Global Information Assurance Certification Paper

Step-By-Step Guide to Configuring an SSL enabled Web Server that Accesses a Backend Database using RedHat 7.0

Introduction
The purpose of this guide is to describe, in detail, the steps required to configure a system designed to host an SSL-enabled, web-based application that requires secure communication to a database hosted on an internal network. For the remainder of this guide, the system will be referred to by its hostname, bastion.

To fulfill the mission requirements, bastion has the following properties:

- Its web-based application serves sensitive information to the Internet, and as such, is configured to only provide encrypted (via SSL) communication.
- It has a database client installed so that the web-based application can query an internal database and generate the appropriate content requested by visitors.
- Since its database queries come from outside the internal network, they will be tunneled through a firewall using SSH.
- It will occasionally need to send email to Internet users as a result of their interactions with the web-based application.

As always, the goal of system security is to provide the necessary and sufficient functionality to fulfill business objectives while minimizing the risk inherent in having a system exposed to the Internet. This guide provides the step-by-step instructions for removing extraneous software and unnecessary services and for properly configuring the operating system and applications required to achieve this goal.

Resources

Network Architecture:
Bastion will be located in the DMZ and accessible to the Internet. This is a typical network configuration where a firewall is in place between the Internet and all systems. The mail and DNS server systems are also located in the DMZ, and the firewall isolates the DMZ subnet from the internal network subnet. Specifically, the firewall allows traffic that initiates a TCP connection for destination port 80 (http) and 443 (hhtps) to bastion from anywhere and traffic for destination port 22 (ssh) to bastion from the internal subnet. The firewall also allows traffic that initiates a TCP connection from bastion outbound to destination port 25 (SMTP) on the mail server and to destination port 22 (ssh) to the MySQL server host on the internal network. The internal network subnet is 10.1.2.0/32 and the DMZ will be 10.1.1.0/32. Other relevant IP address information is as follows:

```
Bastion IP                10.1.1.52
Gateway                   10.1.1.1
Primary DNS server        10.1.1.227
Secondary DNS server      10.1.1.228
```

Hardware list:

- Gateway E-5400 System (relevant system components)
  - IDE Controller - Promise Ultra100 (BIOS version 2.01 b27)
  - Motherboard - Intel OR840 Workstation (BIOS version 245)
  - Network Adapter - Intel Pro/100+ Adapter

- Hard Disk Drive – Quantum Fireball Plus IDE (UltraATA/100) LM30 (27 GB
- Standard Floppy Disk Drive
- CDROM drive
- 3-Button Mouse
- US Keyboard

Software Inventory (primary applications):

- RedHat 7.0 Operating System (2 Discs)
- Apache Web Server version 1.3.19
- ModSSL version 2.8.2
- MySQL client version 3.23.33
- SSH Client and Server version 2.4.0

Step 1 – BIOS and Physical Security

Identify a location in a restricted server room area where bastion will be place after setup is complete. Physical access to a system usually leads to root access, so make sure entry to the room is controlled and audited. In order to install from the CD, the BIOS must be set to boot from CDROM before SCSI. Later, for security reasons, this setting should be changed so that the system will boot from the SCSI device only. Also, set a BIOS password to protect the CMOS settings so that a casual passerby cannot alter them.

Step 2 – Install the Operating System

The installation of the operating system is complicated by the fact that the Linux kernel version on the Operating System Installation Disks (2.2.16) does not support the Promise ATA100 controller installed on the Gateway E-5400. Furthermore, the BIOS on the Intel OR840 motherboard did not support automatic probing for PCI hardware. Detailed instructions for working around the first problem are contained in [1]. To overcome the later problem, the BIOS of the motherboard was updated using software obtained from the Intel website [2].

To install the operating system with the bare minimum number of packages, perform the following steps:

1. Insert the RedHat 7.0 Installation disc number 1
2. Reboot the system
3. At the initial screen entitled "Welcome to Red Hat Linux 7.0!", enter the "expert text ide2=0x1088, 0x1096 ide3=0x1080, 0x1092" at the "boot:" prompt and press Enter.

You will be prompted with a series of screens. Table 1 below summarizes the screens and what actions should be taken on each.

Table 1: Summary of Responses During Red Hat 7.0 Installation (expert text mode)

