TIS Firewall Toolkit

Overview

About The Toolkit and This Documentation

The TIS Firewall Toolkit (hereafter referred to as “the toolkit”) is a set of programs and configuration practices designed to facilitate the building of network firewalls. Components of the toolkit, while designed to work together, can be used in isolation or can be combined with other firewall components. The toolkit software is designed to run on UNIX systems using TCP/IP with a Berkeley-style “socket” interface.

Throughout this documentation, a distinction is made between “configuration practices” and software. A configuration practice is a specific way of configuring existing system software, while a software component of the toolkit is a separate program which may replace or enhance existing system software. Thus, when the documentation refers to the configuration practice applicable to configuring some system daemon in a secure manner, it is assumed that the base operating system in question has existing support for that software, and that it is capable of being configured. The exact details of how to configure various system utilities differ from vendor implementation to vendor implementation and are outside of the scope of this document. In general, most UNIX systems with BSD-style networking will support all the functionality and services referred to herein.

Installing the toolkit assumes practical experience with UNIX systems administration and TCP/IP networking. At a minimum, a firewall administrator should be familiar with installing software and maintaining a running UNIX system. Since components of the toolkit are released in source code form, familiarity with building packages using make is required. The toolkit does not try to provide a “turnkey” network firewall, since every installation's requirements, network topology, available hardware, and administrative practices are different. Depending on how the toolkit is configured, different levels of security can be achieved. The most rigorous security configurations are not for everyone, while for others anything less will not suffice. It is the responsibility of the firewall installer to understand the security policy of the network that is to be protected[1], to understand what constitutes acceptable and unacceptable risks, and to

[1] This may entail helping to develop one, if none exists.

rationalize them with the requirements of the end users. Performing this analysis is the hardest task in implementing any security system. The toolkit, unfortunately, cannot do it for you; it can only provide the components from which to assemble a solution.

The toolkit consists of three basic components:

- Design Philosophy
- Configuration Practices / Verification Strategies
- Software Tools

An individual considering using the toolkit may use any or none of these components, as they see fit.

Design Philosophy

The TIS Firewall Toolkit is designed to be verified for correctness as a whole or at a component level. This appears to be a fairly novel approach for a network firewall, as many existing firewall systems rely on software that is “known to be good” or that is considered trustworthy because it has been used extensively for a long time. One problem with the “known to be good” approach is that historically it hasn't been very reliable. Certain software components such as mailers are frequently exploited in break-ins, no matter how carefully they are maintained. Problem programs are often complex pieces of software, often implemented in several tens of thousands of lines of code, which require system privileges in order to operate. As a step towards addressing this, the firewall toolkit is designed to operate along the basic design principles that:

- Even if there is a bug in the implementation of a network service, it should not be able to compromise the system.
- Hosts on the untrusted network should not be able to connect directly to network services that are running with privileges.
- Network services should be implemented with a minimum of features and complexity. The source code should be simple enough to be reviewed thoroughly and quickly.
- There should be a reasonable way of testing the correctness of the system.

These design principles can be effectively applied to any network firewall architecture.

Configuration Practices

Risk Analysis

Before beginning to design a firewall, it is important to have a clear idea of what the resulting firewall will provide protection against, how it will comply with existing corporate or organizational standards, and how it fits into the overall security architecture of the network. Once the security goals of the firewall have been enumerated, user

requirements and business practices are considered, and (it is hoped) a satisfactory design objective can be derived.

Perimeter Defense

One important consideration in setting up a firewall is that it is, first and foremost, a perimeter defense. Firewalls do not provide any protection once an attacker has gotten past them. Having a firewall is analogous to a large steel door as the front door to one's building — it provides excellent protection against frontal attack. To extend the analogy, if one's security policy is such that a steel front door is required, one should also have steel shutters on the windows, and one should not have connecting doors to the neighboring buildings unless they have steel doors and shutters as well. In practical networking terms, this equates to basic measures such as securing modem pools with passwords, publishing security standards that clearly inform users of their responsibilities, and examining any other networks that share (are “inside”) the security perimeter. A single user who decides to purchase a serial line network connection (SLIP) to an Internet service provider can unintentionally completely circumvent a very expensive security system. Having a firewall that uses strong authentication mechanisms such as one-time passwords or cryptographic calculators while having a modem pool that requires no passwords at all indicates an inconsistent security practice. When establishing a perimeter defense, the administrator must first perform a risk analysis, and then make sure that *all* the entry points into the network are protected equally strongly. If two networks are being connected, and will share a common security perimeter, then both networks should be protected to the same degree, with a consistently enforced shared security policy.

