Issue Date: IBM edition for UNIX
December 2004, Posted On: 12/16/2004

Building Reinforcements

Jaqui Lynch

By default, UNIX* comes with many different daemons (programs that perform a housekeeping or maintenance-utility function without being called by the user) and services enabled. Different UNIX brands have different default behaviors. Not all administrators know this, but some of these daemons and services have known security holes and should be turned off. This step-by-step article explains the services and how to disable them on AIX* while still retaining a functional system.

To best protect your system at the network level, several steps must be taken. First, analyze the /etc/inetd.conf file and determine which services you really need. Make a copy of the file, and then delete all of the unnecessary services from the file—not commented out but deleted. If they're commented out, you run the risk that the next maintenance application will uncomment them, whereas it's rare that a patch would add back a service. Any time maintenance is applied, check the file to ensure that nothing was added back in. Another alternative is to leave the services there but set them all to /bin/false so they never successfully execute. You also need to modify /etc/rc.tcpip to stop certain daemons from starting at boot.

The other steps involve ensuring that the system is fully patched, system logging is set up correctly, TCP Wrappers is installed and tools are implemented to help with monitoring the status (i.e., lsof and port scanners).

Logging

Logging is a critical part of any system-protection strategy. Without logs, it's impossible to know what's been happening on the system. The syslog daemon (syslogd) starts by default on AIX, but the log configuration file isn't set up to actually log. So the first step is to correctly set up /etc/syslog.conf. It's best to set up a separate file system for logs (e.g., /usr/local/logs) rather than use the default of /var/spool. If /var fills up, the system will crash; if your separate file system fills up, it will just stop logging. Although file systems should be monitored, it's still wise to store logs in their own file system to protect against large logs bringing down the system.

Logs can be written to a file, sent to the console, logged to a central host across the network (be wary of this as the traffic can be substantial), e-mailed to an administrator or sent to all logged-in users. The most commonly used method is writing to a file in a file system. Once the file system is set up, code a /etc/syslog.conf file. Here's an example file:

```
*.emerg  /usr/local/logs/syslog
*.alert  /usr/local/logs/syslog
*.err  /usr/local/logs/syslog
auth.notice /usr/local/logs/authlog
mail.debug /usr/local/logs/mailog
daemon.info /usr/local/logs/infolog
```

In this example file, all emergency, alert and error messages are written to the /usr/local/logs/syslog file. Many administrators also have messages go to the console—I've found that most customers now use monitors on a switch so those messages tend to get missed. Therefore, I usually send all of my logs to a file and use scripts to scan them and e-mail or page people as necessary. Authentication messages should go to a separate authlog and mail messages to a separate mailog. Daemon messages are set to go to infolog—daemon.info is where certain daemons, including TCP Wrappers, are configured to log to. This log separation makes it easier to search for patterns and problems. Once the syslog.conf file is coded, each log file used must be created using the touch command (e.g., touch /usr/local/logs/syslog).

Although you can refresh the syslog daemon, I've found this doesn't always work. I tend to stop and start the syslog daemon as follows:

```
stopsrc -s syslogd
startsrc -s syslogd
```

Time Synchronization

Timestamps on all systems must be correct and consistent. Some type of time-synchronization mechanism, such as Network

Time Protocol (NTP), should be implemented. Typically, one server is set up as the timeserver and all of the other servers point to it to synchronize their time. In large networks, it may be necessary to set up a time hierarchy so one server isn't overwhelmed by requests. This helps with any type of analysis you may end up doing later if you have to investigate an event. Cross-referencing multiple logs that are minutes off is a nightmare.

Services
By default, the /etc/inetd.conf includes many services that are insecure or contain bugs. These include ttdbserver (tool talk server), cmsd (calendar server), ntalk, rlogin, rsh, etc. Also, insecure services such as telnetd and ftp are enabled.

