by Peter Collinson, **Hillside Systems**

MICHELLE FRIESENHAHN WILBY

Security on Track

At the end of April, I attended the SANS 97 Conference held in a hotel in the Inner Harbor in Baltimore. I somehow found my way into the security track of the conference and heard a number of very good talks aimed at stopping the bad guys from doing their bad things on my systems. My machines are connected to the Internet, so security is always a hot topic for me. People seem to be continuously finding new ways to exploit holes in systems, and a neat feature can turn into a massive hole overnight. However, things do get fixed because of the activities of organizations like the Computer Emergency Response Team (CERT) Coordination Center, who bring problems into the open. Also, many vendors distribute security patches to all comers, not just their own customers.

There are an increasing number of sites appearing on the Web that are dedicated to security issues. Many of them give information on extant holes, warning systems administrators of problems. Many sites offer prescriptions for trainee system crackers, giving them ways to exploit holes. Many attacks are carried out by naive people who have picked up a set of operations that might work and are using them to experiment. Such people don't often have a deep understanding of what they are doing. I even heard of one case where a cracker successfully became superuser on a UNIX system and then proceeded to type DOS commands.

Even if your systems are sitting behind a firewall, it makes sense to take basic precautions to deter the potential cracker who may be using a prescription to attack your site. This will help limit damage if your firewall is breached because of some oversight or misconfiguration. I can't say that I have heard about successful attacks through firewalls. This may be due to the efficacy of such systems, but then again, if I am running a company that makes firewalls, I will think hard about customer confidence before I broadcast any problem that my system is experiencing. I hope that such companies have enough guts to tell existing customers about problems and fixes.

The inetd Daemon

The first object of my attention on any unprotected system is the `inetd` daemon. To understand what it does, let's take a concrete example. Consider sitting at one machine and running `telnet` to connect to a remote system. The `telnet` program will first create a link to the remote machine using TCP/IP by sending a connection request to a "well-known" port (number 23). On the target machine, some program must have registered an interest in receiving messages that are addressed to port 23. If not, the kernel in the target machine won't know where to send the messages and will reject them. In the early networking systems, the `telnetd` daemon was the program that registered this interest. At system bootstrap time, `telnetd` was run and said to the kernel, "OK, send all requests for port 23 to me." The program then put itself to sleep until such a request materialized.

When telnetd was sent a connection request, it didn't want to deal with the request directly, because it's desirable to have more than one telnet connection to any one machine. Instead, it forked to generate a process that would deal with the connection. Having done its job, telnetd went back to sleep waiting for another connection. The child that it forked got on with the job of communicating with its client using a new port number. TCP/IP allows a server to reply to a connection request by saying: "Yep, I'll talk to you, but please use this port number."

In the early systems, there were many separate services that behaved in a similar way to telnetd. Doing a ps on such a system showed several processes, each offering a different service, sitting there waiting for a connection to be made from some client on the network. Sun co-founder Bill Joy had the bright idea of creating inetd, a single program whose job is to wait for connections to a great many services. When a connection is made, inetd forks and runs the appropriate program, whose identity and parameters are determined by a control file, /etc/inetd.conf. The contents of this file are read by inetd when it starts. The selection of the appropriate program is made by using the protocol type (UCP or TCP) and the port number used in the connection request.

Most vendors ship systems with everything in inetd.conf enabled and ready for use. This perhaps makes sense for installation purposes; systems are operational "out of the box." However, the first job of any system installer should be to go through the entries in the file and disable anything that is not used on that particular system.

There are several categories of programs in the Sun system that you may wish to turn off. First, the easy set, there are several programs that you will not be using. For example, if you don't plan to support uucp over the network, then this line can be removed. I tend to comment lines out of the file by inserting a # character rather than deleting them, which makes it easier to insert the correct lines later. When you change the contents of inetd.conf, you have to tell the running inetd to reread its control file. Find the process ID of inetd and send it a hang-up signal:

```
# kill -1 pid
```

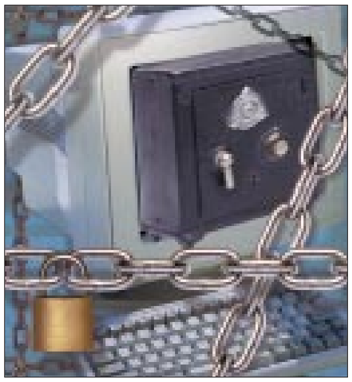
Second, having got rid of the things that you know you don't need, you should examine the set of programs that supply information about the system or its users. For example, I usually turn off remote access to the ps and netstat programs. Many sites now turn off access to the fingerd and talkd programs, largely because they run programs that are suspect from a security point of view (or they did in the past).

Third, on a Sun machine, there are the services provided by Sun RPC (sprayd, rexcd and the like). These are usually turned off because they are not very secure due to weak authentication policies.

