Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Synopsis

There is a security principle that says you should *Configure computers to provide only selected network services* (CERT® Coordination Centre). The basic idea is this: every network service you offer is an opportunity for the bad guys (alternatively a risk to your system). That's not to say that you shouldn't offer any services -- a mail server that doesn't offer mail services isn't very useful. Instead, the principle says you should have a good understanding of network services and you should not offer any service unless there are very good reasons for doing so. This paper is a discussion of network hardening measures for AIX 4.3 -- an application of the security principle.

Some security packages address the problem by stripping all (or nearly all) network services and then instruct you to be careful about what you add to the system. That's a great approach but requires that you "get your hands on" the system before anyone layers anything onto it and you understand what you're adding to the system when you add it back in. Two conditions that don't apply at many sites.

The approach here is different. We consider services offered by the AIX 4.3 operating system, try to explain what each does, note the risks involved with each and make recommendations about what one ought to do to mitigate the risk. For each issue discussed a hardening tool (a shell script) is provided to handle the issue. Typically, the issue is handled by removing a service that you don't need. Several policies are provided to harden typical systems and an implementation tool is provided to enforce the policy.

A traditional Unix tar kit containing all scripts to harden each issue discussed, sample polices and a driver to implement policies is available here.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/02/01

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Synopsis

There is a security principle that says you should *Configure computers to provide only selected network services* (CERT® Coordination Centre). The basic idea is this: every network service you offer is an opportunity for the bad guys (alternatively a risk to your system). That's not to say that you shouldn't offer any services -- a mail server that doesn't offer mail services isn't very useful. Instead, the principle says you should have a good understanding of network services and you should not offer any service unless there are very good reasons for doing so. This paper is a discussion of network hardening measures for AIX 4.3 -- an application of the security principle.

Some security packages address the problem by stripping all (or nearly all) network services and then instruct you to be careful about what you add to the system. That's a great approach but requires that you "get your hands on" the system before anyone layers anything onto it and you understand what you're adding to the system when you add it back in. Two conditions that don't apply at many sites.

The approach here is different. We consider services offered by the AIX 4.3 operating system, try to explain what each does, note the risks involved with each and make recommendations about what one ought to do to mitigate the risk. For each issue discussed a hardening tool (a shell script) is provided to handle the issue. Typically, the issue is handled by removing a service that you don't need. Several policies are provided to harden typical systems and an implementation tool is provided to enforce the policy.

A traditional Unix tar kit containing all scripts to harden each issue discussed, sample polices and a driver to implement policies is available here.

Reg Quinton, *Information Systems and Technology*
2001/01/15-2001/02/01

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Baseline -- What's there?

Before going anywhere you need to know what's on the system and better yet -- how to find that out. There are three tools we find invaluable -- **netstat(1m)** and **rpcinfo(1m)** as provided by the vendor and **lsof(8)** a public domain add on. All can be used to identify network services that your system offers to clients on the network. Services that might be exploited.

 If **lsof** is not installed on your system the sources can be found at Purdue and elsewhere. Get a copy, configure, build and install -- it's easy.

The *netstat(1m)* command

To determine the services that your system offers try this command:

```
[4:32pm wally]# netstat -af inet
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  wally.uwaterl.login    zippy.uwaterloo..1022  ESTABLISHED
tcp4       0      0  wally.uwaterl.22       wally.uwaterloo..1019  ESTABLISHED
tcp4       0      0  *.22                   *.*                    LISTEN
tcp4       0      0  *.7937                 *.*                    LISTEN
tcp4       0      0  *.7938                 *.*                    LISTEN
tcp4       0      0  *.writesrv             *.*                    LISTEN
tcp4       0      0  *.32772                *.*                    LISTEN
tcp4       0      0  *.803                  *.*                    LISTEN
tcp4       0      0  *.802                  *.*                    LISTEN
tcp4       0      0  *.32771                *.*                    LISTEN
tcp4       0      0  localhost.smux         localhost.32770        ESTABLISHED
tcp4       0      0  localhost.32770        localhost.smux         ESTABLISHED
tcp4       0      0  *.smux                 *.*                    LISTEN
tcp4       0      0  *.32769                *.*                    LISTEN
tcp4       0      0  *.klogin               *.*                    LISTEN
tcp4       0      0  *.kshell               *.*                    LISTEN
tcp4       0      0  *.dtspc                *.*                    LISTEN
tcp4       0      0  *.time                 *.*                    LISTEN
tcp4       0      0  *.daytime              *.*                    LISTEN
tcp4       0      0  *.chargen              *.*                    LISTEN
tcp4       0      0  *.discard              *.*                    LISTEN
tcp4       0      0  *.echo                 *.*                    LISTEN
tcp        0      0  *.exec                 *.*                    LISTEN
tcp        0      0  *.login                *.*                    LISTEN
tcp        0      0  *.shell                *.*                    LISTEN
tcp        0      0  *.telnet               *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
tcp4       0      0  *.sunrpc               *.*                    LISTEN
tcp4       0      0  *.6000                 *.*                    LISTEN
tcp4       0      0  *.32768                *.*                    LISTEN
udp4       0      0  *.echo                 *.*
udp4       0      0  *.discard              *.*
udp4       0      0  *.daytime              *.*
udp4       0      0  *.chargen              *.*
udp4       0      0  *.time                 *.*
udp4       0      0  *.sunrpc               *.*
udp4       0      0  *.loc-srv              *.*
udp4       0      0  *.snmp                 *.*
udp4       0      0  *.xdmcp                *.*
udp4       0      0  *.who                  *.*
udp4       0      0  *.syslog               *.*
udp4       0      0  *.ntalk                *.*
udp4       0      0  *.802                  *.*
udp4       0      0  *.803                  *.*
udp4       0      0  *.1515                 *.*
udp4       0      0  *.32770                *.*
udp4       0      0  *.32772                *.*
```

```
udp4       0      0  *.32773                *.*
udp4       0      0  *.32774                *.*
udp4       0      0  *.32775                *.*
udp4       0      0  *.32776                *.*
udp4       0      0  *.32777                *.*
udp4       0      0  *.32788                *.*
udp4       0      0  *.32803                *.*
udp4    1024      0  *.32805                *.*
udp4     900      0  *.32807                *.*
udp4     496      0  *.32809                *.*
udp4   11340      0  *.32827                *.*
udp4       0      0  *.7938                 *.*
```

On a typical AIX system you'll find there's lots going on. The display above is broken into two parts -- UDP services (User Datagram Protocol -- that's a messaging service built on IP) and TCP services (Transmission Control Protocol -- that's a connection orientated service built on IP). UDP and TCP services are Internet services available to the world at large. You should be concerned about both services types (this is complicated further on AIX 4.3 which implements IPv4 and the newer IPv6 -- you'll see UDP4 vs UDP and TCP4 vs TCP).

> Network services are either TCP or UDP based. In the **"netstat -a"** output look for TCP services in a "LISTEN" state and UDP services to determine the services that your system offers to the world.

In the example above it should be apparent that there's a TCP service with the **"ftp"** name (you will discover that's also port 21) in a "LISTEN" state -- it's waiting for a client to connect and the two processes will then start talking to one another. There's also a UDP service at the **"ntalk"** service (again, you will discover that's port 518) -- it's waiting for clients to send it messages (to which it will respond). The mapping of service names -- like **ftp**, **name**, **telnet**, **smtp**, etc. -- to their corresponding number is tabled in the file **/etc/services**. If you search **/etc/services** you will discover the port numbers for the **ftp** and **ntalk** services. The Internet Assigned Numbers Authority (IANA) manages the official names for various IP ports but it's not unusual for Unix systems to have incomplete data in **/etc/services**.

The command **"netstat -a"** shows all active network connections and attempts to map IP numbers to host names and port numbers to service names -- where it cannot do the number to name mapping it will show the number instead. The command **"netstat -an"** will display network connections without any attempt to map numbers to names -- that's another way you'll discover that the **ftp** service is located at port 21 while the **ntalk** service is located at port 518. You should become familiar with this tool; start by reading the manual page.

You may already know that it's the **sendmail(8)** daemon that's providing the **smtp** service (on port 25); or you should know that it's the Apache web server you installed some time ago that's providing the **http** service (on port 80). But in general, determining the process that stands behind a port (alternatively, the process that provides a service) can be difficult. You can rely on documents like this, or you can import a tool to help you. I am not aware of any vendor provided tools on AIX that will do this for you (on Linux the "netstat -ap" command will show the server process by name and number).

The *lsof(8)* command

Lsof(8) is an excellent tool (but it's not part of the AIX distribution) that you can use to determine the processes that provide network services. At our site we install **lsof(8)** on most Unix systems including AIX. Here's how you can use it to examine network services you offer:

```
[4:41pm wally]# lsof -i | egrep "COMMAND|LISTEN|UDP"
COMMAND     PID USER   FD   TYPE     DEVICE SIZE/OFF NODE NAME
dtlogin    2122 root    5u  IPv4 0x70053c00      0t0  UDP *:xdmcp
dtlogin    2122 root    6u  IPv4 0x70054adc      0t0  TCP *:32768 (LISTEN)
syslogd    2730 root    4u  IPv4 0x70053600      0t0  UDP *:syslog
X          2880 root    6u  IPv4 0x70054adc      0t0  TCP *:32768 (LISTEN)
X          2880 root    8u  IPv4 0x700546dc      0t0  TCP *:6000 (LISTEN)
dtlogin    3882 root    6u  IPv4 0x70054adc      0t0  TCP *:32768 (LISTEN)
glbd       4154 root    4u  IPv4 0x7003f300      0t0  UDP *:32803
glbd       4154 root    9u  IPv4 0x7003f700      0t0  UDP *:32805
dtgreet    4656 root    6u  IPv4 0x70054adc      0t0  TCP *:32768 (LISTEN)
i4lmd      5228 root   14u  IPv4 0x7003fe00      0t0  UDP *:1515
i4lmd      5228 root   15u  IPv4 0x70039500      0t0  UDP *:32807
i4lmd      5228 root   16u  IPv4 0x70039400      0t0  UDP *:32809
inetd      5676 root    4u  IPv4 0x700556dc      0t0  TCP *:ftp (LISTEN)
inetd      5676 root    5u  IPv4 0x700552dc      0t0  TCP *:telnet (LISTEN)
inetd      5676 root    6u  IPv4 0x70040edc      0t0  TCP *:shell (LISTEN)
inetd      5676 root    7u  IPv4 0x70040adc      0t0  TCP *:login (LISTEN)
inetd      5676 root    8u  IPv4 0x700406dc      0t0  TCP *:exec (LISTEN)
inetd      5676 root    9u  IPv4 0x70049e00      0t0  UDP *:ntalk
inetd      5676 root   10u  IPv4 0x70049d00      0t0  UDP *:32772
```

```
inetd      5676 root   11u  IPv4 0x70049c00      0t0  UDP *:32773
inetd      5676 root   12u  IPv4 0x70053100      0t0  UDP *:32774
inetd      5676 root   14u  IPv4 0x70049900      0t0  UDP *:32775
inetd      5676 root   15u  IPv4 0x70049b00      0t0  UDP *:32776
inetd      5676 root   16u  IPv4 0x700402dc      0t0  TCP *:echo (LISTEN)
inetd      5676 root   17u  IPv4 0x70042edc      0t0  TCP *:discard (LISTEN)
inetd      5676 root   18u  IPv4 0x70042adc      0t0  TCP *:chargen (LISTEN)
inetd      5676 root   19u  IPv4 0x700426dc      0t0  TCP *:daytime (LISTEN)
inetd      5676 root   20u  IPv4 0x700422dc      0t0  TCP *:time (LISTEN)
inetd      5676 root   21u  IPv4 0x70049a00      0t0  UDP *:echo
inetd      5676 root   22u  IPv4 0x70049700      0t0  UDP *:discard
inetd      5676 root   23u  IPv4 0x70049600      0t0  UDP *:chargen
inetd      5676 root   24u  IPv4 0x70049f00      0t0  UDP *:daytime
inetd      5676 root   25u  IPv4 0x70049500      0t0  UDP *:time
inetd      5676 root   26u  IPv4 0x7003cedc      0t0  TCP *:dtspc (LISTEN)
inetd      5676 root   27u  IPv4 0x7003cadc      0t0  TCP *:kshell (LISTEN)
inetd      5676 root   28u  IPv4 0x7003c6dc      0t0  TCP *:klogin (LISTEN)
inetd      5676 root   29u  IPv4 0x70049800      0t0  UDP *:32777
inetd      5676 root   30u  IPv4 0x7003c2dc      0t0  TCP *:32769 (LISTEN)
rwhod      5934 root    5u  IPv4 0x70049200      0t0  UDP *:who
snmpd      6192 root    3u  IPv4 0x70049100      0t0  UDP *:snmp
snmpd      6192 root   10u  IPv4 0x7003eedc      0t0  TCP *:smux (LISTEN)
dpid2      6450 root    4u  IPv4 0x7003e2dc      0t0  TCP *:32771 (LISTEN)
nsrexecd   7762 root    5u  IPv4 0x7003aadc      0t0  TCP *:7937 (LISTEN)
rpc.statd  8258 root   20u  IPv4 0x7003fa00      0t0  UDP *:802
rpc.statd  8258 root   21u  IPv4 0x70038adc      0t0  TCP *:802 (LISTEN)
rpc.statd  8258 root   22u  IPv4 0x7003f900      0t0  UDP *:803
rpc.statd  8258 root   23u  IPv4 0x70038edc      0t0  TCP *:803 (LISTEN)
writesrv   9038 root    3u  IPv4 0x7003aedc      0t0  TCP *:writesrv (LISTEN)
llbd       9312 root    4u  IPv4 0x7003f400      0t0  UDP *:loc-srv
nsrexecd  10364 root    5u  IPv4 0x70039000      0t0  UDP *:7938
nsrexecd  10364 root    6u  IPv4 0x700542dc      0t0  TCP *:7938 (LISTEN)
i4lmd     10588 root   14u  IPv4 0x7003fe00      0t0  UDP *:1515
i4lmd     10588 root   15u  IPv4 0x70039500      0t0  UDP *:32807
i4lmd     10588 root   16u  IPv4 0x70039400      0t0  UDP *:32809
i4glbcd   11096 root    4u  IPv4 0x70032000      0t0  UDP *:32827
sshd      12068 root    3u  IPv4 0x700372dc      0t0  TCP *:22 (LISTEN)
```

