Global Information Assurance Certification Paper

Case Study In Achieving A Baseline Level Of Computer And Network Security: Recommendations To Student Unix And Linux System Administrators

SANS Security Essentials (GSEC) Practical Assignment

Version 1.4b – Option 2

Completed By: Michael Mitchell

Date Submitted: January 7, 2004

TABLE Of CONTENTS

1.0 Abstract

We are establishing new network and computer security policies and standards that will apply to all the computers on our local network at a major medical school. This paper is written to be a guide to the many student Unix and Linux system administrators on how to set up a baseline level of computer and network security that will implement our new policies. This paper describes the academic and political constraints that affect our policies and standards. Four of my network and computer security philosophy points are enumerated. Eight system and network security software packages are recommended for installation, and configuration hints are given for these eight software packages. I also recommend that the student administrators run a nightly shell script to assess the system security and report a digest of the system logs. Excerpts from these shell scripts are presented to clearly describe the use of the scripts.

2.0 Introduction

I work in a major medical school where many departments exist in a mixed academic and hospital environment. The local network itself (routers, switches, etc.) is mostly flat and professionally managed, but normal system administrators have no control over the network. The virus/worm attacks of last summer and new HIPAA regulations have led to a rethinking of how the network is managed, what policy changes may be needed, and what the minimum operating standards should be for ALL computers that are allowed access to the local network.

As a member of a committee to establish these new policies and standards, I want to provide some method for the more than 100 non-professional (student) Unix and Linux computer administrators to be able to perform a baseline level of daily system checks, to perform intrusion detection checks, and to generate system log digests. Most important of all, the nightly checks should be concise enough so that these student administrators will check the results on a regular (daily) basis.

For several years I have used homegrown shell scripts to examine Unix and Linux computers on a nightly basis in order to generate system log digests, to monitor system resources, to perform intrusion detection, and to mail to various system administrators the nightly results. My intention in presenting this paper is to simplify these scripts, to update them with more modern intrusion detection methods, and to document what these scripts do. I also will describe a number of system security methods and tools that I have found useful over the years, and suggest configurations for a few important server daemons. It is my hope that

this paper, pointers to source or precompiled binaries of security software, and a set of shell scripts will aid the student system administrators in meeting our new network policy objectives.

3.0 Network Security Philosophy

I hope to impart to our student administrators four tenets of my network security policy:

a. Regular disk backups can be the best security of all.
b. Have multiple levels of security. If one level fails you will still be secure.
c. If a network daemon is not running, it cannot be hacked.
d. If hackers cannot get to a network daemon, they cannot hack it.

4.0 Constraints

There are a number of features of the academic enterprise that affect the policies, standards and procedures that we develop:

a. There are approximately 200 faculty owned, Unix and Linux computers on our network with only about 75 receiving professional system administration. The rest of the computers are managed by faculty, by students, or by nobody at all.
b. Many faculty feel that they "own" their computers, and many insist on having full root access to their machines, and on giving their students full root access as well. Use of the Sudo program to restrict root access for faculty and students generally is unacceptable.
c. In the academic environment some computers have enterprise-class system software (volume managers, backup managers) and hardware (tape libraries, hardware support agreements). Unfortunately, most computers have only free or cheap system software and no hardware support agreements.
d. Many faculty do not want to spend money on tape backup hardware or tapes. Many computers have no backup capability at all and many others have only CD burners available.
e. Faculty members often purchase computers without oversight, and without much concern for how the new machines will fit into the existing network and administration structure.
f. Ancient academic politics mean that some faculty want nothing to do with established professional computer support organizations.

Living with the above constraints means that only a few larger file servers and compute servers have adequate backup hardware and software, and receive nightly backups. We encourage users to keep their data on the file servers so

that the data may be properly backed up. Many client computers receive only one or two backups a year. Despite informing all faculty and students that their computers do not receive regular backups, some still keep important data on client computers. Several times a year we are called out to try to resurrect data from crashed disks. In addition, any software we choose to implement the newly established standards and policies will have to be freeware or shareware.

