Global Information Assurance Certification Paper

Table of Contents 1
Matthew_Rice_GSEC.pdf 2

Improving the Network Security of a Computer Science Teaching Laboratory.

GIAC Security Essentials
Certification (GSEC)
Practical Assignment
Version 1.4c

Option 2 - Case Study in
Information Security

Submitted by: Matthew Rice
Location: Minneapolis, MN June 25-30, 2004
Submitted: Feb 3 2005

Abstract

I describe the steps taken to increase the level of network and physical security of a computer science teaching laboratory at a small liberal arts college. The initial state of the laboratory is described and potential weaknesses in the security of the lab are identified. Defense-in-depth (SANS vol. 1.2 64) is used to address these weaknesses and a solution is devised and implemented to reduce the chance of any possible incursion and to detect one if it occurs. Physical security and user education are briefly addressed, and the primary topic of this paper, the configuration of the operating system/software on the lab computers is fully covered. I conclude with a summary of what I believe are the next steps to be implemented in the never ending process of improving system security.

Table of Contents

Abstract..... page 1

1. Introduction..... page 3

2. The original Setup and its Security implications..... page 3

3. A plan for Improvement..... page 4

4. Physical Security..... page 6

5. OS upgrade to Fedora Core 3..... page 6

5a The Kickstart Script..... page 7

5b The Post Install Script..... page 8

6. User Education..... page 12

7. The current state of the Lab..... page 13

Appendix A: The kickstart.cfg file..... page 15

Appendix B: The full post install file: ks-post.sh..... page 16

Appendix C: The reinstall script /root/bin/reinstall.sh..... page 20

Appendix D: The file /etc/cron.daily/suid_chk.cron..... page 21

Appendix E: The edited portion of /etc/aide.conf..... page 22

Appendix F: The file /etc/sysconfig/iptables..... page 24

References:..... page 27

List of Figures

Figure 1. Layout of the CS Laboratory..... page 5

1. Introduction

The subject of this paper is the computer science teaching lab at a small liberal arts college. It consists of three rooms each with 10 to 25 Dell personal computers running Linux for a total of 45 workstations. The architecture of each computer in each room is consistent, with slight variation from room to room. (The computers are all Dell Optiplex GX240's, GX270's or Precision 340's.) The computers are connected in groups of 6 to 8 through Friendlynet ethernet hubs to a Hewlett Packard 2524 switch which connects to the college's network. Each workstation shares files (NFS) and receives authentication (NIS) from a Dell Poweredge 1650 server. The entire lab is on a /24 subnet of a class b network.

2. The Original Setup and its Security Implications.

The initial state of the lab was Red Hat 9 running, unhardened, unmonitored and unpatched on all the machines (workstations and the server). This had to change, as Red Hat 9 had reached it's end of life on the 30th of April 2004. No security or bugfix updates would be supplied by Red Hat after that date, and keeping it patched by hand would be a huge task. In addition, the workstations were not being monitored and what security information that was being gathered was sitting in unopened mail to the root user, or in unread logs, on each machine.

The operating system on the NIS/NFS server had been hardened against an attack in that it was running a firewall, tcp wrappers and had it's logs monitored. However it was still vulnerable to any exploit in its services that was not patched. This server was not physically behind the same switch as the workstations, hence the information exchanged between the workstations and the server (e.g. passwords during a NIS exchange) were in the clear on the college's network.

I identified what I believed to be the major security concerns of this set up. If an attacker could gain access to a workstation they could use processing power for illicit activities or to sniff network traffic. I believe however, that the two major vulnerabilities of the lab were the risk of the workstations being attacked by a student (or an perhaps outsider) just for "fun" with no malicious intent, or someone wishing to use the available storage space to store and distribute illicit material. Ease of infiltration is increased by the state of the operating system (unhardened, and unpatched) and once compromised, the presence of an infiltrator could pass unnoticed for sometime in an unmonitored workstation. I was also very worried about how easy physical access to the workstations was.

Being in a college environment, the attitude to security is more relaxed relative to what it should be. Physical access to the workstations is expected 24 hours a day, and this is expected to be provided to a wide range of people (the small college environment actually means a wider range of people than you might

expect, as students from affiliated colleges often take classes here.) The network itself is also expected to be open with visiting students, faculty and others expecting connectivity. The balance between this openness and maintaining security presents huge problems for IT professionals, and so any measure to increase security, while allowing the users the access that they need or indeed expect, is a step in the right direction.

To summarize the main security issues as I saw them:
1) An attacker may gain physical access to the machines. Having access to the machine may enable full access to it's resources and this needs to be controlled.
2) An attacker may gain access to a machine over the network and fully or partially take control of the machine., and possibly all the machines in the lab.
3) If an attack were to take place, it would go undetected due to the lack of attention paid to the individual machines.

3. A plan for Improvement

Through conversations with the Computer Science faculty and other system administrators at the College, it was decided that the operating system on the workstations would be switched to Fedora Core 3 and the NIS/NFS server would run Red Hat Enterprise Linux ES. The key factor in this decision was the availability of the 2.6 kernel in Fedora Core 3. The faculty are open to switching to a more mature OS in the future. For now thought, their desire for a “cutting edge” OS has an important security upside, and that is that the install of SELinux is straight forward for the 2.6 linux kernel (Security; Wade).

