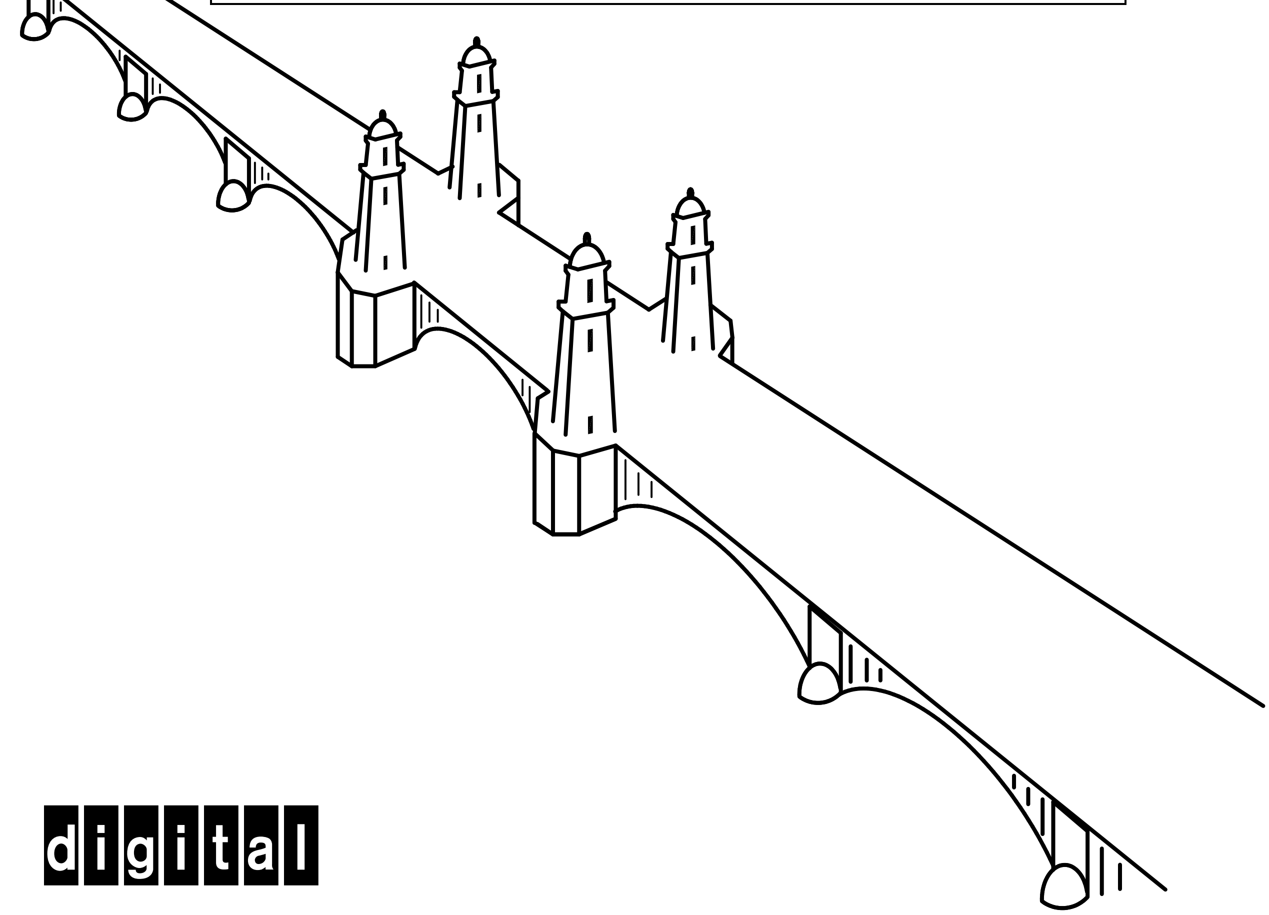
X Through the Firewall, and Other Application Relays

G. Winfield Treese Alec Wolman

Digital Equipment Corporation
Cambridge Research Lab

CRL 93/10 3 May 1993



CAMBRIDGE RESEARCH LABORATORY
Technical Report Series

X Through the Firewall, and Other Application Relays

G. Winfield Treese[1] Alec Wolman[2]

Digital Equipment Corporation
Cambridge Research Lab

CRL 93/10 3 May 1993

Abstract

Organizations often impose an administrative security policy when they connect to other organizations on a public network such as the Internet. Many applications have their own notions of security, or they simply rely on the security of the underlying protocols. Using the X Window System as a case study, we describe some techniques for building application-specific "relays" that allow the use of applications across organizational boundaries. In particular, we focus on analyzing administrative and application-specific security policies to construct solutions that satisfy the security requirements while providing the necessary functions of the applications.