There are several other positive and negative features of our computing and network environment that impact our work:

a. Most users rely on central mail servers.
b. The border router between the University and the Internet catches spoofed IP packets.
c. Ports 137 and 139 are blocked at the border router.
d. For political reasons, RPC traffic is not blocked at the border router.
e. We do not run AIX or HP-UX so we do not have to support all versions of Unix.
f. Most of our client machines and servers are used for academic research purposes, and have fewer than 100 users.

5.0 My Suggestions

Below are eight suggestions that I hope will be adopted by many of the student Unix and Linux system administrators. Some are simply repetition of the obvious (back up your system), and some are recommendations of system software and advice on how to configure the recommended software.

5.1 Back Up All Data

My number one suggestion for the student system administrators is to put all data onto our file servers, which are backed up every night, or onto someone else's file servers that are backed up every night. If the faculty and students absolutely have to have data on their client computers, I recommend that they perform regular nightly backups. Users are far, far more likely to lose data to a crashed disk than to any other cause. In seven years at my current job, I have seldom seen files lost to hacking. I have seen many cases of crashed disks with no backups.

5.2 Turn Off Unnecessary Network Daemons

My second suggestion is that student administrators should not keep unnecessary network daemons running or active. Though many manufacturers are getting better about delivering operating systems with most network daemons

turned off or inactivated, not all do so. In addition, many existing, student managed Unix and Linux machines have older versions of operating systems that still have many unused daemons turned on (time, echo, discard, daytime, chargen, rpc.ttdbserverd, rpc.rexd, shell, exec, talkd, fingerd, etc.). These daemons are usually deactivated by removing their entries (commenting them out) from the /etc/inetd.conf file and rebooting or sending the HUP signal to the inetd process (see tcp-wrappers discussion below for how to send the signal).

5.3 Use Secure Shell

Suggestion number three is use the OpenSSH (http://www.openssh.org) Secure Shell network communication programs ssh, sftp, and scp (password and data are encrypted) in place of the insecure (password, and data pass in the clear over the network) telnet, ftp, rlogin, rsh, and rcp. For some reason, we encounter great resistance to this suggestion. We commonly hear, "We can't do our work if we can't run ftp." Since we currently are not able to force anyone to make this change, we try to chip away at it a little bit at a time. If the faculty and students absolutely have to run ftpd or telnetd to do their work we at least try very hard to get them to run ftpd or telnetd through tcp-wrappers (more on this below). We are not always successful in these attempts.

5.4 Keep Operating Systems Patched

Suggestion number four for the student system administrators is to keep their operating systems patched. It is easier now than in the past to get and install Unix and Linux operating system software upgrades. Patch clusters are often available for system software upgrades, and many of the vendors maintain newsgroups for distributing patch upgrade information.

5.5 Use Tcp-wrappers and/or Xinetd

Suggestion number five is to use tcp-wrappers or xinetd to protect network daemons from unwanted external access. The tcp-wrappers program restricts access to many network daemons based upon the source IP of the incoming network traffic. Good candidate daemons are ones that fork a short-lived daughter process that dies upon completion (telnetd, ftpd). Daemons that are started once at startup and continue running (httpd), are not good candidates. Since our university blocks spoofed IP packets at the border router, tcp-wrappers can allow only local connections to insecure telnet or ftp daemons. It is far better to use Secure Shell instead of telnet or ftp, but the tcp-wrappers program does afford a measure of security. In my own work, I often protect Secure Shell with tcp-wrappers. Have multiple levels of security. If one level fails you should still be secure.

To illustrate this point further, I will relate the following anecdote: the first week in my current assignment a major insecurity in telnetd was publicized and patches released. Since I had just installed tcp-wrappers on every Unix and Linux machine in sight and restricted incoming connections to those originating within the medical school, I went home at a decent hour Friday night fairly sure that we would not be hacked before I could get back in on Sunday to finish installing all the new telnetd daemons.