The install of the OS on each workstation was to be implemented through the use of Red Hat's kickstart utility. Though this is certainly a way to reduce the workload for the system administrator, it has security implications in that the state of the OS on each workstation can be easily changed back to it's original configuration and the workstations can be kept in “sync” with each other, thus enabling easier identification of anomalous behavior. If an incursion were to occur, it will be quickly spotted, followed up, and the workstation can be rebuilt and returned to service quickly. Also, less administrative work for the system administrator means more time to devote to security (e.g. reading logs, etc.). This required the purchase of a dedicated machine to act as the kickstart server. This new server would also act as a local repository for updates.

A third server would be needed as a log server to act as a common collection point for all the security and other information from the 45 workstations.
The hardening of the three servers (NIS/NFS, kickstart and log) is vital. The NIS/NFS stores and exchanges password information, and contains all users home directories and also it's /usr/local slice is shared with the workstations and contains in house software under development. The kickstart server contains the

base OS and update files whose integrity needs to be assured, and the log server is where we will look to identify possible security violations. Hardening of these servers is not addressed in this paper as the setup I have employed is quite standard and can be found in any number of references (Bastille; Mourani; Red Hat Benchmark). One important point that I will draw your attention to is that the NIS/NFS server was physically moved to be behind the switch so that the NIS and NFS data would never get onto the college's network proper. This is important as this data is in the clear, and the fewer hops between the server and client the better.

The final setup of the lab is shown in figure 1.

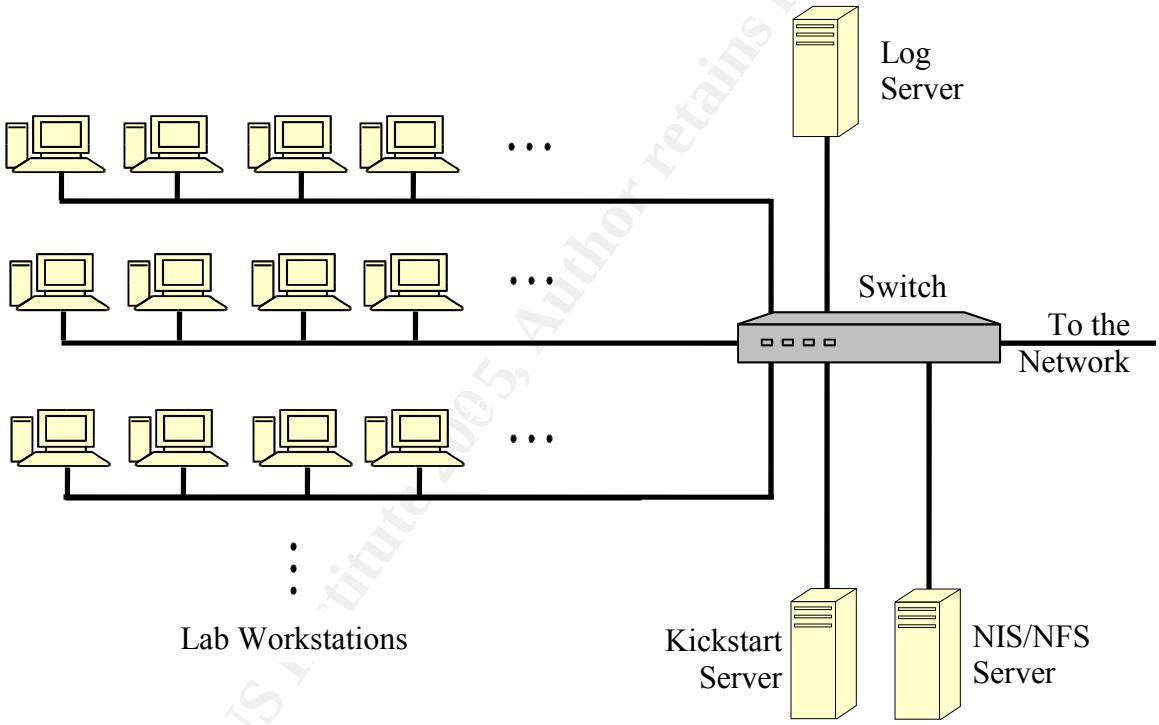

Figure 1
Layout of the CS Laboratory

Restricting access, and limiting exposure when unauthorized access is gained both physically and over the network, and detecting such incidents is discussed in the remainder of this paper.

4. Physical Security

The computer cases are a very possible target for attack (access or theft) as they are situated under the tables in the lab. It is wished that the students have access to the machines to use removable media, so they can not be locked away. The computers are therefore equipped with lockdown cables that attach the computers together in groups and to the furniture in the lab. These lockdown cables also lock the case and prevent it being opened.

The room has been equipped with motion sensors, though these are of limited utility, as students require 24 hr access, but are used during holidays and extended breaks. To allow many students access, but not have the lab open 24 hours a day, the lab doors have been equipped with combination locks, whose codes can be changed frequently. It cannot be guaranteed that someone with malicious intent is denied access to the lab. Indeed, one of the students authorized to be in the lab may be the intruder. I have configured the BIOS on the workstations to boot from the hard drive first and enabled password only access to BIOS settings. The lockdown cables also provide a measure of defense against case access. This means that, if the workstations are to boot, they will boot into the OS on the hard drive (dictated by the bootloader) and not to an OS carried on removable media, which would then give unfettered access to the files on the hard drive. Later we will see how I have locked access to the bootloader, and to the privileged user on the systems.

