Open Computing ‘‘Hands-On’’ Tutorial: September 1994

Internet Firewall Protection

The TIS Firewall Toolkit can be used to control access to your network from the Internet

By Marcus Ranum

Over the past year, the Internet has gone from a network used by the developers and techies to the precursor to the much-hyped Information Superhighway that is populated by a million new users each month. Along the way, businesses are looking into whether they can give their customers new and better services with the Internet. But the thread running through many management information-services directors’ minds is: How can I connect to the Internet and not get burned?

An Internet firewall is the embodiment of a security policy that should implement whatever access controls your organization has deemed necessary and appropriate between your private network and the rest of the world. There are many different types of firewalls and techniques for implementing them. Typically what an organization wants as an Internet access policy is something like: ‘‘let everyone on our network access the Internet, but don’t let anyone from the Internet into our network except authorized users.’’ This access policy relies on two fundamental components: enforcing the access control and determining what an ‘‘authorized user’’ really is.

This tutorial examines using software modules to solve some firewall implementation issues regarding log-in and user authentication. The approach we’ll examine uses some of the proxies from the Trusted Information Systems Inc. (Glenwood, Md.) Internet Firewall Toolkit. The TIS Internet Firewall Toolkit and documentation are available via anonymous FTP (`ftp://ftp.tis.com:pub/firewalls/toolkit`) or via the World Wide Web (`http://www.tis.com`).

Firewall Routers

TIS’ toolkit was designed with several basic philosophies in mind. First, a conservative stance is taken--only services that are permitted may pass through the firewall. Quite a few firewalls are implemented using only routers. Many firewalls are based on the philosophy of permitting everything that is not known to be a threat to pass through. This approach is is more open, but leaves you vulnerable to new attacks.

Security software should be kept simple so it is possible to examine the source code and quickly validate its correctness. Another concept is that organizations would prefer to have a ‘‘crystal box’’ implementing their security policy rather than a ‘‘black box’’ of precompiled vendor-controlled code. The toolkit consists of the following modules, which may be used in different configurations:

- Proxy servers for the `ftp`, `telnet`, and `rlogin` programs for passing these services through the firewall and improving authentication;

- A front end for the `sendmail` program that runs with reduced privileges and collects SMTP (simple mail transfer protocol) requests, passing them to a mail host on the internal network;
- The `netacl` program for securing TCP/IP services started by the Internet daemon (`inetd`), similar to the Tcp Wrapper from Wietse Venema (discussed in the July 1993 Tutorial);
- An authentication server (`authsrv`), which supports a variety of one-time password schemes;
- A system logging daemon (`syslog` replacement), which scans incoming log messages for attack signatures and can issue warnings in real time; and
- Server programs with improved authentication for `ftpd`, `telnetd`, and `login`.

Although establishing a site security policy and setting up the complete toolkit are beyond the scope of this article, we'll try to give a flavor of how the toolkit works. We'll build a hypothetical minimal firewall that shows how you might employ two components of the toolkit: the `telnet` proxy (`tn-gw`) and the authentication server (`authsrv`).

Creating an Air Gap

A good way to conceptualize a firewall built using the toolkit is as an ''air gap'' between two networks that has a small number of software-based bridges known as ''proxies.'' A proxy is a network server that forwards application data through a firewall. The advantage of this approach is that it limits the number of points of attack against your network. The firewall administrator can then focus on making the implementation of each of those services as secure as possible. A completely bare-bones firewall might only relay SMTP-based e-mail. Other interactive services like `telnet`, `rlogin`, and `ftp` are available through proxies included with the toolkit.

Proxies have another major advantage: they're neat, simple places to enforce the second part of our access policy, restricting access to authorized users only. The most important part about authorizing users is *authenticating* them. Recently, there has been a rash of incidents on the Internet where users' passwords have been ''sniffed'' off the network and their accounts broken into. The sad fact of the matter is that ordinary passwords simply aren't a strong enough authentication technology for using across loosely controlled wide-area networks.

There are a number of authentication systems that are commercially available and use one-time passwords or challenge- response schemes to verify a user's identity. These systems all illustrate the principle that information disclosed over a network cannot be reused by an attacker who intercepts it. The toolkit's authentication server can incorporate different types of authentication systems and can interact with the toolkit proxies to supply a transparent interface to authentication and authorization. Once users have authenticated to the proxy, it permits their traffic through the firewall.

Let's imagine we've set the firewall tool-kit up on a workstation that sits between the protected network and the Internet. All traffic between the Internet and our protected network has to go through the firewall somehow, we've been careful to disable routing on it, and we've shut off all nonessential services and daemons that would otherwise run on the system. Further, let's pretend that the only service we (initially) want to run through the firewall is `telnet`, with users not being permitted into the protected network without authenticating first. Outgoing connections, or connections originating from within the protected network, will be permitted without requiring authentication.

Shutting the services down on the system is Unix-version dependent, but the procedure is fairly straightforward. Edit the system start-up files (depending on the Unix version: `/etc/rc`,

/etc/rc.local, or /etc/rc2.d/S*) and comment out anything that isn't strictly necessary. There are other daemons started by inetd or init (depending on the Unix version in /etc/inetd.conf or /etc/inittab) that should be disabled as well.

Battening down the hatches is an iterative procedure: Make a set of changes and reboot the system. Checking the process table and the network tables using netstat and ps will give a clear idea of exactly what is running on the system (Listing 1).

