Global Information Assurance Certification Paper

SuSE Linux on a PowerBook G4 Workstation

by David F. Beck

This paper discusses the installation and configuration of Linux on a Apple Macintosh PowerBook G4 laptop workstation computer ("Titanium"). The actual hardware used in developing the paper had a processor speed of 500 MHz, 1 GB of built-in RAM, and a 30 GB internal ATA hard disk. The primary purpose of the Linux installation is as a portable network testing and security monitor using programs like "tcpdump." In addition, the laptop will be configured for use in a laboratory environment in evaluating MacOS-9 and –X vulnerabilities. Notes on the operating system (OS) software selected can be found under Step 1.1 below.

In the discussions that follow below, actions related to the security posture of the machine are often stated as being an element of a detect-delay-respond (DDR) cycle—the classical elements found in protection system design.[1] Alternatively they might be cast in terms of: protect-detect-react used by SANS; protect, detect, react, defend, reconstitute, and recover used by the DoD; or even prepare, detect, contain, eradicate, recover, and lessons learned used by CERT. Also noted will be actions that provide deterrents and defense-in-depth. Even quality comes into play as it can help avoid mistakes that might otherwise introduce security vulnerabilities or increase risks of existing vulnerabilities.

Step 1. Before Installation

Step 1.1 General considerations

The availability of Linux distributions for a PowerPC-based computer is much more restricted than for the more popular Intel machines. At the time this project began, the only 2.4 kernel available to me on CD was the SuSE Linux 7.1,[2] and is thus the basis of this paper. On August 27, 2001, it was announced that Mandrake Linux 8.0 became available.[3] Reportedly the Yellowdog[4] 2.0 distribution comes with a 2.4.4 kernel, and even others might be found with a more exhaustive search. The latest "Redhat-like" release from linuxppc.org, the so called "2000 Q4" distribution,[5] only comes with a 2.2 kernel. Of course the linuxppc distribution could be installed and updated with a new kernel from their software download site,[6] but then the issues associated with software-

[1] For example, Garcia, Mary Lynn, 2001, *The Design and Evaluation of Physical Protection Systems*, Butterworth-Heinemann, Boston.

[2] http://www.suse.de/us/products/susesoft/ppc/

[3] http://www.linux-mandrake.com/en/fnews.php3

[4] http://www.yellowdoglinux.com

[5] http://linuxppc.com/ or http://cable.linuxppc.org:8080/

[6] http://ftp.linuxppc.org/kernels/titanium/

download security would have to be addressed (or ignored; see section on software updates for more discussion).

Step 1.2 General security considerations

Step 1.2.1 Physical security

The integrated design and use of a LCD display in laptops provide them some security advantages over the typical desktop models deployed today; for example, simple ~$100 in-line keyboard sniffers[7] won't work, and TEMPEST[8] concerns are far less. However, due to the intrinsic and very desired nature of the portability of laptops (i.e., size and weight), they are a prime target for theft, which is the second leading cause of computer losses.[9] In 1997 laptop thefts reached 309,000,[10] in 1999 the number rose to more than 319,000,[11] and in 2000, approximately 387,000 were stolen;[12] based on production rates,[13] this suggests, very roughly, that 1 out of 50 laptops will be subject to theft. While most of these will head straight for a pawnshop, informal surveys indicate that about 10-15% of stolen laptops are taken by criminals intent on selling the data. A number of individual laptop thefts have even been newsworthy.[14] Perhaps most surprising, however, is that more of these thefts occur within the office than outside it, and as much as 75% of all of these thefts, in *and* out of the office, are committed by people with insider information.[15]

Although various firms advertise products that are intended to help track down and recover stolen computers, the best advice is still not to let a laptop out of your sight. The most promising aid for keeping a laptop "in sight" is a portable, two-piece anti-theft system using a key-chain transmitter and a miniature receiver with a 110+ decibel siren that "goes off" if the two are separated by more than ~40 feet.[16] For situations where the laptop must left unattended at home or in the office, storing it out-of-sight in a strong, locked cabinet is probably the best choice. Use of a security cable and locking device can also provide theft deterrence; there is no excuse not to use one at your desktop at least for protection during those "coffee" breaks. A minor security advantage is also available on

---

[7] For example, http://www.zdnet.co.uk/news/2000/12/ns-14347.html

[8] TEMPEST is a "code word" that relates to specific standards used to reduce electromagnetic emanations. See, for example, http://www.eskimo.com/~joelm/tempest.html

[9] http://www.safeware.com/index.htm

[10] http://www.notebookreview.com/security.html

[11] http://www.idg.net/idgns/2000/08/11/StolenLaptopSparksAntiTheftTechnology.shtml

[12] http://www.securityfocus.com/frames/?focus=basics&content=/focus/basics/articles/laptop1.html

[13] http://www.forbes.com/global/2001/0402/024.html

[14] http://www.business2.com/webguide/0,1660,43819|125|0|0|1|a,00.html

[15] http://www.exsys.co.il/it_woffa_overview.htm

[16] For example, http://www.trackitcorp.com/

the PowerBook G4 laptop: a keyboard locking mechanism, when engaged, hinders access to internal components of the computer, including the memory (see boot protection below for the importance of this in addition to outright theft of components).[17]

Step 1.2.2 Initial network security

During operating system (OS) installations, there may be a period of time during which network services may be available before security protections are fully in place or configured.[18] The length of this period of time depends on the installation process itself—how much time the installer devotes to the actual installation and security configuration. This vulnerability may be exploited by attacks that could even escape detection since logging and other security services would likely not yet be functional. The approach taken for this installation is to follow the best security practice of keeping the system disconnected from the network until after the installation and security configurations are complete.

Step 1.3 System boot protection

Apple's current boot firmware[19] is based on IEEE Standard 1275 known as "Open Firmware." Originally developed in 1988 by Sun Microsystems as a way to deal with the maintenance and support problems associated with a wide range of hardware and software configurations, Open Firmware provides processor and system independent boot firmware through a FORTH-based virtual machine. (If you are used to IBM or "clone" PCs, think BIOS.) As implemented by Apple, Open Firmware security options allow a machine to be fully password protected; booting from any device, or executing any boot firmware command for that matter, is not then possible without first entering the password (i.e., it does not matter whether you are using hard drives or removable media, or built-in or external devices, booting requires the password). Apple even went beyond the Open Firmware 1275 specification by adding a progressive delay technique to discourage brute force attacks against the Open Firmware password.

> A word of caution: the old adage applies that if someone has physical access to your computer they can eventually gain access to your data. While "zapping" the PRAM by itself (through use of 'Command' + 'Option' + 'P' + 'R' keys or by use of third-party programs) will not disable or remove the password protection, changing the amount of RAM in the computer followed by three PRAM resets will.[20] Or if they have a user account on the computer, it is possible to retrieve the boot password by reading the contents of the NVRAM itself and decrypting it, such as by using the

[17] 2001, *Getting Started With Your PowerBook G4*, Apple Computer, p.58.

[18] Cf. http://www.infowar.com/iwftp/xforce/advise23.html

[19] http://bananajr6000.apple.com

[20] http://www.msec.net/archives/of_pwd_bypass.html

> freely available FWSucker[21] tool. **However**, it should also be noted that use of an Open Firmware password does provide protection from certain kinds of network attacks—those that require a machine reboot—because the computer will now dutifully wait at the Open Firmware prompt for local user input.

In order to enable password protection for system boot, take the following actions:

1. Boot into Open Firmware by simultaneously pressing the Command, Option, 'O,' and 'F' keys during a system startup. The screen display should present something similar to the following:

```
Apple PowerBook3,2 4.1.8f5 BootROM built on 03/21/01 at 11:49:53
Copyright 1994-2001 Apple Computer, Inc.
All Rights Reserved.

Welcome to Open Firmware.
To continue booting, type "mac-boot" and press return.
To shut down, type "shut-down" and press return.

Ok
0 > _
```

2. At the prompt (0 >) enter the command *password*. You will then be asked to first enter, and then reenter, the desired password.

> A note about passwords: they should not be guessable and should be of a reasonable number of characters in length (say 8). Good passwords have a mix of alphanumeric characters, and even punctuation marks and such taken from the ASCII character set. Theoretically, a completely random string that makes use of all available characters is the best choice.[22] However, due to the propensity of people to write down hard to remember (read random) passwords, I believe the best (read lowest risk) scheme is to think of a short phrase and turn it into an acronym through some character selection scheme (e.g., first, third, or last letter of a word, and the use of some character substitution scheme, like the "3l33t" alphabet, for governing the use of numeric replacements of certain alpha characters). For example, the phrase "Hey diddle diddle, the cat and the fiddle" could become hddtcatf; and if we have a rule of replacing each 't' with a '7' and each 'f' with a '4' and then capitalize the first alpha character, we end up with Hdd7ca74. It goes without saying that your source for phrases should not be obvious, your particular substitution cipher should be secret, you should not have written copies of the password (or at the very least no uncontrolled copies), etc.

---

[21] http://www.securemac.com/file-library/FWSucker.sit

[22] For example, even at this point it would be possible to use a password produced from an external random generator; this could be code running on another UN*X platform (e.g., http://members-http-5.rwc1.sfba.home.net/denisl/passwdgen/download/passwdgen-2.2.tar.gz or http://www.multicians.org/thvv/tvvtools.html#gpw) or on a handheld device (e.g., "passwdGen Pocket" at http://members-http-1.rwc1.sfba.home.net/denisl/passwdgen/).

3. At the prompt (0 >) enter the command string *setenv security-mode full*

4. Reboot the computer, such as by using the Open Firmware command *reset-all*

5. The computer will now always boot into Open Firmware (you no longer have to press the Command, Option, 'O,' and 'F'). The screen display should now present something similar to the following:

```
Apple PowerBook3,2 4.1.8f5 BootROM built on 03/21/01 at 11:49:53
Copyright 1994-2001 Apple Computer, Inc.
All Rights Reserved.

Welcome to Open Firmware.
Full security mode. (emphasis mine to highlight the change)
To continue booting, type "mac-boot" and press return.
To shut down, type "shut-down" and press return.

Ok
0 > _
```

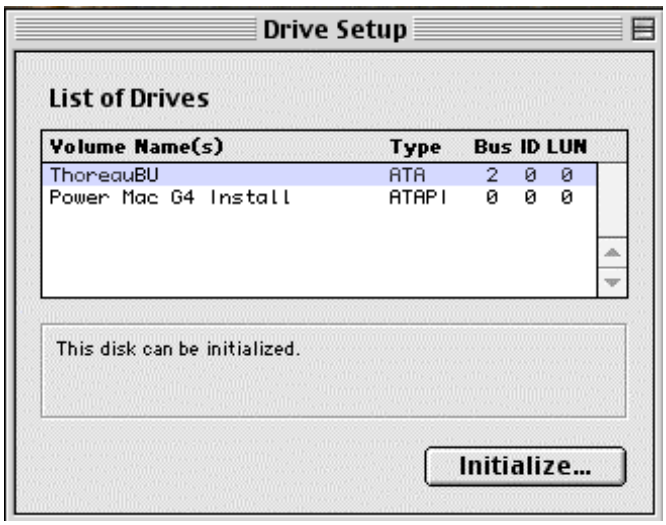
6. Assuming that you still have an intact Mac OS on the computer, follow the Open Firmware instructions and enter the command *mac-boot* to start the boot process. You will then be asked to enter your password before the command is processed.

Step 1.4 Disk partitioning

While your "mileage may vary," the procedure followed herein is for a PowerBook with a single, 30-GB internal hard drive that is to be formatted for three operating systems (OS): the "classic" Macintosh OS-9, OS-X (read 10), and Linux.

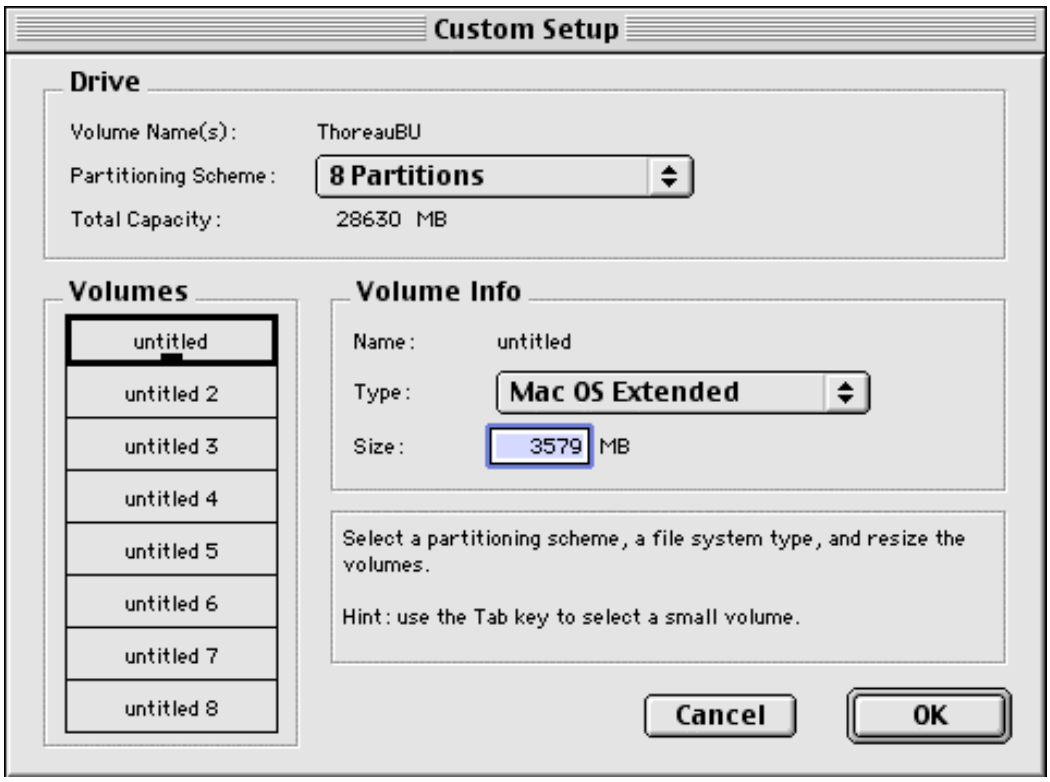
On system start, insert the "PowerBook G4 Software Install" CD into the CD drive. At the prompt (0 >) enter the *mac-boot* command, enter your password, then **immediately** hold down the 'C' key during the boot process. The computer will boot off of the OS it finds on the CD.