Basic Firewall Architecture

When constructing a network firewall, the first configuration decision that must be made is which of the two security models to follow. The two options are:

- That which is not expressly permitted is prohibited
- That which is not expressly prohibited is permitted

When implementing a firewall following the first approach, one identifies the services that will be provided, addresses the security of those services, blocks *all* other services and traffic off, and then enables the selected services only once they have been tested and are believed to be secure. In the second approach, one identifies all the services that are believed to present risks and disables or secures them. The first approach is more conservative, accepting that “what we don't know can hurt us,” but tends to impose limits on the types and number of services that can be provided through the firewall. The second approach is more versatile, since more services are supported, but runs the risk of degenerating into an arms-race between the administrator and system crackers. Another important consideration is the size of the prospective user community on the protected network. As the protected network grows larger and is harder to monitor completely, it becomes increasingly difficult for an administrator to verify that members of the user

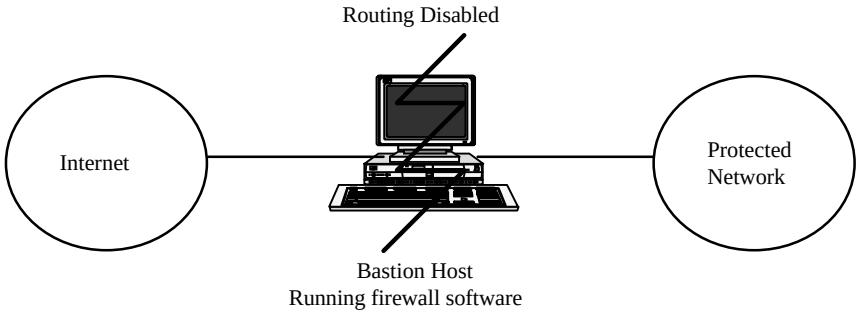
community are not themselves providing services over the network that get around the security of the firewall. An example of such a problem would be a user who decides to provide FTP service on a different port from the standard FTP port (port 23) because the FTP service port is blocked by the firewall but the alternate port is not. Eventually the firewall will need to protect the network from attacks (intentional or accidental) from the inside as well as outside.

The toolkit is designed to support users who want to implement firewalls based on the "that which is not expressly permitted is denied" approach. Generally, when building such a firewall, it is important to have good tools to provide access control and secure service for the few services that are provided. The software components of the toolkit implement security for the most commonly used network services.

Archetypal Firewalls

There are several archetypal firewall configurations that the toolkit is designed to support. For a more in-depth look at various basic forms of firewalls, see [1]. The primary types of firewalls the toolkit is designed to support are dual-homed gateways, screened host gateways, and screened subnet gateways. In these firewalls, the important common factor is a host (a "bastion host") which acts as an application forwarder, traffic logger, and service provider. Maintaining security on the bastion host is of paramount importance, and this is where most of the effort of setting up the firewall is focused.

Figure 1: A dual-homed gateway

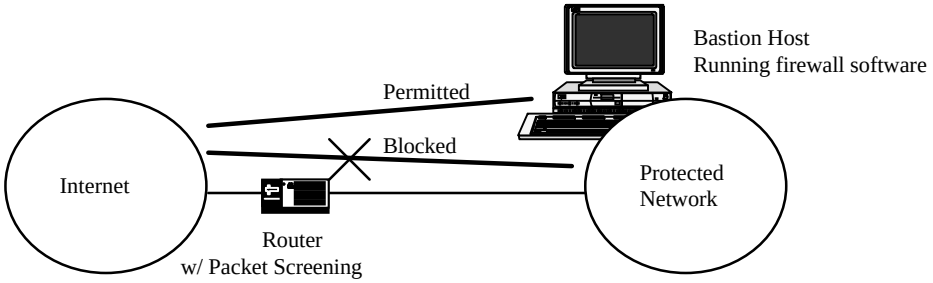


In the dual-homed gateway configuration, the toolkit software is installed on a host with two network interfaces. The toolkit software provides proxy services for common applications like FTP and TELNET, and security for SMTP mail. Since the bastion host is a security-critical network strong point, it is important that the configuration of the software on that system be as secure as possible.

