Global Information Assurance Certification Paper

Practical use of Security tools and methods for AIX 5L Version 5.2 hardening

Practical Assignment for SANS 2004

November 2004

Alexander Stanovoy

Submitted November 2004
to fulfill GIAC GCUX requirements

The main idea of this Practicum was to provide practical guidelines for Unix Security Analysts in making system more secure, using ssh, sudo, AIX IPSec packet filtering feature and some shell scripts.

Abstract

The days when a Systems Administrator was the person responsible for all sides of computerized systems are gone. Today most companies have different departments with different roles and responsibilities. While this separation of responsibilities is more effective and more productive, it makes the control of system consistency more difficult. Through many different publications we can see how important Information Security became and how companies spend millions to keep their systems secure. UNIX systems, as repositories for information, a tool for authentication and authorization, or just as an environment for processes and streams, need to be protected.

Unix Security, as one of Systems Administrators responsibilities, has become a separate discipline. Most companies now have Unix Security departments or groups with Unix Security Analysts.

The purpose of this paper is to provide practical instructions for Unix Security Analysts working in an environment with split roles and responsibilities. This document includes real examples of ssh, sudo, shell scripting and AIX IPSec packet filtering features to help Unix Security Analysts in AIX hardening process.

The main idea is to show how security tools and methods can help in hardening of multipurpose use, Intranet Server.

The Unix Security Analyst's task is to find the balance between Company's business needs and systems security. This balance in most cases is to have the same products (but run them in a secure way), plus disable unneeded services and employ security policies and standards in practice.

In my paper I'll try to implement nine security rules/methods using existing AIX security features and applying some third party security tools.

These nine rules/methods are:

1. Remove unneeded services from the system.

2. Protect the root password and root access.

3. Enforce good password practices and files attributes by users.

4. Hide system information for remote users by removing default systems and services welcome banners.

5. Avoid "clear text" networked services, such as telnet or ftp and use SSH instead.

6. Avoid "clear text" storage of critical data, such as passwords, personal and business information, performing system check using scripts for automation.

7. Allow only controlled access to the system and prevent direct super user (root) access, using SUDO and existing system features.

8. Prevent unauthorized access to system, using packet-filtering feature of AIX IPSec tools.

9. Automate systems review process, automatically creating and sending daily reports, using regularly executable scripts.

Defining the Environment

It is a fact that to make a system completely secure it would be necessary to power it down and put into vault. Our goal, as Unix Security Analysts, is to make our systems secure and at the same time useful to users.

This system that I am going to secure in this paper was preloaded and configured to match my company's requirements and business needs. It includes remote monitoring tools as well as remote administration tools. The physical access to the system is only required in the very first stage of system installation. As soon as hardware is placed and connected, physical access is no longer required. Terminal servers or Ethernet to Serial devices provide console-like access to the system and all that is needed to connect to the system is the IP address and using regular networking services like TELNET or SSH.

As my UNIX system installer promised, I got an IP address, ID and password to access this system. The first logical step to get into the system is to SSH to it and then execute commands to collect information about the system configuration and the products installed.

There is no need to explain how to connect using ssh or telnet client, it is simple and do not require special qualification. The idea of this paper is to show more advanced way of using system tools after the connection was established.

1. Collecting information about system

Checking for network connectivity.

```
$ ifconfig –a
en1:
flags=4e080863,80<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,PSEG,CHAIN> inet 10.10.100.39 netmask 0xffffff00 broadcast 10.10.100.255
lo0:
flags=e08084b<UP,BROADCAST,LOOPBACK,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT> inet 127.0.0.1 netmask 0xff000000 broadcast 127.255.255.255 inet6 ::1/0
tcp_sendspace 65536 tcp_recvspace 65536
```

This command gives information about the IP address and network. The default gateway must be known to access this machine.

```
$ netstat –rn
Routing tables
Destination      Gateway          Flags   Refs     Use  If   PMTU Exp Groups

Route Tree for Protocol Family 2 (Internet):
default          10.10.100.1      UG       7    12438  en1     -  -
10.10.100.0      10.10.100.39     UHSb     0        0  en1     -  - =>
10.10.100/24     10.10.100.39     U        1       13  en1     -  -
10.10.100.39     127.0.0.1        UGHS     6      286  lo0     -  -
10.10.100.255    10.10.100.39     UHSb     0        0  en1     -  -
127/8            127.0.0.1        U        2     6921  lo0     -  -
```

This command gives information about the default gateway for the server. If this machine is not routed or accessible by a workstation, it is possible to use this address and try to connect using this gateway as workstation gateway.

In the current example, the machine is in a routed network and is accessible from all internal networks (any machine in the 10.0.0.0 range).

My workstation is equipped with an SSH client and after the connection was established I was able to switch from a regular user ID to a more privileged ID using su (switch user) command.

1.1 Checking an OS version and patch level

Getting permissions to perform system analysis:

```
$ su –
```

Identifying the OS:

```
# uname –a
AIX hostname 2 5 0003FB5A4C00
```

Verifying the Patch level:

```
#  instfix –ivq | grep AIX_ML
5.2.0.0_AIX_ML Abstract: AIX 5.2.0.0 Release
5200-01_AIX_ML Abstract: AIX 5200-01 Update
```

Kernel information:

```
#  bootinfo –K
64
```

This means that OS is running in 64 bit mode.

1.2 Checking for installed packages

```
# more smit.log
```

There is no reason to include the whole smit.log file in this paper, (as it is about 120 kilobytes) and this paper will not include removal of unneeded packages. Instead I will concentrate on finding a good compromise between what is installed and making the system secure.