Note that browsing all network connections with **lsof(8)** will usually require **root** user privileges. Typically **lsof(8)** is installed with only enough privileges to view network connections attached to your processes -- you aren't allowed to peek at other processes unless you are the **root** user.

You can use **lsof(8)** to determine the services you offer to the network -- look for TCP sockets in a LISTEN state and UDP sockets (they're all in an "Idle" state waiting for data). The **lsof(8)** command also shows the **program**. The listing even shows the process id -- the **sshd** program is running as process id 12068 and that's the process offering a TCP service at port 22. Now we have the information we need:

```
[4:51pm wally]# ps -fp 12068
     UID   PID  PPID   C    STIME    TTY  TIME CMD
    root 12068     1   0   Jan 12      -  2:05 /software/ssh-1/servers/sshd
[4:51pm wally] man sshd

SSHD(8)                         SSH                          SSHD(8)

NAME
     sshd - secure shell daemon

SYNOPSIS
     sshd [-b bits] [-d ] [-f config_file] [-g login_grace_time]
     [-h host_key_file] [-i ] [-k key_gen_time] [-p port] [-q ]
     [-V version]
   ...etc.
```

Once you've determined the program behind the service it's usually not very hard to figure out what's going on -- read the manual page for the program! With a little investigation we've found out that we offer a TCP service on port 22 with the **sshd(8)** program -- the secure shell daemon that peers with the **ssh(1)** and **scp(1)** commands. That port is not listed in my **/etc/services** (it should be, the IANA has named it the "**ssh**" service). **Ssh** services are an important security enhanced replacement for a great wack of older less secure services -- telnet, rlogin, rsh and ftp!

Another good place to find out what a daemon is doing is the AIX - RS/6000 Documentation Library. It's on-line, free and has a good search engine.

Services provided by *inetd(8)*

There's a couple of wrinkles you need to be aware of. First, note in the **lsof(8)** example that it's the **inetd(8)** process that's listening for the **ftp** service -- that's a bit of a surprise, you'd think it would be the **ftpd(8)** daemon. The **inetd(8)** manual page will tell you how the **inetd** process is driven by a configuration file **/etc/inetd.conf** that lists services it should listen for and applications it should run when a connection is made. The application **inetd** invokes when a connection is established to the **ftp** service is in fact the **ftpd(8)** daemon:

```
[4:56pm wally]# grep ftp /etc/inetd.conf
ftp     stream  tcp6    nowait  root    /usr/sbin/ftpd         ftpd
#tftp    dgram  udp6    SRC     nobody  /usr/sbin/tftpd         tftpd -n
```

Note in the **lsof(8)** example that other processes stand on their own -- they're not run configured by **inetd**. In the example you'll see **portmap**, **dtlogin** and **sshd** processes. Those are started at system boot time. You won't find them listed in the **/etc/inetd.conf** configuration file.

Boot Scripts-- a digression: AIX 4.3 is a traditional Unix system in as much as the first process started is **init(8)** and that process is configured by **/etc/inittab**. Unlike Solaris, Linux and many other Unix variants AIX *does not* take advantage of the **/etc/rc.d** directory structure where a driver script runs other scripts found in run level directories (it's there in recent versions but it's not used). Instead you'll find many daemons started directly forn **/etc/inittab**. For example:

```
[10:24am wally] grep -i  daemon /etc/inittab
rctcpip:2:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
rcnfs:2:wait:/etc/rc.nfs > /dev/console 2>&1 # Start NFS Daemons
qdaemon:2:wait:/usr/bin/startsrc -sqdaemon
pmd:2:wait:/usr/bin/pmd > /dev/console 2>&1 # Start PM daemon
```

We'll come back to these shortly. The important thing to bear in mind though is that network daemons are either run out of **inetd(8)** or started at boot time. Here's an interesting exercise -- find the boot script in **/etc/inittab** that starts **inetd(8)**.

The second wrinkle you need to know about is Remote Procedure Call (RPC) services. RPC services are typically layered on top of TCP or UDP services. You will discover that **inetd(8)** is usually configured to provide TCP, UDP and RPC services. Here's an example RPC service offered by **inetd(8)**:

```
[4:57pm wally]# grep rpc /etc/inetd.conf
##      NOTE: The services with socket type of "sunrpc_tcp" and "sunrpc_udp"
#rquotad  sunrpc_udp     udp     wait    root    /usr/sbin/rpc.rquotad rquotad 100011 1
#rexd    sunrpc_tcp     tcp     wait    root    /usr/sbin/rpc.rexd rexd 100017 1
rstatd   sunrpc_udp     udp     wait    root    /usr/sbin/rpc.rstatd rstatd 100001 1-3
rusersd sunrpc_udp      udp     wait    root    /usr/lib/netsvc/rusers/rpc.rusersd rusersd 100002
1-2
rwalld   sunrpc_udp     udp     wait    root    /usr/lib/netsvc/rwall/rpc.rwalld rwalld 100008 1
sprayd   sunrpc_udp     udp     wait    root    /usr/lib/netsvc/spray/rpc.sprayd sprayd 100012 1
pcnfsd   sunrpc_udp     udp     wait    root    /usr/sbin/rpc.pcnfsd pcnfsd 150001 1-2
cmsd    sunrpc_udp      udp     wait    root    /usr/dt/bin/rpc.cmsd cmsd 100068 2-5
ttdbserver      sunrpc_tcp      tcp     wait    root    /usr/dt/bin/rpc.ttdbserver rpc.ttdbserver
100083 1
```

That nasty thing about RPC services is their RPC program number (eg. 100068 in the **rpc.cmsd** example) have *nothing to do with the TCP/UDP port number where the service is found!* Many RPC services are dynamically bound to whatever ports are available when the service is started. Services are registered with the **portmap(8)** daemon and that daemon, if present, should always be found at TCP and UDP ports 111 (the **sunrpc** service). Client RPC applications have to contact the **portmap(8)** daemon to locate the service they're after. Server RPC applications have to tell the **portmap(8)** daemon where they are. The **portmap(8)** daemon is always found at port 111 so clients and servers can find it!

If **netstat(1m)** and **lsof(8)** show strange numbers that don't obviously map to services in **/etc/inetd.conf** you should start looking for a corresponding RPC service. Note that **lsof(8)** may show some RPC services provided by applications other than **inetd** -- the **sunrpc** service is provided by the **portmap** daemon which is started at boot time. Those are easy to figure out, the ones provided by **inetd** are a little harder.

The *rpcinfo(1m)* command

There's a adminstrator's tool to table all RPC services -- see **rpcinfo(1m)**. The tool contacts the **portmap(8)** daemon and gets a "directory" of RPC services that have been registered there. In the example which follows I use the **rpcinfo** command to determine where all the RPC services are -- you'll find program 100068 (version 2-5) is at TCP port 32781. The RPC program numbered "100068" is implemented by the **rpc.cmsd** program found in **/usr/dt/bin** and that program is provided to the client by **inetd** at TCP port 32781.

```
[4:58pm wally]# rpcinfo -p
```

```
program vers proto   port  service
 100000    4   tcp    111  portmapper
 100000    3   tcp    111  portmapper
 100000    2   tcp    111  portmapper
 100000    4   udp    111  portmapper
 100000    3   udp    111  portmapper
 100000    2   udp    111  portmapper
 100001    1   udp  32776  rstatd
 100001    2   udp  32776  rstatd
 100001    3   udp  32776  rstatd
 100002    1   udp  32777  rusersd
 100002    2   udp  32777  rusersd
 100008    1   udp  32778  walld
 100012    1   udp  32779  sprayd
 150001    1   udp  32780  pcnfsd
 150001    2   udp  32780  pcnfsd
 100068    2   udp  32781  cmsd
 100068    3   udp  32781  cmsd
 100068    4   udp  32781  cmsd
 100068    5   udp  32781  cmsd
 100083    1   tcp  32769  ttdbserver
 100003    2   udp   2049  nfs
 100003    3   udp   2049  nfs
 100003    2   tcp   2049  nfs
 100003    3   tcp   2049  nfs
 200006    1   udp   2049
 200006    1   tcp   2049
 100005    1   udp  33000  mountd
 100005    2   udp  33000  mountd
 100005    3   udp  33000  mountd
 100005    1   tcp  32772  mountd
 100005    2   tcp  32772  mountd
 100005    3   tcp  32772  mountd
 100024    1   udp    651  status
 100024    1   tcp    651  status
 200001    1   udp    652
 200001    1   tcp    652
 200001    2   tcp    652
 100021    1   udp  33038  nlockmgr
 100021    2   udp  33038  nlockmgr
 100021    3   udp  33038  nlockmgr
 100021    4   udp  33038  nlockmgr
 100021    1   tcp  32773  nlockmgr
 100021    2   tcp  32773  nlockmgr
 100021    3   tcp  32773  nlockmgr
 100021    4   tcp  32773  nlockmgr
 390113    1   tcp   7937
```

RPC services will often have well known names (which are distinct from TCP and UDP service names) and these are tabled in **/etc/rpc**. In the **rpcinfo(1m)** example above the RPC program numbered 100005 is shown as the service called **mountd** -- the **/etc/rpc** table is used to map the RPC program numbers to RPC service names. You will find several RPC services started from **/etc/inetd.conf**as shown above. Other RPC services correspond to stand alone daemons started in the boot sequence (eg. **rpc.statd** in the **lsof** example). But keep in mind that RPC program numbers have nothing to do with the UDP and TCP port numbers where you find the service -- you need to look at the **RPC** tables to discover that mapping.

> It's important to know about RPC services. The **netstat(1m)** and **lsof(8)** tools will often show strange service numbers with no obvious mention of the number in **/etc/inetd.conf** or **/etc/services**. If you find one of these use **rpcinfo(1m)** to translate TCP/UPD port numbers to RPC service numbers/names which you probably will find in **/etc/inetd.conf**.

Summary: Baseline -- What's there?

The first step in hardening the network connections on a AIX system (or any other computer system for that matter) is to find out what services you offer and what they do. Then you can make a informed risk analysis and impact assessment. The above should have given you the tools to get started so you can determine what's there. Let's summarize:

1. Use "**netstat -a**" to show all your network connections. Look for TCP connections in a "LISTEN" state and UDP connections in an "Idle" state -- those are the network services you offer to the Internet.
2. Many of the services you discover will be found in **/etc/inetd.conf** and are offered indirectly by **inetd(8)** -- for all but RPC services it's easy to find the program that provides the service.