According to the SANS Institute, a cooperative research and education organization providing information, security training and certification, the top 10 UNIX system vulnerabilities are:

1. Bind and DNS
2. Remote procedure calls
3. Apache Web server
4. General authentication due to weak or no passwords
5. Clear-text services such as Telnet
6. Sendmail
7. SNMP
8. Secure Shell (SSH) exploits
9. NFS/NIS misconfiguration
10. OpenSSL exploits

Most, if not all of these, are on by default. The recommendation is to install SSH and scp. Once that's done, disable all non-critical services in /etc/inetd.conf. Disabling them means copying the file as a backup, deleting the entries in it and then refreshing the inet daemon using refresh -s inetd.

Don't logoff during this time until you're sure you can connect to the system remotely or you have direct console access.

Additional services are started from /etc/inittab or /etc/rc.tcpip. The latter should be edited to disable services that aren't needed. For example, if you aren't using SNMP, comment it out and also comment out dpid2 in rc.tcpip. This is also where you comment out the startup of sendmail, which, by default, starts at boot time. Once everything is commented out or deleted, reboot if possible. If a reboot can't be scheduled, stop the services such as sendmail and dpid2, and refresh the inetd daemon so it re-reads its configuration file.

It's also useful to secure disallowed services in case someone accidentally enables them later. An example of this would be tftp. If you code a /etc/tftpaccess.ctl file, any successful tftp login can only access the directories or files listed. An empty tftpaccess.ctl file means there's no access. If you're in an environment that depends on tftp and bootp, specific directories must be enabled in the access file.

You should also regularly scan the system for files that open the system for insecure access. These include .netrc, .rhosts, .shosts and .exrc files. Check the .forward files to ensure they aren't executables, and regularly scrutinize the permissions on .login and .profile files. Also check the /etc/hosts.equiv file (and shosts.equiv) to ensure no one has added remote access without password requirements. I usually delete all of the lines in this file so that it's empty. The same applies to the /etc/hosts.lpd file, unless this is a print server. Both hosts.equiv and hosts.lpd should have group-write access removed in their permissions.

Although many companies have policies against downloading third-party software, this has now become a critical component in securing your AIX system. In fact, now it's virtually impossible to secure a system well without some of these tools. Section 5 of IBM's Redbook, "Additional AIX Security Tools on IBM* eServer pSeries*, IBM RS/6000* and SP*/Cluster" (SG24-5971), has an informative section on some of these tools.

There are several tools that no system should be without, including SSH, sudo, TCP Wrappers and lsof. It's also worth looking at stunnel and portmap.

Invaluable Tools

TCP Wrappers—The purpose of TCP Wrappers is to wrap a service such as Telnet so you can perform security checks before allowing or disallowing access to the service. This program is called by inetd before calling a service, and the wrapper checks two rules files (/etc/hosts.deny and /etc/hosts.allow), logs the attempt, authorizes or denies the attempt and builds an audit trail. It only does this for services that have been told to take advantage of the wrapper. To do this, you must first configure logging and install the tcpd program in /usr/local/bin. Once that's done, the inetd.conf entry for a service such as ftp is changed as follows:

```
Old:
ftp  stream  tcp6  nowait  root  /usr/sbin/ftpd -l  ftpd

New:
ftp  stream  tcp6  nowait  root  /usr/local/bin/tcpd /usr/sbin/ftpd -l  ftpd
```

As you can see, the program tcpd is inserted to execute before the ftpd program. Another option is to actually replace ftpd with the wrapper. However, this causes problems when maintenance is run, so it's best to install the wrapper as a separate program and execute it as shown above.

It's also possible to configure a service to take advantage of the wrapper logging without being able to execute:

```
login   stream  tcp6    nowait  root    /usr/local/bin/tcpd /bin/false  rlogind
```

In the line above, the wrapper will log attempts to use the service, but the service has been set to execute /bin/false rather than the real daemon. Because the wrapper logs for any service that calls it, this is a way to get log entries for attempted break-ins without allowing the service to actually run.