After detailed analysis of the smit log file, it was determined that the system includes:

- BOS with 64 bit Runtime
- All drivers installed by default
- AIX IPSEC package
- SSH version 2 client and server from F-Secure
- sudo version 1.6.8
- Perl 5.8
- Expect with TCL/TK version 8 extension
- C Set ++ Runtime and Preprocessor
- X Window runtimes, drivers and basic utilities
- Tivoli Management Agent (client)
- Tivoli TSM Client
- Auto Path

- OpC Performance analysis tool
- OpenGL runtime
- RPM package manager

1.3 Checking network services

All established connections and services listening for connection:

```
# netstat –na | egrep “LISTEN|ESTABL”
tcp        0      0  *.21                 .                    LISTEN
tcp4       0      0  *.22                 .                    LISTEN
tcp        0      0  *.23                 .                    LISTEN
tcp4       0      0  *.25                 .                    LISTEN
tcp4       0      0  *.111                .                    LISTEN
tcp4       0      0  *.135                .                    LISTEN
tcp4       0      0  *.381                .                    LISTEN
tcp4       0      0  *.383                .                    LISTEN
tcp        0      0  *.514                .                    LISTEN
tcp4       0      0  *.1581               .                    LISTEN
tcp4       0      0  *.32780              .                    LISTEN
tcp4       0      0  *.32786              .                    LISTEN
tcp4       0      0  *.10128              .                    LISTEN
tcp4       0      0  *.17132              .                    LISTEN
tcp4       0     88  10.10.100.39.22      10.11.111.50.2654    ESTABLISHED
tcp4       0      0  *.22201              .                    LISTEN
```

Network services and related processes with PID and owner (very useful for analysis of vulnerable services):

```
# lsof -i | egrep “LIST|ESTAB”
opcctla    90316 root    6u  IPv4 0xf1000089c0190358       0t0  TCP *:32780 (LISTEN)
sendmail  118844 root    7u  IPv4 0xf1000089c017c358       0t0  TCP *:smtp (LISTEN)
inetd     127038 root    4u  IPv6 0xf1000089c0184b58       0t0  TCP *:ftp (LISTEN)
inetd     127038 root    5u  IPv6 0xf1000089c0184358       0t0  TCP *:telnet (LISTEN)
inetd     127038 root    6u  IPv6 0xf1000089c0186b58       0t0  TCP *:shell (LISTEN)
inetd     127038 root    7u  IPv4 0xf1000089c0186358       0t0  TCP *:bgssd (LISTEN)
inetd     127038 root    8u  IPv4 0xf1000089c0185b58       0t0  TCP *:ctmagt1 (LISTEN)
hostAgent 151676 root    3u  IPv4 0xf1000089c0190b58       0t0  TCP *:17132 (LISTEN)
dced      172158 root   10u  IPv4 0xf1000089c0188b58       0t0  TCP *:loc-srv (LISTEN)
llbserver 192606 root    3u  IPv4 0xf1000089c018db58       0t0  TCP *:383 (LISTEN)
```

```
coda_perf 213096 root    4u  IPv4 0xf1000089c0185358       0t0  TCP *:381 (LISTEN)
dsmcad    221296 root    6u  IPv4 0xf1000089c018d358       0t0  TCP *:32786 (LISTEN)
dsmcad    221296 root    7u  IPv4 0xf1000089c01adb58       0t0  TCP *:1581 (LISTEN)
sshd2     233586 root    3u  IPv4 0xf1000089c01b7b58       0t0  TCP *:ssh (LISTEN)
sshd2     249956 root    5u  IPv4 0xf1000089c01b3b58 0t1123908  TCP servername:ssh-
>workstationame:2654 (ESTABLISHED)
```

It showing that all networked services are owned by root. Iit is a traditional system and buffer overflow atack to one of this services, can gain root (priviledged) access to the system.

Checking for running rpc services:

```
# rpcinfo –p
   program vers proto   port  service
    100000    4   udp    111  portmapper
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
```

Checking for System Resource Controller (SRC) status (listing just services which are active):

```
# lssrc –a | grep active
Subsystem         Group            PID          Status
 sendmail         mail             118844       active
 portmap          portmap          122942       active
 inetd            tcpip            127038       active
 xntpd            tcpip            131136       active
 muxatmd          tcpip            135236       active
 DLMManager                        147550       active
 syslogd          ras              114760       active
```

1.4 Listing of processes running by default

ps –ef

```
    UID   PID  PPID  C    STIME    TTY  TIME CMD
   root     1     0  0   Aug 27      -  0:02 /etc/init
   root 53426     1  0   Aug 27      -  0:00 /usr/lib/errdemon
   root 73820     1  0   Aug 27      -  1:21 /usr/sbin/syncd 60
   root 86090     1  0   Aug 27      -  0:00 /usr/sbin/srcmstr
   root 90316     1  0   Aug 27      -  0:04 opcctla -start
   root 94230     1  0   Aug 27      -  0:02 /usr/sbin/cron
```

```
root 106594      1   0   Aug 27      -  0:00 /usr/ccs/bin/shlap64
root 114760  86090   0 19:12:04      -  0:00 /usr/sbin/syslogd
root 118844  86090   0   Aug 27      -  0:00 sendmail: accepting connections
root 122942  86090   0   Aug 27      -  0:00 /usr/sbin/portmap
root 127038  86090   0   Aug 27      -  0:00 /usr/sbin/inetd
root 131136  86090   0   Aug 27      -  0:07 /usr/sbin/xntpd
root 135236  86090   0   Aug 27      -  1:02 /usr/sbin/muxatmd
root 139332      1   0   Aug 27      0  0:00 /usr/sbin/getty /dev/console
root 172158      1   0   Aug 27      -  1:04 /opt/dcelocal/bin/dced -b -t 1440
root 188514  90316   0   Aug 27      -  0:08 opcmsga
root 192606  90316   0   Aug 27      -  0:04 llbserver
root 196704  90316   0   Aug 27      -  0:06 opcacta
root 200802  90316   0   Aug 27      -  0:10 opcle
root 204900  90316   0   Aug 27      -  0:13 opcmona
root 208998  90316   0   Aug 27      -  0:00 opcmsgi
root 213096  90316   0   Aug 27      -  0:34 coda -redirect
root 221296      1   0   Aug 27      -  0:00 /usr/tivoli/tsm/client/ba/bin/dsmcad
root 229380 151676   1                0:00 <defunct>
root 233586      1   0   Aug 27      -  0:01 /usr/local/sbin/sshd2
root 237688 262322   0 18:06:31  pts/0  0:00 -ksh
root 249956 233586   1 18:03:38      -  0:03 /usr/local/sbin/sshd2
user1 262322 249956   0 18:03:43  pts/0  0:00 -sh
root 270584 237688   4 19:03:40  pts/0  0:00 ps -ef
```