3. Many services you discover are started at boot time and will not be found in **/etc/inetd.conf**. If the corresponding program that provides the service is not obvious to you use **lsof(8)** to find the program that offers the service.
4. RPC services are found at strange port numbers not listed in **/etc/inetd.conf** or in **/etc/services**. Use the **rpcinfo(1m)** command to determine the RPC service name/number that's registered at that port. Then you should be able to find the service by name or number in **/etc/inetd.conf**.
5. RPC services not found in **/etc/inetd.conf** will typically correspond to services started at boot time. If the corresponding program that provides the service is not obvious to you use **lsof(8)** to find the program that offers the service.

The strategy given here should work on virtually any Unix system. The recommendations which follow are specific to AIX 4.3 but might be generalized to other Unix systems as well.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/03/13

```
#
#   COMPONENT_NAME: onccmds
#
#   FUNCTIONS: none
#
#   ORIGINS: 24
#
#
# Copyright (c) 1988 Sun Microsystems, Inc.
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
rquotad         100011  rquotaprog quota rquota
sprayd          100012  spray
3270_mapper     100013
rje_mapper      100014
selection_svc   100015  selnsvc
database_svc    100016
rexd            100017  rex
alis            100018
sched           100019
llockmgr        100020
nlockmgr        100021
x25.inr         100022
statmon         100023
status          100024
bootparam       100026
ypupdated       100028  ypupdate
keyserv         100029  keyserver
sunlink_mapper  100033
tfsd            100037
nsed            100038
nsemntd         100039
showfhd         100043  showfh
cmsd            100068  dtcalendar
ypxfrd          100069  ypxfr
pcnfsd          150001
ttdbserver     100083  tooltalk
autofs          100099  automount       #209812
```

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

The AIX Toolset

For those who are familiar with my work (especially the paper on Solaris Network Hardening) you will recognize the previous discussion -- the concepts apply to any Unix system. AIX has unique tools that can help you out. If you use AIX you should be (or become) familiar with these:

lssrc -- list subsystems managed by System Resource Controller

To list subsystems and subsystem groups managed by the System Resource Controller (SRC) use the **lssrc(1m)** command. For example to list all subsystems:

```
[11:47am wally] lssrc -a
Subsystem         Group            PID     Status
 syslogd          ras              2730    active
 portmap          portmap          5418    active
 inetd            tcpip            5676    active
 rwhod            tcpip            5934    active
 snmpd            tcpip            6192    active
 dpid2            tcpip            6450    active
 biod             nfs              3672    active
 rpc.statd        nfs              8258    active
 rpc.lockd        nfs              8516    active
 qdaemon          spooler          8776    active
 writesrv         spooler          9038    active
 llbd             iforncs          9312    active
 glbd             iforncs          4154    active
 i4llmd           iforls           3430    active
 i4lmd            iforls           5228    active
 i4glbcd          iforncs          11096   active
 lpd              spooler                  inoperative
 clvmd                                     inoperative
 gated            tcpip                    inoperative
 named            tcpip                    inoperative
 routed           tcpip                    inoperative
 iptrace          tcpip                    inoperative
 xntpd            tcpip                    inoperative
 timed            tcpip                    inoperative
 sendmail         mail                     inoperative
 dhcpcd           tcpip                    inoperative
 dhcpsd           tcpip                    inoperative
 dhcprd           tcpip                    inoperative
 ndpd-host        tcpip                    inoperative
 tftpd            tcpip                    inoperative
 mrouted          tcpip                    inoperative
 ndpd-router      tcpip                    inoperative
 rsvpd            qos                      inoperative
 i4gdb            iforls                   inoperative
 dtsrc                                     inoperative
 wsmrefserver                              inoperative
 policyd          qos                      inoperative
 nfsd             nfs                      inoperative
 rpc.mountd       nfs                      inoperative
 automountd       autofs                   inoperative
 pxed             tcpip                    inoperative
```

```
 binld           tcpip                   inoperative
 dfpd            tcpip                   inoperative
 secldapclntd    secldap                 inoperative
```

You should be cautious about this tool -- don't make any hasty assumptions. It isn't the case that every subsystem is a network subsystem and there is no requirement that all network daemons (ie. programs that provide some network service) are managed by the System Resource Controller. For example, the Common Desktop Environment (CDE) login service is managed by **dtconfig(1)**.

startsrc|stopsrc -- start/stop subsystems

To start and stop services managed by the System Resource Controller (SRC) use the **startsrc(1m)** and **stopsrc(1m)** commands. You can stop and start subsystems or a whole group of subsystems. For example:

```
[2:06pm wally]# stopsrc -s syslogd
0513-044 The syslogd Subsystem was requested to stop.
[2:06pm wally]# lssrc -s syslogd
Subsystem         Group            PID     Status
 syslogd          ras                      inoperative
[2:07pm wally]# startsrc -s syslogd
0513-059 The syslogd Subsystem has been started. Subsystem PID is 28942.
[2:07pm wally]# lssrc -s syslogd
Subsystem         Group            PID     Status
 syslogd          ras              28942   active
```

You should be cautious about this tool -- don't make any hasty assumptions. Although it will stop and start the subsystems it *does not* remove them from the boot sequence. Other tools are required for that -- eg. **chrctcp** is the tool SMIT uses to manage the **/etc/rc.tcpip** script, **mkitab** and **rmitab** are used to manage **/etc/inittab**, etc.

lssrc -ls inetd -- list subsystems offered by **inetd**

Of the subsystems managed by the System Resource Controller **inetd** can be queried to determine the services it offers:

```
[11:51am wally]# lssrc -ls inetd
Subsystem         Group            PID     Status
 inetd            tcpip            5676    active

Debug         Not active

Signal        Purpose
 SIGALRM      Establishes socket connections for failed services.
 SIGHUP       Rereads the configuration database and reconfigures services.

 SIGCHLD      Restarts the service in case the service ends abnormally.

Service       Command                 Description             Status
 ttdbserver   /usr/dt/bin/rpc.ttdbserver rpc.ttdbserver 100083 1  active
 cmsd         /usr/dt/bin/rpc.cmsd    cmsd 100068 2-5         active
 klogin       /usr/sbin/krlogind      krlogind                active
 kshell       /usr/sbin/krshd         krshd                   active
 dtspc        /usr/dt/bin/dtspcd      /usr/dt/bin/dtspcd      active
 time         internal                                        active
 chargen      internal                                        active
 discard      internal                                        active
 echo         internal                                        active
 time         internal                                        active
 daytime      internal                                        active
 chargen      internal                                        active
 echo         internal                                        active
 pcnfsd       /usr/sbin/rpc.pcnfsd    pcnfsd 150001 1-2       active
 sprayd       /usr/lib/netsvc/spray/rpc.sprayd sprayd 100012 1         active
```

```
rwalld       /usr/lib/netsvc/rwall/rpc.rwalld rwalld 100008 1          active
rusersd      /usr/lib/netsvc/rusers/rpc.rusersd rusersd 100002 1-2       active
rstatd       /usr/sbin/rpc.rstatd     rstatd 100001 1-3        active
exec         /usr/sbin/rexecd         rexecd                   active
login        /usr/sbin/rlogind        rlogind                  active
shell        /usr/sbin/rshd           rshd                     active
telnet       /usr/sbin/telnetd        telnetd -a               active
ftp          /usr/sbin/ftpd           ftpd                     active
```

chsubserver -- manage subsystems offered by **inetd**

The **chsubserver(1m)** command is used to manage the contents of **/etc/inetd.conf**. You can add, modify or delete records. Since it's changing the contents of a file the changes will persist across reboots.

```
[2:39pm wally]# chsubserver -d -v daytime -p udp
[2:42pm wally]# chsubserver -d -v daytime -p tcp
[2:42pm wally]# grep daytime /etc/inetd.conf
#daytime        stream  tcp     nowait  root    internal
#daytime        dgram   udp     wait    root    internal
```

Changes to the file aren't detected by **inetd** unless it's killed and restarted (eg. **stopsrc -s inetd; startsrc -s inetd**) or signalled to re-read the configuration file -- the **chsubserver** command supports a "**-r inetd**" option to signal the daemon. The command **"refresh -s inetd"** is another way to signal **inetd** to re-read it's configuration file -- see the **refresh(1m)** command.

These AIX tools are definitely useful -- they make testing for a service, shutting it off or starting it up very easy. Actually these tools are used by the SMIT interface. But it is important to have some understanding of what's going on them.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/02/01

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Network Hardening Issues

The following issues and recommendations address many of the network services offered by the typical AIX 4.3 system as it arrives "out of the box" -- we make no claim to completeness. The vendor provided configuration offers a great many services on the off-chance that you might need some of them. In today's environment, with most systems connected in one way or the other to the Internet, that configuration is far too promiscuous.

- /etc/inetd.conf -- servers managed by **inetd** daemon.
- /etc/inittab -- servers started by **init** process.
- /etc/rc.nfs -- servers for Network File System and Network Information System.
- /etc/rc.tcpip -- various stand alone servers.
- Miscellaneous -- other odd issues.

In the Hardening Kit section of this paper we'll give you several policies (wrt. these issues) and a tool to apply the policy. The information here is to help you understand services offered so you can pick a policy and/or create a custom policy.

Reg Quinton, *Information Systems and Technology*
2001/01/15-2001/02/06

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Services offered by inetd from /etc/inetd.conf

The **inetd** daemon, started from **/etc/rc.tcpip**, is typically configured to offer a great many services. Those services are easy to find -- they're listed in **/etc/inetd.conf** and you can also see them with **"lssrc -ls inetd"**. Not all of those services are required -- many are historical baggage that the prudent system manager should abandon. Simplify your configuration by getting rid of services you don't need.

1. **Issue/inetd/bootps:** Disable Boot Protocol Service

   The **bootps** service provided by the **bootpd** daemon is used for network booting of various systems -- eg. in the SP environment one might configure all the nodes with this way, or class room of identical systems all booted off a single server. If you have a need of **bootps** services you'd configure a very few systems as servers. Typically **bootps** servers go hand and hand with **tftp** servers -- **bootp** gives boot time parameters to the client (here's your IP number, netmask, router, and your **tftp** server who has your initial boot image). Client systems which rely on **bootps** servers have security concerns of their own -- they trust who ever answers.

   These days with the Dynamic Host Configuration Protocol (DHCP) I can't understand why Unix systems still use **bootps**. Again, this one of those seldom required servers. Don't enable it unless you have to. IBM's Network Install Manager (NIM) uses a **bootps** service to provide configuration data to clients it manages.

2. **Issue/inetd/chargen:** Disable character generator

   The **chargen** service is built into the **inetd(1m)** and typically is configured from **/etc/inetd.conf**. It's available both as a TCP and a UDP service. It's internal built-in function of **inetd(1m)** that spits back a never ending character sequence that might be useful for testing the network (back in 1982).

   ```
   [2:23pm wally] telnet dilbert chargen
   Trying 129.97.128.157...
   Connected to dilbert.uwaterloo.ca.
   Escape character is '^]'.
    !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefg
   !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
   "#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi
    ...etc.
   ```

   This service runs as user root and might be compromised (rather unlikely). More likely it's an opportunity for a DOS attack -- especially since it's a UDP as well as a TCP service. Remember the "ping of death" attacks?

3. **Issue/inetd/cmsd:** Disable calendar manager daemon

   The **rpc.cmsd** daemon is started from **/etc/inetd.conf** -- it's the calendar manager. It runs as root so you can update your calendar and peak at others. Since it runs as root it is a security exposure which might be compromised -- I recall that it has been compromised on other platforms. The Common Desktop Environment (CDE) is usually configured with a "calendar" icon on the control panel. That icon starts up the CDE Calendar Manager **dtcm(1)** which needs to find this daemon -- if you disable the daemon this client application will fail miserably. Of course on back-room database servers you won't have users who need this service.

   You should not leave this service configured on any system unless you have first determined that you require the service. If you need the service you should be especially careful to make sure vendor recommended patches are regularly applied. Historically RPC services have been problematic and should configured with care.

   See also the SANS Top Ten -- this one made it.

4. **Issue/inetd/comsat:** Disable 'you have mail' messages

   The **comsat** service spits a 'you have mail' message to interrupt your interactive login session. The **sendmail(1m)** program receives your mail and gives it to the **bellmail(1m)** program to deliver. That sends a UDP message to the **comsat** daemon who beeps you. That's if your mail is delivered on the same system and you're working as an interactive user at a login session. It isn't going to beep the POP and IMAP clients and it isn't going to beep the users on NFS clients who mount the mail spool. If you're an interactive user you're better off getting your shell to watch your mailbox.