This is a preprint of a paper to appear in the Proceedings of the USENIX Summer Conference, June, 1993.

[1]Also with the MIT Laboratory for Computer Science.

[2]Currently at the University of Washington, on leave from Digital Equipment Corporation Cambridge Research Lab.

1 Introduction

This paper presents some general techniques for making network applications available through secure gateways, based on experience from managing an Internet gateway at Digital Equipment Corporation's Cambridge Research Laboratory (CRL). Like many organizations, Digital operates firewalls at its Internet gateways to isolate the internal network from the rest of the Internet. This isolation is only partial, because communication is the goal of the connection in the first place. Unfortunately, the Internet is not always a safe place, and some form of protection is necessary.

A firewall (sometimes called a "gateway"), such as those described by Cheswick [2], Ranum [11], or Schauer and Wolfhugel [13], is a collection of computers intended to protect an organization connected to a public network. The fundamental premise in the design of firewalls is that it is easier to secure a small number of systems rather than hundreds or thousands. A firewall isolates insecure systems inside the organization from the public network. We discuss the design of traditional firewalls in more detail below.

At CRL, we began with a traditional firewall system. CRL operates one of Digital's connections to the Internet, and the firewall has been in place since that connection was established. At the outset, each person who needed access to the Internet also needed an account on a trusted firewall system. The only exceptions to this rule were those users who only needed to use electronic mail and USENET news, which were forwarded appropriately by a trusted gateway operated as part of the firewall system. Assigning user accounts does not scale very well, however; the administrative costs alone can be prohibitive. Because of this, only a small number of people were allowed to have accounts.

In order to provide some limited Internet access to more people, we began implementing "relays" for various applications, such as FTP, Telnet, and X. A "relay" is an intermediary program, specific to a given application, that permits users on the inside of the firewall to use services available on the public network. The relays run on trusted firewall systems.

The design of application relays is not merely a technical problem. An organization may have an "administrative security policy" that makes some statements about what is allowed and what is not. Relays must respect these policies. The challenge is designing a relay that both provides the needed application functions and satisfies the requirements of the security policy.

In this paper, we discuss some general principles for attacking the problem of designing relays to satisfy both technical and policy requirements. We use our experience with constructing a relay for the X Window System [14] as a case study.

After some brief definitions, we begin with discussions of traditional firewall design and administrative security policies. Next we describe our approach to building the relay for X. This is followed by short descriptions of some other example applications. We compare this approach to some others from the research community and conclude with an evaluation of our experiences and some discussion of possible future work.

2 Definitions

In the sections that follow, an "internal" network is one inside an organization connected to the Internet, and an "external" network is the Internet. Note that "secure" as used here means some reasonable level of security, not an absolute high level. The goal of our work has been to extend services without sacrificing a significant portion of the security that already exists, whatever that might be for a given application.

We use the term "router" to refer to a device that operates on packets at the network layer. A "gateway" operates at the application layer (which may encompass the OSI [18] session, presentation, and application layers).

3 Traditional Firewall Design

A typical firewall configuration may consist of several computer systems. These include routers used to separate the internal and external networks as well as secure hosts that may be used for interactive use. A diagram of a typical firewall configuration is shown in Figure 1.

The routers used in the firewall are often capable of "screening" packets to select desired ones and discard undesired ones. Screening routers can be used to permit direct access between internal machines and external ones. One screening router implementation for UNIX systems is described by Mogul [7] (although not all of these have the flexibility of the one Mogul describes). Similar capabilities are now available in many commercial routers. Such systems allow a system manager to select packets based on some combination of protocol, source and destination IP address, and source and destination port numbers.

Some uses of screening routers include:

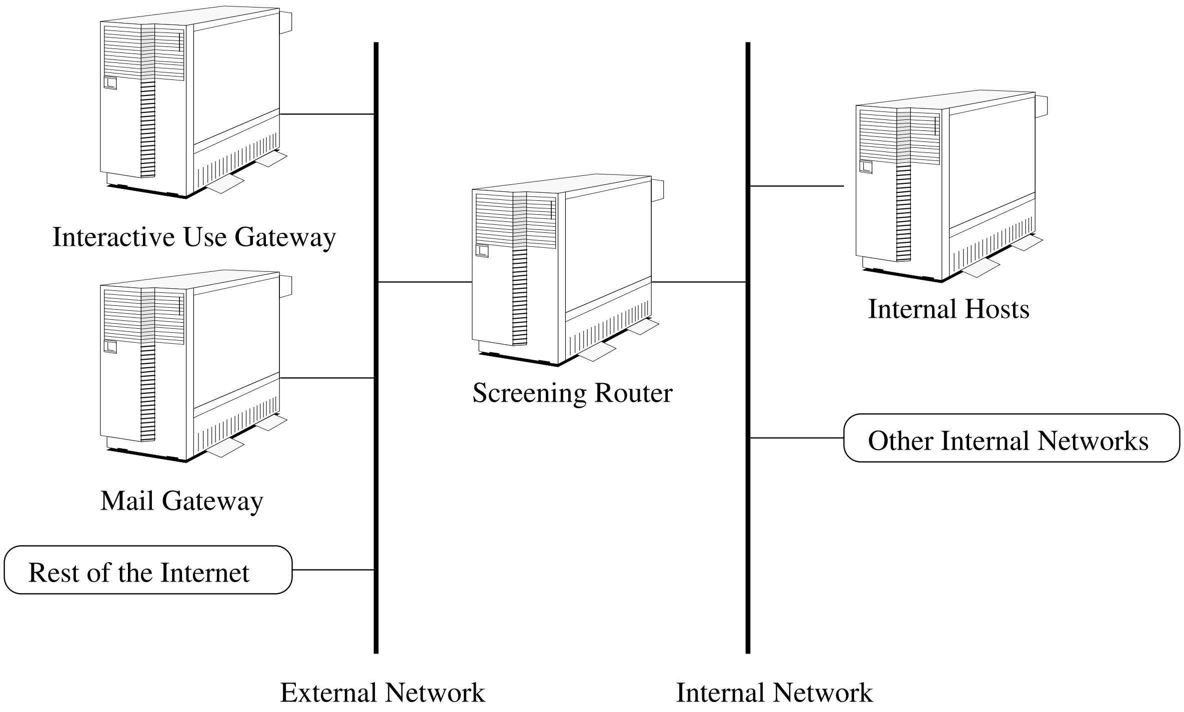


Figure 1: Typical Firewall Configuration

- Protecting firewall systems. Even though a system is part of the firewall, we may wish to prevent it from exchanging certain kinds of packets with external machines. For example, a dedicated mail relay may not need to allow telnet access from external machines. Putting it behind the screening router adds an extra layer of protection.

- Limiting interaction between firewall systems and internal systems. A firewall system probably does not need full access to all internal systems; its access should be limited to the functions that are required. Such a restriction limits the possible actions of an intruder who compromises the firewall system.

- Allowing widespread access for certain services. There may be a few services that are deemed sufficiently safe for all internal systems to use. Such services may include, for example, the FTP and WHOIS services operated by the Network Information Center (NIC). If we believe that the NIC is unlikely to be compromised, then there is little risk posed by allowing such access.

Screening routers can provide substantial protection, but there are many applications that are difficult to support using only screening routers. To circumvent this problem, firewall configurations often include a few privileged hosts that have broader access to the Internet (although their access may not be complete). Rather than allowing direct communication between internal and external machines for applications such as electronic mail and USENET news, the privileged machines accept mail or news messages and deliver them appropriately on the other side of the firewall. An application such as remote login requires a different solution — rather than allowing remote logins between all internal and external machines, a typical configuration would require all users to login to the privileged machine as an intermediate step. These user accounts add some administrative overhead to managing the gateway. This overhead is an important factor in how large the gateway can grow.

Firewall configurations of this nature are currently available from several vendors and consulting services.

4 Administrative Policies

Most organizations connected to the Internet have some sort of policy concerned with the security aspects of the connection. In many cases, common to universities, the policy is that no security should be assumed. Other organizations, such as companies or government agencies, frequently have detailed policies that limit use of the network. We will refer to such policies as "administrative policies." Some examples of issues covered by administrative policies include:

- Who can use the connection. Typically only employees or those individuals otherwise associated with the organization may use the connection. Many organizations further limit the set to those who have authorized user accounts on certain computers associated with the connection.
- What kind of data flow can occur. Many organizations are concerned about potential flow of private information outside the organization (for example, the source code to a critical product). Others are concerned with the flow of data into the organization (for example, employees retrieving proprietary data from another company).
- Which applications can be used. Policies frequently limit the set of applications that can be used with a connection. These limits may be based on lack of a demonstrated need for the application, a belief that the application is unsafe (whether because of known problems, a guess, or a history of security problems), or expected violations of other aspects of security policy. For example, an outgoing file transfer utility may violate the policies surrounding the kinds of data flow that are permitted.

