Creating a Secure and Maintainable Build Environment for Fedora Linux

Rod Hauser
CISSP, RHCE, GCUX

SANS GCUX
Version 3.0
Option 2

20 September 2004

Table of Contents

0.0 Abstract

This practical will lead the reader through creation of a secure build environment that provides the desired security, scalability and maintainability to suit many organizations. The scope of this practical will focus on using Red Hat's Kickstart facility with Fedora Core 2 to create a secure build process with few third party products. Specific Kickstart customizations will be discussed in order to generalize the process and facilitate process ownership by adopters.

0.1 What this paper is *not*

This is not a comprehensive review of the mechanisms, protocols and processes of Red Hat's Kickstart facility, although many specific functions are used. This paper is not a comprehensive description of creating a Kickstart environment, although it does include instructions regarding creating the specific Kickstart server for this environment. Experienced Kickstart users will find clever and elegant enhancements to what is covered here.

0.2 Acknowledgements

I would like to thank my supportive family. General kudos is due to the Open Source Community at large, but specific thanks go to Linus for keeping computing interesting, and making it possible for professionals to tinker in useful ways. Thanks go to Red Hat and to SuSE for proving that enterprise support isn't really that hard to provide, and to businesses everywhere for recognizing the value of Open Source solutions.

1.0 Rationale

The first step in maintaining a secure system is making certain that it is secure from the initial build forward. The integrity of countless systems in production, from enterprises to small businesses, consists of a documented build process and a documented change control process. The variety of change control processes are beyond the scope of this document, but the processes identified here will directly address creating a secure build using Kickstart, and potentially close gaps between the build process and various change control systems.

While this rationale holds for every operating system, this detailed analysis and step-by-step procedures will only be applied to a single operating system. At the time of this writing, Fedora Core 2 (FC2) is the latest freely available version from Red Hat. The specific build process detailed here is accurate and will translate across other versions of Red Hat Linux with only minor changes over time. Why Red Hat Linux? Red Hat Linux is in use in many types of business. Enterprise IT departments, serious home users, application vendors and K-12 and university staff could all benefit from many of the principles and practices detailed here. Also, Red Hat's automated installation mechanism, Kickstart, is a mature and robust tool. Other Linux distributions have automated network installation mechanisms, but many rapidly resort to customized scripting to perform some or all of the function, or are methods that are not supported by the vendor. A distribution-agnostic version of this analysis would be too general and lack the specific technical details that are useful to many. The issue of familiarity is important to a comprehensive discussion of the topic. The author has used Kickstart since Red Hat release 6.1, and has used Red Hat Linux since 3.3, and Linux since approximately kernel 0.99. The author has performed automated installations with other tools, and Kickstart achieves what those tools achieve, and allows integration of post-build hardening into the build process in ways that many other tools do not.

Manual installation from original media can be argued as both the simplest and most secure method of creating an secure system, but manual installations retain a great deal of opportunity for human error, does not scale well, and may result in insecure “shortcuts” when installation requirements increase but staffing remains flat. Replicating an image can scale well, but issues of secure image handling, hardware standards, and image maintenance, quality assurance, and lifecycle must all be dealt with carefully. Automated network-based installations can provide a method for secure builds that provides scalability and can be quickly deployed and easily maintained.

Patches and updates are issues that must be addressed with any build method. Regardless of the frequency of updates, the process of including patches and updates in the build process should be linked to the process of updating live systems with the same patches and updates. If these two tasks are tightly linked, the delta between live systems and new systems will remain small, allowing creation and maintenance of an environment that is both consistent and secure.

2.0 Scope of work

The scope of this document will cover only installation of Fedora Core 2 on a limited selection of hardware. It should be noted that although this document is significantly limited regarding distribution, release, and hardware, the author has extensive experience using Kickstart over multiple releases, with diverse hardware. The process described herein has been performed on different releases and diverse hardware, and each organization should perform testing appropriate to the scope of their desired distribution, release and hardware.

In order to follow this process, you will need a minimum of two systems, an installation server and a target build system. A third system is recommended, so that a functional desktop is available as a workstation while the target build system is being built and rebuilt during configuration testing. There is also a requirement for a CD writer, although speed is not of the essence. The installation server should have a minimum of 4GB available disk space in order to configure it as an installation server for a single distribution. The disk requirement can easily grow to more than 10GB per distribution release, so budgeting for 18GB for each distribution is not unreasonable. It is important to recognize that this disk space is for different files, including downloaded iso images, updates, Kickstart files, and source code as required.

Although this deployment is on a home network, the security requirements remain high, because no self-respecting IT security professional would consider running a blatantly insecure network. Specific segments include a DMZ, a secure build network, and a production user network. Although this discussion will focus on the build process, it is useful to consider the overall architecture in order to clearly see how a securely built system would be deployed from the secure build network to the other networks.

The networks as discussed in this document are:

DMZ:	192.168.1.0/24
build.net:	10.1.2.0/24
prod.net:	172.16.0.0/12

The network infrastructure for all of these are 100Mbps switched. Standard private network addressing is used.[1]

Within these networks, the three required systems are as follows:

Hostname:	IP Address:	Role:
server	10.1.2.4	Installation server

[1] ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt

dmz	10.1.2.6	Target host
works	10.1.2.42	Workstation

A summary of hardware specifications for each system are as follows (from **dmesg** and **lspci**):

Installation server (server):
Dell PowerEdge 6300, four-CPU 400MHz PII, 1024MB RAM, Adaptec 29140 controller with four 18GB 10K SCSI disks, Intel e100 PCI NIC (172.16.2.4), 3Com 3c905C PCI NIC (10.1.2.4), integrated ATI 3D Rage Pro (rev 5c)

Target host, for dmz service (dmz):
Dell GX150, 1GHz PIII, 256MB RAM, 20GB IDE HD, integrated 3c905C NIC, integrated Intel Corp. 82815 CGC display adapter

Workstation (works):
Homebuilt Athlon 1400MHz, 512MB RAM, 60GB IDE HD, SMC epic100 PCI NIC, nVidia Corporation NV18GL [Quadro4 NVS AGP 8x] rev a2 with dual 21" monitors, HP CD-Writer+ 7200, ATAPI CD/DVD-ROM drive

Throughout the documentation, the primary user within the build environment will be 'tester' although this value may be replaced with one or more users for an organization. Note that some specific commands that occur purely in user space may be denoted with a tester@[hostname]:/full/path, tester@[hostname], or simply prefixed with a $.

Initially the installation server may function as a workstation, or the workstation may serve as an installation server in order to install the installation server. The specific details of installing and hardening the installation server will not be covered separately, in order to focus on the procedure of securing builds with Kickstart. The details of manually hardening then automating hardening procedures with Kickstart provide sufficient coverage for hardening the installation server. Details specific to this installation environment will be covered. The installation server does not even have to be a Linux server to provide the services needed for Kickstart.

The entire process of creating a Kickstart environment will be covered. The process of creating a customized boot CD to automate installations for your environment will be detailed, and alternative boot options will be discussed. Installation methods will be discussed, and NFS will be used initially, as a rapid method of creating the build environment with minimal disk space requirements. The target build host will be a server destined for the DMZ, as a bastion host for ssh access.

3.0 Creating the Kickstart installation server environment

The first step for achieving a secure build is to begin with trusted media. Frequently this entails purchasing a shrinkwrapped copy from the vendor, but that is not necessary with Fedora. On the installation server we have created a filesystem specifically for sharing files for Kickstart, called /share. A subdirectory will be dedicated specifically to the Fedora Core 2 release, /share/fc2. Within this directory (and previous and subsequent release directories) we will create a set of directories for specific functions.

Subirectory	Space requirements	Purpose
. (/share/fc2)	2GB for binary iso's 2GB for source iso's	top-level directory of installation share
isolinux	4MB	work directory for creating custom boot CD
ks	1MB	Kickstart configuration files
log	varies, allow 1GB	logging system information at installation time
setup	varies, allow 4GB	environment-specific configuration and

		application files
updates	varies, allow 8GB	updates mirror on the local LAN/WAN

Create the same set of subdirectories for each distribution release that you test or deploy within your environment. There are several benefits to maintaining the hierarchy based on release. Updates and Kickstart configurations are stored in the appropriate directory tree, making manual errors less likely. Removal of an entire distribution is trivial after all relevant systems are upgraded or decommissioned. Creation and testing of Kickstart configuration files is segregated by release, minimizing likelihood of errors over time.

3.1 Establishing trusted installation media

Download the ISO images and md5sum files from http://download.fedora.redhat.com/pub/fedora/linux/core/2/i386/iso/ or from a suitable mirror to /share/fc2. This is also a good time to download the source code CD's and rescue CD iso if desired.

Verify the integrity of the MD5SUM file with GnuPG, or manually verify that the contents of the MD5SUM file matches the hashes on the vendor website – if the integrity is not assured, check the website and mailing lists to determine if a security compromise has occurred. If so, stop now until that issue has been resolved. Check your md5 sums:

```
$ md5sum –c MD5SUM
FC2-i386-disc1.iso: OK
FC2-i386-disc2.iso: OK
FC2-i386-disc3.iso: OK
FC2-i386-disc4.iso: OK
```

The media is now trusted media. Make it difficult to easily change that state:

```
# chown root.root FC* MD*
# chattr +i FC* MD*
# chmod 444 FC* MD*
```

3.2 Sharing the installation files

The installation server will need to provide access to these files. For this configuration, the build network is a small trusted network. We are going to provide access to these files through NFS. To provide this we have added this line to /etc/exports:

```
/share 10.1.2.0/255.255.255.0(ro)
```

We also restart portmap, nfs, and nfslock services, and use chkconfig to set these services to start at each boot.

```
# cd /etc/init.d ; ./nfs stop ; ./nfslock stop ; ./portmap stop
# ./portmap start ; ./nfs start;./nfslock start
# chkconfig portmap on ; chkconfig nfs on ; chkconfig nfslock off
```

Although widely known to have security issues, NFS allows us to install directly from the iso images rather than making a copy of each of the images to another installation directory. Saving less than 3GB of disk space may be inconsequential for some organizations. Choose the installation method (HTTP, FTP, NFS) that is best suited to the environment requirements. HTTP is recommended for most environments. Verify the availability of the share.

```
works:/ # mkdir /share
works:/ # mount server:/share /share
ls /share
```

Edit /etc/fstab to append the following line, making /share NFS-mounted at each boot.

```
10.1.2.4:/share       /share      nfs     rw,soft,intr 0 0
```

It is possible to create an installation server that provides all three installation mechanisms, providing the installation tree via HTTP, FTP, and NFS, simultaneously in order to allow any of those three methods access to the installation tree. This is left as a trivial exercise for the reader.

3.3 Creating boot media for a manual installation

You will need to burn disc1 to a CD-R or CD-RW in order to perform a manual installation. Use **cdrecord**, **xcdroast**, **k3b**, or your choice of CD-burning utilities to create this disc.

```
# cdrecord -v -eject speed=1 dev=1,0,0 –data FC2-i386-disc1.iso &
```

Disc 1 can also typically be used as a rescue CD, and should be retained for that purpose. Inveterate hardware scrounges, and others motivated to use the oldest possible hardware to run Linux, should take care to verify that the CD-R or CD-RW media you select will work to boot the target hardware. After creating the CD, verify the media by booting it and at the CD's initial boot prompt type:

```
linux mediacheck
```

Again, if the media does not pass, scrap it and burn a fresh copy from a verified iso image. Once the bootable CD is verified, note the date that it was created and verified for your records. The physical disc is now trusted media.

4.0 Creating an initial build, manually

In order to automate a build process, it is usually required to perform an installation manually. Expect to follow these steps to perform a manual installation at least once each time you automate the build process for a new release. A manual installation may be required for each hardware platform that is getting an automated install. Over time, experience will allow you to quickly identify required changes to the Kickstart configuration files and rapidly create multiple Kickstart files based on your environment and hardware. With each new release, read the Release notes and perform a manual installation.

Before installing an operating system for a secure host, configure the BIOS settings for the appropriate level of security. Configure a BIOS password if desired, and enable the level of error logging that is supported by the hardware and the organization's support policies. Make certain that you select a boot sequence that is appropriate for the system. Typically this entails booting from the hard disk first, and from any CD-ROM, Floppy or other removable device only through manual intervention and entering a password. If the system in question is going to be headless, disable halts or prompts on keyboard and mouse errors. If the target host is going to be in a secured data center, and support policies do not require BIOS passwords, this may not be necessary. If the system is going to have a remotely accessible console connection, passwords for accessing BIOS information are prudent.

Boot disc 1 of the trusted, verified media on your target host, and at the boot prompt, type

```
linux askmethod
```

Step-by-step details of the installation are beyond the scope of this document, but the 'askmethod' parameter will allow you to select from the available installation methods, Local CDROM, Hard Disk, NFS, HTTP, and FTP. Since we have configured our installation server to support NFS installation from the iso images, choose NFS then install normally. For our first target host, a dmz host, we will select "Minimal installation". Not only is this a sound security principle, it will provide a quick installation for both the manual install as well as for the early testing of Kickstart configuration changes.

When entering the root password, select a temporary build password, or a build password that may be used in a secured build environment for a period of time. Although this password will not be retained in cleartext anywhere, the configuration file (Kickstart) that is created to automate this manual process will contain the hash of this password, and in order for the automated installs to work, this file must be world-readable, so choose a good build password that is not in use anywhere else in your environment.