2. System hardening

Determining the role and risk of the System:

The main purpose of this system is housing business applications, simply saying, it is a multipurpose-use system. The location of the UNIX server is protected by firewalls, switches, VPN devices and isolated by private addresses. Thus the main risk is users, services and other conditions inside of the company's intranet.

2.1 Hardening process

2.1.1 Implementing Process ID and group rule:

To prevent one of the most popular attacks – Buffer Overflow (sometimes known as Stack Overwrite), with possible root privileged access, all applications, which do not require root privileges, must be owned by a separate Application ID and group.
All initialization scripts should switch user id to this application prior to starting the application. As we can see in section "1.4 Listing of processes running by default" the OpC (OpenView OperationsCenter) application is running as root. This is because the

initialization scripts by default are owned and started as super user - root. To fix this situation, the first section of the OpC initialization script must be edited to include the line: "su opcuser – ". The opcuser is an ID with primary group opcgroup and is the owner of this application's files. Adding this line makes all related processes also owned by this ID and group.

2.1.2 Performing changes for super user – root.

The primary target for systems intruders is to get super privileges or root access to the system. A Buffer overflow or stack overwrite attack is the most common way to do this. It is almost impossible to know every vulnerability, but it is possible to prevent this particular attack by restricting and auditing super user access to the system.

These changes need to be performed to make system reliable and secure. The "AIX 5L Version 5.2 Security Supplement" paper provides highly practical and useful directions, one of which is to change the settings, default for most commercial Unix systems.

2.1.3 Create home directory for root:

```
# mkdir /root
# chuser home=/root root
# chmod 0700 /root
```

Linux has had this from the beginning, but instead of /root it is /home/root.

2.1.4 To prevent / or /root filesystem from filling up by smit log file and to stay with smit log file, even if filesystem is filled in, better to change default location of these files to something other than / or /root.

Create smit (System Management Interface Tool) files in /tmp directory:

```
# touch /tmp/smit.log
# touch /tmp/smit.script
# touch /tmp/smit.transaction
```

Change permissions to allow only root access to these files:

```
# chmod 0600 /tmp/smit.*
```

Symbolically link files to root's home directory:

```
# ln –s /tmp/smit.log /root/smit.log
# ln –s /tmp/smit.script /root/smit.script
# ln –s /tmp/smit.transaction /root/smit.transaction
```

2.1.5 Create .profile for root and configure it so that default editor will be vi and the prompt will show the username and the system used. It is very useful for self control and to alert/remind about granted privileges.

```
# vi /root/.profile
set –o vi
export PS1='whoami'@'uname –n'':$PWD #'
```

Then, change file permissions to allow only root access.

```
# cmod 0700 /root/.profile
```

2.1.6 Most companies have security policies saying that remote and direct access to system by root is not allowed. Commercial Unix systems allow this access by default while Linux and friends prevent this by default. We are not an exception and change system settings.

Preventing direct login:

```
# chuser login=false root
# chuser rlogin=false root
# vi /etc/login.deny
root
```

Another interesting feature of AIX is account locking after some unsuccessful tries. For root it is not preferable, as it is impossible to perform system administration (including log files analysis, unlocking other users accounts) without super user privileges and if the root account is locked, who other than root can unlock it?

These changes prevent root account locking, if default settings were changed:

```
# chuser loginretries=0 root
# chuser maxexpired=-1 root
```

2.2 /etc/security/login.cfg changing will prevent automated attacks

2.2.1 The default welcome screen shows the operating system name and version. This information helps attackers to make their activity more targeted. OS scanning tools like Fyodor's nmap can use this feature and it is detailed in the following article: http://www.insecure.org/nmap/nmap-fingerprinting-article.html

Changing the login screen and welcome message makes attacks slower:

```
# chsec –f /etc/security/login.cfg –s default –a herald
"*******************************************************************************

 Systems require authorization; unauthorized access is illegal.
*******************************************************************************
```

```
rlogin:”
```

2.2.2 Enabling a Secure Attention Key (SAK) helps to prevent automated tools from scanning for username and password combinations by asking users to enter key combinations every time (such as Ctrl-X or Ctrl-R). This will also prevent 0622 attributes for tty devices and make them 0600 instead of 0622. The /etc/security/login.cfg entry sak_enabled=true enables this feature.

2.2.3 Number of failed login trials, time that wrong login information was entered, time to re-enable login prompt, interval between wrong login attempts and time for user to enter password must be changed from default system settings 0.