5. OS upgrade to Fedora Core 3

Misconfiguration is a common cause of vulnerabilities so the use of the Red Hat kickstart utility (Red Hat Customization chapter 7) to automate the setup of all the machines can help mitigate this as one can spend more time setting up one system, and mirroring all the other systems to match. The architectures of the workstations are so similar that it only took a small amount of scripting to enable all of the workstations to kickstart from the same configuration file.

The server that acts as the kickstart server is also set up as a mirror for Fedora Core 3 updates so that the lab machines can update from a local source. This box is a Red Hat Enterprise ES server running on a Dell Precision 340 hardened against intrusion (Bastille; Mourani; Red Hat Benchmark).

The final addition to the lab setup was to recycle a Sun Ultra 60 and install Debian to act as a log-server, a collection point and analyzer for all the logs from the workstations (and the kickstart/Fedora mirror and NIS/NFS servers). In the event of a security incident, the validity of the logs is vital, and so this server also needed to be secure. I followed the procedure outlined in Hillier to set up and secure the machine but using syslog-ng (Balázs) as a replacement for the

standard system logger on the server.

As I describe the structure of the kickstart file I use I will detail each step, with particular attention paid to the sections of the configuration that relate to security enhancements. A complete copy of the kickstart script and it's accompanying post install script can be found in the appendices A and B.

In what follows, and in the appendices, I have replaced site specific information with the variables:

```
$SYS_ADMIN
$SYS_ADMIN_EMAIL
$LOG_SERVER_IP
$KICKSTART_IP
$NIS_DOMAIN
$NIS_NFS_IP
$CS_GATEWAY
$CS_BROADCAST
$CS_NETWORK
```

These should be replaced with appropriate values.

5a The Kickstart Script

The file kickstart.cfg is straight forward, based on the kickstart file automatically generated by the anaconda installer for Red Hat distributions. An important point with security implications is that this file contains the superuser (root) and bootloader (grub) passwords for the workstation installs.

```
rootpw --iscrypted $1$ccKok0$6arVKk5eSxhhQ0KK6jNk3.
.
.
.
bootloader --location=mbr --append="rhgb quiet" \
      --md5pass=$1$GHKok0$hEbKsGQTMOoeDqkI17sAL/
```

Admittedly these are not in clear text, but as md5 hashes, but this kickstart file must be stored as world readable, and this information is transmitted over the network at install time. I will address these issues later in this paper, and show how I remove all privileges granted by these passwords, thus rendering them useless to anyone who happened to read or sniff, and decode them.

A firewall iptables is set up as well,

```
firewall --enabled --port=22:tcp
```

but I will actually install a customized version later in the post install script. The

philosophy behind the firewall that I am using is a compromise between filtering and logging. I log possible null scans, but then drop malformed packets to avoid, for example a syn-flood from overwhelming the workstation and logserver (see the discussion of Mitnick vs. Shimomura in SANS vol. 1.3 13). Any incoming packets for non-routable IP addresses are dropped, as are broadcasts on our network. I then drop traffic on common ports so that the logs are not cluttered with attempts to contact services we do not run. When an exploit in one of these services is discovered one can often get overwhelming traffic on these ports for days. Next the ports for the services that we are going to need (e.g. NFS, ...) are opened, and if any other attempts to access the machine over the network get to this point, they are logged and then dropped.

The secure linux kernel wrapper (Security; Wade) is very convenient to install for the 2.6 kernel (which Fedora Core 3 runs), it is chosen as an install option:

```
selinux --enforcing
```

An additional security consideration in the kickstart script is the number of packages installed.

```
%packages
@ office
@ engineering-and-scientific
@ editors
@ xemacs
.
.
.
```

This is dictated by the faculty who want a wide variety of software available for the students, but the more software, the more possible security holes are opened. I mitigate this as much as possible by keeping the system on a tight update schedule (see later in the post install script).

5b The Post Install Script

The post install script is run in the %post section of the kickstart routine, immediately after the OS and packages have been installed. I use it here to set up the lab wide and individual workstation configurations.

The following should let us see what is going on tty3 (using CTRL-ALT-3) and log the results of the post install to a file. I use these during debugging, but disable them in production, believing that the less information around detailing the installation the better.

```
# set -x
# exec 1>/root/ks-post.log 2>&1
```

In the post install script, ks-post.sh, the section between the stars is used to uniquely identify each workstation and to set some variables (e.g. IP number, host name, monitor type, default printer ...) was written by a co-worker of mine. I have included it here for completeness only and do not claim that it is my work. The script would stand without it, just that each machine would have a copy of the script customized for it, stored on the kickstart server with the naming convention:

<IP-addresse>-kickstart.cfg

and

<IP-addresse>-ks-post.sh

The production of each of these scripts could be automated from a template, thus achieving the desired control over the installation.

Many configuration files are set up in advance in a skeleton directory on the server, and these are now copied over to the workstation

```
# copy config files
cp -vpR ${KSROOT}/skel/* /
chmod 440 /etc/sudoers
chmod 600 /etc/aide.conf
```

Included here are the files needed to boot to the install shell and a script to rewrite /boot/grub/grub.conf and perform this reboot. That is, a script that quickly returns the workstation to its initial install state (This can be found in appendix C). Files with security implications that are loaded from this skeleton directory are:

/etc/sudoers
/root/.ssh/authorized_keys
/etc/rc.d/rc.local
/etc/ntp/ntpservers
/etc/cron.daily/suid_chk.cron
/etc/aide.conf
/etc/cron.daily/aide.cron
and
/etc/sysconfig/iptables

Note that the permissions on the sudoers and aide.conf files need to be set so that they can be copied over, but then set to only be accessible to the root user. I will address the implications of each of these files below as I walk through this post install script.