Dual-homed gateways are an appealing firewall, since they are simple to implement, require a minimum of hardware, and can be verified easily. Most Berkeley-based UNIX implementations have a kernel variable *_ipforwarding*, which indicates to the operating system that it should not route traffic between networks, even if it is connected to two (which would normally cause the system to act as a gateway router). By completely disabling routing, the administrator can have a high degree of confidence that any traffic

between the protected network and the untrusted network is somehow passing through an application that is running on the firewall. Since there is no traffic transferred directly between the internal network and the untrusted network, it is not necessary to show any routes to the private network over the untrusted network. This effectively renders the protected network “invisible” to any systems except the bastion host. The only disadvantage of this type of firewall is that it is implicitly a “That which is not expressly permitted is prohibited” firewall — and it's impossible to weaken the firewall's security to let a service through even if one later decides one wants to. All services must be supported via proxies on the firewall.

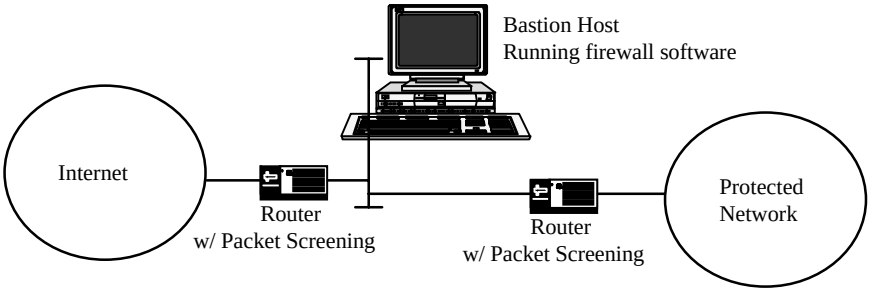
Figure 2: A Screened Host Gateway



A screened host gateway relies on a router with some form of packet screening capability to block off access between the protected network and the untrusted network. A single host is identified as a bastion host, and traffic is permitted only to that host. The software suite that is run on the bastion host is similar to a dual-homed gateway; the system must be as secure as possible, as it is the focal point for attack on the network. Screened host gateways are a very flexible solution, since they offer the opportunity to selectively permit traffic through the screening router for applications that are considered trustworthy, or between mutually trusted networks.

The disadvantage of this configuration is that there are now two security critical systems to be aware of: the bastion host and the router. If the router has access control lists that permit certain services through, it becomes an additional point of complexity to concern the firewall administrator. Verifying the correctness of a screened host firewall is a little more difficult, increasing quickly in difficulty as the number of services permitted through the router grows. Screened host firewalls also introduce management risks — because the capability exists to open “holes” in the firewall for special applications or influential users, the firewall administrator must be careful to resist pressure to constantly be modifying the screening rules in the router.