The “AIX 5L Version 5.2 Security Supplement” recommended to use the next parameters:

logindisable= 10

logindelay=10

loginreenable=20

logininterval=200

logintimeout=60

maxlogins=128

2.2.4 The complete /etc/security/login.cfg file:

```
# cat /etc/security/login.cfg
default:
sak_enabled = true
logintimes =
logindisable = 10
logininterval = 200
loginreenable = 20
logindelay = 10
herald = “**********************************************************************\n\r* Systems require authorization; unauthorized access is illegal.
*\n\r**********************************************************************\n\r\n\rlogin: „
usw:
shells = /bin/sh,/bin/bsh,/bin/csh,/bin/ksh,/bin/tsh,/bin/ksh93,/usr/bin/sh,/usr/bin/bsh,/usr/bin/csh,/usr/bin/ksh,/usr/bin/tsh,/usr/bin/ksh93,/usr/sbin/uucp/uucico,/usr/sbin/sliplogin,/usr/sbin/snappd
maxlogins = 128
```

```
logintimeout = 60
```

2.3 /etc/security/user

Another important feature is extended user attributes located in /etc/security/user file. Properly configuring this file can make an attackers life more difficult and prevent password guessing. The recommendations in “AIX 5L Version 5.2 Security Supplement” are useful in applying third security rule- “Enforce good password practices and files attributes by users.

These are default user access rules:

registry = files , define authentication method and enforce using other than password file authentication.

umask = 077 , enforce created files sharing.

pwdwarntime = 5 , warn users to change password within 5 days.

loginretries = 3 , lock user account after 3 unsuccessful login tries.

histexpire = 26 , allow to use same password just in a half year.

histsize = 20 , define the number of passwords previously used, that cannot be reused.

minage = 1 , define minimum age of password as one week before change.

maxage = 4 , define password age as four weeks.

maxexpired = 2 , give user two weeks to change password.

The password itself must be a minimum of 8 characters and contain a minimum of two alphabetic and two non-alphabetic characters to comply with my company’s security policy. These options defined by:

minalpha = 2

minother = 2

minlen = 8

Some other useful options are the minimum number of identical characters that may be used from the previous password and the maximum number of repeated characters allowed.

mindiff = 4

maxrepeats = 2

The default section of /etc/security/user defines default rules for all users, but each user still can have their own section. This is a minimum requirement for all users:

```
/etc/security/user
```

```
default:
registry = files
umask = 077
pwdwarntime = 5
loginretries = 3
histexpire = 26
histsize = 20
minage = 1
maxage = 4
maxexpired = 2
minalpha = 2
minother = 2
minlen = 8
mindiff = 4
maxrepeats = 2
```


2.4 Automatic control of defined users, groups, passwords and files settings

To automatically control previously defined users, groups, passwords and files settings, AIX 5.2 has a checking mechanism. It is the usrck, grpck, pwdck and tcbck commands. These commands can disable additional access method, but by default asks before locking an account. The –y flag allow Systems Administrators or Security Analysts to make this process automated. This is what must be done:

To check and automatically lock any user account violated defined rules:

```
# usrck –y ALL
```

To check and automatically lock any user account associated with violation of defined group rules:

```
# grpck –y ALL
```

To check and automatically lock any user account associated with violation of defined password rules:

```
# pwdck –y ALL
```

To check and lock any user account associated with violation of system security state

```
# tcpck –y ALL
```

2.5 Disabling unnecessary Networked services

Disabling unnecessary Networked services helps to minimize the risk of unauthorized access to system.

2.5.1 "AIX 5L Version 5.2 Security Supplement" recommends disabling all entries in /etc/rc.tcpip. Commenting all lines beginning with "start…" will disable unneeded network daemons. Only some of them can be enabled because of their secure nature. For example:

```
# Start up socket-based daemons
start /usr/sbin/inetd "$src_running"
```

Allows to use inetd services enabled in /etc/inetd.conf file, which are controlled by tcp wrapper and configured by /etc/hosts.allow and /etc/hosts..deny confuguration files.

```
# Start up Network Time Protocol (NTP) daemon
start /usr/sbin/xntpd "$src_running"
```

Network Time Protocol daemon will be configured to run only as a client and /etc/ntp.conf (NTP configuration file) contain just client settings:

```
server <ntp_server_name>
driftfile /etc/ntp.drift
tracefile /etc/ntp.trace
```

One very important change to networked services is sendmail service. This service is very importand system administration tool; it can send security reports, file system load reports and other important system messages. By default this service is enabled and can accept remote connections making system open to one more possibly vulnerable service.
To make system less vulnerable without losing important functionality, it is good to disable this service in /etc/rc.tcpip, commenting sendmail startup entry. Then configure cron service so that all messages collected in spool will be delivered to their recipients at the same time freeing up system spool. The entry for root crontab will be next:

```
# Send email every 30 minutes
20,50 * * * * /usr/lib/sendmail -q > /dev/null 2>&1
```

Now, sendmail is not accepting connections on TCP port 25, but send messages initiated by local users

2.5.2 It is recommended to disable all unneeded and vulnerable inetd initialized services, commenting entries in /etc/inetd.conf. There will be two inetd services running Tivoli Management agent, Auto Path and all other entries will be commented, even ssh, Tivoli TSM and OpC daemons will be started using own initialization script.
This is what left in /etc/inetd.conf

```
bgssd  stream  tcp  nowait patrol  /etc/bgs/SD/bgssd bgssd -d /etc/bgs/SD
ctmagt1 stream tcp nowait root /opt/ctmagt1/ctm/exe_AIX/p_ctmag p_ctmag -e /opt/ctmagt1/ctm
```