The tcp-wrappers program (http://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz) by Wietse Venema is available for virtually every flavor of Unix and Linux. The tcp-wrappers binary, usually tcpd, can be installed in any directory, but is usually found as /usr/sbin/tcpd or /usr/local/sbin/tcpd. For each daemon to be protected, change that daemon's entry in /etc/inetd.conf to route the incoming network traffic to tcpd. Tcpd then consults the /etc/hosts.deny and /etc/hosts.allow files to determine whether to block the traffic or to route it to the appropriate network daemon. For example on an Irix machine, to add tcp-wrappers protection to the ftpd daemon change the following line in /etc/inetd.conf:

```
ftp stream tcp nowait root /usr/etc/ftpd ftpd -lp
```

to

```
ftp stream tcp nowait root /usr/sbin/tcpd /usr/etc/ftpd -lp
```

Then signal the inetd daemon to reread the /etc/inetd.conf file:

First run "ps -ef |grep inetd" to get the PID of the inetd process

Then signal the process with "kill –HUP PID"

Most Unix and Linux operating systems have commands for directly signaling a process without having to first get its PID. In IRIX the command is: "killall -HUP inetd" and in Solaris the command is "pkill -HUP inetd."

After enabling tcp-wrappers on the daemon of choice, then, if it does not already exit, create an /etc/hosts.deny file containing the line:

```
ALL: ALL
```

Also create a /etc/hosts.allow file containing:

```
ftpd:    LOCAL, .med.school.edu
```

Run the "man hosts.allow" command for more details on how to configure the hosts.allow file.

The xinetd program (http://www.xinetd.org/xinetd-2.3.12.tar.gz), often found in place of inetd in Linux, works in a similar manner to tcp-wrappers. Each daemon to be controlled by xinetd has a file entry in the /etc/xinetd.d directory. The appropriate file in the /etc/xinetd.d directory is edited to turn on or off a daemon or to add IP restrictions. For instance in Red Hat Linux in the /etc/xinetd.d/telnet file change:

disable = no

to:

disable = yes

to turn off a daemon entirely. To add IP restrictions, add a line like:

only_from = 192.168.20.0 143.18.0.0

to only accept network traffic from the 192.168.20.0 or 143.18.0.0 networks.

Some versions of Linux come with xinetd configured to add a tcp-wrappers-like function. With this function active, one can use both /etc/xinetd.d files and /etc/hosts.allow to restrict by source IP what network traffic is granted access to many network daemons. A good tutorial on xinetd by Raynal can be found at http://www.linuxfocus.org/English/November2000/article175.html

The tcp-wrappers program can also be used to protect exported Samba shares (http://www.samba.org) from a Unix or Linux computer. We have certain special projects that require serving Samba shares to Windows clients. There is not a lot of activity on these servers, so we run the Samba netbios daemons out of inetd through tcp-wrappers using entries in the /etc/inetd.conf file like:

netbios-ssn stream tcp nowait root /usr/sbin/tcpd /usr/samba/bin/smbd

and

netbios-ns dgram udp wait root /usr/sbin/tcpd /usr/samba/bin/nmbd

Appropriate smbd: and nmbd: entries in /etc/hosts.allow are also needed. Since we do not have many users in the academic environment, the extra overhead for running Samba shares through tcp-wrappers is inconsequential. I far prefer knowing that only ten machines can access my Samba daemons instead of ten thousand.

5.6 Secure Sendmail

I recommend that the student system administrators secure any Sendmail daemons that they run.

The Sendmail daemon of versions 8.11.x and earlier only has to be running on a Unix or Linux computer if that computer is a mail server. If a user sends an e-mail using a mail client program, the client simply invokes Sendmail directly to send the message. No Sendmail daemon has to be running. When it is necessary to run a mail server using Sendmail 8.11.x on a lightly loaded system, I recommend running Sendmail out of inetd through tcp-wrappers using an /etc/inetd.conf line like:

smtp stream tcp nowait root /usr/local/bin/tcpd /usr/lib/sendmail -bs

Even older, "pre-anti-relaying" versions of Sendmail can be afforded some level of spam protection using this method. Sites with heavy mail traffic probably would not be able to protect Sendmail this way. Sites handling a few hundred to a few thousand mail messages per day, and running older versions of Sendmail through inetd and tcp-wrappers should work fine.