TCP Wrappers uses two files to control such access—/etc/hosts.deny controls the denying of access, and /etc/hosts.allow controls the allowing of access. To keep things simple, it's best to put "all:all" in the /etc/hosts.deny file. This means that all access to the listed services is denied unless it's explicitly allowed in the hosts.allow file. The hosts.allow file can be configured to post a banner for each service, whether the service is granted or not. This is one way to help ensure that users see the acceptable use policy (AUP). It can also be used to execute a command when a connection is denied or to issue an ident lookup when someone attempts a connection. Rules can be coded using IP addresses, domain or system names, or hostmask/ip combinations. There are two special keywords: all, which means anyone; and LOCAL, which means the system itself. If you want someone on the system to Telnet to it, you must include the LOCAL option, IP or name of the server in the allowed list:

```
Sample /etc/hosts.allow
portmap: 123.123.  255.255.255.
ftpd : .abc.com,123.123.123.4
sshd : all
telnetd : 123.123.123.0/255.255.255.0
xmservd : .abc.com,123.123.123.4
rexecd : LOCAL,.abc.com,123.123.123.4
```

Again, keep in mind that you shouldn't disconnect from the system until you're certain the wrappers are working correctly and that you'll be able to log back on.

SSH—This tool was written to replace insecure protocols such as rlogin, rcp, ftp and Telnet with more secure alternatives. SSH provides authentication, encryption and data integrity across the Internet. Since SSH interfaces with TCP Wrappers, the wrappers should be installed and functional before installing SSH. Unlike protocols such as Telnet, SSH encrypts all authentication traffic, helping ensure that user names and passwords don't travel in the clear. SSH also allows the option of standard UNIX passwords or the use of a public/private key pair for authentication. SSH interfaces with TCP Wrappers for logging and access control and has its own built-in access control. The tool includes many features—such as the ability to do secure backups, tunneling and X11 forwarding—all within a secure environment. SSH also comes with a secure FTP server, sftp.

LSOF—This program lists open files on a system. Since all network connections are open files, it's possible to get

information about who's connected to a port. Commands of use are:

```
lsof | grep TCP | more
lsof | grep UDP | more
```

Portmap—A few of the current UNIX vendors still ship a version of portmap that allows anyone to read or modify its tables, and will forward any request so it appears to come from the local system. Wietse Venema, who now works at IBM's Watson Research Laboratory, rewrote portmap to incorporate access control. The replacement product can be used to lock down access to services such as NIS, NFS and other RPC-based services. Its style is similar to the TCP Wrappers package and provides security libraries and access control lists to some of the services.

Stunnel—This tool provides a wrapper utility for encrypting TCP sessions using SSL. It requires users to download and install Openssl and can be used to secure additional daemons such as imapd, pop and ldap. It has built-in TCP Wrappers support and uses the /etc.hosts.allow file for security rules.

Mail Servers—By default, the sendmail server with AIX is enabled and isn't configured correctly for mail-server use. This must be addressed by turning off sendmail, configuring it correctly or replacing it with Postfix, which can be obtained from Venema's Web site. Regardless of whether sendmail is enabled, an aliases file (usually /etc/aliases) should be created so root and postmaster mail have somewhere to go. Once this file is updated, it's necessary to run the newaliases or sendmail -bi command to prompt the mail server to use the new set of aliases.

A Starting Point

It takes little time to secure an AIX system's services. However, one of the biggest issues is the resistance to using open-source software. If a system isn't running TCP Wrappers and SSH and is instead using generic logging and Telnet, then the system is exposed and attempts to access it are most likely not being logged. The recommendations I outline are basic, but they provide a great start for locking down your AIX system and keeping out people who aren't invited.

About the Author(s):

Jaqui Lynch: Jaqui Lynch, an *eServer Magazine, IBM edition for UNIX* technical editor, is a senior systems engineer focusing on pSeries and Linux at Mainline Information Systems. During her more than 26 years in the IS industry, she's been responsible for a wide variety of projects and OSs across multiple vendor platforms, including mainframes, UNIX systems, midrange systems and personal workstations. Jaqui can be reached at jaqui.lynch@mainline.com.