2.5.3 The /etc/inittab entries are not an exception either. All network related entries, such as inetd, rcnfs, writesrv, httpdlite, imnss, imqss, nimclient, with an exception of rctcpip, to disable unused services, must be commented or deleted. Unlike most AIX configuration files, which uses # to comment entries, inittab uses : (colon). This is what left enabled:

```
init:2:initdefault:
brc::sysinit:/sbin/rc.boot 3 >/dev/console 2>&1 # Phase 3 of system boot
powerfail::powerfail:/etc/rc.powerfail 2>&1 | alog -tboot > /dev/console # Power Failure Detection
mkatmpvc:2:once:/usr/sbin/mkatmpvc >/dev/console 2>&1
atmsvcd:2:once:/usr/sbin/atmsvcd >/dev/console 2>&1
load64bit:2:wait:/etc/methods/cfg64 >/dev/console 2>&1 # Enable 64-bit execs
tunables:23456789:wait:/usr/sbin/tunrestore -R > /dev/console 2>&1 # Set tunables
rc:23456789:wait:/etc/rc 2>&1 | alog -tboot > /dev/console # Multi-User checks
fbcheck:23456789:wait:/usr/sbin/fbcheck 2>&1 | alog -tboot > /dev/console # run /etc/firstboot
srcmstr:23456789:respawn:/usr/sbin/srcmstr # System Resource Controller
rctcpip:23456789:wait:/etc/rc.tcpip > /dev/console 2>&1 # Start TCP/IP daemons
sniinst:2:wait:/var/adm/sni/sniprei > /dev/console 2>&1
cron:23456789:respawn:/usr/sbin/cron
cons:0123456789:respawn:/usr/sbin/getty /dev/console
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
l7:7:wait:/etc/rc.d/rc 7
l8:8:wait:/etc/rc.d/rc 8
l9:9:wait:/etc/rc.d/rc 9
diagd:2:once:/usr/lpp/diagnostics/bin/diagd >/dev/console 2>&1
logsymp:2:once:/usr/lib/ras/logsymptom # for system dumps
perfstat:2:once:/usr/lib/perf/libperfstat_updt_dictionary >/dev/console 2>&1
sshkey:2:once:/usr/local/bin/recreate_ssh2_hostkey
rcdce:2:wait:sh /etc/rc.dce core
ntbl_reset:2:once:/usr/bin/ntbl_reset_datafiles
rcml:2:once:/usr/sni/aix52/rc.ml > /dev/console 2>&1
naudio::boot:/usr/sbin/naudio > /dev/null
ctrmc:2:once:/usr/bin/startsrc -s ctrmc > /dev/console 2>&1
pd:2:wait:/etc/pd/pd start > /dev/console 2>&1    # start the PD servers
```

2.5.4 In most UNIX Security related topics it is recommended to disable the NFS service as it is vulnerable and insecure. Previously disabled rcnfs and portmap entries in rctcpip, disabled NFS service and made /etc/rc.nfs file useless.

2.5.5 The AIX “no” (Network Options) command provides additional networked services flexibility, enabling or disabling some networking options.. To make the system more reliable for Network Scanners it is recommended to change default settings:

To disable ICMP echo responses:

```
# no –p bcasting 0
```

To clean partial connections and prevent SYN flood attacks:

```
# no –p clean_partial_cons 1
```

To disable packets routing and minimize different routing attacks:

```
# no –p directed_broadcast 0
# no –p ipforwarding 0
# no –p icmpaddressmask 0
# no –p ipsendredirects 0
# no –p ipsrcrouteforward 0
# no –p ipsrcrouterecv 0
# no –p ipsrcroutesend 0
# no –p nonlocsroute 0
# no –p tcp_pmtu_discover 0
# no –p udp_pmtu_discover 0
# no –p ipignoreredirects 1
```

The –p flag makes these rules permanent and keeps them after next reboots in /etc/rc.no.

2.5.6 After all previous changes were applied is time to check for available services, so that all future hardening procedures be more accurate and reflect current system configuration.

```
#  netstat –na | egrep “LISTEN|ESTABL”
tcp4       0      0  *.22                   .                    LISTEN
tcp4       0     88  10.10.100.39.22        10.11.111.50.2654    ESTABLISHED
tcp4       0      0  *.32780                .                    LISTEN
tcp4       0      0  *.32786                .                    LISTEN
tcp4       0      0  *.381                  .                    LISTEN
tcp4       0      0  *.383                  .                    LISTEN
tcp4       0      0  *.1581                 .                    LISTEN
tcp4       0      0  *.32780                .                    LISTEN
tcp4       0      0  *.32786                .                    LISTEN
tcp4       0      0  *.10128                .                    LISTEN
tcp4       0      0  *.17132                .                    LISTEN
tcp4       0     88  10.10.100.39.22        10.11.111.50.2654    ESTABLISHED
tcp4       0      0  *.22201                .                    LISTEN
```

Only known ports are open for connections and this is what was the target. Now, after all unneeded networked services are denied is time to create "protection wall" so, that all future changes will be known for system administrator.

2.6 Preventing uncontrolled access to the system employing Packet Filtering feature of AIX IPSEC

The main principle of systems protection is to disable everything and then enable only what is really needed. Unlike most command line firewalls, like ipchains, iptables which first deny all rules and then allow, AIX packet filtering uses most commercial firewalls design principle – PERMIT and then DENY. Other words, it checks rules until found a match.

In situations where one person is responsible for system installation, another person is responsible for applications installation and yet another is responsible for system protection, it is almost impossible to guess what the system is going to do in future. The Unix Security Analyst's task is to prevent this situation and develop the system so that all other additions and changes will be under control. The ideal way to do this is through the implementation of IP Packet filters. AIX IPSEC package contain all the utilities needed to create a very effective packet filtering firewall. If by default it was not installed, these packages need to be installed:

- bos.msg.en.US.net.ipsec
- bos.net.ipsec.rte
- bos.net.ipsec.keymgt
- bos.net.ipsec.websm
- bos.crypto
- bos.crypto-priv
- bos.crypto-wt

In my situation, I just requested these packages to be installed on this machine.

Now, let's create an ipsec device to allow us to use this feature. First, remove all rules defined by system:

```
# rmfilt –n all –v4
```

Then, create ipsec device:

```
# smit ips4_start
```

and select options :

Start IP Security

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

```
Start IP Security                    [Now and After Reboot]          +
Deny All Non_Secure IP Packets       [no]                            +
```

Press Enter and after next screen showed up, exit from smit pressing F3.

```
ipsec_v4 Available
```

Default rule for IPv4 in ODM has been changed.