The *Drive Setup* program, found in the *Utilities* directory on the CD, is then used to partition the drive. As shown in the graphical user interface (GUI) illustrated below, the hard drive is selected to initialize (here named "ThoreauBU," yours may be "Untitled," "Macintosh HD," or whatever; the other drive shown in the illustration is the CD-ROM).



After selecting the correct drive in the GUI and "pressing" the *Initialize…* button, a second GUI appears as shown below.

At this point the *Custom Setup…* function should be selected, and a third GUI will appear:



This GUI, under the "Partitioning Scheme" control, will allow selection of the number of partitions the disk is to be initialized to. As shown here, 8 partitions have been selected for the purposes of the project described by this paper, as set forth in the table below. For each volume, the appropriate volume type was selected and the size entered. After pressing *OK*, you are returned to the second GUI, where the *Initialize* function is then invoked.

Again, your names may differ, and you may prefer a different partitioning scheme. The question of how many partitions continues to spark debate within the Linux and broader UN*X community. It is probably fair to say that there are almost as many partition layouts as there are people discussing the issue.

The SuSE recommendation[23] is to have: (1) a large partition for Mac OS (in the case here this becomes two in order to maintain the desired separation between Mac OS-9 and –X); (2) a small MacOS partition of approximately 30 MB in size for a Linux boot partition in HFS format;[24] (3) a swap partition of 4 MB to 128 MB in size (note, however, that most recommended partitioning schemes suggest the swap space should at least equal the installed RAM size as here; some even suggest a swap size of twice the RAM, or more);

---

[23] The recommendations are found in the *SuSE Linux 7.1 PowerPC Edition—the Handbook*, pp. 20-21.

[24] Full support for the Mac OS Extended file system (HFS+ or HFS Plus) in a kernel level module for Linux PPC releases is not yet available; progress in the implementation effort can be tracked at http://www.sourceforge.net/projects/linux-hfsplus. However, there are utilities that allow HFS+ file systems to be accessed in the compressed tarball http://penguinppc.org/projects/hfsplus/hfsplus.src.tar.gz.

and (4), a partition for SuSE Linux itself of recommended size 1.5 GB to 3 GB, or of 6 GB for those who want to install "almost everything" (the 2.783 GB used here is close to the upper end of 3 GB, and was basically selected to allow space for three more, equally and evenly sized partitions as discussed below).[25]

Volume Name	*Volume Type*	*Volume Size (MB)*	*purpose*
untitled	Mac OS Extended	8000	OS-9
untitled 2	Mac OS Extended	9000	OS-10
untitled 3	Mac OS Standard	32	yaboot
untitled 4	A/UX Swap	1000	swap
untitled 5	A/UX Root	2783	/
untitled 6	A/UX User	2600	/usr/local
untitled 7	A/UX User	2600	/var
untitled 8	A/UX User	2600	/home

The reasoning behind the additional three partitions for this project is as follows. First, /var will contain all of the log files, which can grow quite large. Operating systems tend to choke up or even become non-responsive if the file system they are on fills up. So, having /var on a partition that is separate from the root file system (/) can help to avoid this problem. Second, it is envisioned that there may be considerable use or evaluation of software that is not part of the SuSE distribution. It is desirable to keep this software separate from the OS for update, backup, and even system rebuild purposes. For this project all such software will be loaded into the separate /usr/local partition. Third and final, some software development may take place on this machine. This work will be segregated into the /home partition. The sizes used for each of these three partitions (2600 MB) was simply an even division of the available disk space. While it might be argued that more should have been allocated to the root file space, the fact is that the installation discussed below only consumes 670 MB (26%), and so it has plenty of room for growth.

After the initialization was completed, OS-9 was installed on both Mac OS Extended partitions. The machine was then rebooted from the OS-X CD, and OS-X was installed on the second Mac OS Extended partition; after booting in OS-X, the OS-9 system on the OS-X partition was configured to be used by OS-X when running classic Mac programs.

[25] See, for example, *Securing Linux Step-by-step* for additional discussion of this issue. Available at http://www.sansstore.org/

Since OS-X makes changes to OS-9 files, having two OS-9 installations in this manner allows booting in a "virgin" OS-9 environment for testing purposes.

> A further note about Open Firmware: At this point there are two boot systems available. Typing "mac-boot" within Open Firmware will startup the system last pointed to with the *Startup Disk* control panel available under Mac OS. For other options it is necessary to explore the system a little with the Open Firmware commands *printenv* and *devalias*. When OS-9 is the default system to boot, from *printenv* we find that one of the lines reads:
>
> Boot-device /pci@f2000000/mac-io@17/ata-4@1f000/@0:9,\\:tbxi
>
> Here the string of information up to the first colon is the hard drive identifier, which *devalias* then can tell us is also pointed to by the name 'hd.' The '9' after the first colon is the partition number. How convenient (as in easy to remember)! OS-9 on partition number 9, and by inference (although it could be verified in a similar manner by changing the default boot system with the *Startup Disk* control panel), OS-X will be on the 10th partition. Yaboot (the Linux boot script on PPC architectures) will thus be installed on partition 11. (Because Apple makes use of a number of small partitions on a disk, the initialization method used above actually setup partitions 9 through 16!) The "tbxi" at the end of the boot-device string points to a file of type tbxi, which happens to be the Mac OS ROM file.
>
> With this information, from the Open Firmware prompt we can now boot up in OS-9 with the command:
>
> boot hd:9,\\:tbxi
>
> or in OS-X with the command:
>
> boot hd:10,\\:tbxi

While this paper is not about securing Mac OS, at this point in the process, two Mac operating systems are, never-the-less, installed, and it behooves us to take some basic precautions[26] as outlined below.

1. Under OS-9:

   (a) "File Sharing" control panel. Turn file sharing off. Establish owner name and password.

   (b) "Multiple Users" control panel. Enable feature. Under *options* "Other" tab, select "When logging in, users type their names" (i.e., so as not to provide a listing of usernames on login screen) and deselect "allow a guest user account." Under *options* "Login" tab, deselect voice entry or verification

[26] For a start in dealing with Mac-specific security concerns, see, for example, http://www.sans.org/infosecFAQ/mac/ac_list.htm

of any type (no data is available on the security robustness of this feature); a simple warning banner could be entered here if desired. Other user accounts can be set up here as well (if any).

(c) Install and configure a third-party screen saver. Setting Sun[27] is one such program that has proven effective. The desired features are timeout (I set mine to 5 minutes), "hot corners" (to lock screen under password protection if I need to leave my desk for a while, but don't want to completely log out), and a display that does not reveal the desktop or any information about what is being worked on. The timeout feature available under the Mac OS-9 "Multiple Users" control panel only has the timeout feature, and thus the need for a third-party solution.

2. Under OS-X:

(a) Setup users, as necessary, under the "System Preferences," "Users" control panel.

(b) Under the "Login Window" tab of the "System Preferences," "Login" control panel, deselect "Automatically log in" and "Show password hint…." Select "Disable Restart and Shutdown buttons."

(c) In the "System Preferences," "Screen Saver" control panel, under the "Activation" tab, set the slider for the desired timeout (again, I set mine at 5 minutes), and select the "use my user account password" option. Under the "Hot Corners" tab, set up the desired hot corners.

At this point the laptop, still disconnected from the network, is protected from unauthorized use by passwords at system boot time, user login, and at the user desktop.

Step 2. Install Linux

Step 2.1 Install yaboot

Under Mac OS, copy (drag) the "suseboot" directory from *SuSE LINUX 7.1 PowerPC EDITION* CD number 1, to the *Mac OS Standard* partition (the name of the partition is not important; rename it as you see fit). The required files are: Finder, System, vmlinux-2.4, yaboot, and yaboot.conf. All other files and subfolders can be deleted to avoid confusion if desired. If the suseboot folder icon does not appear to be "blessed" (no smiling Mac face superimposed over a file folder icon), drag the "Finder" file out of the folder to the desktop, and then drag it back in. The file yaboot.conf should be edited to match the installation as follows:

```
timeout = 100
default = linux
```

[27] http://www.webthing.net/settingsun

```
bootfolder = suseboot\
# 2.4.2
image=vmlinux-2.4
      label=linux
      root=/dev/hda13
      video=aty128fb
```

In this case, the fact that the 'root' file system will be at /dev/hda13 can be inferred from knowledge about the Mac and other partitions as discussed above. Note the use of Open Firmware notation for the "bootfolder" directory entry. The video directive used here is compatible with the Titanium notebook hardware.

With yaboot in this location, **once Linux is installed** it is booted with the command:

```
boot hd:11,\suseboot\yaboot
```

after which you will soon see the boot prompt (if you are used to Linux on an Intel platform, think "lilo"):

```
Welcome to yaboot version 1.1.1.SuSE
boot: _
```

at which point you will be able to enter `linux` to complete the boot process (other boot parameters could also be passed at this time; e.g., to boot into a "single user" mode type `linux single` or `linux 1`).

Unfortunately, you will also notice the following output appearing before the boot prompt:

```
Config file read, 155 bytes
Config file error: Syntax error near line 4 in file \suseboot\yaboot.conf
Syntax error or read error config
```

In spite of this warning, yaboot will still allow you boot correctly. However, you will have to enter `linux` in order for the boot process to proceed (yaboot cannot find the kernel with a simple return at this point, nor is the timeout feature functional as an indirect result of this error). The problem that yaboot is reporting stems from the "\" in the "bootfolder" directive. However, if the "\" is removed, this version of SuSE yaboot cannot resolve the location of the kernel and, rather that typing in `linux` to boot, you will have to type in the Open Firmware pathname of `\suseboot\vmlinux-2.4` (yaboot does remain it the correct partition, just not in the correct directory). If you choose this method just in order to avoid the warning, the "bootfolder" directive might as well be deleted from the yaboot.conf file altogether.

```
Note: due to the security issues associated with having an
unprotected (except by the Open Firmware security implemented
above) command-line yaboot process that cannot, furthermore,
automatically load the Linux kernel (see the discussion under
```

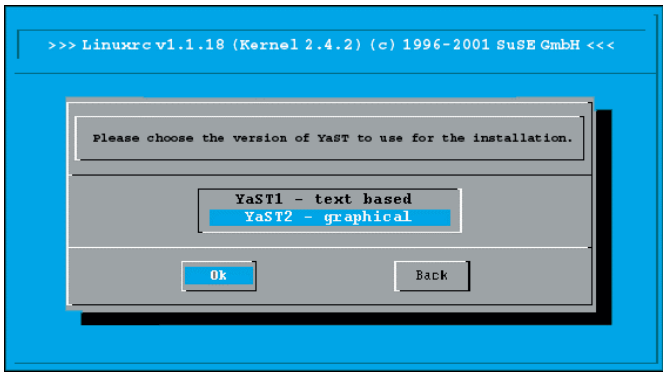
```
Step 2.3), it is strongly recommended[28] that a copy of the
graphical yaboot that comes with the "2000 Q4" linuxppc
distribution be used; no errors, no loosing its way! (Although
you may want to keep the SuSE yaboot somewhere for recovery
purposes.) The yaboot.conf file described above will work;
however, there is no need for the "bootfolder" directive, and it
can be safely removed. If you continue to use the SuSE command-
line version of yaboot, at least always make sure the system has
booted and is presenting a login prompt before walking away.
```

Step 2.2 Install SuSE Linux

Reboot the computer. On system start, if not already mounted, insert the *SuSE LINUX 7.1 PowerPC EDITION* CD number 1 into the CD drive. At the prompt (0 >) enter the *mac-boot* command, enter your password, then **immediately** hold down the 'C' key during the boot process. The computer will boot off of the OS it finds on the CD, which in this case is the SuSE version of yaboot. The screen display should present something similar to the following:

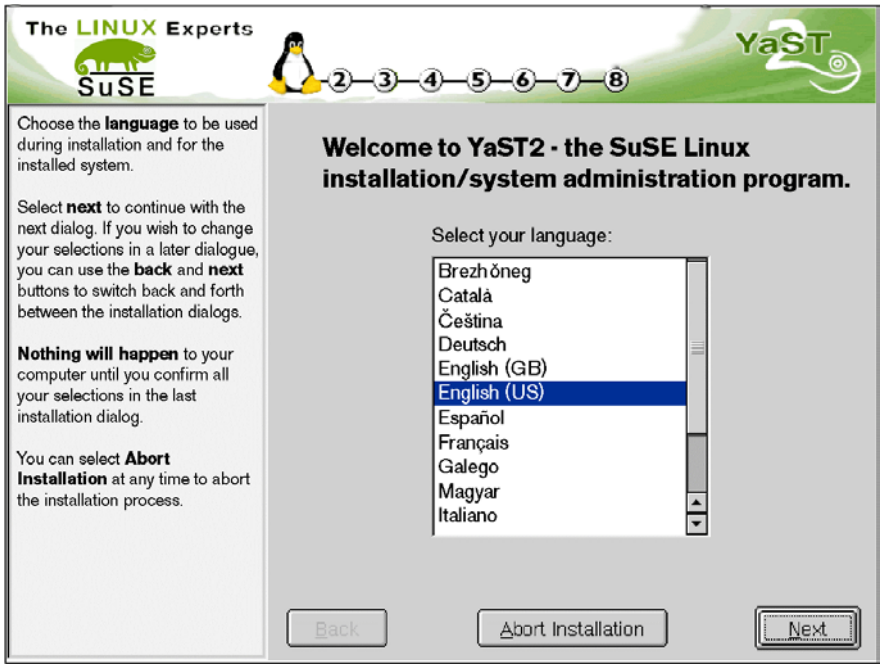
```
Welcome to yaboot version 1.1.1.SuSE
Boot:
```

Press the 'return' key, and the computer will proceed to boot up off of an image of a 'safe' version of Linux located on the CD. After boot up, something similar to the following will be displayed on the screen:



Selecting the graphical, YaST2 installation interface will then present a screen something like the following:

[28] Another option would be to build your own yaboot from the latest source and instructions available at http://www.alaska.net/~erbenson/ybin/



For my own installation, I selected 'English (US)' as highlighted in the figure above. Following this step, there are seven more primary stages in the installation process. As they all have a nice GUI, things are fairly straightforward. However, each stage is summarized below:

- Select keyboard (to match the language selection) and local time zone
- Select 'New Installation'
- Prepare the Hard Disk
    - Select "Custom partitioning" and specify A/UX partition usage (e.g., mount points) as previously determined in Step 1.4 above.
- Select software to install
    - A good choice would be to select the YaST2 option "Standard System."