4.1 Gathering information manually, for the automated installation

When the target installation is complete, reboot the host and log in. Change the root password immediately. Even though this host will be rebuilt very soon, adopt the habit now. The next step is to collect files and information that is necessary for automating the process that has just been performed manually. Most of the information that is necessary to automate an installation is automatically created, and stored in /root, root's home directory. A complete list of packages installed should be generated with:

```
  rpm -qa --queryformat '%{NAME}\n' |sort > /root/rpm-novers.`date +%Y-%m-%d`
```

These files will then be copied to the /share/fc2/log/ directory. With the exception of the file listing packages, each system installation will generate 3 files with identical names, so creating a subdirectory structure within the log directory is useful to organize these without renaming each of them.

```
  tester@server:/share/fc2/log $ mkdir dmz000
  root@dmz:/root $ scp * tester@server:/share/fc2/log/dmz000
```

Depending upon the DNS configuration, you may need to use the IP address for this:

```
  root@dmz:/root $ scp * tester@10.1.2.4:/share/fc2/log/dmz000
```

The primary file among these for automating installations is anaconda-ks.cfg. This is a sample Kickstart file generated anaconda, the system installer. Note that although this installation program is typically run as a GUI, it may be run purely in text mode, and both are called anaconda, and have similar functionality. You may not be able to run the GUI installer if your video card is not recognized by X, and some configuration options, such as LVM, may lag in the text installer.

4.2 Creating a Kickstart configuration file for automated installation

Make a working copy of this Kickstart file in your ks subdirectory, and make the following edits. Modify the initial comment line to reflect that the file is not automatically generated, but edited by the build team.

```
  cp /share/fc2/log/dmz000/anaconda-ks.cfg /share/fc2/ks/dmz001-ks.cfg
  vi dmz001-ks.cfg
  # Kickstart file automatically generated 2004-08-23, modified by Rod
```

Note that if you are creating a Kickstart file to upgrade existing systems, you can create a file in the same manner, but performing an upgrade on an existing system rather than an install. In that case your Kickstart configuration file will begin with `upgrade` rather than `install`. Because this discussion is focused on creating secure builds, the aspect of upgrading an existing system will not be dealt with in depth. From a security standpoint, it would be considered preferable in all cases to back up any data on an existing system, and re-install the system from scratch, rather than potentially retaining any executables from a legacy system. From a functional standpoint, this is a very useful feature of Kickstart, and may be appropriate in some cases where security must be balanced with other concerns. The line immediately following the install directive contains the installation method and parameters. You may edit this line to change your installation method without resorting to another manual installation.

If the target system will not be running any GUI tools, you may comment out the xconfig line, and insert

```
  skipx
```

Insert the line zerombr, and modify the clearpart line to clear all partitions.

```
zerombr
clearpart --all
```

Un-comment the following lines, so that the filesystems will be created at installation time.

```
part /boot --fstype ext3 --size=101 --asprimary
part / --fstype ext3 --size=2020
part swap --size=512
part pv.7 --size=100 --grow
volgroup vg00 pv.7
logvol /usr --fstype ext3 --name=lv00 --vgname=vg00 --size=5432
logvol /var --fstype ext3 --name=lv01 --vgname=vg00 --size=1236
logvol /tmp --fstype ext3 --name=lv02 --vgname=vg00 --size=1024
logvol /home --fstype ext3 --name=lv04 --vgname=vg00 --size=6544
logvol /opt --fstype ext3 --name=lv03 --vgname=vg00 --size=792
```

These lines may vary, depending upon the partition types and sizes selected. While it is possible to create this section from scratch or to migrate a configuration from one hardware type to another, you can easily produce unexpected results. Two systems with different hard disk configurations may also produce different results with the same stanza. Even two systems with identical hardware, but with slightly different hardware RAID configurations may produce different results. Consider specifying disks for each partition with parameters like `–ondisk=sda`, and if a minimum partition size is required, but more disk may be available, consider using the `--grow` parameter with one or two partitions. Test your changes to this file, and do not hesitate to perform a manual installation for different hardware.

Note the %packages and %post sections -- %post is empty in every automatically created anaconda-ks.cfg. We are going to add a single line to the %post, the one step that we did post-install in order to gather full package information.

```
rpm -qa --queryformat '%{NAME}\n'|sort > /var/log/rpm-novers.`date +%Y-%m-%d`
```

4.3 Manually testing the initial Kickstart file

The information gathered from the manually installed system has been copied to the installation server for our reference, in /share/fc2/log. Now it is time to test our new, only slightly modified Kickstart file. Note the full file path, and verify that the permissions are appropriate.

```
tester@server:/share/fc2/ks $ chmod 644 dmz001-ks.cfg
```

Boot disc 1 of the trusted, verified media on your target host, and at the boot prompt, type

```
linux ks:nfs:10.1.2.4:/share/fc2/ks/dmz001-ks.cfg
```

You will see the usual kernel messages as the initial ramdisk loads and hardware is detected. The first screens of the anaconda installer will again be the ncurses colorized text, but a little patience is required. Depending upon the speed of your network and the target host, you should see the GUI installation screen appear in several seconds. You may amuse yourself, or troubleshoot problems by toggling to the other screens of the anaconda installer. Press Alt-F3 and Alt-F4 in order to see stdout and sterr messages from anaconda. 4 out of 5 seasoned Unix administrators agree that file permissions are the leading cause of Kickstart failing to work. Watch the messages as Kickstart uses DHCP to get an initial IP address. Another message will show the success (or not) of reading the specified ks.cfg file. A separate message should show the interface having an IP address re-assigned, not by DHCP but statically, from the Kickstart configuration file. If the GUI does not start, check file permissions and directory permissions, and mount the share from another system and examining files as root, just as the installer does. The file permissions on the iso images themselves have been tested with the initial install using `'linux askmethod'` as have the functionality of the share itself, and the directory permissions. Verify that these have not changed, as well as that the file permissions of dmz001-ks.cfg are correct. The system doing the installing is performing actions on the target hardware as root, and no credentials or authentication methods are configured for the installation CD, so these files must be remotely accessible and world readable. For this reason, it is recommended to do these system builds on a secure network.

Once the manually-initiated Kickstart completes, we are ready to automate further.

5.0 Automating the installation process

The creation of a Kickstart configuration file (dmz001-ks.cfg) has been straightforward, but there are two factors that drive us to take the automation a step further. First, entering of correct ks: parameters at the boot prompt can be tedious and error prone. Second, the default timeout value of Disc1 will cause the installation to proceed manually if the installer is interrupted, for example, by a severe need for caffeine.

The next step is to create a customized bootable CD to speed automated installations for your environment. The CD that is created will not solve every automated install issue – build technicians may still have to press a function key in order to boot the target system from CD and change BIOS settings manually, but it will minimize frustration and error significantly.

5.1 Preparing the files for an auto-install boot CD

The initial disc contains a directory /isolinux with all of the required ingredients for this boot CD. Mount disc1 on the installation server, or mount the iso image through the loopback.

```
  mount -t iso9660 -o loop=/dev/loop1,ro /share/fc2/FC2-i386-disc1.iso
/mnt/fc2-disc1/
  cp -a /mnt/fc2-disc1/isolinux/* /share/fc2/isolinux/
```

Unmount and eject the CD, especially if the CD-ROM precedes the hard disk in the boot sequence. Veteran Red Hat users will recognize that the files within the isolinux directory are very similar to the files previously found on Red Hat's boot floppies.

Edit the file isolinux.cfg to include the IP address and Kickstart directory of the installation server. Those familiar with customizing floppy-based Kickstart configurations will notice that the format of this file is analogous to syslinux.cfg from the diskette images previously included on Red Hat's distributions. Within isolinux.cfg, locate the default stanza, or create a new stanza as the default. Change the append line in this stanza to include the Kickstart command, just as it was manually passed at the boot prompt earlier, to test the Kickstart configuration file and server.

```
  cp isolinux.cfg isolinux.cfg.orig
  chmod 750 isolinux.cfg
  append ks=nfs:10.1.2.4:/share/fc2/ks/ks.cfg initrd=initrd.img
ramdisk_size=8192
```

Note the target Kickstart configuration file carefully. We do not want to burn a new installation CD for every system created, but we want to automate the process as much as possible. The ks.cfg file that is referenced here has not yet been created. Several options are available, and depending upon your environment. For this environment, a symbolic link is appropriate.

```
  tester@server:/share/fc2/ks $ ln -s dmz001-ks.cfg ks.cfg
```

One of the benefits of this method is that the ks directory structure may be shared between system administrators, engineers, and build technicians, with minimal conflict as long as user ownership is preserved and respected. Nothing in this directory structure needs to be owned by root, or remotely writable by root, although the Kickstart configuration files must be readable by everyone at build time. To avoid conflict, have each build technician "own" a symbolic link to delete and create as desired, matching the append filename on their customized installation CD. If you are using links as described, be mindful of the permissions of the underlying –ks.cfg file that is linked.

Although not required, two other changes are recommended. Also within isolinux.cfg, change the value of the timeout line from 600 to a shorter wait, such as 30. The value is in tenths of a second, so the install will start very promptly. You may also want to edit boot.msg to be informative in case this boot CD falls into untrained hands, such as

```
  -  This is a DESTRUCTIVE installation disc.
  -  Untrained monkeys should Power Off NOW!
  -  Please return disc to The Unix Guru, Phone 555-1010
  -  NOT a flying toy.
```

You can even leave the escape codes in place for colorized text, change the colors, or experiment as desired with creating your own graphic for your automated installation boot CD, but those are beyond the scope of this document.

5.2 Creating the auto-install boot CD

All the important edits to the isolinux file structure are finished. Time to make the CD.

```
  $ cd /share/fc2
  $ chmod 640 isolinux/isolinux.bin
  $ mkisofs -o /share/fc2/AutoInst_fc2.iso -b isolinux.bin -c boot.cat -no-
emul-boot -boot-load-size 4 -boot-info-table -R -J -V -T isolinux [2]
  # cdrecord -v -eject speed=1 dev=1,0,0 –data /share/fc2/AutoInst_fc2.iso &
```

Test that it boots and begins the installation process. Make another copy if you are planning on testing in parallel, although you may choose to edit isolinux.cfg again so that each autoinstall boot CD points to a distinct Kickstart configuration file. After testing that the AutoInst CD boots and initiates the Kickstart process, make the iso file less likely to be changed.

```
  # chown root.root /share/fc2/AutoInst_fc2.iso
  # chmod 440 /share/fc2/AutoInst_fc2.iso
```

Follow appropriate change control when creating and modifying this file or files.

5.3 Extending the Kickstart configuration file

After testing the auto-install CD, there is one more step that is prudent to take to make the build process as specific as possible. It is a best practice to specify configuration options, even if the defaults provide the same functionality. The %packages section of the Kickstart file retains just this reliance on defaults, because the required packages are specified elsewhere. We are going to use the file listing packages that we've created for this purpose. Note that although the initial %packages section shows only four packages (grub, kernel, lvm2, and e2fsprogs) there are 228 packages installed when a minimum install is chosen.

```
  wc -l rpm-novers.2004-09-08
  228 rpm-novers.2004-09-08
```

We have already tested the initial Kickstart configuration file, dmz001-ks.cfg, twice – once with a manual invocation of ks:nfs at the boot prompt, and once to verify that the Auto-install CD is functioning properly. Before adding all 228 packages identified in the rpm-novers file, we make a new version of this file, then insert all 228 lines in the %packages section.

```
  $ wc -l dmz00*
       38 dmz000-orig.ks.cfg
       49 dmz001-ks.cfg
```

[2] Red Hat's Kickstart mailing list
Tibbits, Jason L, "Re: modifying isolinux.iso / Red Hat 9" 02 Jul 2003 URL: https://listman.redhat.com/archives/kickstart-list/2003-July/msg00005.html (23 October 2003)

```
      274 dmz002-ks.cfg
  $ rm ks.cfg
  $ ln –s dmz002-ks.cfg ks.cfg
```

The new Kickstart file is tested, and interestingly enough, the graphical installer begins, but shows an error “Comps package not found. Continue [Yes|No]” Continuing allows the installation to succeed as before, but we really do not want to be prompted each time the graphical installer launches. We again make a copy of the Kickstart file to update, dmz003-ks.cfg, edit the file and remove the ‘comps’ line from within the %packages section, update the ks.cfg symbolic link and reboot the target system to verify that the installation launches successfully. While this installation is completing, a bit of research shows the purpose and contents of the comps package. The comps is one of the files that contains the contents of the distribution release, and details which packages are required, as well as other features such as package groups. More about the comps.xml file can be found at http://fedora.redhat.com/projects/anaconda-installer/comps.html.

In addition to reading *about* a package, it is useful to query the package directly:

```
  rpm -qlp Fedora/RPMS/comps-2-0.20040513.i386.rpm
  warning: Fedora/RPMS/comps-2-0.20040513.i386.rpm: V3 DSA signature: NOKEY,
key ID 4f2a6fd2
  /usr/share/comps
  /usr/share/comps/i386
  /usr/share/comps/i386/.discinfo
  /usr/share/comps/i386/comps.xml
  /usr/share/comps/i386/hdlist
  /usr/share/comps/i386/hdlist2
```

The installation completes successfully (without a prompt), and a quick test determines that the comps package was installed, as it was prior to the fully detailed %packages listing. At this point, it has been confirmed that the installation completes and this package is installed, even if left out of the %packages listing of dmz003-ks.cfg. The same practice of iterating the number of the Kickstart files will be continued, in order to track the addition of specific elements to the build process.