   > TRIVIA: Also known as the "biff" service -- apparently "biff" was the name of a dog who barked whenever the letter carrier dropped off mail.

   It's seldom required anymore. It runs as user root and might be compromised. If you don't need it ... don't run it!

5. **Issue/inetd/daytime:** Disable obsolete daytime service

   The **daytime** service (not to be confused with Network Time Protocol) is built into the **inetd(1m)** and typically is configured from **/etc/inetd.conf**. It spits back the local time of day as a string suitable for humans to read. It's available both as a TCP and a UDP service.

   ```
   [2:29pm wally] telnet dilbert daytime
   Trying 129.97.128.157...
   Connected to dilbert.uwaterloo.ca.
   Escape character is '^]'.
   Tue Oct 10 14:29:47 2000
   Connection closed by foreign host.
   ```

   This is a very old service predating NTP and it's seldom if ever used. I know of no client that peers with this service (short of a **telnet** to the port). You definitely don't need it. It runs as user root and might be compromised (rather unlikely). More likely it's an opportunity for a DOS attack.

6. **Issue/inetd/discard:** Disable discard service

   The **discard** service is built into the **inetd(1m)** and typically is configured from **/etc/inetd.conf**. It tosses whatever comes it's way -- a network version of **/dev/null**. It's available both as a TCP and a UDP service. It's a very old service that might have been used for diagnostics at one time. It could be used in a Denial of Service attack (to gobble up sockets). You definitely don't need it.

7. **Issue/inetd/dtspc:** Disable CDE Subprocess Control Service

   The manual page for **dtspcd** page says:

   > "The daemon for the CDE Subprocess Control service, dtspcd, is not intended to be started directly by the user, rather it should be started automatically by the inetd daemon (see inetd(1m)) in response to a CDE client requesting a process to be started on the daemon's host."

   I read that as a method for getting jobs running on your system without your authorization. Certainly if your machine is a backroom server you will not require this at all (if you run CDE you *might* be able to get along without). This looks very dangerous -- yet another reason why people don't trust CDE. I don't run this on my Solaris CDE workstation and I recommend the removal on all back-room servers. I assume the same applies to AIX.

8. **Issue/inetd/echo:** Disable echo

   The **echo** service (not to be confused with the ICMP echo used by **ping(1)**) is an internal built-in function of **inetd** that echos back whatever comes it's way. It's available both as a TCP and a UDP service. This is another very old service that may even predate the **ping(1)** command. It's never used any more but it could be used in a Denial of Service attack (to gobble up sockets). It's also a UDP service so the bad guys might, with a little effort, get it to echo at someone else to get through a firewall or echo to everyone to start a data storm. I'm not aware of any historical compromises but you definitely don't need it.

   It runs as user root and might be compromised. If you're skeptical that anyone would ever exploit such a trivial

service see CERT Advisory CA-98.013 -- ICMP echo is exploited in the infamous "smurf" attacks.

9. **Issue/inetd/exec:** Disable remote execution server

   The TCP **rexecd** daemon allows for remote execution of commands -- it requires that you provide a user name and password. Typically you won't find a Unix command that peers with the server but you will find a library call **rexec(3n)** if you want to build one. But why would you ever want to do that? If **rsh** isn't enough for you there are better tools in the **ssh(1)** suite. I understand there are some PC X-windows clients that use this to start up X11 applications but even then you'd be better off moving your PC users to **ssh**. I don't need this service on my work station nor on any of the back room servers I've hardened. It runs as user root and might be compromised. The userid and password are passed in the clear and can be snooped. I'd be very surprised if you needed this.

10. **Issue/inetd/finger:** Disable remote user information server

    The TCP **fingerd** daemon answers questions about users on your system. There have been historical compromises (remember the "Morris Worm"?) and it's a good tool for hackers to find out about systems and the users on the system. Many sites disable the service because it gives out information about your system and your users:

```
[2:52pm wally] finger @dilbert
[dilbert.uwaterloo.ca]
Login    Name                    TTY     Idle When       Office
dwhitesi Dawn Whiteside           pts/3    38 Tue 13:42  MC3047    3472<
lennox   Bruce Lennox - IST       pts/0  2:59 Tue 09:13
reggers  Reg Quinton              pts/1    14 Tue 09:57
```

    It runs as user root and has been compromised. I'm amazed to discover systems where the manager has disabled the **finger** daemon but leaves services like **rwhod** that give out similar information.

11. **Issue/inetd/ftp:** Disable file transfer protocol server

    The TCP **ftp** service is a primitive file transfer service. On very secure systems you would not offer the service. Instead you'd pull files to the system by running the **ftp(1)** client there or, better yet, rely on public domain tools like the **ssh** suite which includes a **scp(1)** command. There have been **ftpd** compromises (buffer overflows, ftp proxy tricks to get past firewalls and/or to hide your location). Even if those compromises have been attended to the protocol still exposes the userid and password in clear text which might be snooped. You're far better off with **ssh**.

    It runs as user root and has beene compromised -- see the recent CERT Advisory CA-2000-13 and research back to ones like CERT Advisory CA-97.27.

12. **Issue/inetd/imap2:** Disable Internet Mail Access Protocol

    The Internet Mail Access Protocol (IMAP) on TCP port 143 provides client applications (like Eudora, Netscape Communicator, Microsoft's Outlook Express and others) access to mail folders, including the "inbox", stored on the server. It's far richer service than the Post Office Protocol (POP) and was developed to address many of the deficiences of POP. This is a great service ... if your system is a mail server. It's an unnecessary risk if your system isn't a mail server. There have been nasty security compromises with various versions of this server. That's not a surprise as it need's to offer a fairly complex set of services. Userid and password are usually passed in the clear.

    I keep finding back room servers at our site that are running this service when they clearly should not. Don't make that mistake. Don't run the service unless you need it. If you do need it, think about protecting the clear text userid and password with an SSL tunnel -- both Netscape Communicator and Outlook Express understand the SSL tunnel on port 993 that provides the same service. That's usually listed as the **imaps** service. I've not listed it as an issue as I've not seen it on a vendor distribution.

    See also the SANS Top Ten -- this one made it.

13. **Issue/inetd/klogin:** Disable Kerberos login server

    The TCP **klogin** service implemented by the **krlogind** daemon peers with the **krlogin(1)** command and uses Kerberos credentials to authentication -- compare with **rlogin** and **rlogind**. This provides a virtual terminal session with strong encryption -- a rival to the **ssh** tools. If you're not a Kerberos site (we're not), if you're not using Kerberos credentials you have no need of this.

14. **Issue/inetd/kshell:** Disable Kerberos shell server

    The TCP **kshell** service implemented by the **krshd** daemon peers with the **krsh(1)** command and uses Kerberos credentials to authentication -- compare with **rsh** and **rshd**. This allows for the remote execution of commands on a server with kerberos authentication and strong encryption -- a rival to the **ssh** tools. If you're not a Kerberos site (we're not), if you're not using Kerberos credentials you have no need of this.

15. **Issue/inetd/login:** Disable remote login server

    The TCP **login** service implemented by the **in.rlogind** daemon peers with the **rlogin(1)** command and uses a primitive mechanism that permits access from trusted users/sites without a password challenge. Authentication is based on IP address security (which can be spoofed), DNS security (and again that can be spoofed) and the notion of reserved ports (on Unix systems only user root can open the client port). The server can trust a collection of hosts (with **/etc/hosts.equiv**), individual users can trust user/host pairs (with **~user/.rhosts**) and in both cases wild cards are permitted. It's a very convenient system but involves lots of security compromises.

    The server will issue a "**Password:**" challenge if the user/host pair doesn't pass the weak authentication requirements. Unfortunately that data, and everything else, is passed in the clear so you are vulnerable to sniffers watching the network. Like the **in.telnetd** this daemon has to manage a protocol for the psuedo **/dev/tty** device. The **in.rshd** is a much simpler service as there's no **/dev/tty** device.

    These days one ought to use **ssh(1)** instead of **rlogin(1)** for remote access. I strongly urge you to use that instead. Alternatively, if you require rlogin/rlogind services make sure you've at least used the tcp wrapper to stop DNS spoofing (that doesn't stop IP spoofs) and limit your exposure to just those systems you really trust. The CERT Coordination Center has an excellent security improvement module on Installing, Configuring, and Using TCP Wrapper -- the target is the Solaris platform but the module applies equally to AIX.

    > BEWARE: if you require rlogin services you should be especially careful about access to the "root" account -- see the **Issue/misc/root.access** to lock down access to the root account.

    It runs as user root and might be compromised -- if you don't need it ... don't run it! You may recall that **rlogind** was exploited on AIX a several years ago. See CERT® Advisory CA-1994-09.

16. **Issue/inetd/netstat:** Disable network status server

    The TCP **netstat** service allows for remote access to the network status of your system -- what peers are currently connected, what services are provided, etc. It accepts no input and runs the command "**netstat -f inet**". You're giving out too much information to hackers if your run it. Fortunately, it's generally not enabled. You definitely don't need it ... don't run it!

17. **Issue/inetd/pcnfsd:** Disable PC NFS Daemon

    The **rpc.pcnfsd** daemon provides file services to PC's. It's a service built on RPC. Special software is required on the PC. This doesn't really have much to do with NFS services -- it's authenticated file services more like the Server Message Block (SMB) file services of the **Samba** project. PC NFS predates Microsoft's release of SMB specifications as the "Common Internet File System" (CIFS). If you're not using it, disable it. If you're thinking about using it you're probably better off with **Samba** as it works without changing the client. IBM also provides a product for SMB/CIFS clients but we have no experience with it.

18. **Issue/inetd/pop3:** Disable Post Office Protocol

    The Post Office Protocol (POP) on TCP port 110 predates the Internet Mail Access Protocol (IMAP). This

provides client applications (cf. the discussion of IMAP) with access to their "inbox". Userid and password are supplied in the clear. Again, this is a great service if your system is a mail server and you have clients who are using applications that only support POP (if the client application supports IMAP you ought to use that instead). Otherwise, don't run this service unless you need it.

There is an SSL tunneled version of the POP service you may bump into on port 995 -- sometimes it's listed as the **pop3s** service. The SSL tunnel is to protect the clear text userid and password. I've not listed it as an issue as I've not seen it on a vendor distribution.

See also the SANS Top Ten -- this one made it.

19. **Issue/inetd/rexd:** Disable remote execution -- **on(1)**.

    AIX systems arrive with several network services that provide for remote execution of commands -- everyone should know about **rsh(1)** and **rshd(1m)**. Here's yet another one: The **rexd** daemon is started from **/etc/inetd.conf** and peers with the **on(1)** command. It runs as user root and might be compromised. If you don't need it ... don't run it! I've never had occassion to enable this service. I've seen a comment in **/etc/inetd.conf** on other systems saying "The rexd server provides only minimal authentication and is often not run". The entry for this service is commented out of **/etc/inetd.conf** in the vendor configuration. That's probably wise.

20. **Issue/inetd/rquotad:** Disable remote quota reports

    The **rquotad** daemon is started from **/etc/inetd.conf** and provides quota reports to NFS clients. If you're not providing NFS file services you don't need this service. If you're not providing NFS file services with file quotas you don't need this service. And even if you are providing NFS file services with file quotas it's not essential that you also provide this service. You should understand that file quotas on NFS servers are independent of this service -- all it provides is an answer for the **quota(1)** command. Programs will see faults like "hard/soft quota exceeded" and can act on those faults independent of this service. You should not leave this service configured on any system unless you have first determined that you require the service. If you need the service you should be especially careful to make sure vendor recommended patches are regularly applied. Historically RPC services have been problematic.

21. **Issue/inetd/rstatd:** Disable kernel statistics server

    The **rpc.rstatd** daemon gives out performance characteristics of your system and is required for the **rup(1)** command. There are many other services that provide similar data -- the SNMP services, the **rwhod** server as used by the **ruptime(1)** command, the **syslogd** "mark" facility and lots more. If you want to monitor systems you probably ought to be using SNMP and abandon this one.