To the technically-oriented reader, such policies (or variations on them) may seem silly, excessive, or ineffective. In some cases, they are. In other cases, the issues may be more subtle. For example, information can flow out of a company in many ways, especially in the hands of a disgruntled employee, so restricting outbound data transfer may not seem reasonable. On the other hand, consider a thief who breaks into a corporate computers system using a 1200 baud modem. He cannot download a large quantity of source code quickly over such a connection, but he might be able to take advantage of a high-speed network connection to move the code out to some other machine on the network. In this case, restrictions on data transfer serves to limit the damage the thief can cause.

In any case, we are not concerned here with defining or defending particular choices of administrative policies. Given a policy, we want to analyze it in the contexts of particular applications in order to construct an appropriate gateway. These policies are not stated in a formal fashion; part of the problem is constructing a real system that satisfies a vague policy statement (and its issuers).

4.1 Example Administrative Policy

For our case study, we consider the following policy:

- Access is limited to those with authorized user accounts.
- Unauthorized individuals must not be allowed to gain access to internal machines or information.
- It must not be possible for individuals to transfer large amounts of information out of the organization at high speed. Individuals with authorized user accounts are trusted not to abuse their privilege to send out proprietary information.
- The gateway machines should keep reasonable logs about their use.

Like most administrative policies, these policies are general statements and do not prescribe specific details for implementation. Often it is important to understand the intent of the policy as well as its statement in order to construct a system that satisfies the makers of the policy. The policy described above, for example, may be understood to permit electronic mail, even though mail allows some outbound transfer of information. Mail is usually not high bandwidth, and some information about each message is usually logged.

In Section 7 we describe some other applications and variations of the administrative security policy.

4.2 Scope of Protection

In this paper we are primarily concerned with protecting applications. We do not address issues such as spying on individual packets on the Internet, the abuse of protocols to create covert channels for signaling, forging source or destination IP addresses, or "hijacking" a TCP connection by inserting packets with proper sequence numbers into the connection. Many of these potential problems with TCP/IP are discussed by Bellovin [1]. Although we do not consider them here, these potential problems should be considered when designing a real firewall system.

5 A Relay for the X Window System

Some applications are difficult to operate securely with traditional firewall configurations. In our case, the critical ones were applications using the X Window System. After implementing a solution to the X problem, we began to investigate other applications where we could apply some of the same general techniques.

In the X Window System, the basic security model allows a user to control the set of hosts allowed to make connections to the X server. This control only affects new connections, not existing ones. Many users disable the access control entirely for personal convenience when using a more than a few hosts.

Researchers at CRL often collaborate with researchers at universities, and they want to run applications on the university machines with the X display on a workstation at CRL. Because of X's weak security model, simply allowing all X traffic to pass through the screening routers would not meet our security requirements.

Since we had no technical way of enforcing the use of access controls, we had to assume that the internal users would not use the existing access control mechanism carefully. There are also hooks in the X protocol for passing other authentication data to the server; we explain below why that mechanism is not sufficient.

One approach might be to modify the screening routers as necessary to allow connections between specific machines. The main problem with this approach is the management overhead. To keep the screen up to date would require a system manager to take action whenever a new connection was needed or when an old one was no longer necessary. A privileged application to do this would require substantial access controls and authorization for the users.

For quite some time, if a researcher asked for this capability, the answer was that it was simply unavailable. Fortunately, researchers don't like to take "no" for an answer, especially when it has to do with getting their work done, so we began thinking about how to provide this capability. We realized that we needed to solve the following problems:

- Limiting which systems could connect to an internal display. Since the address of the client host is the best information we have about the client, we can use it to restrict which hosts can connect.

- Ensuring that there were no unauthorized connections. Limiting con-

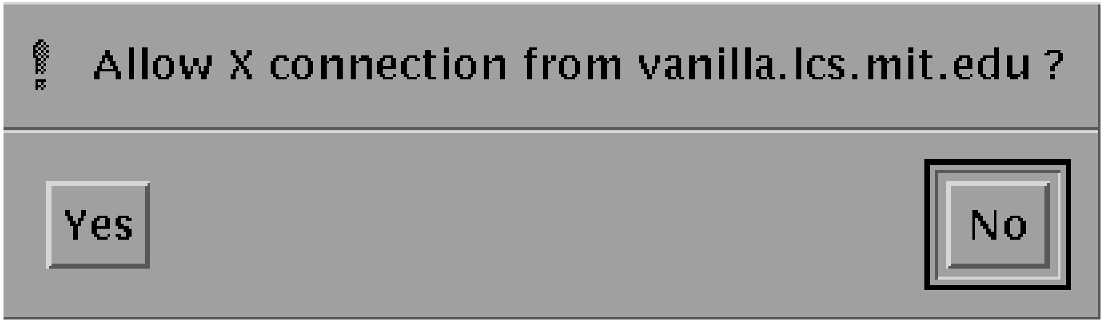


Figure 2: Dialog box for *xforward*

nections to a small set of systems does not eliminate all problems. A hostile user on an allowed system might try to connect to the display. The X protocol allows a client to spy on the entire state of the screen as well as the input stream, so a client can silently eavesdrop on other applications.