The “reboot” parameter is added to the Kickstart file. This allows the system build technician to boot the auto-installation CD, and when the GUI appears and installation proceeds automatically, the CD can be removed and other work performed. Whether the installation takes 5 minutes, or 2 hours and 5 minutes, it has been automated, and until the system reboots, there is nothing further to do. Make certain when you add this parameter that you recognize a possible vicious cycle. If your CD-ROM is your primary boot device, and your auto-installation CD build the system with a Kickstart file that reboots the system, you may see a system boot and install repeatedly until the CD is removed or the boot sequence changed.

There are many other functional tasks that you may choose to add to the %post section of the Kickstart files. A few of these are below, as examples that add functionality in a consistent manner. More of these will be added in the next section, as iterative progress is made to create a more secure system built entirely within the Kickstart process. Note that the commands added to the %post section of the Kickstart file take place as root, in the chrooted environment after the installation occurs. When troubleshooting a specific line of a Kickstart configuration, it is useful to perform the same command manually during a manual installation, or boot Disc 1 in rescue mode and verify that the command works in the installation environment.

5.4 Automating DNS settings

The installation procedure frequently uses a static IP address, rather than a DNS name, because the search suffixes that are needed for the network are not yet in place. In other words, the resolv.conf that results from a build is not suitable for the network. It is possible to specify multiple DNS servers within the Kickstart file, but specifying multiple domains in the search path continues to be easiest done in the %post section.

```
  %post
  # DNS configuration
  cp /etc/resolv.conf /etc/resolv.conf.orig
  LOCALDOM="build.net"
  BUILDHN=`awk -F= '($1=="HOSTNAME") { print $2 }' /etc/sysconfig/network `
  BUILDIP=`awk -F= '($1=="IPADDR") { print $2 }' /etc/sysconfig/network-
scripts/ifcfg-eth0`
  echo "domain $LOCALDOM" > /etc/resolv.conf.build
  echo "search $LOCALDOM" >> /etc/resolv.conf.build
  echo "nameserver 10.1.2.4" >> /etc/resolv.conf.build
  cp /etc/resolv.conf.build /etc/resolv.conf
```

The anaconda installer produces an /etc/hosts file that does not conform to standard format of

```
  IP_address canonical_hostname aliases
```

So we are going to correct this issue with /etc/hosts within the build process. Note that you can't use the command successfully within the %post section, because the hostname is not yet set on

```
  # anaconda doesn't build /etc/hosts correctly. Fix it.
  cd /etc
  cp hosts hosts.orig
  CURRIP=`ifconfig |grep inet|grep -v inet6|grep -v 127.0.0.1|awk -F: '{print
$2}'|awk '{print $1}'`
  HOSTNAME=`hostname`
  cat hosts.orig |sed 's/`$HOSTNAME`//g'>/etc/hosts
  cat hosts.orig |grep -v 127.0.0.1 > hosts
  echo "127.0.0.1   localhost" >> hosts
  echo "$CURRIP    $HOSTNAME.$LOCALDOM $HOSTNAME" >> hosts
```

Keep in mind that the build network is isolated, and each network configuration file that is changed in the Kickstart configuration file may require a separate configuration for deployment when the system moves from the build network to the production or DMZ network. An example of creating separate configurations for build and deployment is below.

```
  echo "domain $PRODDOM" > /etc/resolv.conf.prod
  echo "search $PRODLDOM" >> /etc/resolv.conf.prod
  echo "nameserver 192.168.2.4" >> /etc/resolv.conf.prod
  echo "nameserver 192.168.2.40" >> /etc/resolv.conf.prod
```

Consider creating a simple shell script as well to perform this cutover. You may choose to configure your Kickstart configuration for your organization so that this cutover is not necessary, but that will depend on your organization's network environment. You may also want to customize other files such as ntp.conf, printer configurations, /etc/group and various files that are best kept harmonized through the environment. Since the focus of this discussion is security, the functional example of DNS will be the first and last functional example. The remainder of the entries detailed here will focus on securing the build, and we will consolidate information that is generated during the installation in a directory that we create, /var/log/build. We also copy the three files that need to change for a network change (IP, domain) between build and deployment.

```
  mkdir -p /var/log/build
  cp /etc/hosts /var/log/build/hosts.build
  cp /etc/sysconfig/network /var/log/build/network.build
  cp /etc/sysconfig/network-scripts/ifcfg-eth* /var/log/build/
  rpm -qa --queryformat '%{NAME}\n'|sort > /var/log/build/rpm-novers.`date +%Y-
%m-%d`
```

```
rpm -qa |sort > /var/log/build/rpm-list.`date +%Y-%m-%d`
cp -p /root/* /var/log/build/
```

6.0 Securing the build

Sufficient detail has been given to the creation of an automated build environment. Benefits of automating that process include not only efficiency, but also consistency and reduction of human error, both highly desirable from a security perspective. An automated build is not necessarily a secure build, and the security of a build should be planned from the initial install. The manual installation that was initially performed selected the minimum packages required for Fedora Core 2 to function, which follows the best practice of not installing anything that is not needed. There has been no specification of what services this DMZ host will provide, or how we will secure those services, but that specification will be postponed further, because there are already insecurities in the build.

6.1 Securing the initial services (sshd_config)

Although not specified earlier, the firewall feature was enabled during the manual installation, and ssh traffic allowed to the box, resulting in the following line in dmz001-ks.cfg:

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

The ssh service is widely known to be a secure replacement for telnet, ftp, rsh, rcp, and rlogin, and is of course desired for administrative access to the system. But there are steps to take immediately, because this “secure” default service has flaws. A quick test of the default ssh installation identifies two major problems with the ssh:

```
$ ssh -1 -l root dmz
The authenticity of host 'dmz (10.1.2.6)' can't be established.
RSA1 key fingerprint is c1:c6:39:f4:9e:09:c1:67:4c:ab:a5:d7:2c:a8:61:37.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dmz,10.1.2.6 (RSA1) to the list of known hosts.
root@dmz's password:
Last login: Sun Sep  5 15:12:57 2004 from 10.1.2.42
[root@dmz root]#
```

First, remote root logins were not disabled, and second, ssh protocol v1 remains supported, in spite of a known CRC32 attack. The initial build could be compromised out of the box. The next step is not only to correct the issue, but to capture that correction for the system’s Kickstart process. This two part process, correcting an issue on a live (test) system, and performing a correction within the Kickstart configuration file to correct further builds, is a process that will be repeated many times, and is shown and explained in detail for this specific issue. The corrections for these issues are simple changes to the sshd configuration file /etc/ssh/sshd_config. One line must be changed for each correction. On the live (test) system, this is initially done with **vi**, and the change shown below.

```
# diff sshd_config.orig sshd_config
14c14
< #Protocol 2,1
---
> Protocol 2
37c37
< #PermitRootLogin yes
---
> PermitRootLogin no
```

If you want to support key-based authentication, you should change this line to `PermitRootLogin without-password`. We need to restart sshd and test the corrections (one at a time). Keep in mind that in order to test the non-root login, we will need to create a local account.

```
root@dmz # /etc/init.d/sshd restart
Stopping sshd:                                          [  OK  ]
Starting sshd:                                          [  OK  ]
root@dmz # groupadd –g 777 admins
root@dmz # useradd –u 1234 –g 777 tester
root@dmz # passwd tester
Changing password for user tester.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
tester@works:~$ ssh –l root six
root@dmz's password:
Permission denied, please try again.
```

After verifying that remote root logins are disabled, we need to verify that ssh is working:

```
tester@works:~$ ssh dmz
tester@dmz's password:
[tester@dmz tester]$
```

We also need to test that Protocol 1 has been disabled

```
tester@works:~$ ssh -1 dmz
Protocol major versions differ: 1 vs. 2
tester@works:~$
```

Testing of the (manual) configuration change is complete, but to correct this insecure setting at system build, we need to make a modification to the system's Kickstart file that will prohibit both SSH Protocol 1 and remote root logins. A further implication from the above testing is that you should also know which administrative and user accounts need to be created at system build time, and include them in the build process as well. Familiarity with manipulating text files, either with **awk** and **sed**, or with **perl**, will help to make the Kickstart modifications straightforward.

The interactive portion, setting a password for a user, could be problematic. It is straightforward to script interactive system changes with **expect**, but the build process should not be interactive except where absolutely necessary. Fortunately, the **useradd** command on Linux includes a switch (-p) to allow passing the encrypted password. The encrypted password needs to be generated with **openssl** or with **perl**.

```
$ openssl passwd -1 -salt DoReMiFa L0ngAnd5trong
$1$DoReMiFa$hNm7lzzrHve8kOligtiP5/
```

For consistency, follow the same steps in your Kickstart configuration that you would perform if manually administering the environment. Make a copy of the original file, then make the changes to the desired file. While one might use **vi** to make small changes manually, we need to make the changes within the %post section of the Kickstart file, so we rely on **sed**, and command line options to create the user account. If any changes to the contents of /etc/skel are desired, implement those changes prior to the initial user account creation.

This is one of the chicken and egg issues of an automated build process that must be dealt with carefully. The problem of remote root login has been fixed, but the solution requires that the addition of at least one administrative account be automated as well. We know better than to store cleartext passwords in the configuration file, but even the encrypted administrator password is a risk. Administrators should use a temporary "build process" password, and change it upon first login. If a centralized authentication mechanism is in place, this step can be more elegant, avoiding storage of the account passwords, even if they are hashed and only for temporary use.

For this situation, we're retaining only the hash in the Kickstart file, which is acceptable on the build network. Make sure to escape the dollar signs with backslashes, comment stanzas appropriately, and always test your work.

```
  # Secure sshd. Create admin accounts.
  cp /etc/ssh/sshd_config /etc/ssh/sshd_config.orig
  cat sshd_config.orig |sed 's/#PermitRootLogin yes/PermitRootLogin no/g'|sed
's/#Protocol 2,1/Protocol 2/g'>sshd_config
  groupadd –g 66 admins
  useradd tester -u 1234 -g 66 -p \$1\$DoReMiFa\$hNm7lzzrHve8kOligtiP5/
```

Since this is a build process modification to the system, we do not need to add commands to the Kickstart file to restart sshd. However, this is a prudent time to retest the entire build process and verify that these changes are effected properly from installation. Boot your Kickstart CD on your test system and get a cup of coffee.

Additional steps are needed in order to secure this system. Identifying all of these steps requires consulting with company policy in order to satisfy each technical requirement. The first and largest set of prudent security steps will be identified by using the CIS Scoring Tool, and implemented using Kickstart.

6.2 Assessing system security with CIS Scoring Tool

"The Center for Internet Security (CIS) is a non-profit enterprise whose mission is to help organizations reduce the risk of business and e-commerce disruptions resulting from inadequate technical security controls." [3] Download the CIS Benchmark and Scoring Tool for Linux, from http://www.cisecurity.org/bench_linux.html, and verify the md5sum signature of the tarball prior to unpacking it.

Since this is a third party tool, unpack the tarball in /share/fc2/setup/.

```
  $ md5sum –c cis-linux_tgz_md5.txt
  cis-linux.tar.gz: OK
  $ tar xfvz cis-linux.tar.gz
  cis/
  cis/LinuxBenchmark.pdf
  cis/README
  cis/CISscan-1.4.2-1.0.i386.rpm
  root@dmz # rpm –Uvh /share/fc2/setup/cis/CISscan-1.4.2-1.0.i386.rpm
```

Unfortunately, Fedora is not yet recognized by the CIS Scoring Tool. Participating in CIS is encouraged, and it is likely that only a small amount of development effort would need to be contributed in order to enable the CIS tool to recognize Fedora, but the quickest possible workaround is in order:

```
  root@dmz # cp –p /etc/redhat-release /etc/redhat-release.orig
  root@dmz # echo "Red Hat Linux release 9 (Shrike)" > /etc/redhat-release
  root@dmz # /usr/local/CIS/cis-scan
  root@dmz # scp /usr/local/CIS/cis-most-recent-scan tester@server:/share/fc2/
```

Kindergarten has taught many of us to put things back where we found them.

```
  root@dmz # cp –p /etc/redhat-release /etc/redhat-release.cheat
  root@dmz # cp –p /etc/redhat-release.orig /etc/redhat-release
```

The results of this scan may not be entirely accurate, but performing these steps will provide us with a defined process and a more secure system coming out of the build process. The entire results of this first CIS tool are found in Appendix B. Experience shows that Red Hat, like most other Unix flavors and some Linux distributions, does not make major changes with each revision,

[3] http://www.cisecurity.org

so the evaluation of Fedora Core 2 against the two-releases-previous Red Hat 9 should provide valid work to do, and will result in a more secure system. Individuals and organizations are of course encouraged to extend and customize the security enhancements of their own Kickstart configurations as needed and desired. The cis-scan output is reviewed and corrective actions are taken on the live system and added to the Kickstart configuration, and tested, one at a time.

```
tester@server:/share/fc2 $ grep ^Negative cis-most-recent-log |more
```

Included with the CIS Scoring Tool is a reference that details a method for performing each of the corrections to secure the system. These corrective actions are clearly identified in the LinuxBenchmark.pdf that comes with the CIS Scoring Tool[4]. Read this Benchmark document to determine which corrective actions are needed, and for discussion of each correction – not all actions are necessary or possible for all organizations. The sections of code within the LinuxBenchmark are suitable for performing on a live (test) system. However, not every corrective action will work cleanly within the Kickstart post-installation. Each corrective item identified against the Fedora Core 2 build is included here, in a readable format with backslashes and spaces as appropriate. Notes detail changes that were required to adapt specific corrective actions to the Kickstart environment, and of course the entire Kickstart %post configuration file is in Appendix C.