Window Title	On Screen Question/Option	Action to Perform/Select
Devices	Do you have Driver Disk?	No
Choose a Language	What language should be used during the installation process?	English
Keyboard Type	What type of keyboard do you have?	US
Installation Media	What type of media contains the packages to be installed?	Local CDROM
Devices	Would you like to load any special device drivers?	Done
Red Hat Linux	Welcome and Intro screen	OK
Installation Type	Select Custom System	OK
Disk Setup	Which tool would you like to use?	Disk Druid
Choose Partitions to Format (see below)	What partitions would you like to format?	Select all partitions and check for bad blocks during format
LILO Configuration	If you need to pass boot options to the kernel, enter them now	Deselect linear mode and add "ide2…"
LILO Configuration	Where do you want to install the boot loader?	/dev/hde Master Boot Record (MBR)
LILO Configuration	The boot manager can boot other Operating Systems …	Make no changes and select OK
Hostname Configuration	Hostname:	Enter selected hostname (*e.g.* bastion)
Mouse Selection (Choose correct mouse and "Emulate 3 buttons", if appropriate)	Which model mouse is attached to this computer?	Generic – 3 Button Mouse (PS/2)
Time Zone Selection ("Select Hardware Clock set to GMT", if appropriate.)	What time zone are you located in?	America/New York
Root Password	Pick a root password.	Enter twice
Add user	You should use a normal user account …	Username: www Comment: User for Web Server Password: xxxxxx Password (again): xxxxxx
Authentication Configuration	ONLY Select "Use shadow passwords" and Enable MD5 Passwords	
Package Group Selection		DESELECT ALL PACKAGES
Installation to Begin	A complete log of your installation …	OK
Create a boot disk		Insert a blank floppy and select OK
Complete		OK

Use Disk Druid to partition the disk into the following partitions:

Mount Point	Size (MB)	Filesystem Type	Purpose
/	1000	Linux Native	System files
(N/A)	256	Linux Swap	Swap
/home	3000	Linux Native	Web Server Document Root
/tmp	500	Linux Native	Temporary Files
/var	1000	Linux Native	Logs

The system partition (/) is should be large enough to hold the operating system and application files. The size of the swap partition should be 2 times the RAM of the system. The size of the /home partition should be large enough to hold all of the web server documents with some room to expand the web site in the future. The size of the web server logs will depend on the number of hits received by the web site, and the log partition should be sized accordingly. Temporary files (stored in the /tmp partition) should not be large at all, and 500 MB is generous, but seems appropriate given the size of the available disk. There is approximately 20 GB of disk space left free. Some of it will be used later to back up the system prior to deploying it.

Step 3 – Remove unnecessary packages
Even though no packages were selected during the installation setup, several unnecessary packages are installed. They are *dhcpcd*, *redhat-logos*, *krb5-libs*, *linuxconf*, *quota*, *bc*, *setuptool*, *slocate*, *time*, *apmd*, *at*, *ed*, *hdparm*, *isapnptools*, *kernel-pcmcia-cs*, *kudzu*, *ncompress*, *setserial*, *stat*, *ash*, *authconfig*, *cpio*, *ntsysv*. The functionality provided by each of these packages is not required for bastion to fulfill its mission requirements, however, individual administrators might like to keep one of more of them. Information about a package can be obtained with the following command:

```
[root@bastion]# rpm –qi time
Name        : time                         Relocations: /usr
Version     : 1.7                          Vendor: Red Hat Software
Release     : 9                            Build Date: Mon 22 Mar 1999 01:30:03 AM EST
Install date: Thu 15 Jun 2000 12:02:53 PM EDT    Build Host: porky.devel.redhat.com
Group       : Applications/System          Source RPM: time-1.7-9.src.rpm
Size        : 18921                        License: GPL
Packager    : Red Hat Software <http://developer.redhat.com/bugzilla/>
Summary     : A GNU utility for monitoring a program's use of system resources.
Description :
The GNU time utility runs another program, collects information about the resources
used by that program while it is running and
displays the results.

Time can help developers optimize their programs.
```

It is recommended that all unnecessary packages be removed, since each extra package is one more chance for a vulnerability to exist one the system. Use the Red Hat package manager application to remove the unnecessary packages with the following command:

```
[root@bastion /root]# rpm e dhcpcd redhat-logos krb5-libs linuxconf quota bc setuptool
slocate time apmd at ed hdparm isapnptools kernel-pcmcia-cs kudzu ncompress setserial stat
ash authconfig cpio ntsysv
```

Step 4 – Add Packages Required in Later Steps
A few packages that are needed in later steps did not get installed and should be added now. To do this, insert the first Red Hat 7.0 Installation Disc into the CDROM drive, mount it, and install them with the following commands:

```
[root@bastion /root]# mount /mnt/cdrom
[root@bastion /root]# rpm –ivh /mnt/cdrom/RedHat/RPMS/package_name
```

Replace *package_name* with the name of each specific package to install. The following is a list of packages required and a brief description why they are needed.