Successfully set default action to PERMIT

Now checking for device existence:

```
# lsdev –Cc ipsec
ipsec_v4 Available  IP Version 4 Security Extension
```

Next step is to check which rules exist by default and if some restrictions exist, disable them.

```
# lsfilt –v4 –O
|permit|0.0.0.0|0.0.0.0|0.0.0.0|0.0.0.0|no|udp|eq|4001|eq|4001|both|both|no|all packets|0|all|Default Rule
2|*** Dynamic filter placement rule for IKE tunnels ***|no
0|permit|0.0.0.0|0.0.0.0|0.0.0.0|0.0.0.0|yes|all|any|0|any|0|both|both|no|all packets|0|all|Default Rule
```

This rule means that all ports from inside or outside are allowed/permitted and no restrictions are defined.

Now, it is time to make a decision about what to allow and what to deny.

In section 2.5.6 I listed all the currently listening network services. It is TCP ports 22, 381, 383, 1581, 10128, 17132, 22201, 32781 and 32786 for SSH, Auto Path, Tivoli TSM client, Tivoli Management Agent and OPC services, but to have email service available one more TCP port must be enable, it is SMTP TCP port 25. The rules will reflect these needs, but protect from using everything that is unknown.

First rule is to remove all possible and impossible previously created rules:

```
# rmfilt –n –all –v4
```

Then permit all outgoing connections from localhost with separate rule for SMTP and everything what needs to be allowed for incoming connections:

```
# genfilt -v 4 -a P -s 127.0.0.1  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 22 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 22 –w I -i all
```

```
# genfilt -v 4 -a P -s 127.0.0.1  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 25 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 381 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 383 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 1581 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 10128 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 17132 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 22201 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 32780 –w I -i all
# genfilt -v 4 -a P -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c tcp -O eq -P 32786 -w I -i all
```

Deny everything else:

```
# genfilt -v 4 -a D -s 0.0.0.0  –m 0.0.0.0 -d 0.0.0.0 –M 0.0.0.0 -g Y -c all –w I -i all
```

And activate these rules by the command:

```
# mkfilt –v4 –u
```

To deactivate run the command:

```
# mkfilt –v4 -d
```

These rules protect system from unknown services and all new services will be legally requested instead of running by mistake or any other reason.

To list rules run:

```
lsfilt –v 4 -O
```

2.7 SSH and SUDO as an alternative way to use standard UNIX features

2.7.1 Ssh. Implementing SSH as an alternative for remotely accessed services.

In paragraph 2.6 we implemented packet filtering rules and one of the reasons was X Window. X Window is one of the services which are not active by default, but can be activated by any user. This service uses TCP ports 6000 and up and the authentication mechanism is very primitive (host based or key based). The IP address can be changed manually by any user and the keys mechanism is also a well known problem. So, what to do if a user needs this service, but System Administrator does not want to provide an insecure service?

SSH. Everybody knows this tool, but not everybody knows that it is not just a Secure telnet-like or secure ftp-like utility. One of the most interesting features of ssh is tunneling. It is possible to run insecure services through ssh. How do we do this?

For X Window tunneling sshd (SSH daemon) has an internal function as a parameter in the configuration file (/etc/ssh2/sshd2_config) - AllowX11Forwarding and requires only to export your display to an ssh client's IP address.

SSH daemon configurations file:

```
# grep -v "#" sshd2_config
        HostKeyFile                     hostkey
        PublicHostKeyFile               hostkey.pub
        RandomSeedFile                  random_seed
        BannerMessageFile               /etc/issue
        VerboseMode                     no
        SyslogFacility                  AUTH
        Port                        22
        ListenAddress                   0.0.0.0
        RequireReverseMapping           no
        MaxBroadcastsPerSecond          0
        KeepAlive                       yes
        Ciphers                         Any
        MACs                            AnyMAC
        PrintMotd                       yes
        CheckMail                       yes
        UserConfigDirectory             "%D/.ssh2"
        AuthorizationFile               authorization
        SettableEnvironmentVars
LANG,LC_(ALL|COLLATE|CTYPE|MONETARY|NUMERIC|TIME),PATH,TERM,TZ
        AllowX11Forwarding              yes
        AllowTcpForwarding              yes
        AllowedAuthentications          publickey,password
        PasswordGuesses                 3
        PermitRootLogin                 no
          subsystem-sftp                sftp-server
```

In current situation, with just Secure SHell and non critical networked services, there is no need to have tunnel for other that X Window, but to illustrate possibilities of SSH tunneling

and to show another practical way for secure clear text networked services use, SSH tunnel for SMTP service will be a good example. The command will look like this:

```
# ssh <user>@<server> -L 250:localhost:25
```

This tunneling rule allows user to connect to <server> TCP port 25 not directly, but rather through his local machine's TCP port 250. The same can be done with any other TCP service.

2.7.2 SUDO. Implementing SUDO as an alternative for "su" command and more flexible and advanced way to control access.

Another point of systems security risk is Access control. Big companies have big systems with many different applications running. Many different user groups exist and perform applications maintenance, analysis, etc. Applications have own IDs, users have their own.

For security reasons, Application IDs should not have a shell defined in the password file (instead of /bin/sh they should have /bin/false). There is also no reason for Application ID's to have remote login capability. Some applications, for example Data Bases, use Unix internal authentication mechanism. Disabling login in smit or adding ID to /etc/login.deny file disables the application's own remote communication (for example ODBC). That's why some Application ID's have /bin/false in /etc/passwd, instead of /bin/sh and are allowed to login access.

Our Maintenance team, in order to perform their job, need to be logged in as an Application ID, but without a shell they can not get a command prompt. In this situation the "su" utility is useless and sudo is one of tools which allow the Application ID to attain a shell.