The section on configuring SSH replaces the earlier corrections. This is more comprehensive, although the issue of automatically creating an administrative account remains.

```
# 1.2 Configure SSH
cd /etc/ssh
awk '($1=="Protocol") { print "Protocol 2"; next };
    { print }' ssh_config >ssh_config.new
/bin/mv ssh_config.new ssh_config
/bin/chown root:root ssh_config
/bin/chmod 644 ssh_config
if [ "`egrep -l ^Protocol ssh_config`" == "" ]; then
     echo 'Protocol 2' >>ssh_config
fi
awk '/^#?Protocol/ { print "Protocol 2"; next };
     /^#?X11Forwarding/ \
         { print "X11Forwarding yes"; next };
     /^#?IgnoreRhosts/ \
         { print "IgnoreRhosts yes"; next };
     /^#?RhostsAuthentication/ \
         { print " RhostsAuthentication no"; next };
     /^#?RhostsRSAAuthentication/ \
         { print "RhostsRSAAuthentication no"; next };
     /^#?HostbasedAuthentication/ \
         { print "HostbasedAuthentication no"; next };
     /^#?PermitRootLogin/ \
        { print "PermitRootLogin no"; next };
     /^#?PermitEmptyPasswords/ \
         { print "PermitEmptyPasswords no"; next };
     {print}' sshd_config >sshd_config.new
/bin/mv sshd_config.new sshd_config
/bin/chown root:root sshd_config
/bin/chmod 600 sshd_config
```

[4] http://www.cisecurity.org/bench_linux.html

Item 3.3 is the issue of sendmail running, essentially an unneeded service. From the host, netstat -an shows that sendmail is only listening on port 25 for connections from localhost, so this is acceptable, but we double-check externally with **nmap** from the server and verify that port 25 is not open. The rest of the noted services get shut down with a series of loops, and we add sendmail to the disabled services. Note that one loop is to disable and stop services, one to stop and remove services, and a third to remove services without stopping them, because this installation method (NFS) requires specified services to complete.

```
  # Disable and stop services
  for svc in acpid autofs gpm netfs mdmpd rhnsd rawdevices mdmonitor kudzu
cpuspeed
  do
   chkconfig --level 12345 $svc off
   /etc/init.d/$svc stop
  done
  # Stop and remove services
  for svc in apmd isdn pcmcia smartd
  do
   /etc/init.d/$svc stop
   chkconfig --del $svc
  done
  # For some services, disable them but leave them running during the build
  for svc in portmap nfslock rpcsvcgssd rpcgssd
  do
   chkconfig --level 12345 $svc off
  done
```

Items 4.1 and 4.2 of the Scoring Tool detail safer network interface configuration details.[5] Two recommended configuration settings are already in place in the initial Fedora build. All of the recommended settings are made to /etc/sysctl.conf.

```
  # Restrict network settings for secure operations
  cp -p /etc/sysctl.conf /etc/sysctl.conf.orig
  echo "# Added by ks.cfg" >> /etc/sysctl.conf
  echo "#net.ipv4.ip_forward = 0 " >> /etc/sysctl.conf
  echo "net.ipv4.conf.all.accept_source_route = 0" >> /etc/sysctl.conf
  echo "net.ipv4.tcp_max_syn_backlog = 4096" >> /etc/sysctl.conf
  echo "#net.ipv4.conf.default.rp_filter = 1" >> /etc/sysctl.conf
  echo "net.ipv4.tcp_syncookies = 1" >> /etc/sysctl.conf
  echo "net.ipv4.conf.all.send_redirects = 0" >> /etc/sysctl.conf
  echo "net.ipv4.conf.all.accept_redirects = 0" >> /etc/sysctl.conf
  echo "net.ipv4.conf.all.default.accept_redirects = 0" >> /etc/sysctl.conf
  chown root.root /etc/sysctl.conf
  chmod 0600 /etc/sysctl.conf
```

The CISecurity scan identifies filesystem insecurities with items 6.1 and 6.2. Again the Linux Benchmark recommended actions are added to the Kickstart configuration and tested.

```
  # Correct Filesystem insecurities
  cp -p /etc/fstab /etc/fstab.orig
  # 6.1 Add 'nodev. option to appropriate partitions in /etc/fstab
  awk '($3 ~ /^ext[23]$/ && $2 != "/") { $4 = $4 ",nodev" }; { print }'
/etc/fstab >/etc/fstab.new
  /bin/mv /etc/fstab.new /etc/fstab
  /bin/chown root:root /etc/fstab
  /bin/chmod 0644 /etc/fstab
```

[5] Koconis, David, Jim Murray, Jos Purvis and Darrin Wassam, Securing Linux: A Survival Guide for Linux Security, Version 1.0 SANS Press, 2003.

```
  # 6.2 Add 'nosuid' and 'nodev' option for removable media in /etc/fstab
  awk '($2 ~ /^\/m.*\/(floppy|cdrom)$/) { $4 = $4 ",nosuid,nodev" }; { print }'
/etc/fstab >/etc/fstab.new
  /bin/mv /etc/fstab.new /etc/fstab
  /bin/chown root:root /
  /bin/chmod 0644 /etc/fstab
```

The CIS security scan also identifies mount problems within Pluggable Authentication modules (PAM) in /etc/security/console.perms. The security issue identified is that of granting excessive privileges to the user at the physical keyboard. Although we could justify the location of this system as being in a controlled location regarding console permissions, it is prudent to make the recommended configuration changes unless a mission critical requirement for these permissions exists. Follow the same process of creating a ".orig" file, then follow the cisecurity.org LinuxBenchmark.pdf recommended actions to remove everything else. Backslashes are retained for readability.

```
  # 6.3 Disable user-mounted removable filesystems
  cp -p /etc/security/console.perms /etc/security/console.perms.orig
  cd /etc/security
  awk '($1 == "<console>") && ($3 !~ \
         /sound|fb|kbd|joystick|v4l|mainboard|gpm|scanner/) \
         { $1 = "#<console>" };
        { print }' console.perms >console.perms.new
  /bin/mv console.perms.new console.perms
  /bin/chown root:root console.
  /bin/chmod 0600 console.perms
```

The next issue is identified at the end of the cis-ruler-log, but is kept with the appropriate section for readability and maintainability. Three programs are identified as Non-standard SUID programs, /bin/traceroute, /bin/traceroute6, and /bin/ping6. This identification clearly identifies a change between Red Hat 9 and Fedora Core 2, because the CIS Scoring Tool uses a comparison against a listing of SGID and SUID created at installation. These lists are installed with the tool, at /usr/local/CIS/ in cis_ruler_*. Although we could create an appropriate ruler for Fedora Core 2, the most important aspect of this is the security of the system, and these rulers do not show the most secure locations, they provide a baseline as the OS is installed. A quick check shows that these files are elements of the iputils package.

```
  # rpm –qif /bin/ping6
```

Remediation of this issue includes two elements. The SUID bit is removed from traceroute. This system is not going to be on an IPv6 network, so the IPv6 programs will be removed, and for consistency the symlinks in /usr/sbin will be removed as well.

```
  # Correct Non-standard SUID programs
  chmod -s /bin/traceroute
  rm /bin/ping6
  rm /usr/sbin/ping6
  rm /bin/traceroute6
  rm /usr/sbin/traceroute6
  rm /bin/tracepath6
  rm /usr/sbin/tracepath6
```

Restrictions to cron and at are the next issue requiring remediation. The actions suggested within the LinuxBenchmark.pdf are used verbatim, and test successfully.

```
  # 7.5 Restrict at/cron to authorized users
  cd /etc/
  /bin/rm -f cron.deny at.deny
  echo root >cron.allow
  echo root >at.allow
  /bin/chown root:root cron.allow at.allow
  /bin/chmod 400 cron.allow at.allow
  # 7.6 Restrict permissions on crontab files
  /bin/chown root:root /etc/crontab
  /bin/chmod 400 /etc/crontab
  /bin/chown -R root:root /var/spool/cron
  /bin/chmod -R go-rwx /var/spool/cron
  /bin/chown -R root:root /etc/cron.*
  /bin/chmod -R go-rwx /etc/cron.*
```

Authorized use banners are virtually non-existent in the default installation. Remediation follows the actions suggested within the LinuxBenchmark.pdf, and test successfully. Note that there are many more banners required when other services are enabled, but that is not the case with this minimal installation.

```
  # 7.7 Create appropriate warning banners
  if [ "`egrep -l Authorized /etc/motd`" == "" ]; then
  echo "Authorized uses only. All activity may be monitored." > /etc/motd
  fi
```

Restrictions to root logins are the next issue requiring remediation. The actions suggested within the LinuxBenchmark.pdf are used verbatim, and test successfully.

```
  # 7.9 Restrict root logins to system console
  /bin/cp /dev/null /etc/securetty
  for i in 1 2 3 4 5 6; do
    echo tty$i >>/etc/securetty
    echo vc/$i >>/etc/securetty
  done
  echo console >>/etc/securetty
  /bin/chown root:root /etc/securetty
  /bin/chmod 400 /etc/securetty
```

Item 7.10 of the cis-scan output does not have a non-interactive method detailed, so we create one using **sed** to password-protect the boot selection.

```
  # 7.10 Set GRUB password and permissions
  # Create non-interactive solution
  cd /etc
  cp grub.conf grub.conf.orig
  cat grub.conf|sed 's/default/password=grubpass\ndefault/' > grub.conf.new
  mv grub.conf.new grub.conf
  /bin/chown root:root /etc/grub.conf
  /bin/chmod 600 /etc/grub.conf
```

Once again we have encountered a chicken-and-egg scenario, where automating the securing of a configuration setting places a password in a plaintext file that is world-readable, although only on the build network in this case. GRUB supports encrypted passwords, so the process used for creating an encrypted hash for an automatically adding a user account is repeated for the grub.conf password. You can also use the shell script /sbin/grub-md5-crypt to generate this password hash.

```
  $ openssl passwd -1 -salt DoReMiFa grubpass
  $1$DoReMiFa$C3BVsKoiC1cfg..8Rucbr1
```

The single **sed** line created above is replaced with the following:

```
  cat grub.conf|sed 's/default/password –-md5
\$1\$DoReMiFa\$C3BVsKoiC1cfg..8Rucbr1\ndefault/' > grub.conf.new
```

This item in particular receives a specific test to verify that the bootloader password is working as desired from the generated hash.

Inittab is identified as needing an addition of /sbin/sulogin for single-user mode. Again the standard recommended remediation actions are tested successfully (backslashes retained here for readability).

```
  # 7.11 Require authentication for single-user-mode
  cd /etc
  if [ "`grep -l sulogin inittab`" = "" ]; then
  awk '{ print };
       /^id:[0123456sS]:initdefault:/ \
          { print "~~:S:wait:/sbin/sulogin" }' \
      inittab >inittab.new
  /bin/mv inittab.new inittab
  /bin/chown root:root inittab
  /bin/chmod 644 inittab
  fi
```

Many account parameters are identified as problematic by the CIS Scoring Tool. The recommended remediation actions are added to the -ks.cfg file and tested successfully.

```
  # 8.1 Block system accounts
  for name in `cut -d: -f1 /etc/passwd`; do
    uid=`id -u $name`
    if [ $uid -lt 500 -a $name != 'root' ]; then
      /usr/sbin/usermod -L -s /dev/null $name
    fi
  done
  # 8.3 Set account expiration parameters on active accounts
  cd /etc
  awk '($1 ~ /^PASS_MAX_DAYS/) { $2="90" }
       ($1 ~ /^PASS_MIN_DAYS/) { $2="7" }
       ($1 ~ /^PASS_WARN_AGE/) { $2="28" }
       ($1 ~ /^PASS_MIN_LEN/) { $2="6" }
       { print } ' login.defs > login.defs.new
  /bin/mv login.defs.new login.defs
  /bin/chown root:root login.defs
  /bin/chmod 640 login.defs
  for name in `cut -d: -f1 /etc/passwd`; do
    uid=`id -u $name`
    if [ $uid -ge 500 -a $uid != 65534 ]; then
      /usr/bin/chage -m 7 -M 90 -W 28 $name
    fi
  done
```

The system umask settings and coredump restrictions are the final parameters in this minimal build that are identified by the CIS Scoring Tool. The recommended remediation actions are added to the -ks.cfg file and tested successfully.

```
  # 8.10 Set default umask for users
  cd /etc
  for file in profile csh.login csh.cshrc bashrc
  do
    if [ `egrep -c umask\.\*77 $file` -eq 0 ];
    then
      echo "umask 077" >> $file
    fi
    /bin/chown root:root $file
    /bin/chmod 444 $file
  done
  cd /root
  for file in .bash_profile .bashrc .cshrc .tcshrc
  do
    echo "umask 077" >>$file
    /bin/chown root:root $file
  done
  # 8.11 Disable core dumps
  cat <<END_ENTRIES >>/etc/security/limits.conf
  *               soft    core            0
  *               hard    core            0
  END_ENTRIES
  ## Finished correcting insecurities found with CIS Scoring tool
```

The CIS Scoring Tool has provided both the identification and recommended actions for securing this minimal dmz build. Several of the recommended actions required customization, either for a non-interactive solution, or for requirements specific to the target or because of the build mechanism. Each of these has been integrated into the initial build process with Kickstart and tested successfully.