Fourth, there are several services that are provided internally by inetd. They were originally designed for network testing and have been superseded by ping and traceroute. Such services include echo, which bounces a packet back to its originator, and discard, which acts like a /dev/null for network packets, consigning incoming packets to the great void. These services are not intrinsically risky but can be used by an aggressor who can flood your network in a denial of service attack.

TCP Wrappers

Hopefully, you will be left with a bunch of services that you wish to support on the machine. However, if the service is enabled in /etc/inetd.conf, then it will be available for anyone to use, and you may wish to have a policy that operates a finer grained control. Take for example, the infamous Berkeley "r" programs: rlogin, rsh and rcp. These programs were created as a quick hack for "short-term" use on the Berkeley network and have lingered on. They are convenient, but provide "back-door" password-less access to machines with authentication being provided by a control file in the user's home directory. Because they are convenient, people are often happy to have the programs operating on their local network but wish to prohibit access to them from the outside world. The standard inetd program does not provide this fine grain of access control, although there are versions of the program that do, I believe.



Another large problem with inetd is the lack of logging. With a vanilla system, it's up to the daemons that are forked by inetd to log accesses, and many of them just don't do that, or they only log successful attempts to use their services. So, for example, telnetd will call the login program to allow someone to log in, and a successful login is logged in the normal accounting files. Nothing logs unsuccessful attempts, so you will not see that cracker out there trying to guess passwords. Often the first indication of an attack on a system is a set of attempts to probe the services that are available on that machine, so it can be valuable to have this activity logged.

The current solution to the logging problem is to use a "TCP wrapper," the best of which is tcpd by Wietse Venema of the Department of Mathematics and Computing Science at Eindhoven University of Technology in the Netherlands. Venema calls the program suite: tcp_wrapper.

The idea behind tcpd is simple. For each service, the inetd program runs the wrapper program, rather than the daemon. The wrapper program logs the call to the daemon, optionally does some checks on where the call to the service originated, and then calls the daemon as if nothing had intervened. The name of the daemon and any arguments that it may need are provided as arguments to the wrapper program, so setting up tcpd involves some minor changes to inetd.conf.

Access control to services based on the name of the client machine is provided by a couple of files. The hosts.deny file allows you to specify that hosts can be refused access to certain activities. You can put the names of specific hosts, or use ALL to deny access to all hosts. The hosts.allow file permits access to specified services to certain hosts or networks. Hosts placed in here will override controls in the hosts.deny file. So, for example, I turn off access to the "r" program daemons for all hosts in the hosts.deny file but permit access to the programs from my network or certain other machines as the need arises.

Of course, the access control depends on the integrity of your network. It's important to ensure that packets that purport to come from your local machines have not been sent in from outside. The router that connects my network to the world is set up so that it will ignore any packets that arrive from the outside world pretending to come from a machine on my network. Filtering suspect packets that originate from an illogical place has become a basic precaution. If you don't do this, then you are vulnerable to several different attacks subverting programs that rely on the integrity of the data arriving from the network.

Also, using host names in the deny and allow control files means that you are relying on the integrity of the Domain Name Service (DNS). There have been attacks that used subverted address mapping from the DNS. However, it's not too easy to do a complete subversion job, and DNS is relatively secure. I imagine that in the fullness of time, we will have to move to some DNS system that uses digital keys to ensure data integrity. Sadly, I feel that governmental action in many countries inhibits the free use of encryption, and so really secure DNS is some way off.

The tcpd program logs activity using the standard syslog system. I tend to compile the program so that it logs activity to a local facility number. I can then divert its output to a file that solely contains data from the wrapper program. Once you have done this, you can process the log automatically. I run it through a sed script every day to strip out common activities, and if the resulting file contains any data, I mail the text to myself. This allows me to trap "odd" things happening on the machine. I've caught a few probes onto my network in this way. On both occasions, polite mail to the person named by the NIC as the administrator of the machine has resulted in an explanation of the activity. Incidentally, I wish that the people who made my router would understand the need for full and frank logging of packets that it filters out of the data stream.

It's important to realize that a probe from a machine may not originate from a legitimate user of that system, and that system may have been compromised. Polite mail is essential. You may be dealing with someone whose machine has been cracked and has no idea of what is going on. Generally, my email to the administrator makes the point that their system may have been invaded, largely with the intention that they will deal with the problem quickly before their logs (if any) have timed out.

Passwords

The aim of many crackers is to get a login on your system, because the possibilities of becoming root are vastly improved if the cracker can operate as a user on your machine. The easiest way to get access is to get a password from a person. Many passwords are obtained by “human engineering,” getting on the phone to the appropriate person and asking for a password. It should be site policy never to give passwords out unless you are sure of the identity of the person to whom you are talking. Human engineering takes courage, and so there is considerable focus for crackers to find ways to obtain the password file from your system automatically.

In his talk at SANS 97, Randy Marchany (from the Virginia Tech Computing Center) said that he found several password files from many machines obtained by the cracker that invaded his system. He had no idea where many of these files originated. He advises people to put something in the password file that identifies the home system so that the proper owner can be advised. I’ve decided to put my home URL in the name field in the password file for the `www` login on my machines.