Other prepackaged options are: "Minimal system" (text interface only); "Standard System with Office;" and "Almost everything." For the purposes of the security configuration discussions that follow below, it is assumed that networking and 'X' windows are installed, along with security services like "shadow" and PAM (all included in the "Standard System"). However, a safe workstation configuration will be developed even if you install everything. You can also safely ignore X-specific portions if X is not installed. My personal choice is the "Select single packages …" option (hand pick everything), although this requires considerable work and is heavily dependent upon what the intended use of the computer is. Of course you can always add packages by mounting the appropriate CD (`mount -r -t iso9660 /dev/hdc

/cdrom`) and running `rpm` (rpm -i *package_name*). Or you can delete unwanted packages by using `rpm` (rpm -e *package_name*). One good source for descriptions of the various "rpm" packages can be found on CD#1 in the "docu" directory in the form of the file "pkg_English.ps.gz (`gunzip` it and read it with ghostscript (`gs`), `ghostview` or `gv`). Individual packages can also be queried in order to determine their contents with the command `rpm -qlp *package_name*` and for additional information with `rpm -qip *package_name*`; installed packages can be queried by using the command `rpm -q --filesbypkg *package_name*`. You can determine which packages you have installed by using the command `rpm -qa` (for reference purposes, the final installation for this project is listed in Appendix A, as created using `rpm -qa | sort > packagelist.txt` followed by use of `wc`, `split`, and several applications of `paste` in order to generate the multi-column listing). Note that if you select packages during the installation process based on need, many additional packages will be installed because of various dependencies as determined by rpm (e.g., if you want *cron*, *sendmail* has to be installed, but *sendmail*, in turn, requires *procmail*). Finally, you can determine which package an installed file belongs to with the command `rpm -qf *filename*` or what documentation was installed for it with `rpm -qdf *filename*`. Consult the rpm "man" page for additional options.

- Set up a user account
- Specify the 'root' (system administrator username) password

  Note: use strong passwords for both the user and root accounts. See note about passwords in section 1.3 for a suggested method. Also make sure that the boot password for the machine is different from the root password which is to be different from user passwords and which should be different from those on other machines!

- Confirm the settings
  - Until you confirm with 'Yes – install' at this stage, YaST2 has made no changes and you can safely choose to 'Abort Installation.' After beginning the install, YaST2 will require some interaction in order to fulfill CD change requests.

Step 2.3 Configure security at the desktop

Step 2.3.1 'X' windows

At this point SuSE Linux has been installed with password protected accounts, and is also already configured to require the root password to be entered when changing OS levels to the single user mode.

Note: Yet again comes the warning about physical access. It is still (always!?) possible to boot up a system as root without a

> password. For example, when the SuSE yaboot "boot:" prompt appears enter `linux rw init=/bin/sash` (or place the kernel arguments in an append=" " statement in the yaboot.conf script) and you basically have full run of the root file system. What this does is to tell the kernel to use /bin/sash as the "init" script; since the real `init` is not run, "gettys" & c. are not started and thus "login" is never called to provide access control. This can be very useful for repairing corrupted files. Here I have also specified the "sash" shell since it is a fairly complete, statically linked shell which can be an important trait when trying to recover a "sick" system; however, you could specify a different shell (e.g., bash) if you so desire. Perhaps safer (to allow reliable file system checks during system startup), you could log in with the simpler command `linux init=/bin/sash` ; however, if you need to make changes that involve writing to the file system, you will have to do something like: `mount / -n -o remount,rw` , make the desired changes, and then `mount / -n -o remount,ro` before rebooting. The bottom line from a security standpoint: having the Open Firmware password (or abrogating Open Firmware security as discussed above) is essentially equivalent to having system administrator access to the Linux OS for knowledgeable local users.

The computer is standing by, and the graphical 'k' desktop environment (KDE) login is displayed waiting for a user to login. However, it will be noticed that this KDE login GUI, as configured, displays the account names for all to see and it provides a "button" that allows anyone to shutdown the computer.[29] While it is possible to configure the KDE login in a way that will avoid these security concerns, I prefer to use the 'X' display manager (xdm) and associated xlogin GUI in order to make it a non-issue. This is accomplished as outlined below.

> Note: If for some reason there is a delay in accomplishing this step, a quick, partial solution is to edit /etc/inittab as root to change the line 'id:5:initdefault' to read 'id:3:initdefault' and reboot the machine. This will cause the computer to startup in run level 3, which will display a login prompt on a console interface. ('X' can still be started by logging in and issuing the 'startx' command, but at least the usernames aren't exposed.)

1. Login and 'su' to root in an xterm window. (I always make it a habit to login as a normal user and then 'su' to root, when possible. On a single-user system, this helps to limit accidental privilege misuse. When operating in a multi-user environment, this also helps to maintain accountability through the system logging and accounting mechanisms.)

[29] Granted there are many ways to reboot or shutdown a computer, as discussed later in this paper, but there really are people walking around that have a hard time resisting the temptation to "press a button" just to see what will happen, even when there are **big red** signs posted saying to **only** press in an emergency. Then there are dyslexics and the accident prone that if you make it too easy to shutdown or reboot will do so on occasion quite unintentionally.

```
Note: Most if not all configuration steps that follow below
throughout this paper require the operator to have 'root' or
system administration privileges. In general, the reader will not
be reminded of this again.
```

2. Edit /etc/rc.config to specify DISPLAYMANAGER="xdm" (it is also possible to change the default window manager at this time; I prefer DEFAULT_WM= "fvmw2" because of code size and responsiveness (and perhaps a little because I don't like the default "wintel" look of certain other packages).

3. Edit /etc/X11/xdm/Xsession to add the following line after the 'xsetroot' calls:

   xautolock –time 5 –locker xlock –corners 0+-0 –cornerdelay 1 –cornerredelay 10 –cornersize 10 &

   Since the Xsession script is run as the 'user' during the startup of all 'X' sessions after a login, this will invoke a password protected screen saver ("xlock") after 5 minutes ("-time 5") of no keyboard input or if the cursor is placed in the top right corner of the display (the + in "-corners 0+-0"; note the '-' here means that xlock will *never* be called if the cursor is in the bottom left corner). Read the 'man' pages for 'xautolock' for further details on the configuration options.

4. If desired, a simple warning banner can be placed in the /etc/X11/xdm/Xresources script at this point (the default greeting found under SuSE reads "Welcome at linux" [sic]). Computer usage warning banners are a topic that follows in more detail below.

Step 2.3.2 Virtual terminals

In addition to 'X' running on /dev/tty7, SuSE Linux starts six gettys (actually `mingetty`; see /etc/inittab) that provide six additional virtual consoles or terminal devices. These devices can be accessed under SuSE on the Mac PowerBook with a ctrl-command-fn-F1 through F7 key sequence. Note the 'fn' key should be pressed after 'ctrl' and 'command' are pressed and before 'F*x*' is pressed as it is a modifier key (most 'F*x*' keys are otherwise coded on the PowerBook to control functions like screen brightness). Also note that tty1 is where the boot script was executed. Each of these consoles is a potential point of attack, and must be protected as accomplished for 'X' above. While there is a `vlock` program that can be invoked on a terminal (or `vlock –a` to simultaneously lock all six virtual devices), it is probably safer[30] to simply "^D" (logout) of a console when you are done or have to leave the area. In order to protect these access points into the system from accidentally not logging out—since all but one tty is out of sight this has a high probability of occurrence—the `autolog` program can be called from `cron` to close any inactive login sessions.

1. Edit /etc/autolog.conf to add a control statement that reads:

[30] Bugtrak and other archives have records of security concerns with vlock that were posted several years ago. Searches yielded no clues as to the current status, or of any effort whatsoever to resolve the issue.

```
line=tty[1-6] idle=4 warn grace=60 nomail log
```

Here the 4 minute idle time (as determined from the wtmp log entry) plus the 60 second grace period after a warning is issued gives an effective 5 minute idle time that is consistent with the `xautolock` configuration discussed in Step 2.3.1 above.

2. `Touch` /var/log/autolog.

3. Edit /etc/crontab to add a control statement to have cron run autolog every minute (so this plus the 5 min. above really gives us a variable 5- to 6-min. window):

```
*/1 * * * * root /usr/sbin/autolog
```

Step 2.4 Configure 'XF86'

While 'X' does work following the SuSE installation, it is best to configure the software to match the hardware. An example XF86Config file for the "Titanium" can be found at http://ftp.linuxppc.org/kernels/titanium/XF86Config-TiBook-nakashin2 which makes the task easier. (Call it up on another machine for comparison, print it out, or even configure one of the installed Mac OSs to retrieve the file and copy it over by mounting the disk it is on while running under Linux.) I had five changes to make based on the actual SuSE installation: correct the 'truetype' font path (case error); delete the 'tt' font path; and delete the 'GLcore,' 'glx,' and 'drm' module load calls.

```
Once this point is reached we have a stable and "secure"
standalone (non-networked) Linux workstation. While this security
could be compromised by a knowledgeable individual that has
access to the machine, we have established a minimum number of
barriers that will serve both as a deterrence and as a delay
(assuming there is someone coming around with a reasonable
frequency that might discover their access attempts). And,
without forgetting to use our security cable, we are in a state
that gives us a "warm fuzzy feeling" that it is OK to make a "pit
stop" and grab another coffee or soda.
```

Step 3. Additional Security Configuration Steps

On continuing our effort to configure the various security options available under Linux, there are several things yet to do. Some of these will add additional protection measures for a "standalone" machine (remember, we are not yet networked!). Others will be done in anticipation of networking, yet are accomplished before we ever get to network services per se.

Step 3.1 Other "standalone" single-user considerations

Step 3.1.1 Console rebooting

A line in the /etc/rc.config script that purports to control the behavior of the 'ctrl-alt-del' sequence (CONSOLE_SHUTDOWN="reboot"); setting this variable to "ignore" and running /sbin/SuSEconfig will presumably comment out the

ca::ctrlaltdel:/sbin/shutdown… line in /etc/inittab. To be safe both scripts should be so altered manually as a matter of course. However, it appears that this is simply a leftover from adapting scripts originally developed for "wintel" machines. The default behavior under SuSE on a "PPC" is for the 'ctrl-alt-command-delete' sequence to logout the user from an xsession ('ctrl-alt-del' itself has no effect irregardless). Artifacts are one thing, but, unfortunately, it appears that there is no way to avoid the Mac "three-finger salute:" 'ctrl-command-power' will cause an unconditional, forced reboot!

> That word of caution again: the old adage still applies that if someone has physical access to your computer they eventually can do anything they want. In this case, they could choose to power off or reset your computer. Even if the "three-finger salute" could be circumvented, there is still the reset button on the back of the machine, or one could simply remove the battery and unplug the power cord! While it would be nice to be able to keep someone from maliciously shutting down or rebooting your computer, the options generally available for any platform or OS really only help to protect the computer against inadvertent reboots from people who are too much in the habit of pressing 'ctrl-alt-del' as a way to recover from "hung" programs and the like.

Step 3.1.2 Protecting your identity

An "iron-clad" identity (ID) is based on something you are, something you have, and something you know. Implementing an authentication system with anything less than this combination is going to be with an increased risk that the ID can be forged. Typical computer access is granted based solely on **one** of these three items—something you know; this is generally in a combinational form of username and password. Recognizing this limitation or risk, what can be done to minimize our exposure by only using this single ID factor?