Figure 3: A Screened Subnet Gateway

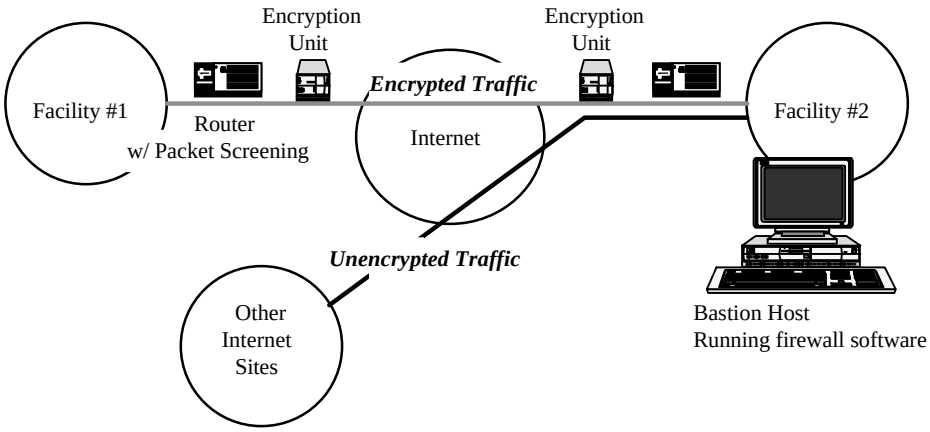


In a screened subnet firewall, a small isolated network is placed between the trusted network and the untrusted network. Access to this network is protected by screening rules in routers, which restrict traffic so that hosts on the screened subnet are the only systems reachable by both networks. Conceptually, this is the dual-homed gateway approach, applied to an entire network. The main utility of this approach is that it permits multiple hosts to exist on the "outside" network (sometimes referred to as the "demilitarized zone"). An additional advantage to screened host subnets is that the firewall administrator can configure network routing, so as to not advertise routes to the private network from the Internet, and to not advertise routes to the Internet internally. This is a powerful means of protecting a large private network, since it makes it very difficult for an outsider to direct traffic at the hidden private network. If the routing is blocked, then, like a dual-homed gateway, all traffic must pass through an application on the bastion host.

Firewalls in a Partitioned Network

Not every network is a single, isolated, network attached to an untrusted network. As use of large-scale networks for business continues to increase, many businesses are forming business partnerships and transmitting corporate sensitive information over public networks. A single corporation may wish to establish a common security perimeter among multiple facilities connected over a public backbone. In this type of situation, a firewall can be effectively combined with network-level encryption hardware (or software) to produce a virtual network, with a common security perimeter.

Figure 4: Common Security Perimeters Over a Public WAN



This figure illustrates how a company might establish a common security perimeter between two facilities, over a public wide area network. In the illustration, the encryption is separate from the router, but need not be, if integrated encrypting routers are available. Currently, there are several products that act as encrypting bridges at a frame level; i.e., they examine the source and destination address of all packets arriving on one interface, and retransmits the packet out the other. If the encrypting bridge/router is configured to encrypt traffic to a specific network, the packet data is encrypted, and a new checksum is inserted in the header. Once the packet is received at the other side, the peer encrypting bridge/router determines that it is from a network with which the router is encrypting traffic, and decrypts the packet, patches the checksum, and retransmits it.

Someone intercepting traffic between the two encrypting networks would see only useless cipher text. An additional benefit of this approach is that it protects against attempts to inject traffic by spoofing the source network address. Unless attackers know the cipher key that is in use, their packets will be encrypted into junk when they go through the encrypting bridge/router. If the encrypting bridge/router gets traffic for a network with which it is not to encrypt, traffic is transmitted normally. In this manner, a firewall can be configured, with encrypted “tunnels” to other networks. For example, a company could safely share files via NFS or use weakly authenticated network login programs, like rlogin, in safety over their encrypted link, and still have a strong firewall protecting access between the corporate perimeter and the rest of the world. A similar approach could be employed between two companies that wished to establish a business connection for proprietary information, in which traffic between the firewall bastion host on one corporate network and the firewall bastion host on the other corporate network was automatically encrypted.

Strong Authentication

Firewalls can be configured to be impermeable from the “outside” network, but sometimes it is important to permit users to access systems on the protected network from the untrusted network. Travelling staff, staff at conferences with network access, or customers may all need to log in over untrusted network links. In these situations, normal

password protection is insufficient, since most applications that use passwords pass them in the clear over the network, where they can be compromised by anyone with a packet-sniffer or equivalent software. In this type of hostile environment, one-time passwords or challenge/response calculators should be used. One-time passwords are passwords that are generated by a user prior to leaving a facility, and which are used sequentially in an agreed-upon manner with the system. If the password is compromised, it is not a security threat, since the password is never re-used. Challenge/response calculators use a secret shared between the system and a portable calculator or smart card to generate an encryption of a random challenge each time the user attempts to log in. Since the only person who could generate the correct encrypted response to the challenge is the user with the specific calculator, the system can authenticate the user without having to exchange a cleartext password.

The toolkit supports one-time passwords or challenge/response systems by including support in the proxy servers for a simple authentication protocol. This authentication protocol is served by a "middleware" authentication server that can embed support for multiple forms of authentication systems simultaneously. The proxy servers can be configured to require authentication (or not) based on the origin or destination of the service request. In this manner, the toolkit provides very flexible configuration of authentication. An administrator typically configures the proxies to require no authentication on outgoing requests, but, perhaps, to require strong authentication for incoming service requests.

Software Components

The following sections list the software components of the toolkit, briefly describing the purpose of each component. These components are all application-level programs, that replace or add to existing software. In some cases, there are other publicly available tools with similar functionality, which can be used instead of or in addition to the firewall toolkit components.