6.3 Securing the automated build, iteratively

Many tools and techniques may initiate a change to systems in order to improve security. The CIS Scoring Tool is a host-based assessment tool that has been invaluable to this process followed so far. It may be desirable to add periodic runs of /usr/local/CIS/cis-scan to cron on each system, but the key element of that implementation are environment-specific. Organizational requirements, business needs and company policy may each affect how the CIS log is parsed and information directed to system administrators. Regardless of these factors for broader implementation, the CIS Scoring Tool should be installed at build time. The unfortunate ugliness of running the Scoring Tool created for Red Hat 9 against a Fedora build are captured here, with aspirations of replacing this workaround with an updated tool duly noted.

```
  # Install and run CISscan package at install
  rpm -i /share/linux/fc2/setup/cis/CISscan-1.4.2-1.0.i386.rpm
  # Remove this workaround when Updated CIS Scoring Tool is available
  cp -p /etc/redhat-release /etc/redhat-release.orig
  echo "Red Hat Linux release 9 " > /etc/redhat-release
  /usr/local/CIS/cis-scan
  cp -p /etc/redhat-release.orig /etc/redhat-release
```

The logs created by the CIS Scoring Tool should also be automatically retained on the build server. For this step, we leverage the initial administrative account created earlier (tester) and the NFS share used in this build environment to transfer all of the installation logs quickly, capturing logs from each Kickstarted system on the build network. Remember that for this build environment, we are creating local logs of scripted tasks in /var/log/build, so installation-specific files are copied there and chown'ed to the administrative user prior to copying to the NFS share, because root can't make the copy. Also note that after the system reboots, running the CIS Scoring Tool will generate output in /var/log, since /usr will be mounted read-only.

```
  # Copy the installation logs to the install server
  CILOG=`ls -1 /usr/local/CIS/cis-ruler-log*|tail -1`
```

```
cp $CILOG /var/log/build/
KSFILE=`ls -al /share/linux/fc2/ks/ks.cfg|awk '{print $11}'`
su -c "mkdir /share/linux/fc2/log/$KSFILE" tester
chown tester /var/log/build/cis-ruler-log*
su -c "cp -p /var/log/build/* /share/linux/fc2/log/$KSFILE/" tester
```

Each time that a security change has been made, the change has been made manually on a freshly built system, added to the %post section of a new -ks.cfg file, and a Kickstart has been performed against the new configuration to verify that the change was successfully integrated into the Kickstart process. For this specific process, we have generated Kickstart files named dmz001-ks.cfg through dmz035-ks.cfg, each with iterative changes. Many of the changes were minor, consisting of only a few lines, while other changes included a stanza of multiple lines. Recommendations for troubleshooting changes within this iterative process are in the following section. Regardless of the scope of changes between Kickstart files, it is the iterative testing process identified here that allows easy maintenance of secure system builds with Kickstart.

The CIS Scoring Tool is not the only useful tool for identifying configuration settings that will improve the security of a system. Run a vulnerability assessment against the system, and evaluate the results. Make recommended changes, and test to verify that functionality is preserved. In some cases, the magnitude of customizations for a specific file may not be well-suited to line-by-line changes with **sed**, **awk**, or **perl** in the Kickstart configuration file, but may be dealt with by pulling a defined file from the build server, then performing minor customization. For a webserver, a standard httpd.conf might be kept on /share/fc2/setup, and a simple **cp** or **wget** would take the place of the many changes required to achieve a more secure httpd.conf.

6.4 Troubleshooting Kickstart %post commands

Eventually even an experienced Kickstart administrator will attempt to implement many changes at once in Kickstart %post section, and some of these may fail. Identifying which commands worked on the live system, but did not work within the installer can be tedious and may require additional troubleshooting steps such as caffeine, sugar, and even in certain extreme circumstances, sleep. Before resorting to all three of those extreme techniques, try these:

a) Within the -ks.cfg file, comment out the reboot line
b) Run the automated installation again (remember, this does allow time for caffeine)
c) When the installation is complete, do not reboot as prompted until errors are examined
d) Press Ctrl-Alt-F3 to view error messages on tty3
e) Manually test Kickstart configuration change commands in tty2

Each of these lines will need to be interpreted and corrected within the Kickstart file

```
* Running kickstart %post script(s)
```

This first line is good to see, because it denotes that nothing within the %post section has scrolled off the screen.

```
Stopping automount:                                  [  OK  ]
Shutting down console mouse service:                 [FAILED]
```

These three lines confirm that the loops dealing with acpi, automount, and gpm are being read and executed. The failure of gpm to stop is of no concern – clearly gpm was not yet running within the installer.

```
Unmounting NFS filesystems                           [  OK  ]
/mnt/source: No such file or directory
Unmounting NFS filesystems (retry):                  [  OK  ]
```

```
  /mnt/source: No such file or directory
  Unmounting NFS filesystems (retry):                      [  OK  ]
  /mnt/source: No such file or directory
```

Although these return success, the "retry" tips us to the fact that the init script is successful but that the NFS files are not unmounting cleanly. It also prompts us to move netfs from the "Disable and stop services" section to the section with portmap and nfslock, because these services will continue to be used by the post-install scripts.

```
  Stopping mdmpd:                                          [FAILED]
  Shutting down sendmail                                   [FAILED]
  Shutting down smartd:                                    [FAILED]
```

Again disregard the failure of services to stop cleanly during the installation. We can remove these errors by moving the services to the "Disable but leave running" section of the Kickstart configuration.

```
  /tmp/ks-script: line 99: 6.2: command not found
```

This immediately identifies where the -ks.cfg file is failing, but note that "line 99" is the 99th line *after* the %post section delimiter. Adding the proper # to this comment line eliminates this error.

```
  awk: cmd. line:2: fatal: cannot open file 'console.perms' for reading (No
such file or directory)
  /bin/chown: cannot access 'console.': No such file or directory
```

These two errors are both the result of failing to either cd to the target directory or fully qualify the path to the target file. Adding 'cd /etc/security' fails to correct these two errors, but fully qualifying the path to the files is successful.

```
  * WARNING - Error code 512 encountered running a kickstart %pre/%post script
  * All kickstart %post script(s) have been run
  * moving (1) to step methodcomplete
  * moving (1) to step complete
```

The final four lines indicate that anaconda is aware that there were errors in the %pre or %post scripts, but it remains up to a human to correct. One of the simplest ways to test a command that has been put into the %post is to toggle to the command prompt of the installer, on tty2. After the installation completes, press Ctrl-Alt-F2, (or if you've already pressed Ctrl-Alt-F3 to examine error output on tty3, simply press Alt-F2) to get to the installer's shell prompt. Keep in mind that the installer works on the target system at /mnt/sysimage, and the %post script executes in a chrooted environment (just as the rescue environment does) so in order to test within that environment, you will have to 'chroot /mnt/sysimage' first, then perform any manual testing.

7.0 Automating updates

Although the CIS Scoring Tool does not complain about updates, there are many current updates to apply to this Fedora Core 2 build. It is also important to make updates regularly. Although there are many tools for performing automated updates, Yellow dog Updater, Modified (**yum**)[6] will be covered here as an example. For Red Hat Enterprise Linux, a similar configuration may be done with up2date. Yum was selected for similarities to up2date. A comprehensive comparison between automatic updating software was not performed, but initial evidence shows

[6] http://linux.duke.edu/projects/yum/

that yum provides ability to reference multiple repositories for updates, on the client, allowing ease of scalability.

7.1 Configuring yum.conf and updating manually

The first recognized step for a secure build environment is to mirror the packages that will be applied as updates. This serves a functional purpose of avoiding direct reliance on an outside site, even if it is mirrored widely and updated regularly. More importantly from a security perspective, there may be verification steps that the organization performs on these packages, prior to releasing them within the organization. Each organization may differ, and detailed implementations will not be covered other than to recognize that separate yum.conf configuration files for test and production could direct test and production systems (respectively) to the correct yum repository. The task of producing both repositories with a set of automatically generated links is left as an exercise for the reader. Keep in mind that the "test and production" referred to here are distinctions made within the organization doing automatic updates, and Fedora releases updates as well as "testing" packages.

Use your favorite tool to create a local mirror of the updates on the build server. If rsync is not already your favorite tool for this, become friends with rsync[7].

```
  tester@server $ rsync -avr --exclude "SRPMS/" --exclude "debug/"
distro.ibiblio.org::fedora-linux-updates/2/i386 /share/fc2/updates/
```

Automate this mirror with cron on the build server. After manually creating a local mirror, manually import the desired GNU Privacy Guard (GPG) keys to verify integrity of packages. In this case, we are going to trust the Fedora packages only, but no other packagers. We are also not going to trust (import) the testing packages.

```
  # rpm --import /usr/share/rhn/RPM-GPG-KEY
  # rpm --import /usr/share/rhn/RPM-GPG-KEY-fedora
```

Users choosing to automatically update Fedora's testing packages should also import RPM-GPG-KEY-fedora-test. Manually test the integrity of the packages in the local mirror – each line should show "OK."

```
  $ rpm -K *rpm
  ...
  kcc-2.3-20.1.i386.rpm: (sha1) dsa sha1 md5 gpg OK
  ...
```

Several configuration changes must be made to /etc/yum.conf. The first set of lines are added to the global section.

```
  retries=2
  releasever=2
  basearch=i386
  gpgcheck=1
  exclude=kernel*
```

The initial parameters show 20 retries, which might be appropriate for a client that does not reference a local repository or that only polls public repositories. Since yum allows multiple repositories, lowering the number of retries will allow the update to proceed to the next repository more quickly. Release version and base architecture are specified manually, although those parameters could be returned dynamically. Adding the gpgcheck=1 line requires packages to pass the integrity check before being applied. The final global parameter that is set is to exclude kernel packages from being updated. For the home network, each kernel upgrade will be performed

[7] rsync http://samba.anu.edu.au/rsync/ is an exceptional tool for performing incremental transfer and mirroring, although it may not play nicely in every firewall environment.

manually[8]. For a larger organization, it may be possible or desirable to skip this exclusion, keeping in mind that the kernel updates need to be carefully migrated from the mirror to the test update repository, then the production update repository, each with appropriate communications to customers and application testers. The other configuration change is to direct the system to the correct internal repository for updates. For this, we replace the baseurl=http line in the [updates-released] stanza with a baseurl pointing to the the NFS mount point that contains updates.

```
  baseurl=file:///share/fc2/updates/i386/
```

Following our earlier procedure, we test the updates manually, prior to integrating them into the -ks.cfg file. The output from the manual update looks like this, with many lines removed `[...]` for readability.

```
  # yum update
  Gathering header information file(s) from server(s)
  Server: Fedora Core 2 - i386 - Base
  Server: Fedora Core 2 - i386 - Released Updates
  Finding updated packages
  Downloading needed headers
  xorg-x11-xauth-0-6.7.0-5. 100% |=========================|  68 kB    00:00
  [...]
  Resolving dependencies
  Dependencies resolved
  I will do the following:
  [update: cups 1:1.1.20-11.1.i386]
  [...]
  Is this ok [y/N]: Y
  Downloading Packages
  Getting cups-1.1.20-11.1.i386.rpm
  [...]
  Running test transaction:
  Test transaction complete, Success!
  libpng 100 % done 1/117
  [...]
  warning: /etc/samba/smb.conf created as /etc/samba/smb.conf.rpmnew
  [...]
  Completing update for cups  - 59/117
  [...]
  Completing update for xorg-x11-libs-data  - 117/117
  Updated:  cups 1:1.1.20-11.1.i386
  [...]
  Transaction(s) Complete
```

7.2 Integrating automatic yum updates into Kickstart

The Kickstart configuration file is edited once more to include the importing of packager keys, repositories for the organization, and to run the updates with yum. The manual edits that were performed in vi make particularly ugly sed commands within the Kickstart file, but are effective for the desired non-interactive task, although we again consider creating a "golden" yum.conf for the environment and storing it on /share/fc2/setup.

```
  # Perform edits on /etc/yum.conf as appropriate
  rpm --import /usr/share/rhn/RPM-GPG-KEY
```

[8] Red Hat's Fedora mailing list
Clint, "Re: yum hangs with "damaged header"?" 24 Dec 2003.
URL: http://www.redhat.com/archives/fedora-list/2003-December/msg04749.html (18 September 2004)

```
  rpm --import /usr/share/rhn/RPM-GPG-KEY-fedora
  cp -p /etc/yum.conf /etc/yum.conf.orig
  cat /etc/yum.conf.orig |sed
's/retries=20/retries=2\nreleasever=2\nbasearch=i386\ngpgcheck=1\nexclude=kerne
l*/g'|sed 's/updates-released]/updates-
released]\nbaseurl=file:\/\/\/depot\/linux\/fc2\/updates\/i386/g'|grep -v
^"baseurl=http://download.fedora.redhat.com/pub/fedora/linux/core/updates" >
/etc/yum.conf
  yum –y update
  ## end yum stanza
```

If performing iterative testing, you may find that commenting out the "yum -y update" line is convenient for avoiding repetitively long test build times, but make sure to include the action on production builds. The `yum -y update` command may be added to **cron** as desired throughout the environment. As mentioned previously, the automatic updating of system packages requires careful thought and testing, as well as communication with users or business units.