First, we should consider if it makes any sense to protect our username. For example, we might be able to reduce the risk of username exposure that could result from distractions during a virtual console login[31] by setting the variable LOGIN_TIMEOUT (say 10) found in the configuration file /etc/login.defs (or as in our changing from KDE environment to xdm). In most organizations of any size, however, usernames are probably fairly trivial to obtain directly or to accurately guess. They are also frequently used as a person's email "handle." (Given an account on a multi-user system, file system structures or world-readable files will also give this information away.) Unless you are able to protect and

[31] Unfortunately, security under SuSE (Linux in general?) is not consistently implemented across all user interfaces. In the example here, LOGIN_TIMEOUT does impact the response of the login program that a user would find on a virtual console, but if, as is most likely, the machine is configured to boot to a run level of 5 and present a xlogin prompt, LOGIN_TIMEOUT has no effect. And this is just one example from the set of parameters that can be found in /etc/login.defs. A better solution, it would seem, would be a complete transition to the paradigm available with pluggable authentication modules (PAM), which are discussed in more detail later in this paper.

use different usernames from published email name lists or other user databases, the risk is very high that your login name can be readily discovered by anyone who cares to try.

More generally, protecting your identity from forgery relies on the use of secret, strong[32] passwords. Even on a single-user machine, threats like "shoulder surfing" or repeated password guessing attempts at the console over an extended period of time can result in the compromise of a login password and thus system. Further exposure can occur if the same password is used on a different computer (shame!). From the field of cryptography comes the answer: one time passwords (OTP). However, while computer login OTP solutions do exist,[33] many are not well developed and none are widely deployed due to the attendant increase in infrastructure costs. And, unless you only plan to logon a very few times, or unless you have an incredible memory (unlike me), implementing OTP will require changing the ID system to use **two** factors: something you know and something you have (be it a list of passwords or a password generator—token—of some type, which will further increase infrastructure costs). While you may later choose to implement some type of two- or even three-factor authentication system, for this project we are relegating ourselves to making do with reusable passwords.

However, even though passwords may be reusable, that does not mean they have to be fixed or permanent. Most if not all computer systems deployed provide some means to change passwords. The tradeoff to be made is between the OTP ideal and the human part of the system. As the frequency of replacement increases the risk decreases until a point is reached where the typical human operator either: (1) finds the burden onerous; or (2), is unable to remember new passwords reliably and will begin to use "aids" (e.g., writing the password down or choosing simple passwords). Further decreasing password life below this inflection point will only serve to increase (probably rapidly) the risk of password compromise. The million-dollar question is, where in the spectrum between a OTP and a permanent password can the human part of the system operate? I am not sure that this question has ever been answered satisfactorily. However, typical policy suggestions for maximum password life range from 30 to 180 days (say 90 days for our purposes; although again note that this is without any quantitative or even semi-quantitative basis).[34]

Unfortunately, it is a human trait to dislike change, and some users will try to revert to their old password. This tendency can be combated in part[35] by having a policy that specifies a minimum number of weeks that a password must be in use before it can be changed again by the *user* (this makes the assumption that, given sufficient time in use, the new password will become acceptable enough to use so that the strong desire to

---

[32] See the introduction to the topic of passwords in Step 1.3 above.

[33] For example, http://www.inner.net/opie as well as various commercial solutions.

[34] Password aging is also important for protecting inactive accounts from compromise. See Step 3.2.4.

[35] Best would be support to prevent password reuse altogether. While this is not possible under the SuSE distribution, the latest versions of PAM do support this feature. Available at http://www.kernel.org/pub/linux/libs/pam/pre/library/Linux-PAM-0.75.tar.gz

change it back will have passed; say 14 days). In an attempt to be more user friendly, policy may also choose to warn users that their password is about to expire (say 7 days). Also, because a user may be gone on travel or what have you on the day their password actually expires, some people consider it acceptable to keep such accounts active for a period of time in order to provide an initial chance to login and change their password when they return without requiring *superuser* action (say 21 days). (An alternative view, that I endorse, is that this additional password exposure cannot be justified; users should change their password with the `passwd` command when they are given a warning. Also, to enable and thus allow users to wait until the system forces them to change their password would seem to only tend to breed a lack of concern and understanding of the importance of a password to security, and will in some measure reduce the reliability of actually having the password changed.)

```
Note: all of these time periods are for illustration purposes
only. The actual periods should be consistent with your password
policy. However, all of the illustrations used here and elsewhere
in this paper for security settings are typical of those found in
the open literature and can be used as a starting point for those
that do not have an existing security policy.
```

While it is possible to enter this data manually into /etc/shadow (shadow passwords are enabled by default under SuSE; see man page for shadow for format information), two UNIX commands are available to make this easier: ‘passwd’ and ‘chage.’ For example, the illustrative policy discussed above might be implemented for a user named “johndoe” as follows (here “johndoe” would be an existing account):

```
passwd –x 90 –n 14 –w 7 –i 21 johndoe
```

or by:

```
chage –m 14 –M 90 –I 21 –W 7 johndoe
```

Other options exist for ‘chage’ (read man page). The configuration file /etc/login.defs also includes global definitions for some of these variables (read man page for login.defs). The commands ‘useradd’ and ‘usermod’ can also be used to specify an expiration date and the inactive time (see man pages).

```
Note: when an account has expired and that person tries to login
under ‘X’, the display (at least under SuSE) will go blank. The
system administrator will have to login under one of the virtual
consoles, set the person’s password to a temporary value, and
stop/start ‘X’.
```

Step 3.1.3 Dealing with forgery attempts in cyberworld

If someone does set about to try and guess your strong password through a console or ‘X’ login prompt (assuming you followed advice similar to that in section 1.3 above, and that they all ready know or are guessing your username as well), and if all is working as planned (don’t forget quality and testing!), the computer will not give them access to any computer resources until it checks their identity (delay); if the ID is determined to be bogus (detect), the login attempt will be rejected (respond).

It is possible for the cyber system to make it tougher on would be identity thieves by inducing human-scale delay into this process. Increased delay generally means there is an increased likelihood of interruption (e.g., of someone observing the activity and calling the guards (detect) who then show up and arrest the perpetrator (respond)). For virtual console login, this can be done by setting the parameters FAIL_DELAY (say 10, as in seconds) in the /etc/login.defs configuration file.

In some cases (such as for this project, where a login denial of service might be a bother but would not be too serious of a situation), a better solution is to simply lockout or prevent further login attempts by that user or until the situation is evaluated and action taken by the system administrator (often called "black listing"). Under the SuSE Linux installation (and most other Linux and some other UN*X versions), most authentication services are implemented using *Pluggable Authentication Modules* (PAM).[36] To effect a black list policy, insert the following two lines in all of the PAM modules[37] found in the /etc/pam.d directory:

```
auth     required       /lib/security/pam_tally.so    no_magic_root
account  required       /lib/security/pam_tally.so    deny=5 no_magic_root
```

Then `touch` and `chmod 600` the log file /var/log/faillog.

By placing these lines in all of the /etc/pam.d configuration scripts, if someone manages to gain access to a shell on your computer because you were careless and stepped away for just a "moment," some protection from password-guessing attacks against other sensitive commands is still available. Related /etc/login.defs configuration entries include LOGIN_RETRIES (say 5) and PASS_CHANGE_TRIES (say 5; although this did not work in the tests conducted, thus requiring the 2nd pam_tally line entry in /etc/pam.d/login).

Step 3.1.4 Another way to deal with forgery attempts in the real world

Since the odds are very-very poor for someone to actually guess a strong password, they will generally have to make many-many attempts (delay). This would imply that if you have and actually use the right system logging tools (detect), the odds are in your favor such that you will notice their attempts and be able to take appropriate action (respond) before they can actually compromise your system. The following suggestions are made for console login protection under SuSE Linux. In /etc/login.defs find the appropriate lines to make the following settings:

---

[36] For reference purposes, the PAM configuration files used in the project reported in this paper are provided in Appendix-B. Further information can be found in the /usr/share/doc/packages/pam directories (e.g., `netscape /usr/share/doc/packages/pam/html/index.html &`).

[37] Actually, because of the way SuSE has implemented the login program, and the way it will be configured below, only the second line should be placed in the /etc/pam.d/login file. Also, due to some (unknown at this time) problems with the SuSE distribution of the xlock and vlock programs, black listing does not work for them, and so these lines might as well be left out of /etc/pam.d/vlock and xlock until the problem is resolved ("pam_tally[1147]: Error opening /var/log/faillog for update").

```
FAILLOG_ENAB      yes
LASTLOG_ENAB      yes
FTMP_FILE         /var/log/btmp
# HUSHLOGIN_FILE
```

In /etc/syslog.conf add the line:

```
 auth,authpriv.*  /var/log/security
```

(Note that while the 'auth' facility is supposedly deprecated, it is still used at least under SuSE Linux; the numerical value of 'auth' and 'authpriv' is actually different, so invoking only one will **not** log information sent to the other facility. Also be warned that although the available documentation indicates that "white" space in syslog.conf can be spaces or tabs, some implementations may have trouble parsing the file unless tabs alone are used.)

As noted previously, however, the login.defs configuration file is for SuSE-unique implementations of certain programs like "login" and is *not* global in scope. In order to get a history of bad login or other password use attempts, add the following line[38] at the end of every PAM module found in the /etc/pam.d directory:

```
session  required       /lib/security/pam_unix.so
```

Make sure the appropriate files exist by doing, for example, a 'touch' on /var/log/lastlog, /var/log/btmp, and /var/log/security (/var/log/faillog was treated in the previous step). Also 'chmod' the permissions on each of them to 600 (in general, only root should have access rights to see the sensitive information that log files contain).

Check these logs with commands like `last`, `lastb`, `faillog`, and `tail /var/log/security`. Console logins will also now display the last time "you" successfully logged in and how many failed login attempts occurred since that time. In a related check, try the command 'last reboot' in order to see if there are any surreptitious machine reboots that may be indicative of someone trying to get control of your computer. A different check is possible by issuing a `cat /proc/device-tree/options/security-#badlogins`. During the initial system startup you can also issue the `printenv` command (before the boot command) and see what value is returned for "security-#badlogins" in order to try and detect password guessing attempts under Open Firmware.

> However, be warned that not all of these logging or reporting mechanisms (e.g., `last` or `lastb`) give a full and accurate summary of all login attempts (good or bad) or even machine reboots (e.g., a `halt` or `reboot` with a '-d' switch will not write the wtmp record). This is especially true of logins made under SuSE through an 'X' interface (xdm/xlogin). The final and best bet seems to be the use of records PAM places in /var/log/security. It is not clear whether this is a SuSE-

[38] The last login module would also be nice to use, but it was not functional under the SuSE distribution (lastlog was unable to resolve symbols pam_sm_authenticate and pam_sm_setcred).

```
specific issue or not. And, of course, if someone has actually
gained access to your computer, the logs may not mean anything.
```

Step 3.1.5 Additional logging considerations

While Step 3.1.4 implemented logging specifically to meet a security threat, it is also worthwhile to add additional system instrumentation (logging) both as a means of supporting general troubleshooting as well as provide information that may help identify attacks. The following are additional suggestions for /etc/syslog.conf (you may want to eliminate redundancies or logging for unimplemented services; also 'touch' and 'chmod' the specified log files as above):

```
#send all "warning"and higher level messages to one file
*.warning                          /var/log/syslog


#send all kernel messages to one log
kern.*                             /var/log/kernel


#print real-time display of security, warning, and kernel
#messages on console
auth,authpriv.*;*.warning;kern.*    |/dev/xconsole


#send emergency messages to all users currently logged on
#system with the 'wall' feature (* in action field)
*.emerg                            *
```

Rather than piping syslog output to 'xconsole,' or in addition to it, you may wish to send these messages to one of the virtual terminal device files (say /dev/tty2).

To activate these changes immediately, I suggest you next execute `/etc/rc.d/syslog stop` and then `/etc/rc.d/syslog start`.

In order to manage these log files that will otherwise grow to very large sizes, it is also suggested that that they be "rotated" (i.e., make a compressed copy and then start over with a new file). Under the default SuSE installation, log rotation is already setup using 'cron'[39] and a 'zip' program.[40] All that has to be done is to make sure the configuration file, /etc/logfiles, accurately reflects the logs you want to rotate (the default file has many logs listed that may or may not reflect your needs—comment them out or delete them as desired). For the log files created above, edit /etc/logfiles to include the following lines:

---

[39] The SuSE version of cron (in what appears to be a modification or adaptation of Paul Vixie's cron) seems to be well suited for the frequent reboots and shutdown periods a machine like the laptop discussed in this paper sees. Other Linux distributions may need to consider other versions like hc-cron.

[40] In contrast, recent RedHat installations are configured to use 'logrotate.'

```
/var/log/security       +4096k      600   root.root
/var/log/syslog         +4096k      600   root.root
/var/log/kernel         +4096k      600   root.root
```

Depending upon your disk space and desires, you may want to change the rotation schedule (here driven by reaching a log size of 4096K bytes). Based upon your own site policy, you may also want to change the record (log copies) retention period. Under SuSE, edit /etc/rc.config and set MAX_DAYS_FOR_LOG_FILES="365" to the desired value. You may also want to periodically make backup copies of these files on removable media.