Smap: SMTP Service

SMTP is implemented using a pair of software tools called smap and smapd. Generally, SMTP mail poses a threat to the system, since mailers run with systems-level permissions in order to deliver mail to users' mailboxes. Smap and smapd address this concern by isolating the mailer so that it runs in a restricted directory via chroot, as an unprivileged user. Smap and smapd do not address any issues relating to mail spoofing or denial of service attacks via mail. Smap's primary purpose is to insulate a notoriously buggy program which has been implicated in many break-ins in the past. The bulk of the real mail-processing work is performed normally by sendmail, requiring no modifications to sendmail or its configuration file. When a remote system connects to the SMTP port, the operating system invokes smap, which immediately chroots[2] to its restricted directory

[2] The *chroot* system call is a means within UNIX of irrevocably isolating a running process within a sub-branch of the file system.

and sets its user-id to an unprivileged one. Since smap requires no system support files, the restricted directory can contain no files other than those that smap creates. There is no risk of smap being tricked into modifying system files (because it is chrooted), and there is no way an interactive session can be obtained via smap (there are no executables in the smap spool directory). Smap's sole purpose is to talk SMTP with other systems, gather mail messages, write them to disk, generate log entries, and exit. The second part of mail processing does not run chrooted to a restricted directory, but still does not require permissions; smapd is responsible for scanning the smap spool directory periodically and submitting the queued messages to sendmail for final delivery. Note that if sendmail is configured normally and smapd is running with the uucp user-id mail can be delivered normally without requiring smapd to run with enhanced permissions. When smapd delivers a message, it clears the file from the spool area.

In this manner, the functionality of sendmail is preserved, while preventing an arbitrary user on the network from communicating directly with sendmail. Analyzing the sendmail program's 20,000 lines of code for bugs is a sizable task when compared to smap's 700 lines.

Netacl: TELNET Service, Finger, and Network Access Control Lists

Inetd contains no provision for access control; i.e., it will permit any system on the network to connect to a service that is listed in inetd.conf. There are several situations where general purpose access control on network services is desirable, and tools to implement such control have been available on the Internet for several years. Netacl is the toolkit component which manages network access control. It permits arbitrary access control specification for each network service based on the client host's network address, and the service requested. Thus, one client (specified either by IP-address or host name) might invoke a different version of telnetd when it connects to the TELNET service port on the firewall.

UDP access control is not provided by netacl. Many “tcp wrappers” available on the Internet (such as Wietse Venema's log_tcp) support access control of a sort for UDP based services. Reliably authenticating the origin of a UDP packet is impossible with the current technology, so netacl does not attempt to address this issue: security for UDP-based services is provided by the draconian means of shutting *all* security-related UDP-based services *off*.

Many services that are disabled by default, such as finger, can be selectively enabled for hosts within the private network. Netacl also plays an important role in the configuration practices employed for other services such as anonymous FTP, since it can be configured to perform a chroot *prior* to invoking a given service. This functionality provides a great deal of flexibility for specifying services that should run under isolation. For example, a finger proxy service can be implemented by configuring netacl to invoke fingerd as an unprivileged user, chrooted to an empty directory. Prior to fingerd's

execution, it will be placed in isolation; if there is a security hole in fingerd,[3] it cannot be effectively exploited, since no file system or executables will be available to the attacker.

In the usual firewall configuration, netacl will be used to block all but a few hosts from being able to attempt to login to the firewall via either telnet or rlogin, as well as being used to possibly block access from aggressor sites in the event of an attack.

The security of netacl relies on IP-addresses and/or host names. For security-critical applications control, IP-addresses should be used, to avoid use of the Domain Name System (DNS) to spoof the source of the connection. Netacl does not protect against attacks using spoofing of IP-addresses via source routing or other means; if such attacks are a concern, a router that is capable of screening source routed packets should be employed. The security of the internet daemon inetd is assumed in this configuration. Netacl is designed to be easy to examine for correctness, consisting of 240 lines of C code (including comments); it is a very simple program. Care should be taken in configuring it.

Ftp-Gw: A Proxy Server for FTP

In order to permit file transfer through the firewall without risking compromising the firewall's security an FTP proxy server is provided. The proxy server supports access control based on IP-address and/or host name, and supports secondary access control permitting any FTP command to be selectively blocked or logged as it is transmitted. Destinations for service can also be selectively permitted or blocked. All connections and bytes transferred are logged.