8.0 Validating a secure build

Once the iterative steps have been completed to the satisfaction of the organization, a "finished" Kickstart configuration file is identified. We put this in a safe place, clean up (keeping the iterative tests in a separate directory) and test the build

```
  tester@server:/share/fc2/ks/
  $ cp dmz042-ks.cfg dmz-release0.99-ks.cfg
  $ mkdir dmz-testing
  $ mv dmz???-ks.cfg dmz-testing/
  $ rm ks.cfg
  $ ln –s dmz-release0.99-ks.cfg ks.cfg
```

We once again boot our automated installation CD on a target box, and get a cup of coffee. If the installation does not complete successfully, we return to the ks/dmz-testing/ subdirectory and our copious notes (some of which are automatically generated in the /share/fc2/log directories) to identify which ks.cfg file was last successful, and identify the error in the current file. A clean build that does not automatically reboot after installation will show the following (with some variation depending upon the %post commands).

```
  * Running kickstart %post script(s)
  Stopping automount:                                [  OK  ]
  * All kickstart %post script(s) have been run
  * moving (1) to step methodcomplete
  * moving (1) to step complete
```

After the installation completes successfully, there is some testing to perform. Run the cis-scan utility again to verify that all of the security fixes identified and integrated into the Kickstart configuration file are indeed being fixed with that process.

Additional security testing may be part of the validated build process. During a Kickstart installation, we run **nmap**[9] to determine if the build process really is secure, or if the installer provides easy mechanisms to compromise the build process.

```
  # nmap -sS 10.1.2.6

  Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
  All 1601 scanned ports on dmz.build.net (10.1.2.6) are: closed

  Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

[9] http://www.insecure.org/nmap/

Everything looks good, as expected. After the target build reboots, we run **nmap** again, as well as analyzing the cis-scan output and performing any other vulnerability assessments that are desired. The **nmap** output shows the expected open port for ssh. A quick look at the latest cis-scan output shows some remaining issues.

```
  $ grep ^Nega cis-ruler-log.20040920-16\:38\:40.1701
  Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/secure_redirects should be set to 0.
  Negative: 4.1 /proc/sys/net/ipv4/conf/lo/secure_redirects should be set to 0.
  Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/rp_filter should be set to 1.
  Negative: 4.1 /proc/sys/net/ipv4/conf/lo/rp_filter should be set to 1.
  Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/accept_redirects should be set to 0.
  Negative: 4.1 /proc/sys/net/ipv4/conf/lo/accept_redirects should be set to 0.
  Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/accept_source_route should be set to
0.
  Negative: 4.1 /proc/sys/net/ipv4/conf/lo/accept_source_route should be set to 0.
  Negative: 4.1 /proc/sys/net/ipv4/tcp_max_syn_backlog should be at least 4096 to
handle SYN floods.
  Negative: 4.2 /proc/sys/net/ipv4/conf/eth0/send_redirects should be set to 0.
  Negative: 4.2 /proc/sys/net/ipv4/conf/lo/send_redirects should be set to 0.
```

Analysis shows that the remediation recommended by the CIS Linux Benchmark for Red Hat 9 is not effective for the newer kernel in Fedora Core 2. Even with this outstanding issue, the overall score from the ruler is satisfying.

```
  Ending run at time: Mon Sep 20 16:38:59 2004
          Final rating = 9.69 / 10.00
```

When the security testing is completed, verify application functionality. For this DMZ bastion ssh system, this means simply **ssh**'ing into the box, and back out. Performing an **scp** to and from the box is reasonable as well. If key-based authentication is used in the environment, test that functionality as well. The application testing requirements are really defined by the purpose of the system, and testing that application may be handed from the build team to the application team for verification. If the application team can create testing scripts to validate each aspect of their requirements, it may be possible to integrate application testing into the build process, for specific application/build requirements.

Once all of the testing has completed, the Kickstart configuration is "golden" and should be treated as such.

```
  tester@server:/share/fc2/ks $ mv dmz-release0.99-ks.cfg dmz-release1.0-ks.cfg
  $ chmod 444 dmz-release1.0-ks.cfg
```

We may make a copy of this file and change only the "network" line in order to create a specific build based upon a validated build.

```
  $ cp dmz_release1.0-ks.cfg dmz_release1.0_ivory-ks.cfg
  network --device eth0 --bootproto static --ip 10.1.2.44 --netmask
255.255.255.0 --gateway 10.1.2.1 --nameserver 10.1.2.4 --hostname ivory
```

8.1 Scaling the build process

It is also possible to move the "network" line to the first line in the file, in order to make changes to the individual -ks.cfg files faster. If your target host has two interfaces, you will have two network lines. Depending upon your hardware configuration, it may be necessary to specify which network interface to install through.

There are several ways to scale the process so that multiple hosts may be built in parallel. The simplest is to create a dedicated Auto-installation CD as specified above for each build technician, so that each installer is responsible for properly editing -ks.cfg files and updating "their" symbolic link appropriately before booting their installation CD on the target machine.

If your system environment or experience includes diskless devices that boot off the network, consider implementing a Pre-eXecution Environment (PXE[10]) to support Kickstart[11]. The details of creating a PXE boot environment are beyond the scope of this document, but a brief overview is in order. Creating a PXE boot environment is in many ways analogous to creating a Jumpstart environment for Solaris. Creating a PXE boot environment involves using DHCP hints to direct hosts to the installation server and initial ram disk, essentially providing all of the information required to boot the system and begin the installation over the network, using PXE instead of local boot media. Unlike Jumpstart, the installation server does not have to reside on the same network segment as the target system, since Kickstart supports HTTP, FTP, and NFS. However, the DHCP hints that direct the target system to the installation server do have to come from the DHCP server that responds (usually on the local network segment). A concern with using PXE to boot an installation is that the build process must then entail configuring (and then de-configuring) each piece of hardware to attempt to PXE boot off the network. If systems are left configured for PXE boot, a single disk failure could result in an unintentional system rebuild occurring automatically within the build environment.

The author has personally had success building a dozen systems within 2 hours, using the process described above, using a single boot CD and a symlinked ks.cfg to direct each successive target system to the appropriate customized Kickstart configuration file. Similarly, the author has installed training classrooms of 25 identical (DHCP) systems in less than an hour, using boot floppies rather than the PXE boot environment. Choose the boot method and installation method that best suited to the environment.

9.0 Conclusions and lessons learned

Kickstart remains a flexible and useful tool, quite customizable for organizations. One of the benefits of creating individualized Kickstart files for each system is that both standardization and customization can be addressed in one place. Additional tools and security mechanisms such as tcp_wrappers and tripwire would be appropriate for an enterprise grade dmz host.

Performing a review of an organization's build process also fulfills the requirement of identifying all of the media necessary for building each type of system in the environment. This is a good time to make certain that copies of each are in a Disaster Recovery Plan location off site. Updates to the DRP location can be made by capturing the installation tree, /share/fc2 in this case, creating separate CD's for each ISO, and for the remaining directories. Documenting the creation of the build environment, using that media, allows a DR plan that can bootstrap quickly, potentially recreating both the build environment and critical servers very quickly on scrounged hardware.

The selection of NFS as an installation method was a poor choice, based primarily on the usefulness of NFS for other tasks in addition to serving the installation iso's. Although this selections saves ~2GB of disk space, little additional effort would have resulted in a more scalable HTTP solution, with significant advantages. First, the web server can be more easily hardened, while NFS remains a less secure service. Second, apache provides logging of activity, which can be useful in maintaining an awareness of installations performed in the environment. Third, performing updates with yum would have been easier pulling the update from an HTTP source rather than creating an NFS mount. The NFS mount is convenient on this particular build network, but the HTTP service could be used for updates on the less secure production network, and could

[10] ftp://download.intel.com/labs/manage/wfm/download/pxespec.pdf
[11] http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/sysadmin-guide/ch-pxe.html

be scaled to multiple sites to optimize both for speed of updates from a local repository, and for redundancy in the event of a local repository failing.

Headless installs are not particularly difficult, although verifying that the system will boot from CD without keyboard or mouse errors is useful, and the amount of time to wait before ejecting the CD may be difficult to determine with a truly headless system. Having the system automatically boot from CD (or other removable media) is not desirable from a security perspective. The only reasonable way to configure headless installs is to have the BIOS boot first from the hard disk, and if that fails, to boot from CD. A blank or wiped hard disk will allow the system to boot and install from CD, but will boot first from the hard disk after the install completes. If you are performing a destructive reinstall with a headless system that is configured with that boot sequence, a simple dd can wipe the first blocks of the hard disk, allowing the system to boot from the second device, the CD, and reinstall.

Once a build network is implemented to allow testing of various Kickstart configuration files and improvements, the process of extending, enhancing and customizing builds for specific tasks takes on new attributes of granularity and manageability. The difference between a dmz webserver and a DMZ mail relay may be very small, small enough to capture on a page or two with a **diff** of two customized Kickstart files. Creating and securing a base OS as we have done with the dmz build allows for extension and customization of that secure foundation for specific purposes. The combination of operating system and applications that comprise a single Open Source distribution release allows for significant customization of builds entirely from vendor-supplied components available at installation, rather than creating extended post-build application installation procedures.

By integrating custom builds into the build process, a rapid customization and deployment paradigm is available to organizations, with little additional overhead. Leveraging older hardware (specifically from hardware upgrade requirements of Windows systems) allows us to create a robust and functional environment inexpensively, using hardware that may have been headed for the dumpster. A full set of build/test/development/production environments that have consistent installation methods and consistent configurations, yet span multiple hardware platforms. It is also possible to fold a review of existing deployments into a "new" build process, simply by taking an inventory of deployed systems' anaconda-ks.cfg files on the build network, and replicating the deployed build to the test network to verify functionality. The same general process could be used to drive software or hardware upgrades, allowing comprehensive yet easy rebuilds of deployed systems for testing and enhancement, which would allow for smooth upgrades with minimal service interruptions.

References

1. Internet RFC's
   ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt

2. Red Hat's Kickstart mailing list
   Tibbits, Jason L, "Re: modifying isolinux.iso / Red Hat 9" 02 Jul 2003 URL:
   https://listman.redhat.com/archives/kickstart-list/2003-July/msg00005.html
   (23 October 2003)

3. Center for Internet Security
   http://www.cisecurity.org

4. Center for Internet Security – Linux Benchmarks
   http://www.cisecurity.org/bench_linux.html

5. Koconis, David, Jim Murray, Jos Purvis and Darrin Wassam, Securing Linux: A Survival Guide for Linux Security, Version 1.0 SANS Press, 2003

6. Yellow dog Updater, Modified (yum)
   http://linux.duke.edu/projects/yum/
   http://www.fedorafaq.org/#WhereCanIGetSoftware
   http://www.fedorafaq.org/samples/yum.conf
   http://www.hut.fi/~tkarvine/yum-package-manager.html#tips

7. rsync http://samba.anu.edu.au/rsync/ is an exceptional tool for performing incremental transfer and mirroring, although it may not play nicely in every firewall environment.

8. Red Hat's Fedora mailing list
   Clint, "Re: yum hangs with "damaged header"?" 24 Dec 2003. URL:
   http://www.redhat.com/archives/fedora-list/2003-December/msg04749.html (18 September 2004)

9. http://www.insecure.org/nmap/

10. Intel Pre-Execution Environment
    ftp://download.intel.com/labs/manage/wfm/download/pxespec.pdf

11. http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/sysadmin-guide/ch-pxe.html

Man pages and source code.

Appendix A: Initial CIS security scan

```
*** CIS Ruler Run ***
Starting at time 20040913-21:49:11

Positive: 1.1 System appears to have been patched within the last month.
Negative: 1.2 sshd_config parameter PermitEmptyPasswords is not set.
Negative: 1.2 ssh_config must have 'Protocol 2' underneath Host *.
Positive: 2.1 inetd/xinetd is not listening on any of the miscellaneous ports
checked in this item.
Positive: 2.2 telnet is deactivated.
Positive: 2.3 ftp is deactivated.
Positive: 2.4 rsh, rcp and rlogin are deactivated.
Positive: 2.5 tftp is deactivated.
Positive: 2.6 imap is deactivated.
Positive: 2.7 POP server is deactivated.
Positive: 3.1 Found a good daemon umask of 022 in /etc/rc.d/init.d/functions.
Positive: 3.2 inetd has been deactivated.
Negative: 3.3 Mail daemon is still listening on TCP 25.
Positive: 3.4 Graphical login is deactivated.
Positive: 3.5 X Font Server (xfs) script has been deactivated
Negative: 3.6 apmd not deactivated.
Negative: 3.6 gpm not deactivated.
Negative: 3.6 isdn not deactivated.
Positive: 3.7 Windows compatibility servers (samba) have been deactivated.
Positive: 3.8 NFS Server script nfs is deactivated.
Negative: 3.9 NFS script nfslock not deactivated.
Negative: 3.9 NFS script autofs not deactivated.
Positive: 3.10 NIS Client processes are deactivated.
Positive: 3.11 NIS Server processes are deactivated.
Negative: 3.12 RPC rc-script (portmap) has not been deactivated.
Negative: 3.13 netfs rc script not deactivated.
Positive: 3.14 printing daemon is deactivated.
Positive: 3.15 Web server is deactivated.
Positive: 3.16 SNMP daemon is deactivated.
Positive: 3.17 DNS server is deactivated.
Positive: 3.18 SQL database server is deactivated.
Positive: 3.19 Webmin GUI-based system administration daemon deactivated.
Positive: 3.20 Squid web cache daemon deactivated.
Negative: 3.21 Kudzu hardware detection program has not been deactivated.
Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/secure_redirects should be set to 0.
Negative: 4.1 /proc/sys/net/ipv4/conf/lo/secure_redirects should be set to 0.
Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/accept_redirects should be set to 0.
Negative: 4.1 /proc/sys/net/ipv4/conf/lo/accept_redirects should be set to 0.
Negative: 4.1 /proc/sys/net/ipv4/conf/eth0/accept_source_route should be set to
0.
Negative: 4.1 /proc/sys/net/ipv4/conf/lo/accept_source_route should be set to
0.
Negative: 4.1 /proc/sys/net/ipv4/tcp_max_syn_backlog should be at least 4096 to
handle SYN floods.
Negative: 4.2 /proc/sys/net/ipv4/conf/eth0/send_redirects should be set to 0.
Negative: 4.2 /proc/sys/net/ipv4/conf/lo/send_redirects should be set to 0.
Positive: 5.1 syslog captures authpriv messages.
Positive: 5.2 FTP server is configured to do full logging.
Positive: 5.3 All logfile permissions and owners match benchmark
recommendations.
Negative: 6.1 /usr is not mounted nodev.
```