Step 3.1.6 Legal notices

Most government and many commercial organizations now require warning banners to be displayed at all login (system user access) points. Think of this like the "no trespassing" sign often seen on fences, gates, doors, or other entry points into controlled areas. These warning banners serve two purposes. First, they can act as a deterrent to the simply curious from going where you don't want them (although conceivably it may also serve to infuriate and even motivate certain types of individuals to violate your space). Second and by far foremost, organizations require the banner in order to give them a "leg up" in prosecuting unauthorized users. Warning banners should be developed under consultation with your legal counsel.[41] However, they typically contain statements about what is considered to be proper use, about system monitoring, and about privacy expectations (read none). For our machine in its present state, warning banners need to appear at console and xlogin prompts. Console logins are easy to take care of: put the desired contents of the warning banner in /etc/issue (you may also choose to put them in /etc/motd, but that is not necessary for our purposes here). For xlogin, a little more work is required. I chose to place the contents of the warning banner in the file /etc/X11/xdm/legalnotice.rtf. I then edited the file /etc/X11/xdm/Xsession and placed the following lines immediately before the 'xautolock' call (see Step 2.3):

```
xmessage -center -file /etc/X11/xdm/legalnotice.rtf \
-buttons "consent:3,          :4,reject:6" -default reject
XMANS=$?
if [ $XMANS -ne 3 ]; then
kill -9 $$
fi
```

The spaces (actually I used 61 of them) and ':4' cause 'X' to display an unlabeled button between the consent and reject buttons just to arrange the message window to suit me. The message appears after login and will restart xdm (and thus give a new xlogin prompt) unless the warning banner is consented to (as indicated by the action of pressing the 'consent' button; the separation of this button from the 'reject' button also helps to avoid

[41] http://ciac.llnl.gov/ciac/bulletins/j-043.shtml includes banners for U.S. Department of Energy systems that could provide a good starting point. The statutory banner from the TITAN security package is another possibility (http://www.fish.com/titan/).

the "I meant to press the 'reject' button" scenario). You could, perhaps, improve on this script by repeating the message if the center button was pressed. You might also "play" around with its location in Xsession in order to have it appear before the login prompt if so desired (but then how do you know that it was not someone else who first consented with the warning but did not log on?).

Step 3.2 Dealing with multi-user concerns

In a multi-user (and network) environment, every account on a system is a potential intrusion point. Compromise of any account will increase the possibility that other accounts could be subverted, data obtained, and the system taken over. (In networked environments we could add that other systems on the network may be at increased risk for compromise.) This suggests that additional measures (beyond the single-user steps above) need to be taken to protect all user accounts.

Step 3.2.1 Additional identity (password) protection

Passwords have to be stored in order for a system to perform authentication. These passwords must be encrypted and should be retrievable only by the system (delay). On Linux and most UN*X platforms this is implemented by using "shadow" passwords—and it is the method employed by the SuSE installation. While the default installation is probably correct, it doesn't hurt to verify 'shadow' is, in fact, being used (the easiest is simply to verify the presence and use of /etc/shadow to store encrypted passwords). It is also an easy matter to verify the appropriate ownership on the files. The file /etc/passwd should be owned by root and belong to the group root. The file /etc/shadow should be owned by root and belong to the group shadow (at least under SuSE; some installations may have /etc/shadow as belonging to root).

In terms of file protections, many UN*X security guides state that the permissions on /etc/passwd should be 444 and that on /etc/shadow should be 400. **Be warned: implementing these file protection levels under SuSE Linux (and maybe others) will break some programs**, including the screen locking program enabled above. Rather, make sure permissions do not allow the "world" any permissions on /etc/shadow, and check that the entries in /etc/group does not inadvertently give permissions to others. Only the file owner (root) should ever have write permissions for either /etc/passwd (say 644) or /etc/shadow (say 640).

However, if a clever attacker somehow manages to obtain a copy of /etc/shadow, the default use of the "crypt" algorithm for encrypting passwords makes it easy with available tools to perform a fairly rapid "brute-force" dictionary attack. Additional delay can be introduced by employing a much harder (more computationally intensive) algorithm. Fortunately PAM includes support for use of the MD5 algorithm for password "encryption" (really a hash). Configuring the system to use MD5 is accomplished by

adding the option "md5" to every "password" rule in the PAM configuration files located in /etc/pam.d.[42]

Unfortunately it is also necessary to add a little quality control (Q) to make sure that all users are using strong passwords (delay). (It was assumed above that a single user/owner would be responsible in following appropriate guidance, but will all of your users abide by such a policy as a matter of course?) The actions to be taken are two fold. In /etc/login.defs find the appropriate lines and make or verify the following settings:

```
PASS_MIN_LEN            8
OBSCURE_CHECKS_ENAB     yes
PASS_ALWAYS_WARN        yes
```

In /etc/rc.config set:

```
PASSWD_USE_CRACKLIB     yes
```

Then run 'SuSEconfig' which will update all "password" rules in the PAM configuration files with the "use_cracklib" option. (Don't ask me why the MD5 option is not set up this way! Maybe in a future release.) These configuration settings are also used by an apparently SuSE-unique PAM module pam_pwcheck.so (the current release of PAM includes the module pam_cracklib.so that performs similar functions).

Besides the 'Q' added to provide assurance that the delay associated with strong passwords as first introduced in Step 3.1 is maintained, we must also revisit one other consideration from that step: password aging. The addition of other users to the system has added another exposure to our passwords. While we have added delay (as above) to provide protection, is our choice for aging (respond) appropriate? We can hope that our instrumentation (e.g., logging, process monitoring, computational loads, etc.) would allow us to detect attempts to "sniff" passwords or gain access to /etc/shadow, but the reality is that if someone is cleaver enough to compromise an account they have a good chance of being capable enough to cover their tracks. This may be a case where you want to assume /etc/shadow is under a dictionary attack ("detect"). Is it then possible to select a maximum password age (respond) that will give favorable odds at thwarting this attack? (I don't have the answer and can still only refer you to the typical policy values given in Step 3.1.2 above. What odds are you comfortable with?)

Step 3.2.2 Tightening authorization with unique user groups

By belonging to a group(s), a user has access to all of the programs and data files belonging to the same group as an account as long as the group permissions on these files permits the access. Thus group access needs to be minimally defined in order to help protect (delay) against an account compromise. For example, only users with full system administrator privileges should be assigned to group 'root.' While the default installation is probably correct, it doesn't hurt to verify the appropriate file protection: /etc/group should be owned by root, belong to the group root, and have permissions no more "open"

[42] In addition to Appendix-B, SuSE provides examples in /usr/share/doc/packages/pam/md5.config

than 644. In addition, all users should be defined in a separate and unique group. By doing this, only the user will have access to their files by default (delay). It is suggested that the group numeric and name should be the same as the user's numeric and name for ease of administration.

Step 3.2.3 Tightening authorization by restricting use policies

Each and every account on a system has a potential—some more and some less—of being exploited. One way to reduce this potential is to place restrictions on what a uses a particular account is authorized for. PAM provides an interface that can be used for such purposes (although not so used in this paper). Several standard PAM modules are available along these lines: pam_time.so and the corresponding configuration file /etc/security/time.conf might be called to restrict the times users can connect (e.g., to normal working hours; although note that the module did not function correctly in the limited tests conducted for this paper); and pam_limits.so with /etc/security/limits.conf can be used to place resource limits on a user.

Step 3.2.4 Tightening authorization by eliminating unwanted "users"

As noted above, user accounts on a system are potential points of exploit. The best way to minimize this potential is to simply eliminate the accounts that are not wanted. Files that are associated with these accounts should also be removed or have ownership changed (especially executables that might be exploited; others simply to ease system administration). This effort is two-fold: initially delete "users" that are part of the default installation; and, in the longer term, delete users that at one time were authorized to use resources on the computer, but are so no longer.

In terms of initial "users," I prefer to manually edit /etc/passwd, /etc/shadow, /etc/gshadow, and /etc/group[43] (make a backup first!). For each user determine which groups they belong to, and which files they are associated with (e.g., `find / -user *username*` and `find / -group *groupname*`). That information, along with some understanding of the system will allow files, groups and users to be safely removed. (In my particular SuSE installation, 39 "users" and associated groups were eliminated! Only the users root, bin, daemon, lp, man, at, nobody, and appropriate local usernames were left in /etc/passwd and /etc/shadow. The entries in /etc/group left were: root, bin, daemon, tty, disk, lp, kmem, mail, uucp, shadow, dialout, audio, at, video, nogroup, and appropriate local user groups.)

While eliminating the initial but unwanted user accounts takes considerable time, in the long term, managing other users will likely become the bigger problem (assuming you actually have multiple user—real people—accounts on your system). These users include those no longer in your organization or company, those on extended absences, and those that have been reassigned to different tasks and no longer require access. That inactive accounts such as those belonging to people on extended absence or reassignment can be a

[43] You can use `grpck` to verify the integrity of group files when you are through.

risk by their existence is well attested to in records of incidents.[44] But insiders in general, and one can only presume that people who have left a company in particular (voluntarily or not), account for roughly one-half of the threat[45] to information technology systems. The key is to somehow work with management and personnel to develop a procedure that will alert system administrators immediately of such changes in status. It would then be incumbent upon system administrators to quickly take appropriate action upon receipt of such notifications. (In general, account deletion should be specified, although account deactivation—e.g., `passwd –l *username*` or otherwise mangle the /etc/shadow password field of an account record, say with =NP= or *LOCK*—is a possibility for cases of extended absence.) While any such procedure will be very specific to your organization and is thus not appropriate to discuss here, it should be pointed out that password aging (see Step 3.1.2) will help to some small degree in reducing this risk (especially for inactive accounts provided the maximum password lifetime is reasonably small).

Step 3.2.5 Noshell

While it might be theoretically possible to reduce system users down to two, 'root' and one local user, most UN*X computers are configured to use additional system process accounts, such as "bin" and "daemon." These system accounts are not intended to be a standard user account—never used for login purposes—and thus should never be configured with a usable password. In addition, to further protect against the possibility of misuse of these accounts—such as may occur from a buffer-overflow exploit—a functional shell (e.g., /bin/bash) must not be started by the system. Often this is accomplished by making an entry such as /bin/false or /dev/null in the shell field of the process account record in /etc/passwd. Of course, if an attacker can establish a real shell program at or linked to such a path, they can exploit this entry.

A better solution is to point to a compiled program (not a shell script!) that will write to 'syslog' if anyone tries to invoke the shell (which should never happen—delay). This could be very useful in identifying exploit attempts for those of you who routinely evaluate your logs. If an entry is found indicating this program was invoked (detect), the system logs should be further scrutinized in order to identify and fix the problem (respond), as well as to look for other unauthorized activity. An example of such a program is 'noshell' from the TITAN security package:[46]

```
/* This tool suite was written by and is copyrighted by Brad Powell, Matt */
/* Archibald, and Dan Farmer 1999                                          */
/* The copyright holder disclaims all responsibility or liability with     */
/* respect to its usage or its effect upon hardware or computer            */
/* systems, and maintains copyright as set out in the "LICENSE"            */
```

[44] For example, Stoll, Cliff, 1989, *The Cuckoo's Egg*, Simon & Schuster, Inc., New York.

[45] For example, Dalton, G., 1998, "Acceptable Risks," August 31, Copyright 1999, CMP Media Inc. (article found at http://www.informationweek.com/shared/printArticle?article=infoweek/698/98prrsk.htm&pub=iwk) places insider threat level at 58% while Rapalus, P., 2000, "Computer Security Institute press release," March 22 (found at: http://www.gocsi.com/prelea_000321.htm) suggests insiders are 38% of the problem.

[46] http://www.fish.com/titan/

```
/* document which accompanies distribution.                                */
/* Titan version 3.0.7 April 25 11:21:02 PDT 1999                          */

#include <syslog.h>
#include <unistd.h>
#include <signal.h>

#ifndef lint
static char sccsid[] = "@(#)noshell.c bpowell- Titan 3.0.7 5/11/99";
#endif /* not lint */

#define perrorexit(s) { perror(s); exit(1); }

main (int argc, char **argv)
{
extern char **environ;
int     i;

/*
 * Fix the environment.
 */
if (environ != 0)
    environ[0] = 0;

/*
 * Fix the signals.
 */
for (i = 1; i < NSIG; i++)
    (void) signal(i, SIG_IGN);

/*
 * Log the login attempt.
 */
openlog(argv[0], LOG_PID, LOG_AUTH);
syslog(LOG_AUTH, "Titan warning: user %d login from a disabled shell", getuid());
exit(0);
}
```

Step 3.2.6 System accounts

In UN*X, the system account known as 'root' requires the most protection as it provides essentially total access to resources on a system. Thus this account needs to be tightly controlled, and giving the root password to more than one person is strongly discouraged. Distributing a password increases its exposure and thus its risk of compromise. (For recovery purposes, having a sealed record of the password locked in a safe under appropriate access controls *may* be acceptable.) If multiple people must be granted system administrator (root) privileges, it is strongly suggested that 'z' accounts be used. A 'z' account is a system account (uid/gid of 0/1) with a 'z' appended to the end of a user name (e.g., *usernamez* or *username_z*). This will not reduce the risk of having a "root" account password inadvertently exposed, as just as many copies of an administrator-level password would be distributed as in the case of simply passing out the real root password. If a copy of the /etc/shadow file is somehow obtained by a threat agent, having an additional account may actually reduce the time required to compromise the system. However, the additional accountability provided by maintaining separate administrator accounts is *thought* to reduce the overall security risk of a system.

Step 3.3 In further anticipation of networking

At this point we are ready to think of security-related configurations that are specific to network-connectivity concerns. All of the single- and multi-user security steps taken in Step 3.1 and Step 3.2 above (with the exception of 3.1.1) are very important for the security posture of a networked system in terms of general *user* authentication and authorization (A&A). The next step before connecting is to actually configure network services and establish *system* and additional *user* A&A controls. Just as in the fact that every user account provides a potential intrusion point, so every service and authorization provided for network connectivity is a potential vulnerability that might be exploited. Measures must be taken to protect any such exposures.