My first step is to lock out root access. This renders the (encrypted) root password, sent over the network and stored, world readable on the server, useless.

```
# Lock out root access
usermod -L root
```

To gain root access on the machines I now have two options. I have created an account and added it to the /etc/sudoers file:

```
$SYS_ADMIN ALL=(ALL) ALL
```

This effectively makes this account the root account, but sudo access and activity is logged, and I can change this password easily after I have used it. Further, I have added a public ssh key to the file /root/.ssh/authorized_keys and hence I can ssh into any box as root if needed using the private key on my workstation, or better, on some removable media I can carry with me. This ssh solution ensures that no passwords are transmitted, and is what I use if I need to access an individual box. Following the principle defense-in-depth, which advocates "multiple layers of protection" (SANS vol. 1.1 58), I then lock out the bootloader password. Again, this has been sent over the network, and is world readable on the server (though as an md5 hash) and so I render it useless:

```
# Edit grub.conf to lockout the password
sed -i -e 's/--md5\ /--md5\ !/' /boot/grub/grub.conf
```

Recall that the machines are physically accessible, and hence rebootable, so this limits the boot options to a choice of kernels, but not allowing single user mode (which would grant root access). I have set a BIOS password, so that the boot order is fixed (hard drive first) so that the bootloader cannot be bypassed. The last line of defense against physical access is the lockdown, locking the case, which stops someone resetting the BIOS, or perhaps adding a second (master) hard drive.

The next set of commands are vital to keep track of what happens on the 45 workstations. Firstly we have all root mail sent to a common account that I check frequently, and secondly we ensure that all log messages get sent from the workstations to a central log server, where they are analyzed using syslog-ng (Balázs) and a summary mail is sent to the same account as the root mail.

```
# Send root mail to admin
echo "root:     $SYS_ADMIN_EMAIL" >> /etc/aliases

# Send all log messages to the log server
echo "# Log all messages to log server" >> /etc/syslog.conf
echo "*.*             @$LOG_SERVER_IP" >> /etc/syslog.conf

# We analyze the logs on the log server, so do not need it
# done locally
rm /etc/cron.daily/00-logwatch
```

Using the principle of least privilege (SANS vol. 1.2 35) unneeded RPMS that are installed by default with the selected packages are removed and unneeded services are turned off. If there is an exploit discovered in one of these, it would be a great pity to be compromised when we don't need to be running them. It is important to edit the /etc/init.d/$SERVICE file as sometimes updates can turn

services back to their default states (on in run level 5).

```
# Remove unneeded rpms
rpm -e pcmcia-cs isdn4k-utils

# Turn off unneeded services
for SERVICE in rpcsvcgssd rhnsd mdmonitor; do
    chkconfig --del $SERVICE
# and keep them off
    sed -i -e 's/2345/-/' /etc/init.d/$SERVICE
    sed -i -e 's/345/-/' /etc/init.d/$SERVICE
done
```

We then enable the nightly updates of the workstations to ensure that they have the very latest patches available. (Our local mirror has been added as the default in /etc/yum.reops.d/.)

```
# Turn on nightly yum update
chkconfig --levels 345 yum on
```

I have also added the line

```
yum -y update
```

to the file /etc/rc.d/rc.local, to ensure that patches get applied at boot time. If a machine is kept off over night, it will be updated at boot time so that no one is able to run the machine unpatched. It may seem rash to have such an aggressive update schedule, but I have found that abuses of exploits happen very fast, and every patch I have applied has been solid. Furthermore, I am prepared to suffer a bad patch (that can be quickly corrected with my kickstart set up) than the chance of being compromised, especially as these workstations are not critical servers.

The GPG keys for Fedora packages are imported to ensure that the integrity of future packages loaded to the machines can be verified. This is done when the packages are loaded onto the mirror, but layered defense dictates that they should be checked again.

```
rpm --import /usr/share/doc/fedora-release-3/RPM-GPG-KEY*
```

It is important to keep the clock on the workstations correct, so that any time stamps made during a break in will be correct and may provide important timing information.

```
# Turn on and ntpd
chkconfig --levels 345 ntpd on
```

An xtras directory has been set up on the kickstart server that contains RPM's and install scripts for other software to be installed on the workstations. Those with security implications are a utility called pwgen (Password) that generates

random, easy to remember passwords, and an anti intrusion detection environment (Lehti) (see more on this below).

```
# Install additional packages
.
.
.
# Install (pre-built) pwgen
cp -vpr ${KSROOT}/xtras/pwgen/pwgen /usr/bin/
cp -vpr ${KSROOT}/xtras/pwgen/pwgen.1 /usr/share/man/man1/

# Install aide
rpm -ivh ${KSROOT}/xtras/aide/*.rpm
```

As mentioned in CERT/CC Intruder Detection Checklist (Intruder), a common technique for intruders is to "leave setuid copies of /bin/sh or /bin/time around to allow them root access at a late (sic) time." I have set up a file that contains the md5 checksum of a list of the setuid files on the system. There is a cron job, /etc/cron.daily/suid_chk.cron (See appendix D), that checks the current list and mails me if there is a difference.

```
find / -user root -perm -4000 -print -xdev | md5sum > \
      /root/suid_chksum
```

I run the intrusion detection system Aide (Advanced Intrusion Detection Environment) (Lehti) on each workstation. This monitors chosen properties (size, md5 check sum, etc.) of selected files and logs changes that it detects.