This is an example of SUDO configuration file,

```
# visudo
"/etc/sudoers.tmp"


# base
User_Alias      AIXDM= user1, user2, user3
User_Alias      USEC=user4, user5, user6
Cmnd_Alias      SUPPAIX=/usr/sbin/shutdown -Fr,/usr/sbin/shutdown -F,\
/usr/sbin/cfgmgr,/usr/sbin/rmdev,/usr/sbin/ifconfig,\
/usr/local/bin/maymap
%support  ALL=NOPASSWD: PERFORMANCE
%system   ALL=NOPASSWD: PERFORMANCE
####################
root      ALL=ALL
USEC      ALL = /usr/bin/chmod
tapelib   ALL = /opt/rr/rr,/opt/rr/xrr
unxadm    ALL = NOPASSWD: /var/adm/issuance/unixupdate
```

```
support  ALL = NOPASSWD: SUPPAIX
########################
# PROC ID SHELL
User_Alias APPSUPP=support, appsupp
Runas_Alias APPID=webappid, monappid
Defaults>APPID targetpw
APPSUPP ALL = (APPID) PASSWD: /bin/sh
```

The section “PROC ID SHELL” illustrates a rule which allows support and appsupp users to execute shell (/bin/sh) as webappid or monappid. Specifying “targetpw” defines the condition where command execution will prompt for the password of the user specified by the -u flag instead of the password of the invoking user.

The command

```
support@server $ sudo –u webappid /bin/sh
password:
webappid@server $
```

provided a command prompt as user webappid without /bin/sh in /etc/passwd file.

3. Post configuration system maintenance

3.1 Checking for files permissions

The final stage of system hardening is checking files permissions with different levels of risks.

For example, a risk is associated with files that are not owned by any group or user because a newly created user or group can take ownership of them. These two commands can find these files and provide information about their settings:

find / -nouser -type f -exec ls -l {} \;

find / -nogroup -type f -exec ls -l {} \;

SUID, SGID files, known as high risk level files, also must be under control. The risk is that users with lower level privileges can possibly execute with higher level of privileges. Next two commands help to find these files in system.

find / -perm -4000 -type f -exec ls -l {} \;

find / -perm -2000 -type f -exec ls -l {} \;

World writable files allow everybody to change their content and are a possible target for Trojan Horse attack or just file damaging or destruction.

These four commands can help find files with world writable or group writable permissions.

find / -perm -0002 -type f -exec ls -l {} \;

find / -perm -0002 -exec ls -ld {} \; | grep drwxrwx

find / -perm -0020 -type f -exec ls -l {} \;

find / -perm -0020 -exec ls -ld {} \; | grep drwxrwx

It is good to perform a check for file permissions a minimum of one time a day. A simple script can do this. First create the file where all information will be placed and change permissions to Read-Write for just owner:

```
# touch filescheck.txt
# chmod 0600 filescheck.txt
```

Then run script for first time to collect information about all files with high risk permissions.

```
# cat filecheck.sh


#!/bin/sh
mv filescheck.txt filescheck.bak
```

```
echo "No user, no group files" > filescheck.txt
find / -nouser -type f -exec ls -l {} \; | egrep -v "/dev|/tmp|/proc" >> filescheck.txt
find / -nogroup -type f -exec ls -l {} \; |egrep "/dev|/tmp|/proc" >> filescheck.txt
echo "SUID and SGID files" >> filescheck.txt
find / -perm -4000 -type f -exec ls -l {} \; |egrep "/dev|/tmp|/proc" >> filescheck.txt
find / -perm -2000 -type f -exec ls -l {} \; |egrep "/dev|/tmp|/proc" >> filescheck.txt
echo "World and groups writable files" >> filescheck.txt
find / -perm -0002 -type f -exec ls -l {} \; |egrep -v "/dev|/tmp|/proc" >> filescheck.txt
find / -perm -0002 -exec ls -ld {} \; | grep drwxrwx |egrep -v "/dev|/tmp|/proc">>
filescheck.txt
find / -perm -0020 -type f -exec ls -l {} \; egrep -v "/dev|/tmp|/proc" >> filescheck.txt
find / -perm -0020 -exec ls -ld {} \; | grep drwxrwx |egrep -v "/dev|/tmp|/proc">>
filescheck.txt
# Finding changes for previous filescheck
diff filescheck.txt filescheck.bak > riskfiles.txt
newriskyfiles = `ll riskfiles.txt | awk '{print $5}'`
if [$newriskyfiles = 0]
then
exit
else
mailx -s "new risky files" user@mail < riskfiles.txt
fi
```

All other times this script will collect and sent e-mail, containing information only about newly created files with high risk permissions.

By scheduling this script with cron we can regulary check for risky files and receive reports regularly.

The Crontab entry to run command every midnight will be next:

0 0 * * * filecheck.sh

The command “filecheck.sh”, showed that there were no files without user or group defined and many files with SUID, SGID permissions, but all of them known applications, like uptime, crontab and other system utilities.

3.2 Checking file for changes

Files to be checked for changes using simple ksh script:

```
# cat check.sh
#!/bin/sh
set -x
touch check.list.sum.prev
cat check.list.sum > check.list.sum.prev
cat /dev/null > check.list.sum
tar -cvf checked_files.‘date +”%d%m$y”‘..tar check.list
while read i
do
tar -rvf checked_files.‘date +”%d%m$y”‘..tar $i ;\
cksum $i >> check.list.sum
done < check.list
 diff check.list.sum check.list.sum.prev > /tmp/check.list.tmp
d=‘ls -l /tmp/check.list.tmp | awk ‘{print $5}’‘
if [ $d=0 ]
then mailx –s “monitored files were not changed in the past 24 hours” usr@dom ; exit
else mailx -s “files were modified 24 hours back from ‘date‘!” usr@dom <
/tmp/check.list.tmp
fi
```

and this is a check.list file:

```
# cat check.list
/etc/inetd.conf
/etc/rc.tcpip
/etc/inittab
/etc/security/login.cfg
/etc/security/user
/etc/sudoers
/etc/rc.nfs
/etc/passwd
/etc/security/passwd
```

```
/var/adm/cron/at.allow
/var/adm/cron/at.deny
/var/adm/cron/cron.allow
/var/adm/cron/cron.deny
/tmp/setuid
/tmp/setgid
/etc/host.equiv
/etc/ssh2/sshd2_config
/etc/.rhosts
/etc/.netrc
/root/.profile
```