> "Securing workstations is primarily a process of eliminating network services… A typical workstation operates on the client end of client-server network communications. There is almost never a need to provide a service."[47]

Securing a workstation includes eliminating internet daemon services like the Berkeley "r" programs (rsh and rlogin) and ftpd. It also includes the elimination of run-time programs like DNS, mail servers, print servers, NFS, and http servers. If a system is installed following after Appendix A, most of these have already been dealt with. However, the steps below will help assure network services are eliminated whichever installation option is selected.

Step 3.3.1 Stop Internet daemon services

The desired configuration for the workstation this paper is concerned with has no need for the services provided by the Internet daemon.

The vulnerabilities associated with these services are eliminated by simply making them unavailable. This is accomplished in several steps as follows.

1. Disable (comment out) all services in /etc/inetd.conf. In the standard SuSE installation, the services to be disabled are: time, ftp, telnet, shell, login, talk, ntalk, pop3, finger, http-rman, and swat.

2. If inetd was installed, remove it with the command `rpm –e inetd`. (A different method—the hard way—is to remove the files by hand; but be warned, the rpm database will not be up to date. First, run `/etc/init.d/inetd stop` and `/bin/rm /etc/init.d/inetd /etc/init.d/rc[0-6].d/*inetd`; the script links could also be removed in this case under SuSE with the command `insserv –r inetd`. In a similar manner remove rusersd and rwhod with their associated rc?.d links. Another alternative is to set START_INETD="no" in /etc/rc.config and run `SuSEconfig`; this will not, however, remove the files.)

[47] The SANS Institute, 2000, *Securing Linux Step-by-step*, version 1.0, p. 18.

3. Finally, any installed inet associated daemons should be removed (`rpm –e *package_name*`), where the packages to remove (depending on the actual install configuration used) may include nkitserv, inetd, proftpd, wuftpd, and ftpd. (A different method—the hard way—is to remove the files by hand; but be warned, the rpm database will not be up to date. First, locate the desired files (e.g., use `which`, `whereis`, and the `find` commands) and delete them (`rm`) in order to prevent inadvertent use (or unauthorized startup). This includes all of the "in.*" files in /usr/sbin and /usr/sbin/vboxd, and any others listed in /etc/inetd.conf installed.)

Step 3.3.2 Disallow Internet daemon connections

Although strictly unnecessary without any Internet services running, it is possible to take several simple steps to—read "no excuse not to"—remove any connectivity authorizations that may exist. This will provide a second layer of protection or "defense in depth" in case the services are somehow installed and started (maliciously or not[48]).

1. To provide protection for TCP wrapped services (SuSE installs by default), remove *all* authorization lines in /etc/hosts.allow and place the line "ALL : ALL" as the sole configuration entry in /etc/hosts.deny.

2. Run `tcpdchk –v` to verify that it is the /etc/inetd.conf file that is in use (Step 3.3.1) and that the configuration is to deny all daemons, clients and access. This should look something like:

```
linux: # tcpdchk –v
Using network configuration file: /etc/inetd.conf

>>> Rule /etc/hosts.deny line 3:
daemons: ALL
clients: ALL
access: denied
```

3. To provide protection from access to Berkeley "r" programs, make sure that no /etc/hosts.equiv or .rhosts files exist on the system (use `ls` and `find` to locate them and `rm` to remove them).

Step 3.3.3 Stop runtime-network daemon services

The desired configuration for the workstation that this paper is concerned with also has no need for runtime-network services.

The vulnerabilities associated with these services are eliminated by simply making them unavailable. This is accomplished in several steps as follows.

1. The first thing to do is remove the following packages, if they were part of a default installation (use `rpm –e *package_name*): lprold, lprng, plp, portmap, nfs-

[48] For example, misuse of `rpm` or other software update program may inadvertently reinstall software that had been removed for security reasons.

server, nfsutils; other possibilities include samba, thttpd, and apache. (A different method—the hard way—is to remove the files by hand; but be warned, the rpm database will not be up to date. First, to shutdown the portmapper and line printer daemons, the easiest method is to set START_PORTMAP="no" and START_LPD="no" in /etc/rc.config and run `SuSEconfig`. Or you can stop the services using the scripts in /etc/init.d and remove the start and stop scripts in /etc/init.d/rc[0-6].d as described above for the inet daemon above. The actual daemons and associated files should be deleted as well. The files related to the standard installation to remove include: /usr/sbin/rpc*; /usr/sbin/pmap*; /usr/bin/rpc*; and /sbin/portmap.)

2. Next, while it would be nice to remove sendmail as this project has no need to send mail, let alone receive it, the SuSE cron package has sendmail dependencies, and so it must be kept. However, make sure that sendmail does not startup in the daemon mode by editing /etc/rc.config and make sure that SMTP="no". I would also suggest that any rc scripts are removed as described above.

3. Finally, while we want to have 'X' available on the local terminal, we do not want it to act as an X server for other network devices; 'X' has proven to be a very vulnerable service (tunnel it over ssh if you really need to). To secure the 'X' server, edit the file /etc/X11/xdm/Xservers and add "-nolisten tcp" to the line ":0 local /usr/X11R6/bin/X :0 vt07" so that it reads:

```
:0 local /usr/X11R6/bin/X :0 vt07 -nolisten tcp
```

After completing these steps run the commands `netstat –at` and `lsof –i +M` to check for listening network services. If you have followed the steps outlined above successfully, you will find that **no** services are listening on any ports.

Step 3.3.4 Disallow network connections

While Internet daemons are generally configured with TCPwrapper controls, PAM supports access control based on both the identity of the network user *and* system (here limiting all logins to local). Edit the file /etc/security/access.conf so that the configuration lines read:

```
+:ALL:LOCAL
-:ALL:ALL
```

and place the command line:

```
account     required    /lib/security/pam_access.so
```

in the xlogin and xdm PAM configuration files. If access takes place locally to the machine (LOCAL), the first line in the configuration enables login (+) for all valid users (ALL). If access takes place over the network, the first line does not match, but the second one does (the 2nd ALL), but login is disabled (-) for all users (the 1st ALL).

It is also considered good practice to restrict root logins to local consoles. This is accomplished with the command line:

```
auth  required    /lib/security/pam_securetty.so
```

that is also placed in the xlogin and xdm PAM configuration files, and by placing a list of the authorized terminals in the /etc/securetty file. If you want 'root' to be able to login at the 'X' and virtual consoles, the /etc/securetty file will have seven lines or entries; one for the 'X' display (:0), and one for each virtual console (tty1 through tty6).

Step 4. Configuring the network interface

Step 4.1 Kernel configuration

The first step in establishing a network connection is to configure the kernel to start the network interface card (NIC). For the Titanium PowerBook G4, edit the /etc/modules.conf and add the line:

```
alias    eth0    gmac
```

When the machine has been booted with this configuration set, you should be able to determine that it has started correctly with the command `cat /proc/net/dev`.

Step 4.2 Network configuration

In order to configure the network software interface, some information must be gathered (here we are configuring a fixed IP address machine). For illustration purposes ***ONLY***, consider the following table.

parameter	value (for illustration purposes ONLY)
hostname	aqua
domain name	waterworks.com
IP address	192.14.64.11
gateway address	192.14.64.254
netmask	255.255.255.0
name server address	192.14.64.1

To configure the system with this data, three choices exist under SuSE: YaST, edit /etc/rc.config and run SuSEconfig, or edit the configuration files manually. YaST is a menu driven system configuration tool; after starting it, select "System administration," "Network configuration," then "Network base configuration." The 'tab' and 'Fx' keys are used to move around through the submenus found at that level to enter in the data. YaST actually writes the data to rc.config and runs SuSEconfig for you. If you wish to edit (or inspect) the rc.config file, look for the following parameters: NETCONFIG, IPADDR, NETDEV, IFCONFIG, and FQHOSTNAME. To verify (or edit manually) the network configuration files manually, consider the file listings that follow below. If the manual method is used, I suggest that you first stop routing and networking (`/etc/rc.d/route stop`

and `/etc/rc.d/network stop`), and then restart them once the configuration is complete (in Step 4.4 below).

/etc/hosts

```
127.0.0.1          localhost
192.14.64.11       aqua.waterworks.com     aqua
```

/etc/resolv.conf

```
nameserver         192.14.64.1
search             waterworks.com
```

/etc/**HOSTNAME**

```
aqua
```

/etc/route.conf

```
127.0.0.0          0.0.0.0     255.255.255.0     lo
192.14.64.11       0.0.0.0     255.255.255.0     eth0
default     134.253.14.254
```

It is now time to connect the Ethernet interface to your local network and restart the network service!

Step 4.3 Monitoring the interface

Once we provide Ethernet connectivity, can reach the computer and its resources through two access points: the terminal (here meant to include all virtual consoles as well) or the network. Terminal access controls (login, xdm, vlock, xlock) include instrumentation (e.g., pam_unix.so and associated entries in /var/log/security) that will let us know when someone is "jiggling the door knob" to try and get in (i.e., password guessing attempts). Instrumentation is also needed for the network interface—we would like to know when someone was "shaking the fence" to try to find a hole to get in. The instrumentation explored here is PortSentry, which is useful to monitor for such things as connection attempts (but we have no network daemons running, so why would someone try?) or port scans (what are they looking for?). PortSentry can be configured to monitor TCP and UDP protocols. Granted that there are other ways to shake the fence (protocols), this will catch the vast majority of the intrusion problem observed in networked systems today. Note that the intent here is to use PortSentry for its logging capability through syslog(). However, PortSentry is also of interest for its ability to actively respond to attacks (through TCPwrapper, `route`, and packet filtering). While not of immediate use for the planned use of this machine—network testing and security monitoring—conceivably the purpose could change and this at-hand functionality could be put to good use.

Step 4.3.1 Obtain the software

Warning: many organizations have security policies that forbid downloading software from external organizations. (This was alluded to in Step 1.1 when discussing the Linux distribution selection.) However, for the purposes of this paper it will be

assumed that your organization has a more moderate stance that allows downloads provided file source and integrity can be verified to some extent. PortSentry is available from http://www.psionic.com/abacus/portsentry. You should obtain a copy of the binary (portsentry-1.1.tar.gz), binary signature (portsentry-1.1.tar.gz.sig), and public key associated with the signature (crowland.new.asc); place them in a suitable directory (I used /usr/local/src/portsentry). (If you *have* to use this computer—the one you are presently configuring—you will have to start the network and routing services as described in Step 4.4 at this point and connect the Ethernet interface to your LAN.)

Step 4.3.2 Verify the file integrity

Before the updates are installed they should be checked to verify their integrity.[49] First, import the key with the command `gpg –import crowland.new.asc`. Then, check the signature by running the command `gpg –verify portsentry-1.1.tar.gz.sig`. The output from gpg should be something like:

```
gpg: Signature made Wed 11 Jul 2001 12:07:59 PM MDT using DSA key
ID BFB08FDA
gpg: Good signature from "Craig H. Rowland
<crowland@psionic.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs
to the owner.
gpg: Fingerprint: AFA2 D8A6 272B 4432 2BF4  FF43 F8A9 9B91 BFB0
8FDA
```

From this you know that the triplet of files belong to each other and that the binary was not somehow replaced or corrupted in transit. However, you have no way of knowing if the key is really the public key of crowland@psionic.com (and thus the warning from gpg). The best thing to do here would be by an "out of band" means—email, telephone, or in person—independently verify the MD5 fingerprint;[50] this will give a higher level of assurance that the server was not spoofed and trojaned software downloaded.

Step 4.3.3 Install the software

The first thing to do when installing PortSentry is to edit the configuration files. The settings for the installation described in this paper follow below.

In portsentry_config.h find and edit the syslog() facility line to read:

```
#define SYSLOG_FACILITY LOG_LOCAL3
```

edit /etc/syslog.conf to add the command:

```
local3.*    -/var/log/portsentry
```

---

[49] For a more general discussion on file signature verification, see http://the.wiretapped.net/security/cryptography/ssh/SSH/HOWTO-CHECK-SIGNATURES

[50] Some software distribution sites use, as an alternative, one or more of the common PGP keyservers (e.g., http://wwwkeys.pgp.net).

and `touch /var/log/portsentry`. This will send the output of PortSentry to a unique log file in order to facilitate analysis of connection or probing attempts.

In order to configure PortSentry to be in a logging mode only (generate no traffic itself nor take any active action against perceived threats), change the appropriate lines in portsentry.conf to read:

```
RESOLVE_HOST = "0"
BLOCK_UDP="0"
BLOCK_TCP="0"
#KILL_HOSTS_DENY="ALL: $TARGET$"
```

Next run `make linux` and `make install`. PortSentry is now ready to use. For the installation described herein, I have configured PortSentry to start automatically in an advanced stealth mode for both TCP and UDP by using the "rc" scripts that run during system startup; a copy of the /etc/init.d/portsentry script is provided in Appendix C. After the script is generated, run the command `insserv /etc/init.d/portsentry` to create the appropriate rc?.d links, and edit /etc/rc.config to add the line:

```
START_PORTSENTRY="yes"
```

Step 4.4 Connect the interface

The system is now in an acceptable state to connect the Ethernet interface to your local network and start the related control and monitoring programs (`/etc/init.d/network start`, `/etc/init.d/portsentry start` and '/etc/init.d/route start`)! This will likely be necessary in order to complete portions of the remaining steps (at least unless you use a different system for the required downloads and "burn" a CD with the software—a more secure way if you can). However, such connectivity should be minimized as a matter of principle until the remaining steps are completed.

Step 5. Software updates

Warning: again note that many organizations have security policies that forbid downloading software from external organizations. The illustration here will be based on an update of the SuSE distribution installed above, and will not address except in passing other variations to this problem (e.g., downloading the latest kernel from linuxppc.org).