Now that we are set the last step in the install is to initialize the aide database. Note that the aide.conf file came over in the skeleton directory, and I have included the relevant part of it in appendix E.

```
rpm -ivh ${KSROOT}/xtras/aide/*.rpm
aide --init
mv -f /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

This data base is checked each night for changes, which are emailed to $SYS_ADMIN_EMAIL and recorded in $LOG_SERVER_IP:/var/log/aide.log. It is also checked at boot from /etc/rc.d/rc.local.

6. User Education

One must never forget that a very common point of weakness in the security of any system lies with the people using it. I have started a campaign to educate people in ways that they can help make the lab more secure.

The first step was to teach people how to tunnel applications that may transmit sensitive material (passwords ...) through ssh. One application that I had many

requests for was to open port 5901 for virtual network connection, vnc traffic (VNC). This can be tunneled through an encrypted ssh tunnel (port 22) (Making) as can many other applications (e.g. reading mail (Hatch)).

Having people choose strong passwords, and to change them frequently is important but difficult to enforce. I have included a simple program, pwgen that users can use to generate passwords that are strong, but easy to remember.

7. The current state of the Lab

As much as I think I have got the final configuration of the workstations complete, there is the occasional configuration file to adjust, or package to install. For example professors may request some new software installed or the aide database may be giving false positives because of a software upgrade, etc. Some of these can be achieved through the mount of /usr/local or /home/setup/profile. Often, however a command or script needs to be run as root on each machine. As I have an ssh key installed, I can perform these actions remotely without needing to log in as root. As the commands usually need to run on all the workstations, the ssh command can be scripted and run as a batch job out to all the workstations.

I have made progress in addressing the three concerns of a local physical intruder, an intruder gaining access over the network, and detection of an intrusion if it should occur.

Setting and securing root, grub and BIOS passwords, setting the boot order and locking down the computer cases has put many hurdles in the way of someone who would try to access the machines physically to use then inappropriately. Setting up a comprehensive firewall, turning off unused services, and keeping an aggressive update schedule has reduced the chance of someone gaining access to the workstations over the net. If an incursion does occur, I now have good access to uncompromised logs for early detection and forensics. I also have two other early detection tools in place (aide and an suid checker).

Network security is a continuous process. The more I think about the vulnerabilities of the lab, and the more reading I do on network security the more improvements I can see making to the computer science lab. Some immediate improvements that I wish to make would include replacing or securing the NIS authentication (Red Hat Security chapter 5.3), improve the functionality of aide, or switch to another program like tripwire or snort. It has been suggested to me that we may also be able to use network monitoring such as nagios to act as a theft detection system. If a workstation were to actually be stolen, then nagios would detect this (as a failure to connect) and signal that the machine was off the network (and perhaps missing).

The college is beginning a switch to a swipe card access system and I have requested that swipe access get added to the the computer science lab. This will act as a much better access control to the lab than a lock or the current combination lock.

Appendix A: The kickstart.cfg file.

```
# Kickstart file automatically generated by anaconda.
# modified for comp-sci labs 05_01_12 -- mjr

network --bootproto=dhcp
install
text
reboot
nfs --server $KICKSTART_IP --dir /software/fedora/3/i386/os
lang en_US.UTF-8
langsupport --default en_US.UTF-8 en_US.UTF-8
keyboard us
mouse genericwheelps/2 --device psaux --emulthree
xconfig --depth 24 --startxonboot  --defaultdesktop gnome
rootpw --iscrypted $1$ccKok0$6arVKk5eSxhhQ0KK6jNk3.
firewall --enabled --port=22:tcp
selinux --enforcing
authconfig --enableshadow --enablemd5 --enablenis \
      --nisdomain=$NIS_DOMAIN --nisserver=$NIS_NFS_IP
timezone --utc America/New_York
bootloader --location=mbr --append="rhgb quiet" \
      --md5pass=$1$GHKok0$hEbKsGQTMOoeDqkI17sAL/

# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
clearpart --all
part /boot --fstype "ext3" --size=256
part swap --size 512
part / --size 2048 --asprimary --fstype ext3 --grow

%packages
@ office
@ engineering-and-scientific
@ editors
@ xemacs
.
.
.

%post

# Mount the kickstart server for post install configuration
# (note that NFS requires the portmapper to be running)
/etc/init.d/portmap start

mkdir /mnt/fedora
mount -t nfs -o ro $KICKSTART_IP:/software/fedora mnt/fedora
#now run the post install script
/mnt/fedora/comp_sci/bin/ks-post.sh

umount /mnt/fedora
/etc/init.d/portmap stop

# done!
```

Appendix B: The full post install file: ks-post.sh

```
#!/bin/sh
#
# ks-post.sh: kickstart post-installation script
#

PATH=$PATH:/usr/sbin:/sbin

# This should let us see what's going on with CTRL-ALT-F3
# set -x
# and log the printout of this script
# exec 1>/root/ks-post.log 2>&1

# the root of all kickstart related stuff
KSROOT=/mnt/fedora/comp_sci

*********************************************************
# the config file syntax is
# host1 host2 ... : VAR=foo VAR2=bar ...
GROUPCONF=${KSROOT}/config/groups
# the config file syntax is
# macaddr: VAR=foo VAR2=bar
# The variables can be room number (for default printer
# selection) and monitor type (for configuring xorg.conf)
NETCONF=${KSROOT}/config/net