- Not modifying any client or X server software. Since we have no control over either the clients or the servers, we could not require any modifications to the software. This ruled out modifying clients and servers to use a strong authentication system such as Kerberos [16] or SPX [17].

The first problem means that we are limited at best to a solution based on IP addresses, not authenticated individuals. That immediately led us to the second problem, of ensuring that no unauthorized connections were made by other users of the remote machines.

Our solution is a modified TCP protocol forwarder called *xforward*, which is run by a user on a privileged gateway machine. A TCP forwarder is a program that listens for data on a port and forwards anything it receives to another system. The screening router is configured to only allow X traffic between the internal hosts and the privileged gateway. *xforward* allows the user to specify a list of hosts that are allowed to connect to the X server. It establishes a relay between a pseudo-display number on the gateway machine and the user's workstation.

Foreign X clients attempt to establish connections to the pseudo-display on the gateway machine, where the list of allowed hosts is checked. When that check is successful, the connection must then be explicitly approved by the user through a dialog box created by *xforward*. The dialog box is shown in Figure 2. This makes the establishment of each connection an intrusive action and ensures that the user is aware of all connections created to the display from the outside, so an intruder cannot spy on the X server. In addition, the list of allowed hosts is specified at relay startup time, and there is no way to disable this mechanism or to specify wildcards. If there is no activity on any connection for a certain period of time, all connections are closed and the relay terminates. For reference, the *xforward* manual page is shown in Appendix A.

This solution may seem rather simple. Indeed, it is not a matter of deep thought. The interesting principle here, however, is that we can use specific knowledge about an application to create a relay that provides the required functions of the application while preserving a desired security policy. In the next section, we describe several examples in which the application semantics differ enough that *xforward* itself would not work, but the general principle of analysis applies.

6 Example Applications

Other applications that we have used in this way include FTP, Telnet, WHOIS, and *sup* (a software distribution program developed by the Mach project at Carnegie Mellon). In each case, we used a similar process of analyzing the application's security policies and the administrative policies to develop a relay that provided the necessary functions while meeting the administrative policies. For comparison, the section concludes with analyses of electronic mail and USENET news in the same framework.

6.1 FTP

The File Transfer Protocol (FTP) [10] is widely used on the Internet for copying files. We may wish to allow many people on internal systems to copy files to and from external systems. With the policies we are currently considering, that is not possible without giving everyone an authorized user account on gateway machines. Suppose, however, that we relax the administrative policy slightly:

- Any employee can copy files from systems on the Internet to internal machines.

This rule does not differ much from the general intent of our original example. It does, however, give us some additional freedom to implement a gateway mechanism. There are two challenges here: one a technical problem and one caused by the administrative policy. The technical challenge is that FTP uses separate control and data connections, so a simple TCP relay is insufficient. The administrative challenge is that this policy does not permit copying files from internal machines to external machines, so even a simple FTP relay that can manage the data and control connections is not sufficient.

The solution in this case is an FTP relay that checks each FTP command before passing it through. Requests to store files are denied, and other requests are passed through. In addition, of course, the FTP relay limits access to internal machines. Such a relay has been implemented; a sample interaction with it is shown in Figure 3.

Given an FTP relay, however, we may be able to expand the offered service with an additional relaxation of the policy:

- An employee may transfer files out if the identity of the employee can be determined and recorded.

Authorized user accounts satisfy this requirement. We can also satisfy the requirement by using a strong authentication system, such as Kerberos [16], SPX [17], or "smart cards." In fact, a system has been implemented based on the the SecureNet SNK-004 from Digital Pathways, Inc. An individual with a registered key is allowed to transfer files after authentication. Figure 4 gives a sample dialog with this service.

Hence, with an understanding of both the administrative policy requirements and the details of the FTP implementation, we can devise a functional relay. If we understood the policy and had only a vague understanding of how FTP works, we might conclude that such a relay is not possible. If, on the other hand, we understood the FTP implementation and did not fully understand the policy, we might conclude that such a relay is not permissible.