```
Negative: 6.1 /var is not mounted nodev.
Negative: 6.1 /tmp is not mounted nodev.
Negative: 6.1 /home is not mounted nodev.
Negative: 6.1 /boot is not mounted nodev.
Negative: 6.1 /opt is not mounted nodev.
Negative: 6.2 Removable filesystem /mnt/cdrom is not mounted nosuid.
Negative: 6.2 Removable filesystem /mnt/cdrom is not mounted nodev.
Negative: 6.2 Removable filesystem /mnt/floppy is not mounted nosuid.
Negative: 6.2 Removable filesystem /mnt/floppy is not mounted nodev.
Negative: 6.3 PAM allows users to mount removable media: <floppy>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <cdrom>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <pilot>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <jaz>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <zip>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <ls120>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <camera>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <memstick>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <flash>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <diskonkey>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <rem_ide>.
(/etc/security/console.perms)
Negative: 6.3 PAM allows users to mount removable media: <rio500>.
(/etc/security/console.perms)
Positive: 6.4 password and group files have right permissions and owners.
Positive: 6.5 all temporary directories have sticky bits set.
Positive: 7.1 rhosts authentication totally deactivated in PAM.
Positive: 7.2 /etc/hosts.equiv and root's .rhosts/.shosts files either don't
exist, are zero size or are links to /dev/null.
Positive: 7.3 FTP daemons do not permit system users to use FTP.
Positive: 7.4 X11 Server is blocked from listening on TCP port 6000.
Negative: 7.5 Couldn't open cron.allow
Negative: 7.5 Couldn't open at.allow
Negative: 7.6 The permissions on /etc/crontab are not sufficiently restrictive.
Negative: 7.7 No Authorized Only message in /etc/motd.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: vc/7.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: vc/8.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: vc/9.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: vc/10.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: vc/11.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: tty7.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: tty8.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: tty9.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: tty10.
Negative: 7.9 /etc/securetty has a non console or tty 1-6 line: tty11.
Negative: 7.10 GRUB isn't password-protected.
Negative: 7.11 /etc/inittab needs a /sbin/sulogin line for single user mode.
Positive: 7.12 NFS server restricts clients to privileged ports.
```

```
Negative: 8.1 bin has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 daemon has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 adm has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 lp has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 mail has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 news has a valid shell of /bin/sh.  Remember, an empty shell
field in /etc/passwd signifies /bin/sh.
Negative: 8.1 uucp has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 operator has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 games has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 gopher has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 ftp has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 nobody has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 rpm has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 vcsa has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 nscd has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 sshd has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 rpc has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 rpcuser has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 pcap has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
```

```
Negative: 8.1 mailnull has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Negative: 8.1 smmsp has a valid shell of /sbin/nologin.  Remember, the
/sbin/nologin shell, when found in /etc/shells, leaves a user potentially able
to use FTP.
Positive: 8.2 All users have passwords
Negative: 8.3 User rod should have a minimum password life of at least 7 days.
Negative: 8.3 User rod should have a maximum password life of between 1 and 90
days.
Negative: 8.3 /etc/login.defs value PASS_MAX_DAYS = 99999, but should not
exceed 90.
Negative: 8.3 /etc/login.defs value PASS_MIN_DAYS = 0, but should not be less
than 7.
Negative: 8.3 /etc/login.defs value PASS_MIN_LEN = 5, but should be at least 6.
Positive: 8.4 There were no +: entries in passwd, shadow or group maps.
Positive: 8.5 Only one UID 0 account AND it is named root.
Positive: 8.6 root's PATH is clean of group/world writable directories or the
current-directory link.
Positive: 8.7 No user's home directory is world or group writable.
Positive: 8.8 No group or world-writable dotfiles in user home directories!
Positive: 8.9 No user has a .netrc file.
Negative: 8.10 Current umask setting in file /etc/profile is 000 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/profile is 000 -- it should
be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/csh.login is 000 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/csh.login is 000 -- it should
be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/bashrc is 022 -- it should be
stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/bashrc is 022 -- it should be
stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/csh.cshrc is 002 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /etc/csh.cshrc is 002 -- it should
be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.bash_profile is 000 -- it
should be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.bash_profile is 000 -- it
should be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.bashrc is 000 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.bashrc is 000 -- it should
be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.cshrc is 000 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.cshrc is 000 -- it should
be stronger to block group-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.tcshrc is 000 -- it should
be stronger to block world-read/write/execute.
Negative: 8.10 Current umask setting in file /root/.tcshrc is 000 -- it should
be stronger to block group-read/write/execute.
Negative: 8.11 Coredumps aren't deactivated.
Preliminary rating given at time: Mon Sep 13 21:49:12 2004
```

```
      Preliminary rating = 6.25 / 10.00

Positive: 6.6 No non-standard world-writable files.
Negative: 6.7 Non-standard SUID program /bin/traceroute
Negative: 6.7 Non-standard SUID program /bin/ping6
Negative: 6.7 Non-standard SUID program /bin/traceroute6
Ending run at time: Mon Sep 13 21:49:13 2004

      Final rating = 6.41 / 10.00
```

Appendix B: Kickstart configuration file

```
# Kickstart file automatically generated by anaconda, modified by Rod

install
nfs --server=10.1.2.4 --dir=/share/fc2
lang en_US.UTF-8
langsupport --default en_US.UTF-8 en_US.UTF-8
keyboard us
#xconfig --card "Intel 815" --videoram 16384 --hsync 30-121 --vsync 48-160 --
resolution 1024x768 --depth 24
skipx
network --device eth0 --bootproto static --ip 10.1.2.6 --netmask 255.255.255.0
--gateway 10.1.2.1 --nameserver 10.1.2.4 --hostname dmz
rootpw --iscrypted $1$3S0XP9DU$JH.fAvDjc5hrXIMa17UpP0
firewall --enabled --port=ssh:tcp
#firewall --disabled
selinux --disabled
authconfig --enableshadow --enablemd5
timezone America/Chicago
bootloader --location=mbr
# 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=101 --asprimary
part / --fstype ext3 --size=2020
part swap --size=512
part pv.7 --size=100 --grow
volgroup vg00 pv.7
logvol /usr --fstype ext3 --name=lv00 --vgname=vg00 --size=5432
logvol /var --fstype ext3 --name=lv01 --vgname=vg00 --size=1236
logvol /tmp --fstype ext3 --name=lv02 --vgname=vg00 --size=1024
logvol /home --fstype ext3 --name=lv04 --vgname=vg00 --size=6544
logvol /opt --fstype ext3 --name=lv03 --vgname=vg00 --size=792

reboot

%packages
```

(complete listing in Appendix C)

```
%post
# DNS configuration
cp /etc/resolv.conf /etc/resolv.conf.orig
echo "domain build.net" > /etc/resolv.conf.build
echo "search build.net prod.net" >> /etc/resolv.conf.build
echo "nameserver 10.1.2.4" >> /etc/resolv.conf.build
echo "nameserver 10.1.2.40 " >> /etc/resolv.conf.build
cp /etc/resolv.conf.build /etc/resolv.conf
# anaconda doesn't build /etc/hosts correctly. Fix it.
cd /etc
cp hosts hosts.orig
CURRIP=`ifconfig |grep inet|grep -v inet6|grep -v 127.0.0.1|awk -F: '{print
$2}'|awk '{print $1}'`
HOSTNAME=`hostname`
cat hosts.orig |sed 's/`$HOSTNAME`//g'>/etc/hosts
cat hosts.orig |grep -v 127.0.0.1 > hosts
```

```
echo "127.0.0.1   localhost" >> hosts
echo "$CURRIP    $HOSTNAME.$LOCALDOM $HOSTNAME" >> hosts
# Create directories for build process
mkdir -p /var/log/build
mkdir /share
mount 10.1.2.4:/share /share
# Capture the initial package listing
rpm -qa --queryformat '%{NAME}\n' |sort > /var/log/build/rpm-novers.`date +%Y-
%m-%d`
rpm -qa |sort > /var/log/build/rpm-list.`date +%Y-%m-%d`
# Capture the initial build network configuration
cp /etc/hosts /var/log/build/hosts.build
cp /etc/sysconfig/network /var/log/build/network.build
cp /etc/sysconfig/network-scripts/ifcfg-eth* /var/log/build/
# Capture the initial log files generated by anaconda
cp -p /root/* /var/log/build
## Correct insecurities identified with CIS Scoring Tool
# 1.2 Configure SSH
cd /etc/ssh
awk '($1=="Protocol") { print "Protocol 2"; next };
    { print }' ssh_config >ssh_config.new
/bin/mv ssh_config.new ssh_config
/bin/chown root:root ssh_config
/bin/chmod 644 ssh_config
if [ "`egrep -l ^Protocol ssh_config`" == "" ]; then
     echo 'Protocol 2' >>ssh_config
fi
awk '/^#?Protocol/ { print "Protocol 2"; next };
     /^#?X11Forwarding/ \
         { print "X11Forwarding yes"; next };
     /^#?IgnoreRhosts/ \
         { print "IgnoreRhosts yes"; next };
     /^#?RhostsAuthentication/ \
         { print " RhostsAuthentication no"; next };
     /^#?RhostsRSAAuthentication/ \
         { print "RhostsRSAAuthentication no"; next };
     /^#?HostbasedAuthentication/ \
         { print "HostbasedAuthentication no"; next };
     /^#?PermitRootLogin/ \
        { print "PermitRootLogin no"; next };
     /^#?PermitEmptyPasswords/ \
         { print "PermitEmptyPasswords no"; next };
     {print}' sshd_config >sshd_config.new
/bin/mv sshd_config.new sshd_config
/bin/chown root:root sshd_config
/bin/chmod 600 sshd_config
# Create initial administrators, one at a time
groupadd -g 66 admin
useradd tester -u 1234 -g 66 -p \$1\$DoReMiFa\$DtDg5bsz3X2c1HEvZXF5r.
# Disable and stop services
for svc in autofs cpuspeed kudzu mdmonitor rhnsd rawdevices rpcsvcgssd rpcgssd
do
 chkconfig --level 12345 $svc off
 /etc/init.d/$svc stop
done
# Stop and remove services
for svc in apmd isdn pcmcia smartd
```

