external gateway. These are connected by a private link. The internal machine provides a few carefully-guarded services to the external gateway. This configuration helps protect the internal internet even if the external machine is fully compromised.

1. Introduction

The design of a Corporate gateway to the Internet must deal with the classical tradeoff between security and convenience. Most institutions opt for convenience and use a simple router between their internal internets and the rest of the world. This is dangerous. Strangers on the Internet can reach and test every internal machine. With workstations sitting on many desks, system administration is often decentralized and neglected. Passwords are weak or missing. A professor or researcher often may install the operating system and forget it, leaving well-known security holes uncorrected. For example, a sweep of 1,300 machines inside Bell Labs around the time of the Internet Worm found over 300 that had at least one of several known security holes.

When we first obtained a connection to the ARPAnet, Dave Presotto configured our gateway machine (named) as an application-level gateway. For two years this machine was the sole official link to the Internet for AT&T. Until its disconnection a little while ago, this VAX 750 handled all the Internet mail traffic and other services for the company. had Ethernet connections to both the inside and outside Internets, just like a router. It could also make and accept calls on our corporate Datakit network.

Dave took a number of steps to make our gateway more secure. He turned off IP forwarding in the kernel so packets could not travel between the Internets. He installed a kernel modification that limited TCP connections from to the inside network to *smtp*, *uucp*, *named*, and *hostname* ports. And he rejected the *sendmail* mailer as too complicated and dangerous: the Upas[1] mailer was installed in its place. We removed a number of non-essential daemons, including the *finger* server.

To give insiders access to the Internet, a *gate* service was installed on . Insiders could call this service and supply an Internet address. The gate connected to a socket of a remote Internet host and then copied bytes between the two connections. It was easy to provide *atelnet*, a version of *telnet* that used the gate service. *Aftp* supplied FTP services: it was the standard FTP modified so both the command and data connections were initiated from the inside. (The standard *ftp* would have tried to make the data connection from to the inside, a connection prohibited by 's kernel.)

This configuration successfully resisted the Internet worm. We ran neither *sendmail* nor *fingerd*, the two programs exploited by the worm.[2] The internal internet was spared the infection. (Actually, there was a sec-

ond, unguarded IP link to the Outside. We got lucky: only a few machines at the other end knew of the link, and their machines were shut down before the worm could creep across.)

Had been infected, the worm could have reached the inside machines. The initial *smtp sendmail* connection was permitted, and the worm's second connection would have been initiated from the inside target machine into , the permitted direction.

2. The new gateway

All of 's protection has, by design, left the internal AT&T machines untested—a sort of crunchy shell around a soft, chewy center. We run security scans on internal machines and bother system administrators when holes are found. Still, it would be nice to have a gateway that is demonstrably secure to protect the internal machines. For peace of mind, the gateway design should not rely on vendors' code more than absolutely necessary. We would like the internal machines protected even if an invader breaks into the gateway machine, becomes root, and creates and runs a new kernel.

We had to replace . The VAX 750 ran with typical load averages of seven to twelve jobs throughout the day. When the load average hit about fifteen, the old Datakit driver expired, wedging the Datakit ports and requiring a reboot.

A new machine gave the opportunity for a clean start. We could re-think the security arrangements to improve on 's shortcomings.

Our new gateway machine, named , is a MIPS M/120 running System V with Berkeley enhancements. Various daemons and critical programs have been obtained from other sources, checked, and installed.

We store nothing vital or secret on , since we assume that it may be defeated in unforeseen ways. It does not currently run *uucp*—systems files and dialers could fall into the wrong hands. There are few system administration accounts, and user accounts are discouraged. is not used for other tasks. It is backed up regularly, and scanned for unauthorized changes and common system administration mistakes. Though we don't trust , we protect it as much as we can.

has a single Ethernet port which is connected to a router on JVNCnet, our external regional network. It also has a connection to Datakit. We have configured our Datakit controller to force all connections from to a single internal machine, named . can redial, or splice connections to other internal machines. provides a limited set of services to for reaching internal machines. The list of services are:

1. connection to an approved machine's *smtp* port,
2. connection to a login or trusted-login Datakit destination after passing a challenge-response test, and
3. connection to a logging service.

The key to the arrangement is a restricted channel from to . This private channel was easily constructed using stock features of our research Datakit controller. Other connection schemes could be implemented using a simple multiplexed protocol over some back-to-back connection between the machines, or a simple Ethernet would suffice. If the last approach is used with TCP, the internal machine must supply differing TCP services to its two Ethernet interfaces. (I am not sure this is possible with standard TCP/IP implementations. It wouldn't be too hard to modify *inetd* to do this.)

These functions do not load the internal machine too much; it could have other uses like *uucp*, *mail*, or even normal user jobs. But the services it provides the external machine are the key to security, and must be protected well.