The ftp-gw poses no threat to the security of the firewall system itself, since it runs chrooted to an empty directory and does no file I/O other than reading its configuration file. Since emulating the FTP protocol is somewhat complex, ftp-gw is approximately 1,300 lines of source code and is slightly less easy to verify by examination than other components of the toolkit. The FTP gateway is only intended to provide access to FTP services, and does not address issues of who is authorized (or not) to export files. Support for strong user authentication can be attached to the gateway, to require user authentication prior to exporting or importing files. The FTP gateway is a useful tool, which should be implemented in accordance with a consistent site data security policy[4]. The toolkit includes source code for a modified version of the FTP daemon which permits an administrator to provide both FTP service and FTP proxy service on the same system.

Telnet-Gw: A Proxy Server for TELNET

In order to permit remote terminal access through the firewall, without risking compromising the firewall's security, a TELNET proxy server is provided. The proxy server supports access control based on IP-address and/or host name, and supports secondary access control permitting any destination to be selectively blocked. All

[3] The Morris internet worm took advantage of a loophole in fingerd to compromise some systems.

[4] For example, it is fruitless to mandate blocking outgoing export of files through the FTP gateway when electronic mail is permitted.

connections and bytes transferred are logged. Once users have connected to telnet-gw, they are presented with a simple menu of options to assist in connecting to a remote host.

The telnet-gw poses no threat to the security of the firewall system itself, since it runs without permissions, chrooted to a restricted directory. The source code for telnet-gw is easily reviewed, consisting of approximately 1,000 lines of code. The telnet-gw menu processing is entirely in-memory and no subshells or programs are invoked; no local file I/O is performed other than reading the configuration file. Thus, telnet-gw cannot provide access to an interactive login on the firewall system.

Rlogin-Gw: A Proxy Server for Rlogin

Terminal access via the BSD rlogin protocol may be supported via the rlogin proxy. The rlogin proxy supports permissions checking and access control in the same manner as the TELNET gateway. Rlogin clients can specify a remote system as part of the initial connection to the proxy, eliminating the need for user interaction if authentication is not required.

Plug-Gw: A TCP Plug-Board Connection Server

Certain services such as Usenet news are often provided through a firewall. In such situations, the administrator has the choice of either running the service on the firewall itself, or installing a proxy server. Since running news directly on the firewall exposes the system to any bugs in the news software, it is safer to use a proxy to gateway the service onto a "safe" system on the protected network. Plug-gw is a general purpose proxy that "plugs" two services together transparently. Its primary purpose is for supporting Usenet news, but it can be employed as a general-purpose proxy if desired.

In general, plug-gw is installed on the firewall, so that when a remote system connects to the NNTP[5] port it automatically reconnects to a designated news server on the inside network. If the news server on the inside network connects to the news port on the firewall, plug-gw automatically reconnects to a designated news server on the outside network. This reciprocal connection is based on the IP-address of the originally connecting host. Once plug-gw is connected, it simply copies data until one side or the other shuts down the connection, at which point it exits. Plug-gw is configurable to selectively permit or deny connections based on IP-address/host name. All connections and bytes transferred are logged.

Plug-gw can act as a general portal between the protected network and the outside network; therefore, it should be used sparingly and with caution. Since it acts only as a data pipe, it performs no local disk I/O, and invokes no subshells or processes. In the general case of use for news, plug-gw provides excellent security, since it permits an outside system bi-directional communication with a single port on an internal news server,

---

[5] NNTP: Network News Transfer Protocol, a TCP-based protocol for transmitting Usenet news articles in bulk.

while still blocking all other traffic. In a sense, plug-gw is similar to adding a configuration rule to a router that permits traffic only between two systems on a single port, except that it logs all transactions.

Optional Components and Configuration Practices

The following components and configuration practices are optional.

Authd: Network Authentication Service

The network authentication server authd provides a generic authentication service for network applications. Its use is optional, required only if the firewall proxies ftp-gw and tn-gw are configured to require authentication. Authd's purpose is to provide a generic interface to multiple forms of authentication. For large organizations, where several forms of authentication challenge/response cards are in use, authd can link them all together to use a single database. A simple administrative shell is included that permits the authentication database to be manipulated over a network, with optional support for encryption of authentication transactions. The authd database supports a basic form of group management; one or more users can be identified as the administrator of a group of users, and can add, delete, enable, or disable users within that group. Authd internally maintains information about the last time a user authenticated to the server, how many failed attempts have been made, and can automatically disable accounts that have multiple failures. Extensive logs are maintained of all authd transactions. Authd is intended to run on a secured host, such as the bastion host, since its database is a possible point of attack.