The script uses AIX cksum command to generate a unique string for each file and compares this cksum value with the previous recorded cksum value. If changes were found it sends an e-mail message with the diff command output, similar to this:

```
2c2
< 1476426758 1983 /etc/sudoers

3606404487 1978 /etc/sudoers
```

It is very simple to add more details, such as what exactly was changed, but there was no need for this. It is enough to have copies of these files in /opt/checkfiles/bak/ for future analysis. To have this check both regularly and automatically it should be scheduled through cron:

```
# crontab –e
0 0 * * * /opt/checkfiles/check.sh
```

Every midnight important files changes report will be generated.

One more recommendation - keep all scripts and related files protected, this will prevent inaccurate and maybe not informative reports generation.

4. Resume

The main idea of this Practicum was to provide direction for Unix Security Analysts in making system more secure and provide practical recommendations. Some tools like ssh, sudo, no, IPSEC were involved in the hardening process and some practical methods of their usage were shown. For example, SSH as a tunneling tool, sudo v.1.6.8's new feature – Defaults> and "no" as a tool for network scanning control. Packet Filtering was introduced as a way of system hardening and preventing unauthorized networked services startup.

Targets reached:

1. Remove unneeded services from the system:

Cleaning of networked services configuration files, like inetd.conf, rc.tcpip, etc., combined with packet filtering, created minimized access to the system through networked services.

2. Protect root password and root access:

The password is protected by system password encryption mechanisms and direct root login being disabled. Implementing the rule for all applications to run by a unique Application ID and group and changing of initialization scripts to switch to this ID and group helps to prevent Buffer Overflow or Stack Overwrite attacks.

3. Enforce good password practices and files attributes for users:

Creating new rules for password, changing /etc/security/login.cfg and /etc/security/user and then implementing these rules checking mechanisms using usrck, grpck, pwdck and tcbck utilities makes the system more reliable and hard for automated password scanning attacks.

4. Hide system information for remote users by removing default systems and services welcome banners:

Changed the default welcome screens for system and network services, protected system and made automated OS detection and scanning tools like nmap, useless.

5. Avoid "clear text" networked services, such as telnet or ftp and use SSH instead.

Implementing secure communication methods makes sniffing useless. SSH tool with SSL libraries were involved as the main communication protocol. For other, "clear text" services, ssh tunneling was recommended and a real example was presented.

6. Avoid "clear text" storage of critical data, like passwords, personal and business information

All system passwords are protected by Unix crypt library and file permission checks are going to protect system from world read/write files creation.

Two scripts are involved to this process: check.sh to check for changes of important system files and filescheck.sh to check for high risk files permissions. This makes the system protected from unauthorized access to system data and processes.

7. Allow only controlled access to the system and prevent direct super user (root) access, using SUDO and existing system features.:

Disabling services like rsh, rlogin (and other “r” services), tftp and others which do not require authentication prevents unauthorized access to the system and employs a better of system access control.

SSH is the only way to access system and direct rot login is disabled, plus SUDO or “restricted super user access” rules for those who need some administration privileges.

8. Prevent unauthorized access to system, using packet-filtering feature of AIX IPSec tools.

IPSEC as AIX Packet filtering feature let System Administrator control and be informed of all future network related system changes

9. Automate systems review process, automatically creating and sending daily reports, using regularly executable scripts.

Controlling of system status and performing regular system checks makes the system more protected. For companies with many systems in different locations it is almost impossible to check all logs every day. An automated checking mechanism was implemented to cover this problem. Every night special scripts perform system analysis and send a message about the most important changes to the system made within the last twenty four hours.

The last and very important recommendation for Security analysts is to be always informed.. Read works recognized in the Unix Security field and check bugtraq mailing list. Reading of www.securityfocus.com and being subscribed to bugtraq mailing list (to subscribe just send an email to bugtraq-subscribe@securityfocus.com) helps to be informed as soon as new bugs are discovered and ensures hat your knowledge of Unix Security is always fresh and updated.

5. References

1. Nemeth, Evi, et al. Unix System Administration Handbook. 3rd ed. Prentice Hall. 2000.

2. Frisch, Aeleen. Essential System Administration 2nd ed. O'Reilly & Associates, Inc. 1995

3. Garfinkel, Simson and Gene Spafford. Practical UNIX and Internet Security 2nd ed. O'Reilly & Associates, Inc. 1995

4. MOREnet, "Unix Security Best Practices" URL: http://www.more.net/security/best/unix10.html

5. SecurityFocus URL: http://www.securityfocus.com

6. Steven Tuttle, Gabriel Pizano, Chris Smith. AIX 5L Version 5.2 Security Supplement, IBM International Technical Support Organization, Nov. 2003, URL: ibm.com/redbooks

7. Yoshimichi Kosuge, Francois Armingaud, Lip-Ping Chew, Leonie Horne, Timothy Witteven. AIX 4.3 Elements of Security: Effective and Efficient Implementation. IBM International Technical Support Organization, Aug 2000

8. Sandor W. Sklar. AIX Packet Filtering with IPSEC, e-mail: ssklar@stanford.edu

9. David Batten, Antonio Joglar, Linda St. Clair, Susan Schreitmueller, Rebecca Sanchez. Strengthening AIX Security: A System Hardening Approach, Mar 26th 2002