3. Outbound services

It is quite easy to implement most outbound services to the Internet. has a small program, named *proxy* (a descendant of 's *gate*), that makes calls to the Internet on behalf of an inside machine and relays bytes between the inside Datakit connection and the outside Internet TCP connection. *Proxy* can also listen to a non-privileged socket and report connections to an inside process. Several outbound services are implemented using *proxy*, and more are easy to create. In all cases, it appears to the remote Internet hosts that our gateway machine is making the calls.

may be reached over the Datakit. But how do internal machines reach over the Ethernet? responds to two IP addresses: its own, and an internal IP address for . (Dave Presotto implemented this after a trivial change to the Tenth Edition Research Unix connection server.[3]) Calls to certain TCP ports on this internal IP address invoke *dcon*, a program that simply relays the bytes between the TCP port and Datakit connections on .

I have replaced the old *aftp* and *atelnet* with *ptelnet* and *pftp*. They work in the same manner, but the

new routines call a portable implementation of *ipcopen*, a piece of the connection server. *Ipcopen* hides the details of a connection (TCP sockets or Datakit), simplifying the application program. For example:

```
ptelnet tcp!toucan
```

connects to machine on our internet, and

```
ptelnet proxy!ernie.berkeley.edu
```

connects to on the external Internet. `proxy!` is the default. The *ipcopen* implementation is not flawless: some socket features such as out-of-band data and the urgent pointer are missing because they are not supported by Datakit. *Ptelnet* was stripped down to avoid these features.

Pftp provides FTP access in a similar manner. It is an updated version of *aftp* from . The *ipcopen* routines allow it to work over Datakit.

Outgoing mail is sent to via *smtp* over either Datakit or the internal Internet. It is stored and forwarded from there. Upas performs the mail gateway functions.

4. Inbound services

We provide incoming login and mail service. For incoming file transfer, provides an anonymous FTP service.

We do not trust our passwords to the Internet: it is too easy to eavesdrop or steal packets. See [4] for a discussion of these security problems. Login service requires a hand-held authenticator (HHA). These are calculator-sized devices that contain DES encryption and a manually-loaded 64-bit key. They cost about $50.

Inbound login service is provided through an authentication manager on . A session is shown in figure 1. To connect, the following sequence occurs:

- The Internet caller uses *telnet* to connect to (a.k.a) via *telnet*. The login name is `guard`.
- The `guard` login connects to the authentication manager on over the Datakit. It spends the rest of the connection relaying bytes between the two connections.
- The authentication manager on requests a login name.
- sends a random challenge number, which the caller supplies.
- The user enters the challenge into his HHA.
- The HHA encrypts the challenge using a preloaded DES key, and displays the response.
- The user types the response. He has three tries to answer a challenge correctly, and is disconnected if he fails.
- The authorization manager prompts for a Datakit destination.
- When the user enters the destination, the manager sends a redial request to the Datakit controller with the given destination and a service of 'dcon'. For machines that trust , the 'dcon' service bypasses further logins and avoids further passwords.
- The redial request transfers the call, switching out of the connection. In non-Datakit implementations, would probably have shuttle bytes between the two connections.

Each user requires a DES key, and keys have an expiration date. The keys are stored on a separate passwd/key server machine connected to . The keys in this machine may be changed or examined only from its console.

Inbound mail is delivered directly to . checks the destination. If it is a trusted machine (i.e. its *smtp* is trusted), a connection request is sent to . If not, the mail is relayed through an accessible internal machine. will permit connections only to trusted *smtp* implementations. The list is short because most internal machines run *sendmail*.

5. Protecting INET

The preceding precautions might imply that we expect our gateway to be compromised at some point. In fact, we are taking great pains to protect the machine, including the usual good system administration steps needed to secure any system[5]: directory and file permissions are checked, backups performed regularly, etc.

We have taken some steps to avoid denial-of-service attacks. For example, the logs, the spool directory, and the publically-accessible FTP directory are each on separate file systems. If a stranger fills the public FTP directory, there is still room for the logs.

Here are some other steps taken:

```
$ telnet research.att.com
Trying...
Connected to research.att.com.
Escape character is '^]'.


RISC/os (inet)

login: guard
RISC/os (UMIPS) 4.0 inet
Copyright 1986, MIPS Computer Systems
All Rights Reserved
Security Authentication check

login: ches
Enter response code for 90902479: 818b71fe

Destination please: coma
OKYou have mail.
coma=; date
Tue Nov 14 10:52:37 EST 1989
coma=;
Eof
Connection closed by foreign host.
$
```

Figure 1: A connection session through the guard.

- All the important executable files are periodically checksummed and checked for changes.
- Most user accounts do not have passwords to be checked. They obtain permission to login based on the source of the call.
- Non-essential network daemons have been removed: we don't need to trust them.
- *Inetd(8)* handles all network connections. Certain modifications allow *telnetd*, *smtpd*, and *ftpd* to run without special permissions:[5] *inetd* handles the privileged stuff.
- There is extensive logging of network activity, including connection and login attempts. A write-only log server is planned that will keep a copy of these logs off-machine and inaccessible to any network.
- Since the network daemons are so important to the security of the machine, we obtained the latest BSD versions and examined, modified, and installed them.