6.2 Telnet

Telnet [9] is widely used on the Internet for remote logins. General telnet access between internal and external machines can serve many purposes, supporting cooperative work projects, customer support, and traveling employees who need to read their electronic mail. The relaxed version of the

```
% ftp ftp-gw
Connected to ftp-gw.
220-FTP passthrough server
220-To connect to a remote FTP server give your login as:
220-user@server.serverdomain (EG: anonymous@host.cs.mumble.edu)
220-
220-NOTE: All file transfers are logged by the relay, and are
220-      most likely also logged by the system at the other end
220       of the connection.
Name (ftp-gw:): anonymous@gatekeeper.dec.com
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
    ...
README.nfs
    ...
226 Transfer complete.
448 bytes received in 0.18 seconds (2.4 Kbytes/s)
ftp> get README.nfs
200 PORT command successful.
150 Opening ASCII mode data connection for README.nfs (2799 bytes).
226 Transfer complete.
local: README.nfs remote: README.nfs
2853 bytes received in 0.54 seconds (5.2 Kbytes/s)
ftp> put README.nfs
200 PORT command successful.
530 Operation denied by FTP gateway
ftp> quit
221 Goodbye.
```

Figure 3: Sample interaction with FTP relay.

```
% ftp ftp-gw
Connected to ftp-gw
220-FTP passthrough server
220-To connect to a remote FTP server give your login as:
220-user@server.serverdomain (EG: anonymous@host.cs.mumble.edu)
220-
220-NOTE: All file transfers are logged by the relay, and are
220-      most likely also logged by the system at the other end
220       of the connection.
Name (ftp-gw:): treese@somewhere.edu
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> ftp> put fubar
200 PORT command successful.
530 Operation denied by FTP gateway
ftp> quote authorize treese
331 Enter response code for 88146:
ftp> quote response 35233348
230 Accepted and authorized, 305336
ftp> put fubar
200 PORT command successful.
150 Opening data connection for fubar (192.58.206.2,3441).
226 Transfer complete.
local: fubar remote: fubar
20929 bytes sent in 0.035 seconds (5.8e+02 Kbytes/s)
ftp> quit
221 Goodbye.
```

Figure 4: Sample interaction with FTP relay using authentication.

policy described for FTP is sufficient to permit a telnet relay implementation using a simple TCP relay and dialog to set up the connection, given some kind of strong authentication mechanism.

With "outbound" telnet (from an internal machine to an external one) we can take an extra precaution, however. In the abstract, we are supporting interactive remote terminal service, not arbitrary data connections. Hence, it may be reasonable to limit the data rate on the outbound connection to (say) 9600 bits per second. This is surely fast enough to handle a person typing; we are not interested in anything more.

6.3 WHOIS

WHOIS [6] is a TCP-based query/response service, often used as a directory service. The Internet Network Information Center (NIC) has historically maintained a name lookup service for individuals on the Internet (although the Internet is now so large as to make maintaining a centralized directory impossible).

Given the somewhat relaxed policy described for FTP and telnet above, we consider access to the NIC WHOIS server. For this application, access to the NIC is sufficient because it holds the directory of interest. Widespread access to WHOIS servers throughout the network is a different issue. In this case, the protocol itself is not a means of data transfer, and we wish to provide the service for all internal users.

A simple TCP relay that only connects to the NIC suffices for WHOIS. It should limit access to internal systems, but no access control beyond that is necessary. It does not transfer significant data, and an intruder would need to compromise both the NIC and an internal machine to hijack the relay.

6.4 Sup

sup [15] is a software distribution program developed by the Mach project at Carnegie Mellon University. It is used to distribute updates to the Mach software to interested parties who are cooperating with CMU. Suppose we are interested in receiving updates to Mach on an internal system. *sup* is clearly designed for data transfer, so we cannot provide widespread and unauthenticated access. Since only one internal group needs to use *sup*, we can use a TCP relay that admits connections only from their machine and connects only to a designated machine at CMU. An intruder must compromise both ends of the connection to move data through the relay.

6.5 Electronic Mail and USENET News

Relays are nothing new to electronic mail and USENET news; "store-and-forward" systems have been relaying mail through organizational boundaries for years. We believe that these applications fit well into our framework. The security policy is relaxed to allow use by anyone for inbound or outbound messages. Messages "touch down" on a gateway machine, where sender and destination can be logged. The applications are not high in bandwidth because of the usual processing overhead.

7 Other Approaches

Application relays are not the only possible means to secure connected but distrustful networks. In this section we discuss some alternatives that operate at the network layer, concluding with a comparison to our work at the application layer. The approaches described here all interact with the routing of a packet through the network.

Routing determines the path a packet will take through a network. The path may traverse several networks and routers. Each router decides what action to take for each packet. This action may be to forward it to the ultimate destination, forward it to another router, or reject the packet for some reason (possibly with a notification to the sender). Most routers today try to select a route that minimizes some metric of the path, such as delay or number of routers traversed. Routers could use other considerations in making these decisions. For example, the router might permit only certain senders, or it may forward packets of certain users over higher-speed links. The alternative approaches discussed here vary primarily in what information is used to make routing decisions.