```
do
 /etc/init.d/$svc stop
 chkconfig --del $svc
done
# For some services, disable them but leave them running during the build
for svc in acpi gpn mdmpd netfs nfslock portmap sendmail smartd
do
 chkconfig --level 12345 $svc off
done
# 4.1 Network Parameter Modifications
cp -p /etc/sysctl.conf /etc/sysctl.conf.orig
cat <<END_SCRIPT >> /etc/sysctl.conf
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
END_SCRIPT
/bin/chown root:root /etc/sysctl.conf
/bin/chmod 0600 /etc/sysctl.conf
# 4.2 Additional Network Parameter Modifications
cat <<END_SCRIPT >> /etc/sysctl.conf
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
END_SCRIPT
/bin/chown root:root /etc/sysctl.conf
/bin/chmod 0600 /etc/sysctl.conf
# Correct Filesystem insecurities
cp -p /etc/fstab /etc/fstab.orig
# 6.1 Add 'nodev. option to appropriate partitions in /etc/fstab
awk '($3 ~ /^ext[23]$/ && $2 != "/") { $4 = $4 ",nodev" }; { print }'
/etc/fstab >/etc/fstab.new
/bin/mv /etc/fstab.new /etc/fstab
/bin/chown root:root /etc/fstab
/bin/chmod 0644 /etc/fstab
# 6.2 Add 'nosuid' and 'nodev' option for removable media in /etc/fstab
awk '($2 ~ /^\/m.*\/(floppy|cdrom)$/) { $4 = $4 ",nosuid,nodev" }; { print }'
/etc/fstab >/etc/fstab.new
/bin/mv /etc/fstab.new /etc/fstab
/bin/chown root:root /
/bin/chmod 0644 /etc/fstab
# 6.3 Disable user-mounted removable filesystems
cp -p /etc/security/console.perms /etc/security/console.perms.orig
awk '($1 == "<console>") && ($3 !~
/sound|fb|kbd|joystick|v4l|mainboard|gpm|scanner/) { $1 = "#<console>" };{
print }' /etc/security/console.perms >/etc/security/console.perms.new
/bin/mv /etc/security/console.perms.new /etc/security/console.perms
/bin/chown root:root /etc/security/console.perms
/bin/chmod 0600 /etc/security/console.perms
# 6.7 Correct Non-standard SUID programs
chmod -s /bin/traceroute
rm /bin/ping6
rm /usr/sbin/ping6
rm /bin/traceroute6
rm /usr/sbin/traceroute6
rm /bin/tracepath6
rm /usr/sbin/tracepath6
# 7.5 Restrict at/cron to authorized users
/bin/rm -f /etc/cron.deny /etc/at.deny
echo root >/etc/cron.allow
echo root >/etc/at.allow
/bin/chown root:root /etc/cron.allow /etc/at.allow
/bin/chmod 400 /etc/cron.allow /etc/at.allow
# 7.6 Restrict permissions on crontab files
/bin/chown root:root /etc/crontab
/bin/chmod 400 /etc/crontab
/bin/chown -R root:root /var/spool/cron
/bin/chmod -R go-rwx /var/spool/cron
/bin/chown -R root:root /etc/cron.*
/bin/chmod -R go-rwx /etc/cron.*
# 7.7 Create appropriate warning banners
if [ "`egrep -l Authorized /etc/motd`" == "" ]; then
echo "Authorized uses only. All activity may be monitored." > /etc/motd
fi
# 7.9 Restrict root logins to system console
cp -p /etc/securetty /etc/securetty.orig
/bin/cp /dev/null /etc/securetty
for i in 1 2 3 4 5 6; do
echo tty$i >>/etc/securetty
echo vc/$i >>/etc/securetty
done
echo console >>/etc/securetty
/bin/chown root:root /etc/securetty
/bin/chmod 400 /etc/securetty
# 7.10 Set GRUB password and permissions
# Create non-interactive solution
cd /etc
cp grub.conf grub.conf.orig
cat grub.conf|sed 's/default/password .-md5
\$1\$DoReMiFa\$C3BVsKoiC1cfg..8Rucbr1\ndefault/' > grub.conf.new
mv grub.conf.new grub.conf
/bin/chown root:root /etc/grub.conf
/bin/chmod 600 /etc/grub.conf
# 7.11 Require authentication for single-user-mode
cd /etc
cp -p inittab inittab.orig
if [ "`grep -l sulogin inittab`" = "" ]; then
awk '{ print }; /^id:[0123456sS]:initdefault:/{ print "~~:S:wait:/sbin/sulogin"
}' inittab >inittab.new
/bin/mv inittab.new inittab
/bin/chown root:root inittab
/bin/chmod 644 inittab
fi
# 8.1 Block system accounts
for name in `cut -d: -f1 /etc/passwd`; do
  uid=`id -u $name`
  if [ $uid -lt 500 -a $name != 'root' ]; then
    /usr/sbin/usermod -L -s /dev/null $name
  fi
```

```
done
# 8.3 Set account expiration parameters on active accounts
cd /etc
awk '($1 ~ /^PASS_MAX_DAYS/) { $2="90" }
     ($1 ~ /^PASS_MIN_DAYS/) { $2="7" }
     ($1 ~ /^PASS_WARN_AGE/) { $2="28" }
     ($1 ~ /^PASS_MIN_LEN/) { $2="6" }
     { print } ' login.defs > login.defs.new
/bin/mv login.defs.new login.defs
/bin/chown root:root login.defs
/bin/chmod 640 login.defs
for name in `cut -d: -f1 /etc/passwd`; do
  uid=`id -u $name`
  if [ $uid -ge 500 -a $uid != 65534 ]; then
    /usr/bin/chage -m 7 -M 90 -W 28 $name
  fi
done
# 8.10 Set default umask for users
cd /etc
for file in profile csh.login csh.cshrc bashrc
do
  if [ `egrep -c umask\.\*77 $file` -eq 0 ];
  then
    echo "umask 077" >> $file
  fi
  /bin/chown root:root $file
  /bin/chmod 444 $file
done
cd /root
for file in .bash_profile .bashrc .cshrc .tcshrc
do
  echo "umask 077" >>$file
  /bin/chown root:root $file
done
# 8.11 Disable core dumps
cat <<END_ENTRIES >>/etc/security/limits.conf
*               soft    core            0
*               hard    core            0
END_ENTRIES
## Finished correcting insecurities found with CIS Scoring tool

## Automate updates with yum
# Perform edits on /etc/yum.conf as appropriate
cp -p /etc/yum.conf /etc/yum.conf.orig
rpm --import /usr/share/rhn/RPM-GPG-KEY
rpm --import /usr/share/rhn/RPM-GPG-KEY-fedora
cat /etc/yum.conf.orig |sed
's/retries=20/retries=2\nreleasever=2\nbasearch=i386\ngpgcheck=1\nexclude=kerne
l*/g'|sed 's/updates-released]/updates-
released]\nbaseurl=file:\/\/\/share\/fc2\/updates\/i386/g'|grep -v
^"baseurl=http://download.fedora.redhat.com/pub/fedora/linux/core/updates" >
/etc/yum.conf
yum -y update
## end yum stanza

# Install and run CISscan package at install
rpm -i /share/linux/fc2/setup/cis/CISscan-1.4.2-1.0.i386.rpm
```

```
cp -p /etc/redhat-release /etc/redhat-release.orig
# Remove this workaround when Updated CIS Scoring Tool is available
echo "Red Hat Linux release 9 " > /etc/redhat-release
/usr/local/CIS/cis-scan > /var/log/build/cis-scan.`date +%Y-%m-%d`
cp -p /etc/redhat-release.orig /etc/redhat-release
# Copy the installation logs to the install server
KSFILE=`ls -al /share/linux/fc2/ks/ks.cfg|awk '{ print $11 }'`
su -c "mkdir /share/linux/fc2/log/$KSFILE" tester
CILOG=`ls -1 /usr/local/CIS/cis-ruler-log*|tail -1`
cp $CILOG /var/log/build/
chown rod /var/log/build/cis-ruler-log*
su -c "cp -p /var/log/build/* /share/linux/fc2/log/$KSFILE/" tester
chmod 750 /var/log/build
```

Appendix C: Package listing

acl-2.2.7-5
acpid-1.0.2-6
anacron-2.3-30
apmd-3.0.2-22
ash-0.3.8-18
aspell-0.50.3-19.1
aspell-en-0.51-7.1
at-3.1.8-53
attr-2.4.1-4
authconfig-4.6.2-1
autofs-4.1.2-2
basesystem-8.0-3
bash-2.05b-38
bc-1.06-16.1
beecrypt-3.1.0-3
bind-libs-9.2.3-13
bind-utils-9.2.3-13
bzip2-1.0.2-12.1
bzip2-libs-1.0.2-12.1
chkconfig-1.3.9-1.1
comps-2-0.20040513
coreutils-5.2.1-7
cpio-2.5-6
cracklib-2.7-27.1
cracklib-dicts-2.7-27.1
crontabs-1.10-6
cyrus-sasl-2.1.18-2
cyrus-sasl-md5-2.1.18-2
cyrus-sasl-plain-2.1.18-2
db4-4.2.52-3.1
dev-3.3.13-1
device-mapper-1.00.14-3
devlabel-0.42.05-3.1
dhclient-3.0.1rc12-4
diffutils-2.8.1-11
dos2unix-3.1-17
dosfstools-2.8-12
dump-0.4b33-3
e2fsprogs-1.35-7.1
ed-0.2-35
eject-2.0.13-5
elfutils-0.95-2
elfutils-libelf-0.95-2
ethtool-1.8-3.1
fbset-2.1-15
fedora-logos-1.1.24-1
fedora-release-2-4
file-4.07-4
filesystem-2.2.4-1
findutils-4.1.7-25
finger-0.17-21
ftp-0.17-19
gawk-3.1.3-7
gdbm-1.8.0-22.1
glib-1.2.10-12.1.1
glib2-2.4.0-1
glibc-2.3.3-27
glibc-common-2.3.3-27
gmp-4.1.2-14
gnupg-1.2.4-2.1
gpm-1.20.1-49
grep-2.5.1-26
groff-1.18.1-34
grub-0.94-5
gzip-1.3.3-12
hdparm-5.5-1
hesiod-3.0.2-29.1
hotplug-2004_04_01-1
hwdata-0.118-1
info-4.6-3
initscripts-7.53-1
iproute-2.4.7-14
iptables-1.2.9-2.3.1
iputils-20020927-13
irda-utils-0.9.15-5
isdn4k-utils-3.2-13.p1.1
jwhois-3.2.2-3
kbd-1.12-1
kernel-2.6.5-1.358
kernel-utils-2.4-9.1.131
krb5-libs-1.3.3-1
krbafs-1.2.2-2.1
kudzu-1.1.62-1
less-382-3
lftp-2.6.12-1
lha-1.14i-14
libacl-2.2.7-5
libattr-2.4.1-4

```
libgcc-3.3.3-7
libpcap-0.8.3-3
libselinux-1.11.4-1
libstdc++-3.3.3-7
libtermcap-2.0.8-38
libuser-0.51.7-7.1.1
libwvstreams-3.70-13.1
libxml2-2.6.8-1
libxml2-python-2.6.8-1
lockdev-1.0.1-2.3.1
logrotate-3.7-4.1
logwatch-5.1-3
lrzsz-0.12.20-18
lsof-4.68-2
lvm2-2.00.15-2
mailcap-2.1.15-1
mailx-8.1.1-32
make-3.80-3
MAKEDEV-3.3.13-1
man-1.5m2-6
man-pages-1.66-2
mdadm-1.5.0-3
mingetty-1.07-2
minicom-2.00.0-18.1
mkbootdisk-1.5.1-1.1
mkinitrd-3.5.22-1
mktemp-1.5-7
modutils-2.4.26-16
mtools-3.9.9-8
mtr-0.54-5
mt-st-0.7-13.1
nano-1.2.3-1
nc-1.10-20
ncurses-5.4-5
netconfig-0.8.20-1.1.1
netdump-0.6.9-3.1
net-tools-1.60-25
newt-0.51.6-2.1.1
nfs-utils-1.0.6-20
nscd-2.3.3-27
nss_ldap-217-1
ntsysv-1.3.9-1.1
openldap-2.1.29-1
openssh-3.6.1p2-34
openssh-clients-3.6.1p2-34
openssh-server-3.6.1p2-34
openssl-0.9.7a-35
pam-0.77-40
pam_krb5-2.0.10-1
pam_smb-1.1.7-3.1
parted-1.6.9-3
passwd-0.68-8.1
pax-3.0-8
pciutils-2.1.99.test3-1.1
pcmcia-cs-3.2.7-1.5
pcre-4.5-2
perl-5.8.3-18
perl-Filter-1.30-5
pinfo-0.6.8-4
policy-1.11.3-3
policycoreutils-1.11-2
popt-1.9.1-0.3
portmap-4.0-59
ppp-2.4.2-2
prelink-0.3.2-1
procmail-3.22-13
procps-3.2.0-1.1
psacct-6.3.2-29
psmisc-21.4-2
pyOpenSSL-0.5.1-21.1
python-2.3.3-6
python-optik-1.4.1-5
pyxf86config-0.3.18-2
quota-3.10-2
raidtools-1.00.3-8
rdate-1.3-3.1
rdist-6.1.5-32
readline-4.3-10.1
rhnlib-1.5-1.1
rhpl-0.143-1
rmt-0.4b33-3
rootfiles-7.2-7
rpm-4.3.1-0.3
rpm-python-4.3.1-0.3
rp-pppoe-3.5-14
rsh-0.17-21
rsync-2.6.2-0
schedutils-1.3.0-6
sed-4.0.8-4
sendmail-8.12.11-4.6
setarch-1.4-1
setserial-2.17-15
```

setup-2.5.33-1
setuptool-1.15-1
shadow-utils-4.0.3-21
slang-1.4.9-3.1
slocate-2.7-9
specspo-9.0.92-1.1
star-1.5a25-5
statserial-1.1-34
stunnel-4.05-1
sudo-1.6.7p5-26
symlinks-1.2-21
sysklogd-1.4.1-16
syslinux-2.08-3
system-config-mouse-1.2.6-2
system-config-network-tui-1.3.16-1
system-config-securitylevel-tui-1.3.12-1
SysVinit-2.85-25
talk-0.17-23
tar-1.13.25-14
tcpdump-3.8.2-3
tcp_wrappers-7.6-36
tcsh-6.12-8
telnet-0.17-28
termcap-11.0.1-18.1
time-1.7-24
tmpwatch-2.9.0-2.1
traceroute-1.4a12-21.1
tzdata-2003d-2
unix2dos-2.2-21
unzip-5.50-37
up2date-4.3.19-1
usbutils-0.11-4
usermode-1.70-2
utempter-0.5.5-4
util-linux-2.12-18
vconfig-1.8-2
vim-minimal-6.2.457-1
vixie-cron-3.0.1-87
wget-1.9.1-5
which-2.16-2
wireless-tools-26-4
words-2-22
wvdial-1.53-13
ypbind-1.17.2-1
yp-tools-2.8-3
yum-2.0.7-1.1
zip-2.3-22
zlib-1.2.1.1-2.1