    The **rup(1)** display looks an awful lot like that produced by **ruptime(1)**:

```
[1:53pm wally] rup
hooper.uwaterlo    up 22 days, 22:25,    load average: 0.16, 0.23, 0.37
hebron.uwaterlo    up 152 days,  3:09,    load average: 0.11, 0.17, 0.18
lan3.uwaterloo.    up  6 days, 10:45,    load average: 2.46, 2.57, 2.12
dilbert.uwaterl    up  5 days,  3:24,    load average: 0.39, 0.38, 0.43
xpurt.uwaterloo    up  6 days, 10:45,    load average: 0.12, 0.19, 0.18
 ....etc.
[1:55pm wally] ruptime
alvin         up 133+20:28,     9 users,  load 0.92, 0.59, 0.48
amy           up 173+04:03,     2 users,  load 0.06, 0.07, 0.06
bert          up  7+03:26,     2 users,  load 1.20, 1.13, 0.99
ernie         up 139+04:08,     0 users,  load 0.00, 0.00, 0.00
 ...etc.
```

    The data for **rup** is obtained by canvassing **rstatd** servers; the data for **ruptime** is data collected by the **rwho** daemon who broadcasts similar data to the subnet.

    See also the SANS Top Ten -- this one made it.

22. **Issue/inetd/rusersd:** Disable network username server

The **rpc.rusersd** gives out a list of current users on your system and peers with the **rusers(1)** command -- a command I had never used. When run from my work station I discover:

```
[3:10pm wally] rusers
Sending broadcast for rusersd protocol version 3...
cool.uwaterloo.c mike mike
waynesworld.uwat daveb
trek.uwaterloo.c reggers picard wally
ist.uwaterloo.ca support tcampbel gregs roger...etc.
dilbert.uwaterlo paul
Sending broadcast for rusersd protocol version 2...
tazmania.uwaterl ron ron ron ron ron ron ron
```

It looks like the service is implemented by a broadcast on the local network but I would not be surprised if the same information discovered above can as easily be discovered by bad guys at remote sites. The bad guys like to attack systems where they'll not be noticed -- if no one is on **waynesworld** or only one person is ever there then it's a safe bet that it's a desktop Unix system where a late night compromise would not be noticed.

I have seen lots of systems where they lock down the **finger** service but leave this one running. It's not an essential service, it's giving away information that you don't need to share and it runs as user root. If you don't need it ... don't run it!

23. **Issue/inetd/rwalld:** Disable network rwall server

    The **rpc.rwalld** daemon writes messages to every logged in user -- messages received are handed off to **wall(1m)**. It receives **rwall(1m)** requests (write to all users on remote systems). NFS servers will use **rwall** to notify client systems of an impending reboot. If your system has interactive users they'll be notified.

    But these days Unix systems have very few interactive users. Unix systems excel at providing E-mail, Web, News and Database services where the user is no longer an interactive user on a **/dev/tty** device. The user is sitting at a PC workstation and isn't going to see the **rwall** messages.

    This is not an essential service, it runs as user root and might be compromised. If you don't need it ... don't run it!

24. **Issue/inetd/shell:** Disable shell/rshd service

    The **rsh** service implemented by the **in.rshd** daemon which peers with the **rsh(1)** command uses the same primitive authentication mechanism as **in.rlogind**. The daemon allows for the the remote execution of commands by users. All of the issues which apply to **in.rlogind** apply here as well. The same recommendation applies -- use **ssh** instead and disable the service. If you must have the service use the TCP Wrapper to stop spoofing and limit your exposure.

    Disabling **rsh** services also disables **rcp(1)** and **rdist(1)** commands. Those commands are layered services on top of **rsh**. Note the **rlogin** services are not layered that way -- the **in.rlogind** server peers with the **rlogin** command. That's because there's no **/dev/tty** device required for an **rsh** session.

    > BEWARE: at our site rsh/shell services are *required* for xhier -- **rdist(1)** is the fundamental file distribution method that xhier builds on and **rdist** uses **shell** services. That's the reason why I've never had occassion to disable **inetd** entirely -- we always need the **shell** service..

25. **Issue/inetd/sprayd:** Disable spray daemon

    The **rpc.sprayd** daemon receives RPC packets sent by **spray(1)** -- that's a **ping(1)** like tool used to test network connectivity and RPC reliablity (as might be required if you're having NFS problems). When I test the service from my work station I find:

    ```
    [2:03pm wally] spray ratbert
    sending 1162 packets of length 86 to ratbert ...
            317 packets (27.281%) dropped by ratbert
            53 packets/sec, 4590 bytes/sec
    ```

    Here's another service that runs as user root and might be compromised. If you don't need it or have never used it

... don't run it! I had never used this application before writing this paper -- I've only diagnosed NFS problems by pinging with large packets.

26. **Issue/inetd/systat:** Disable system status server

    The **sysstat** service allows for remote sites to see the process status on your system -- what jobs are currently running. It's a simple client that accepts no input and runs the command "ps -ef". I know of no client other than **telnet(1m)** that peers with this server. You're giving out way too much information to hackers if you enable this -- it's way worse than the **finger** service. Fortunately, it's disabled in **/etc/inetd.conf** and you would be very foolish indeed to enable it. Why it remains as a commented entry is a mystery to me.

    You definitely don't need it ... don't run it!

27. **Issue/inetd/talk and inetd.ntalk:** Disable talk and ntalk services

    Unix systems often are configured to provide a **talk** service -- that's a UDP service at port 517 that peers with the the **talk(1)** command. The "new talk" is at port 518 and implemented by the **ntalk(1)** command. The both "talk" commands establish a split screen connecting two users of the net. Each user writes in one screen and sees what the other writes in the other screen -- they get to "talk" to one another. The service is configured in **/etc/inetd.conf**.

    On multi-user systems this may be a nice tool to offer your users but these days few users actually login to Unix machines any more. It's certainly not a required service -- your system will run fine without it. On back room database servers it's not required at all. Given the limited application I'd recommend that the service be disabled until such time as you've determined that there is a real need.

    It runs as user root and might be compromised. If you don't need it ... don't run it!

28. **Issue/inetd/tftp:** Disable Trivial File Transfer Protocol

    The **tftp** service (Trivial File Transfer Protocol) is seldom required. It's a UDP service you will find at port 69 (if enabled). When required it's often configured in error and bad guys use it to pick up data that you thought was restricted. It runs as user root and might be compromised. If you don't need it ... don't run it! IBM's Network Install Manager (NIM) uses a **tftp** service to provide the initial boot image to clients it manages.

    If you're providing boot services to a collection of diskless or dataless stations you will need TFTP services on some server. Alternatively, if you're providing boot services to a collection of X11 stations (eg. NCD work stations boot that way). Often times routers, switches, terminal servers and other network infra-structure use TFTP services to upload configuration files -- but be careful about exposing sensitive data which might be found in those files.

    If you do need to offer TFTP services you should restrict access to a very limited portion of your filing system. For example, a properly configured server might have:

    ```
    [10:54am wally] grep tftp /etc/inetd.conf
    tftp   dgram   udp    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
    ```

    The usual convention is that you should restrict access to the **/tftpboot** directory. The directory does not exist on the AIX distribution -- you need to create it and populate it with the files you need to offer by TFTP. Be careful about the data you make available (should anyone be able to read it?) and be aware that you can use TFTP to upload files to a system -- cf. the **anon.ftp.write** issue. Uploading files in that manner is often done to backup the configuration of routers, switches or other network devices. Be careful if you do that.

29. **Issue/inetd/telnet:** Disable telnet service

    The **telnet** service implemented by the **in.telnetd(1M)** daemon supports remote login sessions using the **telnet(1)** client -- the user is given a login name and password challenge and a pseudo **/dev/tty** device is delivered for the login session. Unfortunately the credentials, and everything else, is passed in the clear -- that's a great opportunity for sniffers. People sometimes get around that issue with one-time passwords, eg. SecureID is

very popular, but there are better ways.

These days one ought to use **ssh** instead of **telnet(1)** for remote access. The **ssh** protocol uses strong encryption to protect your credentials and everything else during the entire session. If you are at all concerned about security, configure **ssh** and get rid of the **telnet** service.

The **telnet** service runs as user root and might be compromised. If you're skeptical that anyone would ever exploit the service see the recent CERT Incident Note IN-2000-09 regarding SGI/IRIX systems. If you don't need it ... don't run it!

> BEWARE: if you require telnet services you should be especially careful about access to the "root" account -- see the **Issue/misc/root.access** to lock down access to the root account.

30. **Issue/inetd/time:** Disable obsolete time service

The **time** service (not to be confused with Network Time Protocol) is an internal built-in function of **inetd(1m)** that spits back the time (as a 32bit number). It's available both as a TCP and a UDP service. It's a very old service predating NTP and seldom used any more -- it is used by the **rdate(1m)** command that's sometimes integrated into the boot sequence. These days we should use **ntpdate** to synchronize clocks at boot time.

31. **Issue/inetd/ttdbserver:** Disable ToolTalk database server

The **rpc.ttdbserverd** daemon is the "Tool Talk Data Base Server". It *seems* to be a required service if you are running the "Common Desktop Environment" (CDE) but I've found my Solaris work station runs fine without it -- I assume the same for AIX. There have been security problems with this tool -- you should not run this in any environment where security is a concern. You may want to run this service on CDE desktops or multi-user systems with X11 client stations but should not run this on back room servers. Even in CDE environments I would encourage you to try running without it -- if I can survive without it on my Solaris work station then you probably can too on your AIX work station.

The **rpc.ttdbserverd** daemon runs as user root and might be compromised. If you don't need it ... don't run it! If you're skeptical that anyone would ever exploit this service see CERT Advisory CA-98-11. See also the SANS Top Ten -- this one made it.

32. **Issue/inetd/uucpd:** Disable UUCP daemon

The **uucpd** service is for traditional UUCP style networking over the IP network. Why anyone would do that today beats me -- there's very few geezers around anymore that even remember the UUCP network (I count myself as one of them). These days hardly anyone needs this service.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/02/06

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Services started from /etc/inittab

Many of the services configured in **/etc/inittab** can be safely removed in their entirety. Some may require more work.

1. **Issue/inittab/dt:** Disable CDE DeskTop Login

   The **/etc/rc.dt** script in **/etc/inittab** is used to bring up the "DeskTop" environment of the notoriously insecure Common DeskTop Environment (CDE). This starts an X11 server on the console and the **dtlogin** process. That daemon is responsible for the login panel where you fill in your credentials to gain access to the console. The **dtlogin** daemon also supports the X11 Display Manager Control Protocol (xdcmp) so that other X11 stations can login to the same machine as well -- eg. a lab full of inexpensive NCD stations. On AIX systems I find **dtlogin** listening at UDP port 177 (the **xdmcp** port) and TCP port 32768 (for the X11 server running on the console -- see the **lsof** example above).

   > Watch out! Things get complicated. You can have a simple "dt" entry, a "dt_nogb" (no graphical boot) or a vanilla xdm entry. I'll give you scripts for each but you need to be careful about how you use them. You can, of course, use SMIT to manage this issue.

   If you're not providing CDE login services you'll be more secure if you get rid of the **dtlogin**. Is anyone watching for prowlers using **dtlogin** as a password cracker? Are there any packet sniffers watching for credentials submitted to the service? I use CDE services on my personal workstation but disable many of the "DeskTop" components. At our site I keep finding headless servers where these services have not been disabled. For backroom servers I recommend that you remove the service (unless you're intentionally providing **xdmcp** to a room full of X11 stations).

   Beware: removing this entry from **/etc/inittab** gets rid of only some parts of the CDE environment -- don't assume it removes all of CDE components. See the discussion of **inetd.ttdbserver**, **inetd.dtspc** and **inetd.cmsd**.

2. **Issue/inittab/dt_nogb:** Disable CDE DeskTop Login (No Graphical Boot)

   That's a subtle variation on the CDE boot sequence -- you don't get a graphical screen until the machine is all the way up. AIX geeks like it this way, naive users don't.

3. **Issue/inittab/httpdlite:** Disable the "docsearch" Web Server

   The **httpdlite** process, as the name suggests, is a web server. It's the default web server for the "docsearch" engine found in the **IMNSearch** package. At most sites the prudent strategy would be to allocate a single system for documentation of this sort and use instead -- there's not need to replicate the service on several system. In any case, it's generally a bad idea to mix web services with other production services.

Don't run this service unless you really need to. If you do need it try to find it on another server -- let them assume the risk.