7.1 Screening Routers

As we have discussed, a screening router can be an important component of a firewall system. A screening router permits an organization to implement a variety of security policies [8]. Unfortunately, a screening router alone cannot always satisfy the administrative policy for a given application. Because it has no real knowledge of the application, it cannot perform application-specific actions such as those we use for *xforward*.

7.2 Policy Routing

Policy routing, such as the architecture described by Clark [3], enables routers to make decisions based on resource policies, including security as well as other considerations such as link speed or cost. Clark proposes an architecture in which policy routes are synthesized by the source host and its administrative domain. The routers along the path are responsible for enforcing the selected policy.

In practice, the main problem is that it requires wide deployment of both end systems and routers capable of creating and enforcing policy routes. Therefore we could not build a system based on policy routing.

7.3 Visa Protocols

Visa protocols [5] are another approach to managing the flow of packets between organizations. A "visa" is an unforgeable cryptographic stamp attached to a packet. In the same way that a passport visa grants permission to visit a country, a packet's visa grants it permission to enter or leave an organization. Routers at organizational boundaries can check the visa of a packet before forwarding it to verify that it has the appropriate permissions. Visas are issued by trusted access control servers in the source and destination organizations.

Visa protocols solve many of the problems we have considered so far. However, the routers still cannot take action based on the application, which is required for some of the policies we have considered. For example, a router would not be able to create the *xforward* prompt for a new connection, since it has no knowledge of the protocols it is routing.

7.4 Why the Application Layer?

Estrin and Tsudik [4] argue that the network layer is the appropriate place for inter-organizational access controls. This conclusion is based on the design principle of the end-to-end argument [12]. The basic end-to-end argument is that controls should be placed at the highest layer of the network at the actual endpoints of the communication, since only the highest layer will actually be able to ensure properties such as reliability and security.

Estrin and Tsudik note that one may consider entire organizations as endpoints in their own right, because the network resources of the organization must be protected as well as the end systems within the organization.

Because the network layer is the highest normally handled by an organization's border routers, the end-to-end argument places responsibility for security at that layer. We have gone a step beyond, however, using application relays on border gateways. Given the application relays, the end-to-end argument moves the responsibility for security to the relays.

Application relays give us the additional flexibility to perform application-specific functions — *xforward* can interactively request approval for a connection, for example.

8 Conclusions and Future Work

Building secure and useful inter-organizational networks is a complex challenge. By building application-specific relays to satisfy particular administrative policies, we have been able to expand the services available to users lacking full access to the Internet. These services preserve the desired security of the internal network as well. Not every application is suitable for this kind of relay, but the techniques work in a surprising variety of situations.

In addition to the security issues, these techniques can be useful when routing issues would normally prevent access between networks. For example, if an organization uses a single network number for its internal network, it is limited to a single primary route from the Internet into the organization's network, even if it has several points of connection. The application relays hide this problem from the end systems.

As the range of services available on the Internet grows, we hope to be able to make many new services available on protected networks by applying these techniques. Although the implementation often varies from application to application, making it difficult to reuse source code, each new application is easier to construct because it builds upon the cumulative experience. Much work remains to be done, however, in making this process more straightforward and automatic.

Although the telnet and ftp relays do eliminate the need for user accounts on the gateway, the current version of *xforward* does not. In order to eliminate the need for user accounts, we need a method for users to invoke the X relay remotely, without having a user account on the firewall. We might accomplish this in two ways: first, we can build a listener that will accept *xforward* invocation requests from internal machines, on the theory that anyone on an internal machine should have authorization to use the X relay. If this does not meet the administrative security requirements, in addition we can make use of "smart cards", to verify that the requester is in fact authorized to remotely invoke the X relay.

All of the relays we have implemented use TCP. For TCP applications, the relevant security decisions can usually be made at connection creation time. Application-specific relays can also be built for applications that use UDP. For UDP applications, the relay's decision whether or not to forward would have to be made packet by packet, rather than per connection. This would require detailed information about the application specific protocol layered on UDP. This is similar in spirit to what the FTP relay does when it checks each FTP command before passing it through.

On a more practical note, we are planning to extend *xforward* to support compression of the X protocol for use over low-bandwidth network connections. We also plan to add an interactive control panel that will display status information and allow users to modify the access lists during execution.