# automagical network configuration, right here.
# step 1: get our mac address
MACADDR=`/sbin/ifconfig | grep -i hwaddr | awk '{print $5}'`
# step two, find out if that tells us who we are via our
config/net file
if grep -q "^$MACADDR" "$NETCONF"; then
        eval `grep "^$MACADDR" $NETCONF | cut -d';' -f2-`
      # set networking config for next boot
      SHORTHOST=`echo $HOSTNAME | sed -e 's/\..*//'`
      cat << EOF > /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=$HOSTNAME
GATEWAY=$CS_GATEWAY
NISDOMAIN=compsci
EOF
      cat << EOF > /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=$CS_BROADCAST
HWADDR=$MACADDR
IPADDR=$HOSTIP
NETMASK=255.255.255.0
NETWORK=$CS_NETWORK
ONBOOT=yes
TYPE=Ethernet
EOF
fi

# source any config variables for groups of machines
if grep -qw "^[^:#]*$SHORTHOST" "$GROUPCONF"; then
      eval `grep -w "^[^:#]*$SHORTHOST" $GROUPCONF | cut \
            -d: -f2-`
```

```
else
      eval `grep '^:' $GROUPCONF | cut -d: -f2-`
fi
************************************************************
# copy config files
cp -vpR ${KSROOT}/skel/* /
chmod 440 /etc/sudoers
chmod 600 /etc/aide.conf

# Lock out root access
usermod -L root

# Edit grub.conf to lockout the password
sed -i -e 's/--md5\ /--md5\ !/' /boot/grub/grub.conf

# Send root mail to admin
echo "root:      $SYS_ADMIN_EMAIL" >> /etc/aliases

# Send all log messages to the log server
echo "# Log all messages to log server" >> /etc/syslog.conf
echo "*.*              @$LOG_SERVER_IP" >> /etc/syslog.conf

# We analize the logs on the log server, so do not need it
# done locally
rm /etc/cron.daily/00-logwatch

# Remove unneeded rpms
rpm -e pcmcia-cs isdn4k-utils

# Turn off unneeded services
for SERVICE in rpcsvcgssd rhnsd mdmonitor; do
    chkconfig --del $SERVICE
# and keep them off
    sed -i -e 's/2345/-/' /etc/init.d/$SERVICE
    sed -i -e 's/345/-/' /etc/init.d/$SERVICE
done

# Turn on nightly yum update (note that yum.conf
# has been set to point to our local mirror
chkconfig --levels 345 yum on

# We need the public key loaded to do gpg sig checks on
# packages installed:
rpm --import /usr/share/doc/fedora-release-3/RPM-GPG-KEY*

# Turn on and ntpd
chkconfig --levels 345 ntpd on

# Add entries to /etc/fstab to mount /home and /usr/local
echo "$NIS_NFS_IP:/home        /home   nfs     defaults \
      0 0" >> /etc/fstab
echo "$NIS_NFS_IP:/usr/local   /usr/local      nfs      \
      defaults,ro        0 0" >> /etc/fstab

# Add line to /etc/profile to include local setup
# parameters from $NIS_NFS_IP
echo "source /home/setup/profile" >> /etc/profile

# Optimize resolution for the Monitor size (The monitors
```

```
# are the one thing that the OS does a poor job of
# detecting.  Sample xorg.conf files are moved in with
# the skel copy, and are now linked for each architecture.)
mv /etc/X11/xorg.conf /etc/X11/xorg.conf.kickstart
ln -s /etx/X11/xorg.conf.$MONITOR /etc/X11/xorg.conf

# Configure the various printers (note that the drivers are
# loaded into /etc/cups/ppd from the skel upload, named
rm1_pr, rm2_rp and rm3_pr.)
chkconfig --add cups
/etc/init.d/cups start
for pr in rm1_pr rm2_pr rm3_pr; do
    lpadmin -p "$pr" -v socket://$pr:9100 -P /
etc/cups/ppd/$pr.ppd  -E
done
# Set default printer
lpadmin -d "$PRINTER"
/etc/init.d/cups stop

# Install additional packages
# (RPMs, install scripts, direct copying ...  Some examples
# are shown.)

# java runtime and sdk
rpm -ivh ${KSROOT}/xtras/java/*.rpm

# link the java plugin for firefox
ln -s /
usr/java/jre1.5.0_01/plugin/i386/ns7/libjavaplugin_oji.so \
      /usr/lib/mozilla/plugins/libjavaplugin_oji.so

# Install flash plugin
# (The modification to the setup script was to remove
# attempts to write to /usr/local.
cp -vpr ${KSROOT}/xtras/flash-plugin /usr/lib
sh /usr/lib/flash-plugin/setup_mod

# Install Acrobat reader and it's plugin
cp -vpr ${KSROOT}/xtras/Acrobat5/ /usr
ln -s /usr/Acrobat5/bin/acroread /usr/bin/acroread
ln -s /usr/Acrobat5/Browsers/intellinux/nppdf.so \
      /usr/lib/mozilla/plugins/nppdf.so

# Install (pre-built) pwgen
cp -vpr ${KSROOT}/xtras/pwgen/pwgen /usr/bin/
cp -vpr ${KSROOT}/xtras/pwgen/pwgen.1 /usr/share/man/man1/

# Install aide
rpm -ivh ${KSROOT}/xtras/aide/*.rpm
.
.
.

# Two final security measures

# Create a file containing the md5 checksum of the list of
# the suid files on the system.  There is a cron job that
# checks the current list and emails me if there is a
# difference. (cron job loaded with the skel)
```