4. **Issue/inittab/i4ls:** Disable Licence Manager services

   Licenced products, eg. the C, C++ and Fortran compilers, will require licence manager services -- at a minimum the **i4llmd** subsystem of the **iforls** SRC group. Client systems which use licences managed elsewhere will require a great wack of services. For example:

```
[9:38am wally] lssrc -a | grep ifor
 llbd             iforncs          6208    active
 glbd             iforncs          6450    active
 i4llmd           iforls           6708    active
 i4lmd            iforls           7224    active
 i4glbcd          iforncs          7482    active
 i4gdb            iforls                   inoperative
```

   While it's clear you need the licence services *when* you're using the licenced product you generally will not need any licence services (for languages at least) in a production environment. We try to make a distinction between production and development environments -- they have different requirements.

   The policies in the kit will leave the licence manager services untouched on all but a back-room database server. You should carefully consider whether or not you need to accept the exposure of these several servers. If you don't need them, don't run them. If you need them only occasionally, then you should run them only occassionally.

5. **Issue/inittab/imnss:** Disable "docsearch" **imnss** daemon

   Again, this is part of the "docsearch" engine found in the **IMNSearch** package. You'd be better off with a couple of systems devoted to this task. Don't run it unless you really need to.

6. **Issue/inittab/imqss:** Disable "docsearch" **imqss** daemon

   Again, this is part of the "docsearch" engine found in the **IMNSearch** package. You'd be better off with a couple of systems devoted to this task. Don't run it unless you really need to.

7. **Issue/inittab/lpd:** Disable BSD Line Printer Daemon

   The TCP **lpd** service accepts print jobs from other systems using the protocol developed for BSD Unix (strange enough I'm not aware of any SystemV counterpart -- if you print between Unix systems it's always done by the BSD protocol). If your system doesn't have any printers, or if you don't want others printing jobs on your printers then you don't need this service.

   I've discovered on Linux and Solaris systems that you can safely disable print services and still send print jobs to other servers. I can confirm that you can safely disable this daemon on AIX and still send print jobs to a print server.

   Beware: there's a nasty confusion on AIX. You'll find an entry in the **/etc/rc.tcpip** file that leads one to think that **lpd** is started there -- it's not. It's managed here in **/etc/inittab**.

8. **Issue/inittab/nfs:** Disable NFS (and NIS) services

   The Network File Systems (NFS) services started from **/etc/rc.nfs** are problematic and should be considered carefully. Unfortunately the boot script includes several Network Information Services

(NIS) services. NIS was formerly known as "Yellow Pages" hence the naming of things like **ypupdated**. NFS and NIS services are built on UDP/RPC and that's a messaging protocol with no synchronization between client and server. Authentication is minimal -- security is based largely on the idea of a "trusted" host and that works well enough in a controlled environment. But back room database servers should have no need of these services and shouldn't trust them in any case. If you can, the ideal is to disable the whole lot of them.

If you can't disable the entire set of services see the various **nfs.*** issues below where I try to tease out the various subsytems managed by the **/etc/rc.nfs** script.

9. **Issue/inittab/piobe:** Disable Printer IO BackEnd

   The **piobe** daemon is the Printer IO BackEnd. It handles the scheduling, spooling and printing of jobs submitted by the **qdaemon**. I'm not familiar with AIX printing and can't comment much beyond this -- if you're not doing any printing you won't need it. See also the discussion at **tcpip.lpd** -- you don't need to offer print services to others.

10. **Issue/inittab/qdaemon:** Disable Queueing Daemon

    The **qdaemon** submits print jobs to the **piobe** daemon -- they come as a pair of related jobs. I'm not familiar with AIX printing and can't comment much beyond this -- if you're not doing any printing you won't need it. See also the discussion at **tcpip.lpd** -- you don't need to offer print services to others.

11. **Issue/inittab/uprintfd:** Disable Kernel Messages

    The **uprintfd** daemon is generally not required according to Farazdel, Genty et al. (see their RedBook in the references below). This apparently has something to do with getting kernel messages to consoles.

12. **Issue/inittab/writesrv:** Disable real-time write(1) to tty

    The **writesrv** daemon is the server for the **write(1)** client. This allows users on the same system to send real time messages to one another of the sort "John, you there? Wanna do lunch?". The messages are blasted onto your terminal without much ado. These days users aren't typically logged in as interactive Unix users (unless they're geeks) so there's no compelling need for this service. If they need this service why don't they use talk, or e-mail, or pick up the phone?

    On other Unix systems the **write(1)** command searchs out the tty device and writes directly to it. The use of a intermediary daemon on AIX seems unduly complex.

13. **Issue/inittab/xdm:** Disable traditional X11 Display Management

    AIX supports the traditional X11 display management -- see **xdmconf** if you want to try it. Security folks think that CDE and traditional X11 are both security compromises. I'd not run **xdm** on a back room server.

BEWARE: these issues are all handled by commenting in/out the appropriate entry in **/etc/inittab**. That isn't the way SMIT does it and probably will cause confusions if you manage the issue with SMIT -- you might end up with two entries for the same service. It might be better to use **rmitab** and **mkitab** to add and delete entries but that requires you keep track of what ought to be there.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/03/13

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Services started from /etc/rc.nfs

If you cannot eliminate the **/etc/rc.nfs** issue you can address components that are started by the script. Not all are required -- eg. NFS client systems don't need all the processes that a server needs, likewise a server may not require all the processes that are provided by default. On high risk back room servers you should avoid NFS entirely or otherwise minimize your dependence -- get rid of the components you don't need or aren't using.

1. **Issue/rc.nfs/automountd:** Disable NFS automounter

   The **automount(1m)** daemon is started from **/etc/rc.nfs** and mounts Network File System (NFS) file systems when and as required using data in NIS maps (or **/etc/auto*** files if you're not using NIS). It's a kind of "just in time service" sometimes used to manage the **/home** directory. At our site we do use NFS but hardly anyone uses the automounter. This would not be required for a back-room database server and should not be trusted in that environment. This might be required on desktop systems with a highly distributed NFS. At our site we prefer large centralized NFS servers and avoid the hodge-podge file system that the automounter maintains.

2. **Issue/rc.nfs/biod:** Disable NFS Block I/O Daemon

   The **biod** daemon is started from **/etc/rc.nfs** and provides block I/O services for NFS clients. You don't need this unless you're an NFS file server. If you disable this you should disable the **biod**, **nfsd** and **rpc.mountd**.

3. **Issue/rc.nfs/keyserv:** Disable NIS Key Server

   The **rpc.keyserv** daemon is started from **/etc/rc.nfs** and is used to manage the keys required for secure RPC. I believe this comes from the Kerberos model and is important for NIS+ which never seemed to get further than Solaris ... and isn't Sun abandoning NIS+ for LDAP directory services? If you're not using NFS or NIS you won't need this. Even if you are using NFS and NIS you probably can get away without it.

4. **Issue/rc.nfs/nfsd:** Disable NFS server

   The **rpc.nfsd** daemon is started from **/etc/rc.nfs** and provides remote access to the local filing system from clients who have done an NFS file mount. Obviously this service is only required on NFS file servers. Authentication is very weak -- essentially this is trusted host authentication. Over the years there have been lots of compromises -- stack frame crashing. If you disable this you should disable the **biod**, **nfsd** and **rpc.mountd**.

   See also the SANS Top Ten -- this one made it.

5. **Issue/rc.nfs/rpc.lockd:** Disable NFS File Lock Daemon

   The **rpc.lockd** daemon is started from **/etc/rc.nfs** and is used to implement file locks across the

NFS. If you're not using any NFS you certainly won't need the service. If you are using NFS but don't need to use file locks over the network you won't need the service. I occassionally use NFS services from my personal work station and I have disabled this service with no loss of functionality. The **statd(1m)** and **lockd(1m)** daemons are mentioned in the SANS Top Ten Security Threats.

6. **Issue/rc.nfs/rpc.mountd:** Disable NFS mount services

   The **rpc.mountd** daemon is started from **/etc/rc.nfs** and peers with the **mount(1)** command when a client system requests an NFS file mount. Obviously this service is only required on NFS file servers. Authentication is very weak -- essentially this is trusted host authentication. Over the years there have been lots of compromises -- stack frame crashing. If you disable this you should disable the **biod**, **nfsd** and **rpc.mountd**.

7. **Issue/rc.nfs/rpc.statd:** Disable NFS statd (file locks)

   The **rpc.statd** daemon is started from **/etc/rc.nfs** and is used to implement file locks across the NFS (it's used to recover locks which might be lost across system failures). Same comments as for **rpc.lockd**.

8. **Issue/rc.nfs/rpc.yppasswdd:** Disable NIS Password Daemon

   The **rpc.yppasswd** daemon is started from **/etc/rc.nfs** and is used to manipulate the local password file. In the NIS environment only the "master" server has the real password file, everyone else has a copy distributed by NIS services. The service isn't required if you're not using NIS and it's not required at all unless your server is the NIS master.

9. **Issue/rc.nfs/ypupdated:** Disable NIS Update Daemon

   The **rpc.ypupdated** daemon is started from **/etc/rc.nfs** and is used to receive NIS data base maps (for **/etc/passwd** and other files) pushed from an NIS master. NIS requires a master server that owns the maps and distributes updates to slave servers which act as backups and redundant servers. The service isn't required if you're not using NIS and it's not required unless your server is an NIS slave to some master.

BEWARE: Most of these components can be controlled by the System Resource Controller -- ie. you can stop and start the daemons. However, I never found the AIX tools to manipulate **/etc/rc.nfs** changes so that the changes would be persistent across system reboots. I've done it with simple minded edits -- probably a little too simple minded.

Reg Quinton, *Information Systems and Technology*
2001/01/15-2001/03/13

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Services started from /etc/rc.tcpip

Many network daemons, including the **inetd**, are started from the **/etc/rc.tcpip** script. Not all are required, many can be safely disabled.

1. **Issue/rc.tcpip/autoconf6:** Disable IPv6 Configuration

   The **autoconf6** process started in **/etc/rc.tcpip** is used to configure IPv6 network interfaces. While IPv6 is out of the laboratory it's not in production at many sites. We have no need for this.

2. **Issue/rc.tcpip/dhcpcd:** Disable DHCP Client Daemon

   The Dynamic Host Configuration Protocol (DHCP) provides boot time configuration information to clients who issue DHCP requests. I'm completely confused by the notion of a DHCP Client Daemon. A DHCP client should just issue a request and get back an answer. I don't know what this daemon is doing but I can make this observation -- if your host isn't using DHCP to get boot time configuration data you won't need this. Back room servers should not rely on DHCP in any case -- it's too easy to set up a rogue DHCP server and trick unsuspecting systems into trusting you.

3. **Issue/rc.tcpip/dhcpsd:** Disable DHCP Server Daemon

   This is the DHCP Server Daemon -- **dhcpsd** is started from **/etc/rc.tcpip** and answers DHCP requests from client systems which broadcast a request at boot time. The answer, from tables on the server, gives the client configuration information -- IP name, number, netmask, router, broadcast address, etc. The nasty thing about DHCP is anyone can run a DHCP server and clients who use DHCP will naively believe the first response they get.

   Since DHCP is a broadcast protocol it's typically not passed between subnets (unless the router is configured with a helper address). Even at very large sites you're probably better off with a central server.

   If you don't need it, don't run it. And even if you do need it think about taking advantage of a central service instead of building your own.

4. **Issue/rc.tcpip/dhcprd:** Disable DHCP Relay Daemon

   This is the DHCP Relay Daemon -- **dhcprd** is started from **/etc/rc.tcpip** and might be used if you wanted to act as a DHCP relay. A relay grabs DHCP broadcasts and packs them off to a server on another network. This is the kind of service often found on a router.

   Hardly anyone has a need for this, of those who do their need would be better met by special purpose routers sold by vendors like Cisco.

5. **Issue/rc.tcpip/dpid2:** Disable (Legacy) SNMP Daemon

   According to Farazdel, Genty et al the **dpid2** daemon started from **/etc/rc.tcpip** is a legacy version

of the SNMP daemon. If you need SNMP don't use this version.

6. **Issue/rc.tcpip/gated:** Disable IPv4 Gated Router

   Where the traditional RIP protocol is inadequate sites can use **gated** instead. But you'd be better off using a special purpose router. Hardly anyone has a need for this, of those who do their need would be better met by special purpose routers sold by vendors like Cisco.

7. **Issue/rc.tcpip/inetd:** Disable inetd

   The **inetd** process started out of **/etc/rc.tcpip** offers many services as configured by **/etc/inetd.conf**. If you have very thoroughly hardened your AIX system you could find yourself with no services left that need to be offered by **inetd**. At our site it would be unusual indeed to see a system that didn't require the **inetd** daemon -- remote **shell** services are *required* for xhier and that service is provided by **inetd**. Nevertheless, for completeness, I've included a script to check and eliminate that server.