Telnetd: Network Login Service to the Firewall

For administrative reasons it is sometimes desirable that a systems manager be able to login to the firewall itself to perform maintenance. Optionally, the TELNET server process telnetd can be configured to run on the firewall, for this purpose. The standard configuration practice is that the TELNET proxy server telnet-gw runs on the TELNET TCP service port (port 23) and administrators must login on the console. Optionally, telnetd can be configured in inetd.conf to provide access. It is recommended as a configuration practice that access to telnetd be protected using netacl, with a limited number of hosts having permission to connect to the service. Additionally, it is a recommended configuration practice that the login procedure on the firewall require a one-time or changing password with challenge/response for authentication.

Login: User Authentication

The toolkit includes a login program named login-sh for user authentication. This version contains support for a variety of authentication protocols using token authenticators such as Security Dynamics' SecurID, Digital Pathways' SecureNet Key, and the Racal Watchword. The login program replaces the user's initial shell, and requires them to authenticate prior to invoking their normal command interpreter. This approach provides good login security without requiring modifications to the vendor-provided login

program, and can be activated on a per-user basis. The toolkit version of login provides enhanced logging over the standard version, logging successful logins as well as failed ones.

Ftpd: Anonymous FTP Service

The toolkit provides a configuration practice for the FTP service. Traditionally, ftpd permits users to login as “anonymous” or “ftp” to gain guest access to the system. In this case, the user is then chrooted into the FTP account's home directory, where presumably access is controlled using standard configuration practices for installing anonymous FTP (see ftpd for details). The toolkit's configuration practice takes advantage of netacl to determine if the FTP request is from a system on the “outside” network, and if it is, ftpd is invoked *after* having already been chrooted to a restricted area. This is to obviate any bugs that may be in the implementation of ftpd[6] and is consistent with the overall configuration practice of preventing any system on an untrusted network from being able to directly connect to a privileged application running in an unrestricted file system. Despite the security this practice provides, due care must be spent in configuring ftpd properly, though combining ftpd with netacl does not require any changes from the standard way of installing anonymous FTP that is documented in the ftpd manual pages.

Syslogd: System Logging

The firewall toolkit includes a version of the system logging daemon which permits specification of regular expression search patterns in its configuration file, and the ability to invoke specified programs when a specified log entry is received. This permits real-time scanning of system logs, and real-time alerts. The invocation of programs based on log entries and patterns is a powerful tool for permitting an administrator to trigger a shutdown or redirect messages to a beeper or electronic mail if a security-critical event is detected.

---

[6] One popular ftpd implementation that is in widespread use turned out to have a security hole that permitted anyone on the network root access to the root file system of the host running it.

For Further Reading

[1] Marcus J. Ranum, “Thinking About Firewalls,” proceedings of the Second World Conference on Systems Management and Security (SANSII), 1993 Available for FTP from ftp.tis.com: /pub/firewalls/firewall.ps.Z

[2] Bill Cheswick, “The Design of a Secure Internet Gateway,” USENIX proceedings. Available for FTP from research.att.com: /dist/secure_internet_gateway.ps

[3] Cliff Stoll, “The Cuckoo's Egg”

[4] Smoot Carl-Mitchell and John Quarterman, “Building Internet Firewalls,” UNIX World, February 1992

[5] Simson Garfinkel and Gene Spafford, “Practical UNIX Security,” O'Reilly and Associates, June 1991

[6] Dan Farmer, “COPS and Robbers, UN*X System Security,” Internet software. Available for FTP from cert.sei.cmu.edu: /pub/cops

[7] Bill Cheswick, “An Evening with Berferd in which a cracker is Lured, Endured, and Studied,” USENIX proceedings, Jan 20, 1990 Available for FTP from research.att.com: /dist/internet_security/berferd.ps

[8] Marcus J. Ranum, “An Internet Firewall,” proceedings of World Conference on Systems Management and Security, 1992 Available for FTP from tis.com: /pub/firewalls/dec-firewall.ps.Z