```
find / -user root -perm -4000 -print -xdev | md5sum > \
      /root/suid_chksum

# Now that we are set initialize the aide
# database (note that the aide.conf file came over in the
# skel copy, and a check is a nightly cron, and rc.local
# job.)
rpm -ivh ${KSROOT}/xtras/aide/*.rpm
aide --init
mv -f /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz

#done!
```

Appendix C: The reinstall script /root/bin/reinstall.sh

```
#! /bin/sh

# This script rewrites the grub.conf file and reboots the
# system so that it reinstalls.  If something goes wrong
# (before anything is written to the hard drive)it should
# reboot to the last good kernel.  (vmlinuz-install, and
# initrd-install.img are copied over with the skel
# directory.)

# usage /root/bin/reinstall.sh

OS=Fedora
GRUB_CONF=/boot/grub/grub.conf

ROOT=`grep -m1 -A1 $OS $GRUB_CONF | grep root | \
      awk '{print $2}'`
cat <<EOF > /tmp/new.grub
default=1
timeout=5
title Fedora Core 3 Reinstall
        root $ROOT
        kernel /vmlinuz-install  \
      ks=nfs:165.106.23.209:/software/fedora/comp_sci/\
      kickstart_cs.cfg
        initrd /initrd-install.img
EOF
grep -m1 -A4 $OS $GRUB_CONF >> /tmp/new.grub
mv $GRUB_CONF /boot/old.grub
mv /tmp/new.grub $GRUB_CONF
echo savedefault --default=0 --once |  grub
sleep 5
reboot
```

Appendix D: The file /etc/cron.daily/suid_chk.cron

```
# this script runs an md5 check sum of the list of suid
# root files on the system and mails a warning if there
# is a change (from setup)
if [ -e /root/suid_chksum ] && `find / -user root -perm \
-4000 -print -xdev | md5sum --status --check \
root/suid_chksum`; then
        exit 0;
fi
#If all is well, we should not get to here
mail -s "Set UID checksum error!  Possible intrusion" \
     $SYS_ADMIN_EMAIL
exit 1
```

Appendix E: The edited portion of /etc/aide.conf.

I have removed some directories (/boot, /lib, and /usr) to cut down on noise from package upgrades, but kept some common /usr/bin and /usr/sbin attack targets.

```
# These are the default rules.
#
#p:      permissions
#i:      inode:
#n:      number of links
#u:      user
#g:      group
#s:      size
#b:      block count
#m:      mtime
#a:      atime
#c:      ctime
#S:      check for growing size
#md5:    md5 checksum
#sha1:   sha1 checksum
#rmd160: rmd160 checksum
#tiger:  tiger checksum
#haval:  haval checksum
#gost:   gost checksum
#crc32:  crc32 checksum
#R:      p+i+n+u+g+s+m+c+md5
#L:      p+i+n+u+g
#E:      Empty group
#>:      Growing logfile p+u+g+i+n+S

# You can create custom rules like this.

NORMAL = R+b+md5

DIR = p+i+n+u+g
#/boot   NORMAL
/bin    NORMAL
/sbin   NORMAL
#/lib    NORMAL
/opt    NORMAL
#/usr    NORMAL
/root   NORMAL

# Check only permissions, inode, user and group for /etc,
but
# cover some important files closely.
/etc    p+i+u+g
!/etc/mtab
/etc/exports  NORMAL
/etc/fstab    NORMAL
/etc/passwd   NORMAL
/etc/group    NORMAL
/etc/gshadow  NORMAL
/etc/shadow   NORMAL