8. **Issue/rc.tcpip/mrouted:** Disable Multi-cast Router

   According to Farazdel, Genty et al the **mrouted** daemon started from **/etc/rc.tcpip** is used to route multi-cast packets between network segments (cf. traditional routing by **routed** and **gated**). Multi-cast services haven't really made their mark, if they do I'd much prefer that the routing issue were handled where it should be -- by special purpose routers. Hardly anyone has a need for this, of those who do their need would be better met by special purpose routers sold by vendors like Cisco.

9. **Issue/rc.tcpip/named:** Disable DNS Name Server

   The **named** daemon started from **/etc/rc.tcpip** provides the Domain Name Service (DNS). Typically at a site you would have a few DNS name servers and all clients are configured, with **/etc/resolv.conf**, to use those servers. Historically there have been lots of security problems with the DNS service. I often see systems (especially Linux) configured with a DNS server running -- but the server has no data and nobody is using it. You are much better off letting other people run the DNS service.

   See also the SANS Top Ten -- this one made it.

10. **Issue/rc.tcpip/ndp-host:** Disable IPv6 Host

    According to Farazdel, Genty et al this has something to do with IPv6 and you can safely avoid the issue until IPv6 becomes a production network at your site.

11. **Issue/rc.tcpip/ndp-router:** Disable IPv6 Router

    I understand from Farazdel, Genty et al that this has something to do with IPv6 routing. My comments would be that IPv6 is certainly not production and if it were I'd rather rely on a special purpose router. Hardly anyone has a need for this, of those who do their need would be better met by special purpose routers sold by vendors like Cisco.

12. **Issue/rc.tcpip/portmap:** Disable RPC directory services

    The **portmap** process started from **/etc/rc.tcpip** is the RPC directory server. It's found at well known port number which will not vary. RPC servers register themselves with the **portmap** daemon and clients who need to locate RPC services ask the **portmap** daemon to tell them where a

particular service (an RPC service number) is found. I have been able to strip systems of all RPC services save for **portmap** and the Legato backup service. The **nsrexecd** process registered as RPC program number 390113/tcp is the Legato client required for backup/recovery. I've not had the time to determine if the Legato backups would work without the portmap daemon -- they're always at the same port number so there's a chance it might.

Some RPC services are at well known ports and don't care if **portmap** is around. But many RPC services will fail if you get rid of **portmap** -- the clients will not be able to locate the server. You should be very cautious about disabling the **portmap** daemon. However, if you've managed to reduce RPC services to the extent that the only one remaining is **portmap** then you can safely get rid of it as well:

```
[1:51pm wally] rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  rpcbind
    100000    3   tcp    111  rpcbind
    100000    2   tcp    111  rpcbind
    100000    4   udp    111  rpcbind
    100000    3   udp    111  rpcbind
    100000    2   udp    111  rpcbind
```

Note that RPC services you have removed from a running system will remain in the **rpcinfo(1m)** listing -- killing off a daemon doesn't automatically remove it from the RPC directory. Don't be surprised if **rpcinfo(1m)** shows programs running at ports after you've killed them off. After hardening and rebooting your system you should check for RPC services. If the results are as above, it's safe to disable the **portmap** directory service.

13. **Issue/rc.tcpip/routed:** Disable IPv4 RIP Router

    The traditional RIP protocol is adequate for most network environments (it requires a common netmask for all subnets managed by RIP). You could use an AIX machine to route IP packets between networks and you'd need the **routed** daemon started from **/etc/rc.tcpip**. However, you'd be better off using a special purpose router. Hardly anyone has a need for this, of those who do their need would be better met by special purpose routers sold by vendors like Cisco.

    The static route that an AIX system should use is maintained by the Object Data Manager (ODM) -- use the command **"lsattr -E -l inet0"**.

14. **Issue/rc.tcpip/rwhod:** Disable Remote Who Daemon

    The **rwhod** process started from **/etc/rc.tcpip** is used to collect and broadcast data to peer servers on the same network. The data collected and dispersed is the kind of information you would expect in an "open" environment. For high risk or otherwise sensitive systems you probably don't want to be broadcasting out who is on, what they're doing, etc. That's why you disable services like the **finger** daemon. If you're interested in monitoring a system this is a pretty crude tool -- you'd be much better of using SNMP. I don't recommend this service but at our site many people still like to see it -- even on back room servers.

15. **Issue/rc.tcpip/sendmail:** Disable sendmail/smtp services

    The **sendmail(1m)** daemon is started at boot time and provides the **smtp** service (Simple Mail Transfer Protocol). You should know that it has a long history of security breaches -- too long to

summarize here. It runs as user root so of course it's a dandy target if you're searching for a way to break in. We can safely assume that as new features are added sendmail will continue to be a security problem where vigilance is required.

A back-room database server often has no need of a stand-alone sendmail/smtp service -- mail goes out of the machine but none needs to come in. If you're configuring that kind of server, you might want to disable sendmail services. If you do there's a couple of issues you should address:

1. A machine not running an SMTP daemon should have a **cron** job running periodically to ensure that messages don't get stuck in the queue. A crontab entry like this run the **sendmail** daemon at 23 minutes after each hour to clear the queue:

   ```
   23 * * * * /usr/lib/sendmail -q
   ```

2. It would be wise to configure DNS services so the mail for your server is delivered to some other system. That requires an MX record to direct the mail elsewhere, eg. a DNS entry something like this:

   ```
   db3.uwaterloo.ca. IN A 129.97.86.34
                     IN MX 0 ist.uwaterloo.ca
   ```

   This requires that the **ist.uwaterloo.ca** mail server be configured to understand that mail received for **user@db3.uwaterloo.ca** should be delivered to the local user and not forwarded on. There's lots of ways to do that.

There are some who recommend that you bail out of **sendmail(1m)** entirely and replace it with something else. I've never been keen about that recommendation. There is a very large community who use **sendmail(1m)** -- that's one reason so many bugs have been discovered. It's also the reason why it works so very well. I'm especially thankful for the way sendmail is able to handle spammers. If you must run a mail server I'd recommend you stick with **sendmail(1m)** but be vigilant about vendor patches.

See also the SANS Top Ten -- this one made it.

16. **Issue/rc.tcpip/snmpd:** Disable Simple Network Monitor Daemon

    The Simple Network Management Protocol (SNMP) is implemented by the **snmpd** process started from **/etc/rc.tcpip**. The idea behind SNMP is to provide a standard way to monitor (and control) various system parameters on different systems scattered about a network -- the kind of thing that a "Network Operation Centre" would do. A good idea but if you haven't got SNMP configured properly you may leave a security problem. If you're not monitoring the system using SNMP tools then why have those processes at all -- you're providing information to who knows who. You may require SNMP on important servers; but you probably don't need to monitor workstations. If you do need SNMP services make sure you restrict the service to just those machines that need to monitor your machine and make sure you keep up with recommended patches.

    See also the SANS Top Ten -- this one made it.

17. **Issue/rc.tcpip/syslogd:** Disable System Log Daemon

    The System Log Daemon is implemented the **syslogd** process started from **/etc/rc.tcpip**. It will accept UDP messages on port 514 from anyone on the network -- as such it's prone to denial of service attacks and message forgeries. It also accepts messages on the **/dev/log** socket -- that's how

local processes are supposed to communicate messages. You'd be foolish indeed if you disabled this service, however a script is provided to do so (if only so you can determine that you have a daemon running).

There is an option, **-r**, to suppress logging of messages received from remote hosts. That would help but I've not scripted that.

18. **Issue/rc.tcpip/timed:** Disable (Old) Time Daemon

    The **timed** process started from **/etc/rc.tcpip** is an old time daemon -- for keeping clocks on systems in synchronization. These days the preferred time server is **xntpd**. Don't use this version.

19. **Issue/rc.tcpip/xntpd:** Disable (New) Time Daemon

    The **xntpd** process started from **/etc/rc.tcpip** is the new time daemon -- for keeping clocks on systems in synchronization. Most sites these days have several systems they've identified as their top-most time servers. The top most time servers either have an external clock that picks up the broadcast time signal or they talk to other hosts at other sites who have clocks. At many sites all Unix systems are configured as **xntp** servers. I recommend instead that you configure a few systems as time servers and let other systems synchronize to them with a periodic cron job that invokes **ntpdate**. While every site should have some **xntp** servers you should not make the hasty assumption that all systems should be **xntp** servers.

These issues are all handled by the AIX **chrctcp** tool -- it handles starting and stopping the services as well as changes to the boot script. There was no need for us to construct edit scripts to manipulate the startup file as we did for the issues discussed in the NFS/NIS Section.

Reg Quinton, *Information Systems and Technology*
2001/01/15-2001/02/06

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Miscellaneous Network Issues

Some issues don't fall so neatly into categories with similar problems. The following is a collection of miscellaneous issues all related to the network environment.

1. **Issue/misc/dt.restrict:** Restrict CDE login to console

   By default the **dtlogin(1)** service is configured to provide CDE (Common Desktop Environment) login services to the world -- anyone can approach an AIX system across the network and ask for a CDE session (ie. the login panel you get when you boot a typical AIX system with a console). This probably isn't what you want. Is anyone watching for prowlers using **dtlogin** as a password cracker? Are there any packet sniffers watching for credentials submitted to the service? What kind of gunk are the prowlers tossing at **dtlogin** to try to break it?

   - If the system is a personal work station you probably don't want to provide CDE login services to anything other than the console attached to the system.
   - If the system is a back-room database server you probably don't want to provide **dtlogin** services at all -- strip out all CDE services!
   - There are occassions where you would want to provide **dtlogin** services to others -- a lab of X11 stations attached to a large multi-user server is a good example. But even then I doubt that you want to provide the service to the world.

   The ideal is to get rid of the **dtlogin** service; if you can't do that restrict access to just the machines that require CDE login to the system.

2. **Issue/misc/ftp.anon:** Disable anonymous ftp

   The anonymous ftp services allows guest access to the ftp service (you authenticate as "anonymous" or "ftp"). The **Issue/misc/ftp.anon** script makes sure you don't offer an anonymous ftp service -- if you have a user "ftp" it renames the user to "Noftp" and that disables the service. You should not provide an anonymous ftp service unless you really need to. It's very easy to make configuration mistakes that cause nasty problems (eg. Q: "Where did my disk space go?" A: "Oh the hackers are using me as their exchange site for warez tools!"). Further, it's hard (even with logging) to track who did what -- there's no accountablity. In any case, in the year 2000 why would anyone offer anonymous ftp services instead of a Web service? If you want to serve up files to the public then the Web is a far better way.

   This issue doesn't apply, of course, if you're not offering the ftp service.

3. **Issue/misc/ftp.anon.write:** Disable anonymous ftp writes

   If you really need anonymous ftp you should not let people upload files into your site. If you do you'll soon have the bad guys using your site as a public repository for their stuff. Worse yet, the bad guys, knowing that you've done such a poor job with anonymous ftp, will start searching your site for other exploits. The **Issue/misc/ftp.anon.write** script makes sure you don't offer an anonymous ftp service

for uploading files to your system. It scans the anonymous ftp tree looking for files that belong to user "ftp" (if any are found they changed to owner root), files that belong the login group for user "ftp" (if any group writeable files are found the group write is removed) and files that are world writeable (those are fixed as well).

This issue doesn't apply, of course, if you're not offering the anonymous ftp service.

4. **Issue/misc/ftp.restrict:** Disable ftp to system accounts

   AIX systems can be configured to not allow ftp access to the root account and other system accounts -- the file **/etc/ftpusers** denies ftp access to any userid listed. This doesn't prevent those users from using the ftp command to push/pull files from the system. What it prevents is remote access to those accounts using ftp.

5. **Issue/misc/netoptions:** Set Network Options

   This issue is a little outside the norm of this paper. Generally, we've been concerned with removed network services that we don't need. There are more fundamental problems with the IP, ICMP, UDP and TCP services on which one builds a client/server application. You may be familiar with attacks like the "Ping of Death", denial of service attacks involving "half-open" connections or attacks that subvert routing. There is a good discussion of network options you can set to better protect your AIX system by Farazdel, Genty et al at Section 8.4 of their Redbook. We can verify that the options they recommend have no ill effect in our environment and would highly recommend you use their settings -- we make similar recommendations for Solaris.