Two Sendmail daemons need to be running on a computer for a user to be able to send a mail message using 8.12.x and newer versions of Sendmail. One non-privileged daemon accepts user messages for processing on port 587, and one privileged daemon runs as root bound to port 25 to deliver e-mail. Despite all assurances that Sendmail will not forward SPAM or cannot be hacked, I still like to have restrictions on who can get to the Sendmail daemons (multiple layers of security philosophy). Since most users of our student administered Unix and Linux computers will only want to send out e-mail, the Sendmail daemons on these client computers only need to allow access to localhost. This can be achieved by a pair of "daemon_options" entries in the Sendmail ".mc" configuration file:

DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1,Name=MTA)
DAEMON_OPTIONS(`Port=587,Addr=127.0.0.1,Name=MSA,M=E)

Then rebuild the sendmail.cf file. The Sendmail daemons then will ignore all external network connection attempts. Directions for making these modifications are widely available. One specific source is http://www.sun.com/bigadmin/features/articles/config_sendmail.html. For example, on Irix machines, edit the /etc/sendmail.mc file and run "configmail mc2cf" to rebuild the /etc/sendmail.cf file. On Solaris 9 machines, edit /usr/lib/mail/cf/main-v7sun.mc file and run "make sendmail.cf" to rebuild the /etc/sendmail.cf file. You will have added another layer of security to an important daemon. I run a single Sendmail server for receiving the nightly security reports and periodic MD5 file checks from all the supported Unix and Linux machines. It is an 8.11.x version run out of inetd through tcp-wrappers, and it only accepts mail from the few machines whose IP numbers are in the

/etc/hosts.allow file. Sometimes I record lots of attempts to connect to this Sendmail daemon from computers outside the university. All the attempts fail. If a new exploit for Sendmail is discovered and released, Sendmail daemons configured as described above probably will not be hacked because the hackers probably will not be able to access the Sendmail daemons. An extra layer of security will be in place.

5.7 Use Secure Portmap Or Rpcbind

I recommend that our student system administrators run either a secure portmap or rpcbind program. Since RPC traffic is not blocked at the border router between our university and the Internet, it is important to implement some sort of host-based blocking of RPC traffic. Where possible, I recommend that RPC traffic be turned off entirely in client Unix and Linux computers. Unfortunately many of our Unix and Linux computers run NFS and NIS and need RPC turned on. If we have to allow RPC traffic, and have to allow it for as wide a network domain as for all of .edu, we still will have inhibited a world of possible hackers.

Secure versions the portmap (http://ftp.porcupine.org/pub/security/portmap_4.tar.gz) and rpcbind (http://ftp.porcupine.org/pub/security/rpcbind_2.1.tar.gz) daemons restrict external access to RPC information based upon the source IP address of an incoming network packet. Many versions of Linux have secure portmap or rpcbind capability built into the operating system. For instance, Red Hat Linux runs a portmap daemon that refers to “portmap:” entries in the /etc/hosts.allow file to control access to RPC information. FreeBSD supplies a version of rpcbind that refers to “rpcbind:” entries in /etc/hosts.allow.

On Solaris machines, we use a locally compiled versions of the secure rpcbind program, that refers to “rpcbind:” entries in /etc/hosts.allow to allow external network access to RPC port assignments. On Tru64Unix machines we use the secure portmap daemon which refers to “portmap:” entries in the /etc/hosts.allow file. On Irix machines, the vendor supplied portmap program reads, at startup, the /etc/config/portmap.options file for RPC traffic control. Entries in /etc/config/portmap.options file to allow RPC access are of the form “-a netmask,network” where “network” can be a single IP address or an IP network entry.

Unlike tcp-wrappers, where daemon access is completely controlled, secure versions of rpcbind and portmap do not completely block RPC traffic. The secure versions of rpcbind and portmap only inhibit reporting of what port a particular RPC service is assigned to. Hackers are still free to perform RPC port sweeps looking for vulnerabilities. We frequently see these port sweeps from the Internet, and cross our fingers, hoping that our intrusion detection will find any

successful hacks. We also frequently log improperly configured NIS clients broadcasting for NIS servers. Every RPC connection attempt is not malicious.