6. Gateway alternatives

There are several much simpler alternatives for an Internet gateway. The simplest is a router, which just lets the packets through. Some routers, like Cisco's, provide packet filtering that can block various types of access to an institution.

We did not choose the router. Though the filtering is quite good, it's not clear whether a clever worm could get through the permitted ports. Can we trust the router? If *telnet* access is allowed from the outside, inside machines are exposed to password-guessing attacks. If *telnet* access is not allowed, an alternative is needed anyway, requiring additional provisions. The router does not provide logging to detect invasion attempts. And mail gating must be provided by a machine somewhere: it is unreasonable to expect each internal machine to be configured to handle all the varieties of external mail addressing.

Many Internet sites use a gateway machine like a Sun. These machines forward IP packets in both directions, and provide a mail gateway service. The packet flow is still dangerous, though filtering is available. Many internal machines may trust the gate machine, leaving them further exposed if the gate machine

is compromised.

7. Performance

The mail throughput of the new gateway has been gratifying, though a VAX 750 is an easy act to follow. In many cases, we have had replies to cross-country mail return in less than a minute. It sometimes seems that the mail must have bounced. has little else to do, and a MIPS M/120 is a fast machine.

Pftp transfers are fastest over Datakit, since they avoid the *dcon* gateway in . File transfers range from 17 to 44 Kb/sec. TCP transfers through run at 9 to 16 Kb/sec. By comparison, *ftp* on runs at about 60–90 Kb/sec. Clearly, security has its costs. But these are top speeds. The limiting factor is often the external net or host. In any case, several users have expressed satisfaction about the throughput.

8. Conclusions

The new gateway achieves a useful balance of utility and security. Most internal users seem to be happy with *pftp* and *ptelnet*. Some have asked for *talk*, resolver service and other UDP-based protocols. These could be provided with non-*proxy* services on accessible through Datakit.

There are certainly limits to our security. If and are subverted, the inside machines could be attacked.

Insiders can easily import trouble such as Trojan horses or programs infected with viruses. Our best defense is continued scanning of internal machines for security holes in case such a program gets loose.

There is now a second AT&T internet gateway. Its configuration is similar to 's. These two front doors provide reasonable security to an isolated internal internet. But AT&T is a large company, so we keep a constant watch to assure that no other links are made to the external Internet. A locked front door is useless if the back wall of the house is missing.

The incoming guarded *telnet* service is not perfect. The remote *telnet* may be insecure, and the TCP connection itself could be stolen after login is complete. Most internal AT&T machines do not accept 's judgement that the user is valid, and require their own login passwords. These passwords travel over the Internet in the clear.

Our solution does have some drawbacks. We rely on two machines and Datakit to keep things working. This yields three points of failure, while the simpler approaches have (in some sense) only one point of failure. The use of TCP-level gateways does lower throughput. Though most users seem to be content with the *pftp* response, it would be nice to speed it up some.

This paper is not an invitation to come test the security of our gateway. It is management's policy to call the authorities when intruders are detected.

9. Acknowledgements

Many people have contributed to the support of these gateways. Steve Bellovin did most of the initial work to get talking to the ARPAnet and Datakit. Dave Presotto has supplied much of the software and most of the paranoia to provide a secure gateway. Howard Trickey implemented earlier versions of *ptelnet* and *pftp*. Dennis Ritchie has kept a watchful eye and stepped in when things broke. Steve Bellovin and others have provided numerous suggestions and warnings on various networking and security topics. Jim McKie ported many useful Research Unix[6] functions and the INCON Datakit driver to our MIPS computers, making life much easier for me.

References

[1] David Presotto. *Upas - a simpler approach to network mail*. USENIX Summer Conference Proceedings, pps. 533–538, June 1985.

[2] Donn Seeley. *A Tour of the Worm*. USENIX Winter Conference Proceedings, Jan. 1989.

[3] David Presotto and Dennis Ritchie. *Interprocess Communication in the Ninth Edition UNIX System*. Unix Programmer's Manual, Tenth Edition. A. G. Hume and M. D. McIlroy, Editors. AT&T Bell Laboratories, Murray Hill, NJ. 1990.

[4] Bellovin, S.M. *Security Problems in the TCP/IP Protocol Suite*. Computer Communications Review, Vol. 9, No. 2; April, 1989, pps. 32–48.

[5] Dennis M. Ritchie. *On the Security of UNIX*. Unix Programmer's Manual, Tenth Edition. A. G. Hume and M. D. McIlroy, Editors. AT&T Bell Laboratories, Murray Hill, NJ. 1990.

[6] Unix Programmer's Manual, Tenth Edition, Volumes One and Two. A. G. Hume and M. D. McIlroy, Editors. AT&T Bell Laboratories, Murray Hill, NJ. 1990.