While it may seem excessively paranoid to reduce a perfectly useful system to the level of a dedicated processor, it's a conservative approach and reflects an admission that what you don't know can hurt you. Anyone who feels this approach is too extreme should read prior CERT and vendor security advisories; there are too many examples of ''harmless'' network servers that have enabled attackers to gain a foothold into a system. We prefer to shut everything off and turn things on when we know we can trust them (which is sometimes never) than to have to constantly scramble about closing holes.

Because we want to enable telnet through the firewall, there has to be a telnet proxy listening on the telnet service port. The proxy is designed to be invoked directly from inetd so that there is one active proxy per connected user. An entry for authsrv is also needed, making for a fairly abbreviated /etc/inetd.conf (Listing 2).

Authentication Rules

All of the toolkit components read their configuration information and access policies from a master configuration file called netperm-table. The netperm-table consists of a list of rules, each tagged with the name of the application that rule applies to. Rules are interpreted in order from the beginning of the file to the end, a feature that is important when specifying an access policy. For example, Listing 3A contains the rules we might define for the telnet proxy.

The tn-gw: on the side indicates the rules apply to the telnet gateway. The welcome-msg rule specifies some configuration options, namely, the path name of a file to display when a site connects to the proxy.

Permit-hosts rules define the proxy's access policy, and it is here where the order of the rules becomes important. When defining Permit-hosts rules, wildcards can be used to apply a rule to an entire group of systems. Both IP-addresses and domain names can be used, but we prefer to use IP-addresses because DNS (Domain Name System) information is vulnerable to spoofing by a sophisticated attacker.

Reading the rules in the example, we see that the first line permits all hosts in network ''192.33.112'' to connect to the proxy. The second line permits all other systems to use the proxy. You'll notice, however, that the rules contain different parameters on each line. The second line contains an option -auth added at the end. This option effectively extends our access policy to the level of the individual user by informing the proxy to permit access only from users who can prove that they are authorized. Taken together, these two configuration lines implement the access policy postulated at the beginning of this discussion: ''let everyone on our network access the Internet, but don't let anyone from the Internet into our network except authorized users.''

For a hypothetical user at a desktop inside our network, accessing a remote site on the Internet looks like what appears in Listing 3B. When the proxy received a connection from the user's workstation, it read `netperm-table` and applied each of the access control rules based on the workstation's address, accepting the first matching rule. In this case, the first matching rule was the one that doesn't require authentication, so a command prompt was immediately displayed to the user. Users accessing the firewall from the outside see a somewhat different dialog, shown in Listing 3C.

Logging All Calls

Before the proxy permits users to connect to another system, they first have to authenticate to it. While it looks a lot like logging into a system, it's important to realize that users really aren't logging into anything. They don't actually have an account on the firewall system itself, they're just connected to a small application that is forwarding data back and forth. This strategy strengthens the firewall considerably because the administrator doesn't have to permit general command-level user access to the firewall host itself. Another advantage proxies give the administrator is detailed audit information (Listing 3D).

The first set of audit records show the outgoing connection, which didn't require authentication. When users authenticate on the incoming connection, the audit records reflect the additional information, including on the summary lines, where it clearly associates the individual users with the amount of traffic they passed through the firewall, their source, and their destination.

When the user in the example authenticated to the firewall proxy, the proxy used a form of challenge-response authentication based on a handheld calculator. Initially, when the user responded to the ‘‘`Username:`’’ challenge, the proxy passed the user ID to the authentication server, requesting an authentication dialog. The authentication server keeps a database of each user ID and the type of authentication to perform with that user, along with some other information such as number of failed log-in attempts, full name, and so forth.

In the example, the authentication server determined the user has a challenge-response calculator and generated a challenge, which it relayed to the proxy for presenting to the user. Someplace on the other side of a network, the user read the challenge, pulled out his or her challenge-response calculator, unlocked it with a secret PIN, entered the challenge, and typed the response back to the proxy. The proxy relays the response to the authentication server for validation and accepts the user if the authentication server indicates it should.

While all the dialog between the proxy and the authentication server sounds fairly complex, it's transparent to the user and turns out to be extremely convenient because there is only one piece of software that needs to include support for different authentication systems. It also provides the ability to mix and match authentication methods depending on your user base. For example, a network having 500 users might only have 100 who will ever need access from the ‘‘outside’’ through the firewall. The administrator only needs to maintain authentication entries for those users. Suppose of the 100 users, 20 of them travel frequently and the remaining 80 only travel once a year. In that case, the administrator can issue challenge-response calculators to the 20 frequent travelers and can issue the remaining 80 users software-based one-time passwords, which are less convenient but very inexpensive.

Creating user records in the authentication server is done by assigning a user name, an authentication system to use, and initializing the authentication system as necessary by using the `authsrv` program on an internal host (Listing 4).

In this example, a new user is created and assigned an authentication system: a challenge-response calculator. The administrator then enters the calculator's shared secret into the database so that the responses can be verified. The newly created user can now log in through the firewall.

This brief overview of some toolkit components can only give you an example of the things you can do to implement your security policy with these tools. The security policy appropriate for a site depends on the assets to be protected, the cost of downtime due to a network intrusion, and the types of connectivity to the Internet required. There are no simple guidelines for creating policy-but these are fairly simple tools for enforcing many aspects of policy.

The toolkit includes extensive documentation that explains the concepts behind the toolkit and the design of firewalls. Other good sources of information for firewall implementors are the Firewalls mailing list (mail to `majordomo@greatcircle.com`) saying ``subscribe firewalls'' in the message body to join) or Bill Cheswick and Steve Bellovin's recent book, *Firewalls and Internet Security--Repelling the Wily Hacker,* published by Addison-Wesley (ISBN 0-201-63357- 4).

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content

Search

Last Modified: Tuesday, 22-Aug-95 16:20:26 PDT