Step 5.1 Find and download security-related updates

Even though you just purchased the latest Linux release available on CD, that does not mean that you have the latest version of the software. SuSE updates can be found and retrieved through the web at http://www.suse.com/us/support/download/updates/index.html by selecting the correct distribution (version and platform); in our case the correct choice is the 7.1 PPC release. This will bring up a listing of all of the update rpm packages available, including source. Of particular interest here are those updates that are red-tagged "Security-Update!" Compare this list against the installed packages (`rpm –qa`),

and download the needed updates. (The standard location is /usr/src/packages/RPMS/ppc for the binaries and /usr/src/packages/SRPMS for the source.)

It should be noted that if you are interested in the topic of updates in general, you cannot rely solely on the SuSE site. A **small** sampling of other but important sites include: www.kernel.org; www.linux.org; www.penguinppc.org; www.gnu.org; and www.x.org. In addition, many of the individual packages installed under SuSE also have their own home pages. One good way to help understand this complex web of sources that goes into a Linux distribution is to check out the information available at www.linuxfromscratch.org. However, before installing software from any of these other (non-SuSE) sites, you would be well advised to make sure that they are compatible (i.e., SuSE may have made changes to the base code such that they are no longer compatible with the standard open-source-tree distributions and development activities; it would be nice if SuSE had this documented and readily available).

Step 5.2 Verify the updates

Before the updates are installed they should be checked to verify their integrity. This requires the use of "gpg" and the public key tied to the SuSE build; try `gpg –list-public-keys` to verify that "<build@suse.de>" shows up, or else import the copy of the public key found in the root directory of CD#1 into gpg. (The public key is also available on the SuSE web site; the more paranoid could compare the pre-installed key with the CD key with the web key in order to increase confidence that they really have a valid SuSE key.) Next change your working directory to the location where the binaries were downloaded. Then run `rpm –v –checksig *.rpm` to verify all of the files that were retrieved are intact as created by SuSE, and thus are unlikely to be Trojans. (Note: you may see warnings about an untrusted key, just as in Step 4.3.2 above, but the files should all verify. If you want to eliminate the warning, you can sign the SuSE key yourself—`gpg –gen-key`, `gpg –edit-key build@suse.de`, `trust`, `4`—say on the basis of comparing the CD and web site keys.)

Step 5.3 Install the updates

The now-verified files can be installed by running the command `rpm –U *.rpm`.

Step 6. "Material" controls

The security configurations that have been discussed up to this point fit principally into the category of protection elements—functions designed to keep unauthorized personnel away from protected assets (information or information processing capabilities). They do little or nothing to protect said assets from users who have access granted to them by the system; whether they enter with valid or forged credentials, the system still thinks of them as authorized users (it knows no difference). The general class of techniques used to protect assets from people that have access to them are called material controls; they include functions like providing independence of control from controlled (think separation of system administrator from users), two-person rules, auditing, and inventorying.

Step 6.1 Protecting the installed software base from unauthorized change

The two *very much* related functions of auditing and inventorying are introduced here. (In this context, auditing is *not* referring to functions like reviewing access logs generated by PAM library calls discussed above.) The packages introduced provide a detect function for the type of attacks that require system files to be changed, added or deleted (e.g., the installation of back doors, Trojans, or scripts, in order to gain certain user privileges or exploit system resources). The packages introduced **do not**, however, provide any capability to detect other classes of attack, such as data theft by copying (for a start, this would require some form of a complete file access journaling program).

Step 6.1.1 System software inventory

What is there to inventory? The installed software, of course. While there may be some software on the system that did not originate with the SuSE distribution (e.g., PortSentry and user data), by far the majority of the system software, at least, was installed using rpm packages. Because of the design of rpm—a program that creates a database of all software it installs as an aid to manage said software—the system can be inventoried to determine if the original software is still installed. The database is a list of files (including directory structures and links) along with information that includes (but is not limited to) such things as file size, MD5 sum, permissions, type, owner, and group. A system software inventory is conducted by running `rpm –Va`. The output at this point in our setup will run somewhere around a hundred lines, and look something like:

```
…
S.5....T c /etc/hosts
S.5....T c /etc/hosts.deny
S.5....T c /etc/inetd.conf
S.5....T c /etc/pam.d/other
S.5....T c /etc/securetty
…
```

As shown, the output format is a string of 8 characters, a possible ‘c’ denoting a configuration file, and the file name. Missing files are simply reported as “missing” along with their file name. Only files that failed in some way are output. A ‘.’ in the output indicates one of the eight tests passed. Other characters in the string indicate which test or tests failed, as follows:

S	file size
M	mode (permissions or file type)
5	MD5 sum
D	device
L	symlink
U	user
G	group
T	mtime

I suggest that the first time you use rpm to check the inventory of your system software, you save a copy (say as /root/*run_date*.rpmverify.report) to check against (`diff`) in future runs. The first time you run the report you should also take the time to understand the test failures rpm is complaining about (the majority if not all of the cases will be due to changes in configuration files).

6.1.2 System-wide data audit

In the context here, data audit refers to data of any and all forms (programs, databases, etc.) as stored on the computers mounted, read-write mass storage devices (e.g., hard drives) that are being audited to identify unauthorized changes. By far the most popular program of this type is probably "Tripwire," which was developed at Purdue in 1992. The software was eventually commercialized,[51] and is available for a number of platforms. Recently, an updated, open-source Tripwire development project[52] was launched that specifically targets the Linux community. Unfortunately the latest source version (2.3.1-2 as of this writing, release date of 03MAR01) will not compile on the PPC platform (it complains when trying to compile the cryptographic libraries that both BIG_ENDIAN and LITTLE_ENDIAN are defined), and we are left to use the Purdue version 1.2 (patch level 2) as supplied with the SuSE distribution (it is in the installed base listed in Appendix A). However, from a pure functionality standpoint, the biggest difference between version 1 and 2 was added support to cryptographically protect the database that Tripwire produces as a baseline for the file system; some form of protection is needed to prevent tampering that may occur as part of an attack to cover "tracks" that may otherwise have been left. Solutions to this problem do exist for version 1: store the database on write-protected removable media or use a program such as gpg to encrypt the database files (both of which are discussed below).

The first step is to create a configuration file, /var/adm/tripwire/tw.config, that will tell Tripwire which files to monitor. The details needed for creating this file can be found in the tw.config man page; however, an example file from this project is provided for reference purposes in Appendix D (the objective being to eliminate hits by Tripwire from simply rebooting the machine while covering a significant fraction of the system file space). Second, run the command `/var/adm/tripwire/bin/tripwire –initialize` in order to create the initial database (about a five minute process for the hardware/software used in this paper), which will appear as /var/adm/tripwire/bin/databases/tw.db_*yoursystemname*. Move this database file to the directory /var/adm/tripwire/db. You are now ready to begin monitoring your system for changes. This is done simply by running the command `/var/adm/tripwire/bin/tripwire` (or `/var/adm/tripwire/bin/tripwire > *diff.file.name*`). The output on the display will show something similar to:

```
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
```

[51] http://www.tripwire.com

[52] Source available at http://sourceforge.net/projects/tripwire and precompiled binaries for several platforms are available at http://www.tripwire.org

```
### Phase 4:   Searching for inconsistencies
###
###                     Total files scanned:            62616
###                           Files added:              0
###                           Files deleted:            0
###                           Files changed:            59278
###
###                     After applying rules:
###                           Changes discarded:        59278
###                           Changes remaining:        0
###
### Phase 5:   Generating observed/expected pairs for changed files
###
```

Here the "Changes remaining: 0" means that, under the conditions of the configuration file, no reportable changes were found (and *diff.file.name* will be empty).

In terms of protecting the databases, I suggest that you (1) encrypt them, removing the originals except when conducting an integrity test, and (2) copy the encrypted files to a removable media for backup purposes.[53] The commands `gpg –c tw.config` and `gpg –c tw.db_*yoursystemname*` from within the appropriate directories will create encrypted files with a .gpg extension—you will be asked to supply a pass phrase[54]—while to un-encrypt the files use `gpg tw.config.gpg` and `gpg tw.db_*yoursystemname*.gpg` with the same pass phrase when prompted. Before encrypting, you may want to compress the files with a program such as "bzip2;" this will save considerable space that could be important when saving the files on removable media.

Step 6.1.3 Data backup for recovery

In general, information security is concerned with many types of threats, including hardware failures, supporting utility failure, natural disasters, human errors, and neighboring hazards. For most of these threats, backup and recovery are key elements in reducing the risks associated with the potential data loss that could occur. For the threat specifically addressed in this step—tampering—the same holds true. But why recovery and not response? While it may be possible that inventory and audit (I&A) activities could discover an attack in progress and so be able to interrupt the activities, it is not clear that I&A can discriminate between an attack that is in progress from one that has been completed. Trying to reconstruct the events is the subject of incident response, and is beyond the scope of this paper. However, on the other side of such activities is the eventual need to restore the system, and the ability to do that relies on data backup.

For the purposes of the project described in this paper, I rely on three methods of recovery, depending upon the nature of the problem. First, if it appears from the reports

---

[53] Unfortunately, the only writeable drives—CD and DAT tape—available to me are "FireWire" devices which I have yet to get working under SuSE Linux. My choices are to mount the MacOS Standard format hard drive, copy the files over, and write the media while booted under MacOS, or to use the "scp" client (ssh copy command) to move the files to another machine where they can be written to appropriate media.

[54] I suggest using a unique one that is different from the root password in order to provide protection in the case that the root password was what was compromised in order to gain the access being exploited.

that only a few rpm packages are affected, I prefer to cleanly install said packages from the original distribution CDs (or CD containing downloaded updates). If, rather, it appears that there is extensive corruption of the OS across the file structure, I prefer to use a CD image of the installed system. Finally, if the problem seems to be related to a particular branch of the file structure, I rely on a tarball for that branch; this last technique is also useful for providing backups of user data. (Note: the Linux source files kept in /usr/src are always recovered from CD, and are not included in the system CD image or tarballs.) Creation of the CD image and tarballs is discussed below. Several things should be noted: (1) other backup methods exist (e.g., ufsdump/ufsrestore, dump, dd, and cpio); (2) backups should be coordinated with I&A; (3) the frequency of backups should be tied to changes in the data (for the present use of this machine—network testing and security monitoring—the installed base is stable); and (4), store the backup media in a secure location that is not subject to the same set of physical threats if at all possible.

Creating an image of the system was done within the /tmp directory by issuing the command `mkisofs -o cd.iso -log-file cd.iso.log -R -x /usr/src -x /proc /`. (The installed system, as described in this paper, easily fits within the capacity of a single CD.) Tarball images are created directory by directory (bin, boot, dev, etc, home, lib, opt, root, sbin, usr, var). To create the image, change your location (`cd`) into the directory of choice (say bin for the example here) and enter the command `tar –zcvpf /bin_archive.tar.gz .`. Here this places a compressed tar image of the bin directory in the root directory using relative paths and preserving ACL and permission information. (Using a relative path is important if you want to untar the file in a temporary location, and not simply write over all of an existing bin tree.) For the usr directory, I also add the switch –exclude=src after the "." to keep from including the Linux source tree in the tarball. Finally, move the newly created archive files off of the machine to appropriate media where they will be safe and available, but hopefully never needed.[53] Backup media should also be read before this task is considered complete, so as to provide assurance that recovery can really take place.

Step 7. Final thoughts

If all of the recommendations preceding this step were successfully implemented, you should have a fairly high confidence that you now have a secure installation. However, there are several additional actions that you can take to *maintain* that confidence, as outlined below.

Step 7.1 Staying abreast

Information security is a rapidly changing field. One way to keep current—I suggest a minimal way—would be through subscriptions to the SuSE security mailing lists found at http://www.suse.com/us/support/mailinglists/index.html and through a periodic review of the listings at www.susesecurity.com. A good list of general Linux security-related newsletters to consider can be found at http://www.linuxsecurity.com/resources/forums-1.html; the weekly summaries come highly recommended. A number of other popular lists (including Linux specific) can be found at http://www.securityfocus.com/cgi-bin/forums.pl.

Step 7.2 Log review and processing

An important part of maintaining a secure installation is actually making use of the instrumentation that tells you the protection measures are working. This means that all of the logging implemented in preceding steps must be reviewed. If you find this to be an intolerable burden, several tools are available that might help. These include swatch[55] and logcheck.[56]

Step 7.3 Testing

Throughout the preceding steps, at times explicitly and at times not, each and every security-related feature was tested for functionality. Such testing should be repeated periodically, especially after any system software upgrades, in order to make sure the security functionality remains. The testing can be extended, however, to include network-based vulnerability testing with nmap, Nessus, CyberCop, SATAN, SARA, or SAINT.

Step 7.4 Reevaluating security design

Any time the purpose a computer is used for changes, all of the assumptions that went into assembling a suite of protection elements need to be reevaluated. For example, if the machine that was the subject of this paper were to be reapplied to serve as a typical workstation, an additional layer of protection could be gained by adding a stateful packet-filtering firewall like `iptables`, and linking it up with PortSentry. Periodic reevaluation of the design is called for in any case, however, due to the speed with which IT technologies are changing.

Step 8. Postscript

The introduction to this paper alluded to the fact that the primary purpose of the Linux installation discussed herein was to setup a PowerBook G4 laptop is a *portable* network testing and security monitor. The principle program to support this function is called tcpdump, which is able to capture and print out the headers of packets seen by a network interface. Using tcpdump is as simple as entering the command `tcpdump`, although it is also powerful enough to screen traffic based on many parameters including source or destination network, host and port, and protocol (the man pages provide good documentation). The SuSE distribution provides a copy of tcpdump, although you may want to check out the latest at www.tcpdump.org.