Your machine will provide many services that deliver files over the network, and these represent attack points to obtain your password file. Some of the services run with superuser privilege and so can be used to send the shadow password file should your system use this method of defense. The prime targets in the past have been `sendmail` and your Web server. It’s a good idea to replace the current version of `sendmail` on your machine with the latest version from `http://www.sendmail.org`. Eric Allman takes security breaches very seriously and works hard to eliminate them from his code.

The CGI script directory is a common source of security holes on your Web server. Remove or disable (by ensuring that the file is unreadable) any CGI script that you are not using. A recent spate of attacks have used a script called `nph-test-cgi` that is distributed with several `httpd` programs. You should instantly check whether you have this and disable or fix it (see the appropriate CERT advisory notice for details).

If someone gets hold of your password file, then there are several programs available over the Net that can be used to break passwords. The main technique is brute force, the programs use a dictionary to supply a basic set of words, encrypt each word and see if the encryption matches the password in the file. The best cracking programs apply common transformations of the words. For example, they check for capitalizations or try replacing the letter “I” with the digit one or the letter “O” with zero.

Perhaps the best password cracking program for UNIX is `Crack`, written by Alec Muffett. It’s currently at Version 5.0, I’ve come across several older versions sitting on machines on the Net. The program has been around for several years, and I recall the misplaced furor that ensued on Usenet when Muffett first made the program freely available.

If you obtain and run this program on your machine, and I suggest that you do, I predict that you will crack about 10% of your user’s passwords. People don’t often understand the need to use strong passwords. They see the password as an annoyance. If you insist on using password aging on your system, then I predict that you will get more hits from the `Crack` program. I have said for years that password aging is a bad idea. It seems hard to get people to believe that aging systems force people to create a personal algorithm used to generate their current password. The algorithm is bound to result in weak passwords. I said in my article "Passwords," February 1994, Page 24: "It seems a good idea to change passwords regularly. It doesn't seem a good idea to force this on people. Password aging systems are enthusiastically stupid and I believe that they may result in worse security." I was sent email from several people who didn't believe me.

It really is better to insist that users generate a good password and stick with it. Thinking of a good, memorable password is quite hard. Once you have one that is not easily broken, then why change it?

It really is better to insist that users generate a good password and stick with it. Thinking of a good, memorable password is quite hard. Once you have one that is not easily broken, then why change it?

Of course, all the password security in the world doesn't help us much when logging into remote machines over the Internet because the passwords travel down the wire as plain text. Many people have access to that text. There has been at least one incident where crackers stored a packet sniffing program on a network router and captured passwords as they flew by. If you are logging in from a public terminal at a conference or perhaps visiting a Cybercafe to read your mail, then you need to be deeply suspicious about typing any passwords.

I use two forms of defense. The first is the S/KEY system by Phil Karn, who conceived S/KEY when working for Bellcore in New Jersey. The code implements a challenge/response login system for UNIX, providing a one-time password. The idea of a challenge/response system is that when a user logs in, the host types something and the user responds. The "something" is different every time and the user has to compute the response in some way using the challenge as a key. The computation can be done with a handheld unit. The user types the challenge into the unit and the unit returns a number that is the response. The response is typed into the machine. I use my Psion 3a machine as my handheld unit. You can get S/KEY to print a list of challenges and their passwords. You then travel around with a bit of paper that you need to take great care of, print it using a small font and keep it in a safe place.

I have started to use a more recent system for security. The `ssh` system provides encrypted communications between hosts. The system consists of a client that replaces programs such as `rlogin` and `rcp` with a secure encrypted communications path, talking to a daemon, `sshd`, running on the remote machine. The `ssh` system was created by Tatu Ylonen from Finland and is an excellent piece of software.

Security Resources

Thanks to Randy Marchany for two excellent talks at SANS 97 that gave me the idea to write this article. I wrote an article that described the role of `inetd` some two years ago ("TCP/IP and UNIX," September 1995, Page 26).

You can always find helpful security information on the CERT site: `http://www.cert.org`. Details about the `nph-test-cgi` problem can also be found on the CERT site. Following the link `Cert Advisories` puts you into the FTP site, and the file that you want is `CA-97.07.nph-test-cgi_script`. In addition, the CERT FTP site features many tools that can help you ensure the integrity of your system. You'll find the current version of `Crack` here too, see `ftp://ftp.cert.org/pub/tools/crack`.

The `tcp_wrapper` suite is available from `ftp://ftp.win.tue.nl/pub/security`; the current version is 7.6. S/KEY is a trademark of Bellcore and is available from `ftp://ftp.bellcore.com/pub/nmh/skey`.

Finally, the latest version of `ssh` is available for testing, see `http://www.cs.hut.fi/ssh`.

Peter Collinson *runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever... He writes, teaches, consults and programs using Solaris running on a SPARCstation 2. Email:* `pc@cpg.com`.