In addition to the above restrictions, on Sun Solaris machines we recommend running the syslogd daemon with the "-t" option so that syslogd only accepts log entries from localhost.

5.8 Use Intrusion Detection Software

To upgrade intrusion detection software on our student managed machines, I recommend using the program AIDE (Advanced Intrusion Detection Environment) by Lehti and Virolainen (http://www.cs.tut.fi/~rammer/aide.html). AIDE checks the integrity of system files. Most system files should have checksums that do not change from day to day. Changes in the checksum values may indicate that a machine has been hacked.

The AIDE (http://www.sourceforge.net/projects/aide/aide-0.9.tar.gz) program creates a database of file and directory descriptors, which can include permissions, owner (user and group) time modified, time changed, md5 checksum, and many others. The administrator can direct AIDE to save any combination of file system descriptors. Once the database is created, the file system can be checked again, and any differences in file or directory descriptors will be reported.

When AIDE is run, a report of file system differences is created and can be sent to standard output. AIDE reads a configuration file which delineates what files or directories are to be included or excluded from the database, where the database is located, and where the report should be sent.

Directing AIDE to check all files and directories in busy file systems will generate voluminous output. To keep the report to a manageable size, AIDE users have to skip checking parts of file systems that rapidly change. A beginner's AIDE configuration file (aide.conf) may only contain:

```
database=file:/usr/security/aide.db
database_new=file:/usr/security/aidb.new
report_url=stdout

/       p+md5

!/tmp
!/proc
!/var/tmp
```

With the above aide.conf file, command line options for AIDE include:

/usr/security/aide init

/usr/security/aide check

/usr/security/aide update

The init and update options will create a new database of file system descriptors at /usr/security/aidb.new. The check option will compare the current file system descriptors with the descriptors recorded in /usr/security/aide.db. The above aide.conf file directs that permissions and md5 checksums should be checked for all file systems (/ p+md5). However, the /tmp, /proc, and /var/tmp directories will not be checked. Further details on AIDE operation and configuration can be found in the distribution or man pages.

An obvious problem with AIDE is that anyone who can read the aide.conf file will know what parts of the file system are not being checked. Hackers can hide malicious programs in the unchecked directories. Another obvious problem is that, once a machine is hacked, the hacker can substitute his own version of aide.conf or AIDE. At this point, we plan to serve up a read-only distribution of the programs and scripts for each client computer to use.

As a further increase in intrusion detection capability, I recommend running chkrootkit (http://www.chkrootkit.org) on a nightly basis. The program requires little configuration, and when run checks the / file system and all lower file systems and subdirectories for signs of a rootkit hack. Among other things, chkrootkit checks for modified system binaries such at ps or ls, and checks for truncated or modified system logs. Unless one of the chkrootkit tests returns "INFECTED" as a result, the system is probably OK.

6.0 Nightly Script Excerpts

The first decision in updating the old nightly system check scripts was to create separate shell scripts for each operating system. Attempting to fold the directives for all operating systems into one script was unwieldy, hard to modify, and hard to debug. The second decision was to install GNU versions of some commands (such as "find") to enable using the same arguments in the scripts for all operating systems. As outlined above, I am recommending that student system administrators use precompiled binaries of the new system and security programs (chkrootkit, tcp-wrappers, portmap, rpcbind, etc.). Most of the student administrators are far more interested in performing academic research than in administering computers. I am trying to convince skeptical people to get on board with a new policy, and everything I can do to help things go more smoothly can only help. I also decided to use sh and csh shell scripts instead of Perl,

because every Unix or Linux has those shell interpreters. Not all machines will have Perl correctly installed.

Most of what the scripts do is to compare today's log files with cached versions of yesterday's log files and report differences. The overall philosophy in the scripts is to report everything that is not explicitly ignored. There are points at which the scripts look for a particular pattern in the logs such as "refused," "connect," "sendmail," etc.). But the same logs are also searched with much less restrictive searches.