# Added a few files -- mjr 05_01_26
/usr/sbin/inetd NORMAL
/usr/bin/lsof   NORMAL
```

```
/usr/bin/md5sum NORMAL
/usr/bin/w      NORMAL
/usr/bin/who    NORMAL
/usr/bin/md5sum NORMAL
```

Appendix F: The file /etc/sysconfig/iptables

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]

#####################################################
# INPUT POLICY
#####################################################
-P INPUT DROP

#####################################################
# accept from lo
#####################################################
-A INPUT -i lo -j ACCEPT

#####################################################
# DROP
#####################################################

#*****************************************************
# Kill malformed packets. Add logging here if you want

# Block NULL packets
-A INPUT -p tcp --tcp-flags ALL NONE -j LOG --log-prefix
"[iptables] NULL scan : "

# SYN/FIN
-A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP

# SYN/RST
-A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP

# FIN only bit set, with no accompanying ACK
-A INPUT -p tcp --tcp-flags ACK,FIN FIN -j DROP

# PSH only bit set, with no accompanying ACK
-A INPUT -p tcp --tcp-flags ACK,PSH PSH -j DROP

#*****************************************************
# Block common private/non-routable address blocks
# (possible spoofing)
-A INPUT -s 10.0.0.0/8 -j DROP
-A INPUT -s 172.16.0.0/12 -j DROP
-A INPUT -s 192.168.0.0/16 -j DROP
-A INPUT -s 0.0.0.0/8 -j DROP
-A INPUT -s 169.254.0.0/16 -j DROP
-A INPUT -s 192.0.2.0/24 -j DROP
-A INPUT -s 204.152.64.0/23 -j DROP
-A INPUT -i eth0 -s 127.0.0.0/8 -j DROP

# Drop multicast traffic
-A INPUT -s 224.0.0.0/4 -j DROP
-A INPUT -d 224.0.0.0/4 -j DROP

# Drop local broadcasts
-A INPUT -d xxx.xxx.xxx.255 -j DROP
-A INPUT -d xxx.xxx.255.255 -j DROP
```

```
-A INPUT -d 255.255.255.255 -j DROP

#*****************************************************
# Drop packets of the following services.
# There is much traffic on the net to these ports,
# and it can overwhelm the logs.

# port 80 = HTTP
-A INPUT -p tcp --dport 80 -j DROP

# ports 1433, 1434 = MSSQL
-A INPUT -p tcp --dport 1433 -j DROP
-A INPUT -p udp --dport 1433 -j DROP
-A INPUT -p tcp --dport 1434 -j DROP
-A INPUT -p udp --dport 1434 -j DROP

# ports 137, 138, 139 = NetBIOS
-A INPUT -p udp --dport 137 -j DROP
-A INPUT -p tcp --dport 137 -j DROP
-A INPUT -p udp --dport 138 -j DROP
-A INPUT -p tcp --dport 138 -j DROP
-A INPUT -p udp --dport 139 -j DROP
-A INPUT -p tcp --dport 139 -j DROP

# port 554 = RTSP
-A INPUT -p udp --dport 554 -j DROP
-A INPUT -p tcp --dport 554 -j DROP

# port 631 = ipp
-A INPUT -p udp --dport 631 -j DROP
-A INPUT -p tcp --dport 631 -j DROP

#####################################################
# ALLOW rules
#####################################################

# allow established/related connections
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow nfs from $KICKSTART_IP
-A INPUT -s $KICKSTART_IP -p udp --sport 2049 --dport   \
      600:1023 -j ACCEPT
-A INPUT -s $KICKSTART_IP -p tcp --sport 2049 --dport   \
      760:800 -j ACCEPT
-A INPUT -s $KICKSTART_IP -p udp --sport 111 --dport    \
      600:1023 -j ACCEPT
-A INPUT -s $KICKSTART_IP -p tcp --sport 111 --dport    \
      600:1023 -j ACCEPT

# Allow ICMP from other CS machines
-A INPUT -p icmp -s xxx.xxx.xxx.0/24 -j ACCEPT

# Allow new ssh connections
-A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT

# allow Network Time Protocol
-A INPUT -p udp -m udp --dport 123 -j ACCEPT

#####################################################
```

```
# OUTPUT policy
##################################################
-A OUTPUT -j ACCEPT

##################################################
# Default drop policy
##################################################
-A INPUT -j LOG --log-prefix "[iptables] Default drop : "
-A INPUT -j DROP
COMMIT
```

References:

Bastille Linux, 27 Jan. 2005. Bastille Linux. 28 Jan. 2005. <http://www.bastille-linux.org/>.

Hatch, Brian. SSH Port Forwarding. 6 Jan. 2005. SecurityFocus. 17 Jan 2005. <http://www.securityfocus.com/infocus/1816>.

Hillier, Vincent. Setting up a Debian Log Server. 7 Nov. 2002. 1 Dec. 2004. <http://www.pantz.org/os/linux/security/debianlogserver.shtml>.

Intruder Detection Checklist. 20 July 1999. CERT Coordination Center. Carnegie Mellon University. 17 Jan. 2005. <http://www.cert.org/tech_tips/intruder_detection_checklist.html>

Lehti, Rami. Aide. 17 Jan 2005. <http://www.cs.tut.fi/~rammer/aide.html>.

Making VNC more secure using SSH. 1999. AT&T Laboratories Cambridge. 20 Oct. 2004. <http://www.uk.research.att.com/archive/vnc/sshvnc.html>.

Mourani, Gerhard. Securing and Optimizing Linux. 2000. Open NA. 21 Jan. 2005. <http://www.faqs.org/docs/securing/>.

Password Generator. 2005. SourceForge. 17 Jan. 2005. <http://sourceforge.net/projects/pwgen/>.

Red Hat Enterprise Linux 3: Security Guide. 2004. Red Hat Inc. 1 Feb. 2005. <http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/security-guide/>.

Red Hat Enterprise Linux Benchmark v1.0. 17 Jan. 2005. The Center for Internet Security. 21 Jan. 2005. <http://www.cisecurity.org/tools2/linux/RH-LinuxBenchmark.pdf>.

Red Hat Linux 9: Red Hat Linux Customization Guide. 2003. Red Hat Inc. 17 Nov 2004. <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/custom-guide/>.

SANS Institute, Track 1 - Security Essentials Volume 1.1. SANS Press. June 25, 2004.

SANS Institute, Track 1 - Security Essentials Volume 1.2. SANS Press. June 25, 2004.

SANS Institute, Track 1 - Security Essentials Volume 1.3. SANS Press. June 25, 2004.

Scheidler, Balázs. Syslog-ng Reference Manual. 2000. Balabit. 1 Dec 2004. <http://www.balabit.com/products/syslog_ng/reference/book1.html>.

Security-Enhanced Linux. National Security Agency. 28 Jan. 2005. <http://www.nsa.gov/selinux/>.

VNC 4.0 Documentation. 2004. Real VNC. 20 Oct. 2004. <http://www.realvnc.com/documentation.html>.

Wade, Karsten. Fedora Core 3 SELinux FAQ. 21 Jan. 2005. Fedora Project. Red Hat Inc. 28 Jan. 2005. <http://fedora.redhat.com/projects/selinux/>.