9 Acknowledgements

The authors would like to thank John Kohl for implementing the original TCP forwarding code now used in *xforward*; it had no particular security features. Brian Reid forced us to carefully work through the security issues surrounding *xforward*. Marcus Ranum designed and implemented the FTP and Telnet relays and has provided much helpful discussion on these issues. Victor Vyssotsky gave us the freedom to experiment carefully with the Internet gateway at Digital's Cambridge Research Lab. Neil Fishman and Ted Wojcik assisted with our experiments at CRL's gateway. We would like to thank Murray Mazer, Larry Stewart, Jim Miller, and the USENIX referees for helpful comments on this paper.

References

[1] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 9(2):32–48, April 1989.

[2] Bill Cheswick. The design of a secure internet gateway. In *Proceedings of the USENIX Summer Conference*, 1990.

[3] David D. Clark. Policy routing in internetworks. *Internetworking: Research and Experience*, 1:35–52, 1990.

[4] D. Estrin and G. Tsudik. An end-to-end argument for network layer, inter-domain access controls. *Internetworking: Research and Experience*, pages 71–86, June 1991.

[5] Deborah Estrin, Jeffrey C. Mogul, and Gene Tsudik. Visa protocols for controlling inter-organization datagram flow. *IEEE Journal on Selected Areas in Communication*, 1989.

[6] K. Harrenstien, M. K. Stahl, and E. J. Feinler. NICNAME/WHOIS. RFC 954, Network Information Center, October 1985.

[7] Jeffrey C. Mogul. Simple and flexible datagram access controls for UNIX-based gateways. In *Proceedings of the USENIX Summer Conference*, pages 203–221, Baltimore, MD, June 1989.

[8] Jeffrey C. Mogul. Using *screend* to implement IP/TCP security policies. Technical Note TN-2, Digital Equipment Corporation Network Systems Lab, 1991.

[9] J. B. Postel and J. K. Reynolds. Telnet protocol specification. RFC 854, Network Information Center, 1983.

[10] J. B. Postel and J. K. Reynolds. File transfer protocol. RFC 959, Network Information Center, 1985.

[11] Marcus J. Ranum. A network firewall. In *Proceedings of the World Conference on System Administration and Security*, Washington, D.C., July 1992.

[12] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2:195–206, 1984.

[13] Hervé Schauer and Christophe Wolfhugel. An Internet gatekeeper. In *UNIX Security Symposium III Proceedings*, pages 49–61. USENIX, 1992.

[14] Robert W. Scheifler and James Gettys. *X Window System*. Digital Press, Bedford, MA, 3rd edition, 1991.

[15] Steven Shafer and Mary Thompson. The SUP software upgrade protocol. Available by anonymous FTP from mach.cs.cmu.edu:/pub/sup/sup.doc.

[16] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication system for open network systems. In *Proceedings of the USENIX Winter Conference*, pages 191–202, January 1988.

[17] J. Tardo and K. Alagappan. SPX: Global authentication using public key certificates. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 232–244, Oakland, CA, May 1991.

[18] Hubert Zimmermann. OSI reference model — the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, Com-28(4):425–432, April 1980.

A *xforward* Manual Page

NAME

xforward – provide user-level X forwarding service

SYNOPSIS

xforward [options]

OVERVIEW

xforward provides a user-level X11 forwarding service which can be useful if there are IP network topologies which provide non-transitive routing (e.g. routers which implement policy packet screening).

OPTIONS

–display *display*

Specifies the destination display where the user wants applications to appear. Without this argument, xforward will use the DISPLAY environment variable.

–allow *allowed-host1 [allowed-host2 ... allowed-host16]*

Only connections from *allowed-hosts* are permitted. At least one *allowed-host* must be specified, and at most sixteen are allowed.

DESCRIPTION

xforward will choose an unused port for the local display, and listen for connections on the local host at that port. **xforward** informs the user which port to use as the local display, when **xforward** is first invoked. When it receives a connection, it will create a confirmation pop-up on the destination. If the user confirms the connection request, it will create a separate socket and connect it to the destination, and then commence data piping between the two connections. **xforward** can handle multiple simultaneous connections.

If there is no activity on a connection for 90 minutes, the connection is closed. If the X server at the destination does not have access control enabled, then xforward will report an error.

If a connection is closed by either end, any buffered data is drained to its destination before **xforward** will close the corresponding socket on the other end.

SEE ALSO

accept(2), bind(2), connect(2), listen(2), select(2), socket(2)

BUGS

Out-of-band data is ignored/thrown away.

If the initial connection to the destination fails for some reason, the client who connected to the local-display will get an open and immediately closed TCP connection, which may cause some difficulty for programs that expect some sort of server response or an error code indicating failure to connect.

AUTHORS

John Kohl, MIT Project Athena and Digital Equipment Corporation
Modifications for X by Win Treese and Alec Wolman