I show below a number of excerpts from these shell scripts. For clarity, only the most important lines of shell code are shown. Some system checks are strictly housekeeping such as checking for stuck mail queues or checking for overfull disk partitions. The typical script establishes logfile on stdout and then:

Syncs system time with campus clock servers

```
ntpdate clock1.med.school.edu
```

Prints out OS version and hardware

```
uname -a or uname -R; arch
```

Checks mail queues

```
/usr/bin/mailq or /usr/lib/sendmail -bp

When running Sendmail 8.12.x:

/usr/lib/sendmail -bp
/usr/lib/sendmail -Ac -bp
```

Checks for full partition sizes:

```
df -k  | grep "/dev/" | sed '/.*1..%.*/\!d'
df -k | grep  "/dev/" | sed '/.*9.%.*/\!d'
```

Extract today's logs:

Solaris

```
diff /usr/adm/messages /usr/adm/messages.old > /usr/adm/messages.diff
diff /usr/adm/sulog /usr/adm/sulog.old > /usr/adm/sulog.diff
diff /var/log/syslog /var/log/syslog.old > /var/log/syslog.diff
last > /var/log/last.current; diff /var/log/last.old /var/log/last.current
```

Irix

```
diff /usr/adm/SYSLOG /usr/adm/SYSLOG.old > /usr/adm/SYSLOG.diff
```

```
diff /usr/adm/sulog /usr/adm/sulog.old > /usr/adm/sulog.diff
```

Linux:

```
diff /var/log/messages /var/log/messages.old > /var/log/messages.diff
diff /var/log/maillog /var/log/maillog.old > /var/log/maillog.diff
```

Checks message logs for interesting things and, at some point, report everything that is not explicitly ignored (Solaris example):

```
egrep 'logi|su:|SU:|endmail|vent|sysctlrd|exit|arnin|nable|hutdown|
        eboo|NFS|such|hang|setui|therne|ailed|rror|amed|estar|niti|le0|rrno|
        ecc|conn|ANONY|PAS|denie|ailure|pbind|lock'
        /usr/adm/messages.diff >> $logfile

egrep -v  "pted pub|word req|iled pass|KAPWEXPIRED|t use ill"
                                        /var/log/syslog >> $logfile

grep -v sendmail /var/log/syslog.diff | egrep "popper|comsat|
        loopback" >> $logfile

grep sendmail /var/log/syslog.diff | cut -c1-110 >> $logfile
```

(Irix example):

```
egrep 'logi|su:|SU:|endmail|vent|ysctlr|aile|arnin|nable|hutdown|eboo||NFS|
        uch|ang|etui|herne|rror|amed|estar|itial|le0|rrno|ecc|ANONY|PASS|
        denie|ailur|pbind' /usr/adm/SYSLOG.diff >> $logfile
```

Checks su log

```
cat /usr/adm/sulog.diff >> $logfile
```

Checks ftpd log, where applicable (Red Hat Linux example):

```
diff /var/log/xferlog /var/log/xferlog.old  >> $logfile
```

Checks for non-local tcp-wrappers entries in logs and for refused RPC requests (Solaris example):

```
grep refused /var/log/syslog.diff >> $logfile

grep connect /var/log/syslog.diff |grep -v refused >> $logfile
```

Checks mail log if applicable

(Solaris example):

```
grep sendmail /var/log/syslog.diff | cut -c1-110 >> $logfile
```

(Linux example):

```
cat /var/log/maillog.diff >> $logfile
```

Checks for changes in passwd file:

```
diff /etc/passwd /usr/local/security/passwd.save >> $logfile
```

Checks for changes in shadow file, if applicable (no sense printing the encrypted password to the security log file):

```
diff /etc/shadow /usr/local/security/shadow.save | awk –F":" '{print
    ":"$1":":"$3":"$4":"$5":"$6":"$7":"$8":"$9}'>> $logfile
```