   This issue is handled by extending the **/etc/rc.tcpip** boot script -- we add these options to the end:

```
[9:38am wally] tail -22l /etc/rc.tcpip
# BEGIN: Net Options
# Harden off various Network Options/Attributes. REQ 5-Feb-2001
#
# See Section 8.4 of "Additional AIX Security Tools on IBM eserver
# pSeries, ..." by Farazdel, Genty, et al at http://ibm.com/redbooks
#
/usr/sbin/no   -o clean_partial_conns=1
/usr/sbin/no   -o bcastping=0
/usr/sbin/no   -o directed_broadcast=0
/usr/sbin/no   -o ipignoreredirects=1
/usr/sbin/no   -o ipsendredirects=0
/usr/sbin/no   -o ipsrcroutesend=0
/usr/sbin/no   -o ipsrcrouterecv=0
/usr/sbin/no   -o ipsrcrouteforward=0
/usr/sbin/no   -o ip6srcrouteforward=0
/usr/sbin/no   -o icmpaddressmask=0
/usr/sbin/no   -o nonlocsrcroute=0
/usr/sbin/no   -o tcp_pmtu_discover=0
/usr/sbin/no   -o udp_pmtu_discover=0
/usr/sbin/no   -o ipforwarding=0
#
# END: Net Options
```

   Beware: these options are not suitable if your system is used as a router between networks -- the **ipforwarding** is disabled with these options.

6. **Issue/misc/root.access:** Lock down root access

The default configuration allows **telnet** and **rlogin** access to the root account. This can be configured in the **/etc/security/user** file -- set the **rlogin** option to "false" for all system accounts. System managers should login to their account and then **su** so we have an audit trail.

7. **Issue/misc/snmpd.readWrite:** disable SNMP readWrite communities

   The default SNMP configuration includes these "**readWrite**" communities:

   ```
   [9:38am wally]# grep readWrite /etc/snmpd.conf
   #    readOnly, writeOnly, readWrite.  The default permission is readOnly.
   community       private 127.0.0.1 255.255.255.255 readWrite
   community       system  127.0.0.1 255.255.255.255 readWrite 1.17.2
   ```

   These seem dangerous indeed -- especially since they have such well known names. I recommend you disable those communities (if you haven't already disabled the **snmpd** daemon). You probably should restrict the "**public**" community to just those systems that really need to monitor your system.

8. **Issue/misc/syslog.conf:** configure syslogd(8)

   The default **syslogd(8)** configuration does nothing -- you won't get any important messages logged unless you configure the file **/etc/syslog.conf**. This issue is handled by adding these configuration settings if **syslogd(8)** hasn't been configured:

   ```
   [9:38am wally]# tail /etc/syslog.conf
   # BEGIN: Syslog Config
   *.err;kern.debug;daemon.notice  /var/log/messages
   mail.debug;*.info               /var/log/syslog
   auth.debug                      /var/log/authlog
   *.crit                          *
   # END
   ```

   Beware: if you choose to configure the system logs by this method (or any other for that matter) you need to set up **cron(1)** jobs to roll the log files you create. Else one day you'll arrive to discover there's no space left on some file system.

9. **Executeable Stack:** No Solution!!

   One of the biggest network security problems on many Unix systems is the stack overflow vulnerablity -- poorly written programs can take well crafted input that overruns the space allocated for the data in such a way that a subroutine/function return results in supplied code being executed on the stack. Solaris and Linux are especially vulnerable systems. If a network daemon has this vulnerablity then it can be exploited -- cf. the recent **LPRng** exploits on Linux, the **rpc.statd** bugs on Solaris.

   It is our understanding that AIX 4.3 on the RS/6000 platform is *vulnerable* to these attacks -- the recent vulnerablities with DNS bind and the emergency fixes released by IBM lead us to believe that AIX on RS/6000 can be exploited. We are not aware of any system configuration parameter that will protect you from these attacks. The prudent system manager should make sure that services offered are always patched and up to date.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/03/13

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Network Hardening Kit


A traditional Unix tar kit containing all scripts to harden each issue discussed, sample polices and a driver to implement policies is available here.

1. When you retrieve the kit you will have a Unix tar file to unpack:

```
[4:23pm wally] ls *.tar
aix-network-harden.tar
```

2. Unpack the tar file into a like named directory:

```
[4:26pm wally] mkdir aix-network-harden
[4:26pm wally] cd aix-network-harden
[4:26pm wally] tar xf ../aix-network-harden.tar
[4:26pm wally] ls -l
total 136
-r--r-----   1 reggers  other        635 Feb  6 13:38 BEWARE
-r-xr-x--x   1 reggers  other        918 Jan 30 10:21 Implement
drwxr-x--x   7 reggers  other       4096 Feb 13 16:15 Issues
-r--r-----   1 reggers  other       1296 Feb 13 15:52 Makefile
-r-xr-x--x   1 reggers  other       1165 Feb 13 16:04 Monitor
-r--r-----   1 reggers  other       4776 Feb  6 10:13 Policy-DBServer
-r--r-----   1 reggers  other       4806 Feb  6 10:14 Policy-Default
-r--r-----   1 reggers  other       4738 Feb  6 10:14 Policy-Everything
-r--r-----   1 reggers  other       4699 Feb  6 10:15 Policy-WKStation
-r--r-----   1 reggers  other       4821 Feb 13 11:03 Policy-XhierServer
-r--r-----   1 reggers  other        650 Feb  6 13:39 README
-r--r-----   1 reggers  other        445 Feb  6 13:45 RTFM.html
```

3. The **Issues** directory contains scripts we've developed for hardening off the issues developed in this paper.

```
[4:26pm wally] ls -l Issues
total 40
drwxr-x--x   3 reggers  other       4096 Jan 29 11:17 inetd
drwxr-x--x   3 reggers  other       4096 Jan 30 09:49 inittab
drwxr-x--x   3 reggers  other       4096 Jan 29 14:50 misc
drwxr-x--x   3 reggers  other       4096 Jan 30 09:06 rc.nfs
drwxr-x--x   3 reggers  other       4096 Jan 30 09:54 rc.tcpip
[4:26pm wally] ls -l Issues/misc
total 80
-r-xr-x--x   1 reggers  other       1324 Jan 29 14:33 ftp.anon
-r-xr-x--x   1 reggers  other       2231 Jan 29 14:33 ftp.anon.write
-r-xr-x--x   1 reggers  other       2209 Jan 29 14:50 ftp.restrict
-r-xr-x--x   1 reggers  other       2543 Feb  6 09:19 netoptions
-r-xr-x--x   1 reggers  other       1318 Jan 29 14:35 root.access
```

   The script names correspond to the names of the issues as discussed in this paper.

4. Each script can be run to deal with a particular issue, to check an issue (the "**-c**" option) or to undo

an issue (the **"-u"** option). The undo option is important if you change your mind later. For example, here's some work with the **comsat** service:

```
[4:37pm wally]# ./Issues/inetd/comsat -c
Ok: comsat disabled (already)
[4:26pm wally]# ./Issues/inetd/comsat -u
Ok: comsat enabled
[4:26pm wally]# ./Issues/inetd/comsat -c
ERR: comsat not disabled!
[4:25pm wally]# ./Issues/inetd/comsat
Ok: comsat disabled
```

It should be clear that you need to be user "root" to enable/disable **comsat** services or any other issue addressed by the kit.

 **Beware:** before you do any hardening make sure you keep copies of **/etc/inetd.conf**, **/etc/inittab**, **/etc/rc.nfs** and **/etc/rc.tcpip**. You need to save your baseline in case you need to restore it!

5. The **Implement** tool is a driver you can use to implement a network hardening policy -- as you can see several are provided. You can use the driver to check on all issues for which we've provided hardening scripts:

```
[4:26pm wally]#  ./Implement -c Policy-Everything
ERR: bootps not disabled!
ERR: chargen not disabled!
ERR: cmsd not disabled!
Ok: comsat disabled (already)
Ok: daytime disabled (already)
 ...etc.
Ok: routed disabled (already)
ERR: rwhod not disabled!
Ok: sendmail disabled (already)
ERR: snmpd not disabled!
ERR: syslogd not disabled!
Ok: timed disabled (already)
Ok: xntpd disabled (already)
```

6. You can use the driver to enforce a policy (your own or perhaps you'd like to use one of the samples provided):

```
[4:26pm wally]# ./Implement Policy-Default
Ok: bootps disabled (already)
Ok: chargen disabled
Ok: cmsd disabled
Ok: daytime disabled
Ok: discard disabled
 ...etc.
Ok: ndp-host disabled (already)
Ok: ndp-router disabled (already)
Ok: routed disabled (already)
0513-044 The rwhod Subsystem was requested to stop.
Ok: rwhod disabled
0513-044 The snmpd Subsystem was requested to stop.
Ok: snmpd disabled
Ok: timed disabled (already)
Ok: xntpd disabled (already)
```

The Policy files are readable lists of issues that should be applied. They're readable for a reason -- read them carefully before enforcing a policy!

7. You will need to determine the policy required for your system -- the samples provided are nothing more than samples. However, once you have established a policy enforcing it is pretty easy. If you want to catch deviance from your policy a cron job to check it is easy too -- see the **Monitor** script we've provided in the kit.
8. Watch out for vendor patches! They're notorious on all platforms (AIX is no exception) for undoing work that you've done to lock the system down. After applying patches make sure the system still conforms to the policy you've established.

Beware: Network Hardening is but one part of an overall security policy. It's only concerned with locking down services to keep the bad guys out and this kit will help (but bear in mind it's not complete -- there may be services on your system that aren't addressed). Do not assume your job is done when you've applied this kit! There's lots more work required to secure a system.

Reg Quinton, *Information Systems and Technology*
2001/01/15-2001/05/04

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

See Also

Some further reading for the brave or curious:

- IANA Port Number Assignments
- AIX - RS/6000 Documentation Library (IBM)
- The **lsof** utility is found at Purdue and mirrored at many other sites.
- CERT® Security Improvement Modules, Carnegie Mellon Software Engineering Institute
- Frequently Asked Questions about AIX and the IBM RS/6000 (Usenix posting). Also a pdf version for printing.
- IBM Redbooks. See especially these excellent papers:
  - AIX 4.3 Elements of Security Effective and Efficient Implementation (by) Kosuge, Arminguad, Chew, Horne & Witteveen 18-Aug-2000. Also a pdf version for printing.
  - Additional AIX Security Tools on IBM pSeries, IBM RS/6000, and SP/Cluster, (by) Farazdel, Genty, Kerouanton & Khor 20-Dec-2000. Also a pdf version for printing.
  - Exploiting RS/6000 SP Security: Keeping It Safe, (by) Farazdel, DeRobertis, Genty, Kreuger & Wilkop 21-Sep-2000. Also a pdf version for printing.
- SANS Top Ten Security Threats (consensus).
- Our Security How To ... documents -- various topics. Including:
  - Solaris Network Hardening 19-Sept-2000.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/05/17

Security Review: AIX 4.3 Network Hardening
Information Systems and Technology
University of Waterloo

Credits

If you are so foolish as to think this paper is the work of a single individual you are very much mistaken. This paper follows from a similar paper on Solaris Hardening where, with very good input from Bruce Lennox and many other colleagues at Waterloo, we developed the principles you find here. Martin Timmerman was instrumental in twisting my arm to do the same for AIX and Ron Hosler is the AIX guru who provided details about features that I'd never seen. All of my colleagues who made this paper possible are owed a sincere thanks.

My thinking has been influenced by my involvement with the YASSP Project and it's major contributors Jean Chouanard and Sean Boran. That project is sponsored by the SANS Institute -- their "Step-by-Step" Consensus guides for Linux and Solaris are excellent hardening documents (it's unfortunate that they aren't in the public domain).

It should be apparent that I've read two AIX Redbooks on AIX Security: AIX 4.3 Elements of Security Effective and Efficient Implementation (by) Kosuge, Arminguad et al (18-Aug-2000) and Additional AIX Security Tools on IBM pSeries, IBM RS/6000, and SP/Cluster (by) Farazdal, Genty et al (20-Dec-2000). You should too if you are at all interested in AIX security.

This paper was prepared for the inaugural USERblue Conference, February 25-28, 2001 in Long Beach California. USERblue is a SHARE forum for IBM users of Unix Technologies.

Reg Quinton, Information Systems and Technology
2001/01/15-2001/01/31