---

[55] http://www.oit.ucsb.edu/~eta/swatch/ See also discussion at http://www.enteract.com/~lspitz/swatch.html

[56] http://www.psionic.com/abacus/logcheck/

Appendix A – installed packages

3ddiag-0.149-13
aaa_base-2001.9.1-0
aaa_dir-2001.1.23-2
aaa_skel-2001.2.20-0
acct-6.3.5-101
allman-2001.1.15-3
arpwatch-2.1a10-8
asl-1.42build9-15
autoconf-2.13-271
autolog-0.35-197
automake-1.4-267
base-2001.1.9-8
bash-2.04-76
bc-1.06-15
bdflush-1.5-272
bindutil-8.2.3-81
binutils-2.10.0.33-12
bison-1.28-13
bwbasic-2.20.2-104
bzip-1.0.1-13
calctool-2.4.12-281
catdoc-0.90.3-165
compat-2001.1.24-6
compress-4.2.4-275
cpio-2.4.2-283
cracklib-2.7-241
cron-3.0.1-295
ctags-2000.5.18-12
db-3.1.17-15
ddd-3.2.1-135
devs-2001.1.2-9
diffutils-2.7-12
dump-0.4b20-12
e2fsprogs-1.19-15
e2fsprogs-devel-1.19-15
ed-0.2-266
eject-2.0.2-173
elm-2.4.60-208
fbset-2.1-180
fetchmail-5.6.5-3
file-3.32-8
fileutils-4.0.35-12
findutils-4.1.6-14
flex-2.5.4-282
freetype-1.3.1-142
freetype2-2.0.1-13
fvwm2-2.2.4-70
gawk-3.0.6-14
gcc-2.95.2-28
gccmesg-2.95.2-28
gdb-5.0-14
gdbm-1.8.0-211
gettyps-2.0.7j-95
glib-1.2.8-90
glibc-2.2-13
glibc-devel-2.2-13
glibc-nssv1-2.2-13
glibc-profile-2.2-13
gperf-2.7.2-37
gpg-1.0.6-0
gpgaddon-1.1-8
gpm-1.18.1-157
gpp-2.95.2-28
gppshare-2.95.2-28
groff-1.16.1-13
gs_fonto-5.50-140
gs_fonts-5.50-140
gs_lib-5.50-140
gs_x11-5.50-140
gsview-1.5-254
gv-3.5.8-276
gzip-1.3-12
hfsutils-3.2.6-123
hwinfo-1.95-3
hwinfo-devel-1.95-3
icons-100.0-5
indent-2.2.6-12
iputils-20001110-6
kbd-1.03a-13
lclint-2.5q-8
less-358-13
libelf-0.7.0-157
libgpp-2.95.2-28
libjpeg-6.2.0-138
libmikmo-3.1.9-60
libpng-2.1.0.8-9
libtiff-3.5.5-82
libtool-1.3.5-54
libz-1.1.3-264
lsof-4.52-13
lukemftp-1.5-8
lvm-0.9.1_beta4-5
lx_suse-2.4.2.SuSE-1
mailx-8.1.1-258
make-3.79.1-61
makedev-2.5.3-81
makewhat-2001.1.24-5
man-2.3.10d69s-162
man-pages-1.34-3
man9-98.2.14-55
mesa-3.4-13
mesasoft-3.4-13
metamail-2.7.19-269
mgdiff-1.0-306
mkisofs-1.9-5
modutils-2.4.2-9
mol-0.9.58-1
mtools-3.9.7-57
ncurses-5.2-16
ncurses-devel-5.2-16
net-tools-1.57-13
netcat-1.10-263
netcfg-2000.12.14-2
netscape-4.70-92
nkitb-2001.8.14-0
openmotif-2.1.30MLI4-15
openssh-2.9p1-6
openssl-0.9.6a-14
pam-0.72-156
pam_devperm-2000.12.1-9
patch-2.5.3-232
pciutils-2.1.8-66
pcmcia-3.1.22-13
pdisk-0.8a-28
pdksh-5.2.14-195
perl-5.6.0-16
perl-Digest-MD5-2.12-13
perl-HTML-Parser-3.13-13
perl-MIME-Base64-2.11-13
perl-Storable-0.6.11-13
perl-URI-1.09-15
perl-gettext-1.01-13
perl-libnet-1.0703-13
perl-libwww-perl-5.48-15
perl_tk-800.014-195
pmake-2.1.33-181
popt-1.6-0
procmail-3.15.1-6
ps-2001.1.24-6
rawio-2-166
rpm-3.0.6-19
sash-3.4-160
scslog-2.2-7
sendmail-8.11.2-15
sh-utils-2.0-13
shadow-20000902-61
sharutils-4.2c-14
strace-4.2-110
sudo-1.6.3p6-5
syslogd-1.3.33-186
sysvinit-2.78-133
tcl-8.3.2-12
tclx-8.3-12
tcpd-7.6-99
tcpdump-3.4a6-284
tcsh-6.09.00-138
termcap-2.0.8-278
terminfo-5.2-16
texinfo-4.0-158
textutils-2.0.10-13
timezone-2.2-13
tix-8.1-12
tk-8.3.2-12
tripwire-1.2-263
ttmkfdir-noversion-109
type1inst-0.6.1-12
udf-0.9.1-115
unzip-5.41-13
usbmgr-0.4.1-9
util-linux-2.10q-9
vim-5.7-21
vlock-1.3-20
xautolck-pl10-280
xaw3d-1.5-192
xbanner-1.31-8
xcolors-91.10.4-265
xdevel-4.0.2-15
xdmbgrd-0.4-113
xf86-4.0.2-15
xf86_3x-3.3.6-115
xfbdev-3.3.6-115
xfine-2.8-184
xfnt100-4.0.2-15
xfntscl-4.0.2-15
xgrabsc-2.41-260
xkeycaps-2.46-14
xli-1.16-304
xloader-4.0.2-15
xlock-4.16-106
xmods_3x-3.3.6-115
xmodules-4.0.2-15
xosview-1.7.3-133
xpdf-0.91-10
xsfb-2.8-52
xshared-4.0.2-15
xtermset-0.5-11
xv-3.10a-280
y2t_inst-packages-2.1.28-11
y2t_inst-update-2.1.26-11
y2t_inst-x11-2.1.25-11
y2t_lan-2.1.32-3
y2t_menu-2.1.19-3
y2t_online_update-2.0.25-4
y2t_prnt-2.1.21-3
y2t_rc_config-2.0.24-3
y2t_snd-2.1.23-7
y2t_spkg-2.1.20-3
y2t_update-2.0.24-3
yacc-91.7.30-267
yast-1.09.3-1
zip-2.3-147
ziptool-1.2-201

Appendix B – PAM configuration files

```
#%PAM-1.0
#/etc/pam.d/chfn
auth       required     /lib/security/pam_unix.so
auth       required     /lib/security/pam_tally.so      no_magic_root
account    required     /lib/security/pam_tally.so      deny=5 no_magic_root
session    required     /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/chsh
auth       required     /lib/security/pam_unix.so
auth       required     /lib/security/pam_tally.so      no_magic_root
account    required     /lib/security/pam_tally.so      deny=5 no_magic_root
session    required     /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/login
auth       required     /lib/security/pam_unix.so
auth       required     /lib/security/pam_securetty.so
auth       required     /lib/security/pam_nologin.so
account    required     /lib/security/pam_tally.so      deny=5 no_magic_root
account    required     /lib/security/pam_unix.so
account    required     /lib/security/pam_access.so
session    required     /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/other
auth       required     /lib/security/pam_warn.so
auth       required     /lib/security/pam_deny.so
account    required     /lib/security/pam_deny.so
password   required     /lib/security/pam_warn.so
password   required     /lib/security/pam_deny.so
session    required     /lib/security/pam_deny.so

#%PAM-1.0
#/etc/pam.d/passwd
auth       required     /lib/security/pam_unix.so
auth       required     /lib/security/pam_tally.so      no_magic_root
account    required     /lib/security/pam_tally.so      deny=5 no_magic_root
account    required     /lib/security/pam_unix.so
password   required     /lib/security/pam_pwcheck.so    md5 use_cracklib
password   required     /lib/security/pam_unix.so       md5 use_first_pass use_authtok
session    required     /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/su
auth       required     /lib/security/pam_unix.so
auth       required     /lib/security/pam_tally.so      no_magic_root
account    required     /lib/security/pam_tally.so      deny=5 no_magic_root
account    required     /lib/security/pam_unix.so
session    required     /lib/security/pam_unix.so
```

```
#%PAM-1.0
#/etc/pam.d/vlock
auth        required        /lib/security/pam_unix.so
#vlock can't seem to handle pam_tally
#"pam_tally[1147]: Error opening /var/log/faillog for update"
#auth       required        /lib/security/pam_tally.so          no_magic_root
#account    required        /lib/security/pam_tally.so          deny=5 no_magic_root
session     required        /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/xdm
auth        required        /lib/security/pam_unix.so
auth        required        /lib/security/pam_securetty.so
auth        required        /lib/security/pam_nologin.so
auth        required        /lib/security/pam_tally.so          no_magic_root
account     required        /lib/security/pam_tally.so          deny=5 no_magic_root
account     required        /lib/security/pam_unix.so
account     required        /lib/security/pam_access.so
session     required        /lib/security/pam_unix.so

#%PAM-1.0
#/etc/pam.d/xlock
auth        required        /lib/security/pam_unix.so
#xlock can't seem to handle pam_tally
#"pam_tally[1147]: Error opening /var/log/faillog for update"
#auth       required        /lib/security/pam_tally.so          no_magic_root
#account    required        /lib/security/pam_tally.so          deny=5 no_magic_root
session     required        /lib/security/pam_unix.so
```

Appendix C—PortSentry startup script

```
#! /bin/sh
#/etc/init.d/portsentry
# System startup script for Portsentry:
# /usr/local/psionic/portsentry/portsentry
#
### BEGIN INIT INFO
# Provides:       portsentry
# Required-Start: $local_fs $syslog network
# Required-Stop:
# Default-Start:  2 3 5
# Default-Stop:   0 1 6
# Description:    Start port monitoring
### END INIT INFO

# Source SuSE config (looking for START_PORTSENTRY="yes" or "no")
. /etc/rc.config

# Determine the base and follow a runlevel link name.
base=${0##*/}
link=${base#*[SK][0-9][0-9]}

# Force execution if not called by a runlevel directory.
test $link = $base && START_PORTSENTRY=yes
test "$START_PORTSENTRY" = yes || exit 0

PORTSENTRY_BIN=/usr/local/psionic/portsentry/portsentry
TCP_PARAMS=atcp
UDP_PARAMS=audp

test -x $PORTSENTRY_BIN || exit 5

# Shell functions sourced from /etc/rc.status:
#      rc_check         check and set local and overall rc status
#      rc_status        check and set local and overall rc status
#      rc_status -v     ditto but be verbose in local rc status
#      rc_status -v -r  ditto and clear the local rc status
#      rc_failed        set local and overall rc status to failed
#      rc_reset         clear local rc status (overall remains)
#      rc_exit          exit appropriate to overall rc status
. /etc/rc.status

# First reset status of this service
rc_reset
```

```
# Return values acc. to LSB for all commands but status:
# 0 - success
# 1 - misc error
# 2 - invalid or excess args
# 3 - unimplemented feature (e.g. reload)
# 4 - insufficient privilege
# 5 - program not installed
# 6 - program not configured
# 7 - program is not running
#
# Note that starting an already running service, stopping
# or restarting a not-running service as well as the restart
# with force-reload (in case signalling is not supported) are
# considered a success.

case "$1" in
   start)
          echo -n "Starting Portsentry"
          $PORTSENTRY_BIN -$TCP_PARAMS
          $PORTSENTRY_BIN -$UDP_PARAMS

          # Remember status and be verbose
          rc_status -v
          ;;
   stop)
          echo -n "Shutting down Portsentry"
          ## Stop daemon with killproc(8) and if this fails
          ## set echo the echo return value.

          killproc -TERM $PORTSENTRY_BIN

          # Remember status and be verbose
          rc_status -v
          ;;
   *)
          echo "Usage: $0 {start|stop}"
          exit 1
          ;;
esac
rc_exit
```

Appendix D—Tripwire configuration file

```
/bin
/boot
!/cdrom
/dev R-mc
!/dev/pts
!/dvd
/etc R-mc
!/etc/ioctl.save
!/etc/mtab
!/floppy
=/home
/lib
=/lost+found
!/mac
!/macboot
!/macos
!/mnt
/opt
!/proc
=/root
/sbin
=/tmp R-mc
/usr
/usr/local
/usr/local/psionic/portsentry/portsentry.blocked.atcp R-mc
/usr/local/psionic/portsentry/portsentry.blocked.audp R-mc
/usr/share/zoneinfo R-mc
/var R-mc
!/var/adm/tripwire
/var/adm/tripwire/bin/siggen
/var/adm/tripwire/bin/tripwire
/var/adm/tripwire/bin/twdb_check.pl
=/var/cache R-mc
=/var/lib/xdm/authdir/authfiles R-mc
=/var/lock R-mc
/var/log L
/var/log/boot.msg L-i
/var/log/boot.omsg L-i
=/var/lost+found R-mc
=/var/mail R-mc
=/var/run R-mc
=/var/spool R-mc
=/var/tmp R-mc
```