Checks for changes in hosts.allow and hosts.deny files:

```
diff /etc/hosts.allow /usr/local/security/hosts.allow.save >> $logfile

diff /etc/hosts.deny /usr/local/security/hosts.deny.save >> $logfile
```

Checks for changes in inetd.conf file:

```
diff /etc/inetd.conf /usr/local/security/inetd.conf.save >> $logfile
```

Checks last users log:

```
last > /var/log/last.current; diff /var/log/last.old /var/log/last.current >>
    $logfile
```

Checks file systems for core files, ownerless files, "group-less" files, .rhosts files, and hosts.equiv files (Example for /usr file system):

```
find /usr -xdev -mount \( -nouser -o -nogroup -o –name …  -o -name core -
o -name .rhosts -o -name hosts.equiv \) -exec /bin/ls -ladF {}\; >> $logfile
```

List set UID and set GID files:

```
find /usr -xdev -type f -a \( -perm 4000 -o -perm 2000 \)
    -exec /bin/ls -ladF {}\; >> $logfile
```

Finds and reports files and directories with world write permission:

```
find /usr -xdev -mount \( -type f -o -type d \) -perm 002
        exec /bin/ls -ladF {}\; >> $logfile
```

Performs AIDE intrusion detection:

```
/usr/security/aide -check -config=/usr/security/aide.conf >> $logfile
```

Performs chkrootkit check for installed rootkits:

```
/usr/security/chkrootkit >> $logfile
```

7.0 Conclusion

We are establishing new network and computer security policies, standards, and procedures that will affect which computers are allowed access to the local network. We need for many of our student Unix and Linux system administrators to maintain their computers so as to comply with our new policies and standards. This paper contains some of my personal network security philosophies, eight suggestions to the system administrators for installation of software to increase system security, and explanatory excerpts from shell scripts designed to be run nightly to assess the security of a computer system. Installing the system security software and monitoring the output from the nightly log digests should result in a baseline level of system security consistent with our newly emerging security policies.

I intend to distribute this paper, pointers to precompiled system software binaries, and sample shell scripts for student system administrators to use to perform nightly system security monitoring and log digests.

I hope that this short explanation of some of our security philosophies, and examples of the tools we use will prove helpful for your work as well.

8.0 References

8.1 Campbell, A., Beck, B., Friedl, M., Provos, N., de Raadt, T., and Song, D., “OpenSSH”, September 16, 2003. http://www.openssh.com The latest version of OpenSSH is 3.7.1. Binary versions are available with many operating systems and from many web sites such as http://www.sunfreeware.com and http://freeware.sgi.com/

8.2 Lehti, R., Virolainen, P., “Aide – Advanced Intrusion Detection Environment,” July 31, 2003. http://www.cs.tut.fi/~rammer/aide.html

The latest version of AIDE is available at
http://www.sourceforge.net/projects/aide/aide-0.9.tar.gz

8.3 Murlilo, N., Steding-Jessen, K., “Chkrootkit,” November 12, 2003. Homepage at http://www.chkrootkit.org. Source tarball is available at ftp://ftp.pangeia.com/pub/seg/pac/chkrootkit.tar.gz

8.4 Raynal, F., “Xinetd,” http://www.linuxfocus.org/English/November2000/article175.html

8.5 Rinker, E., “Configuring Sendmail On The Solaris 9 Platform,” http://www.sun.com/bigadmin/features/articles/config_sendmail.html

8.6 Tridgell, A., “Samba,” December 15, 2003. Version 3.0.1 is available from many mirror sites with links found in http://www.samba.org/.

8.7 Tsirigotis, P., Braun, R., “Xinetd,” August 5, 2003, http://www.xinetd.org/xinetd-2.3.12.tar.gz

8.8 Venema, W., “Portmap,” May 31, 1996, http://ftp.porcupine.org/pub/security/portmap_4.tar.gz

8.9 Venema, W., “Rpcbind,” April 11, 1998, http://ftp.porcupine.org/pub/security/rpcbind_2.1.tar.gz

8.10 Venema, W., “TCP-wrapper”, available at http://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.tar.gz