1. ftp-0.17-6.i386.rpm (for anonymous FTP connections to retrieve package updates and other software)
2. bind-utils-8.2.2_P5-25.i386.rpm (for nslookup and hostname to IP address resolution)
3. tcp_wrappers-7.6-15.i386.rpm (SSH and other listening services uses these wrappers to control access to the host)
4. openssl-devel-0.9.5a-14.i386.rpm (while compiling apache with mod_ssl, the SSL headers in this package are required)
5. dialogue-0.9a-3.i386.rpm (autorpm requires this package)
6. gcc-2.96-54.i386.rpm (this and the remaining packages are needed to compile the kernel and other applications)
7. binutils-2.10.0.18-1.i386.rpm
8. cpp-2.96-54.i386.rpm
9. glibc-devel-2.1.92-14.i386.rpm
10. make-3.79.1-5.i386.rpm
11. kernel-headers-2.4.0-0.26.i386.rpm
12. dev86-0.15.0-5.i386.rpm
13. patch-2.5.4-4.i386.rpm
14. ncurses-devel-5.1-2.i386.rpm (buildkernel needs the header files in this package to make menuconfig)

Note that packages numbered 5-12 are only required for compiling the kernel and other applications and should be removed once the system setup has been completed and the system is deployed.

Step 5 – Configure GNUPG

The installation packages from Red Hat are signed using Red Hat's public key. This key is included on the installation CD and can be used to validate the signature on any further packages downloaded from the Red Hat updates site or any mirror site. To enable signature verification of RPM packages, the gnupg package must be configured. Use the following commands:

```
[root@bastion]# gpg
gpg: /root/.gnupg: directory created
gpg: /root/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new options file
[root@bastion /root]# gpg --import /mnt/cdrom/RPM-GPG-KEY
gpg: /root/.gnupg/secring.gpg: keyring created
gpg: /root/.gnupg/pubring.gpg: keyring created
gpg: key DB42A60E: public key imported
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: Total number processed: 1
gpg:               imported: 1
[root@bastion]# rpm --checksig /mnt/cdrom/RedHat/RPMS/patch-2.5.4-4.i386.rpm
/mnt/cdrom/RedHat/RPMS/patch-2.5.4-4.i386.rpm: md5 gpg OK
```

The last command is used to verify that the setup has been completed correctly, and any package on the CDROM may be selected.

Step 6 – Stop unnecessary services and configure them not to restart on system boot
Services that will not be used should be turned off. The chkconfig command is used on Red Hat to control the starting and stopping of system services.

```
[root@bastion]# chkconfig –list
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
keytable        0:off   1:off   2:on    3:on    4:on    5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on    5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
random          0:off   1:off   2:on    3:on    4:on    5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on    5:on    6:off
syslog          0:off   1:off   2:on    3:on    4:on    5:on    6:off
sendmail        0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

The scripts used to start and stop these services are located in the `/etc/rc.d/init.d` directory. The name of the script matches the name of the service. There is a short description of the service inside each script. For bastion, netfs (mounts and umounts all network filesystems. SAMBA, NFS, NCP) and rawdevices (maps raw devices to block devices, used by ORACLE) should be shut off using the following commands.

```
[root@bastion]# chkconfig –level 01234556 netfs off
[root@bastion]# /etc/rc.d/init.d/netfs stop
[root@bastion]# chkconfig –level 01234556 rawdevices off
[root@bastion]# /etc/rc.d/init.d/rawdevices stop
```

In addition, since bastion is not a mail server, the sendmail service should be configured not to run in daemon mode. This is done by editing the file `/etc/sysconfig/sendmail` to read

```
DAEMON=no
QUEUE=10m
```

After saving the file, restart the sendmail service with the following command:

```
[root@bastion]# /etc/rc.d/init.d/sendmail restart
```

The service will no longer listen to port 25 for incoming mail and, therefore, will not be vulnerable to remote sendmail exploits. The service will still deliver mail by clearing the queue every 10 minutes. The web application requires the ability to send mail to fulfill its mission requirements.

Step 7 – Configure and start networking
At this point, bastion should have no open ports listening for incoming requests from the network. It is safe to configure the networking parameters, plug in the Ethernet cable and start up networking services. To configure the network interface, create the a file with the following contents and save it as `/etc/sysconfig/network-scripts/ifcgf-eth0`:

```
DEVICE=eth0
IPADDR=10.1.1.52
NETWORK=10.1.1.0
BROADCAST=10.1.1.255
NETMASK=255.255.255.0
GATEWAY=10.1.1.1
```

Edit the `/etc/resolv.conf` file and add the IP addresses for DNS servers so it looks like

```
nameserver 10.1.1.227
nameserver 10.1.1.227
```

Plug in the Ethernet cable to an active socket and start up networking with the following command:

```
[root@bastion]# /etc/rc.d/init.d/network start
Setting network parameters:                                [ OK ]
Bringing up interface lo:                                  [ OK ]
Bringing up interface eth0:                                [ OK ]
```

Verify the host is up and that there are no open ports by scanning the system from another host with nmap.

```
[dave@dhcp18]# nmap –sT bastion
Starting nmap V. 2.53 by fyodor@insecure.org ( www.insecure.org/nmap/ )
All 1523 scanned ports on  (10.1.1.52) are: closed
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Step 8 – Download and configure autorpm and retrieve updated packages

AutoRPM is a tool that can be configured to check a specified updates location on a regularly scheduled interval. It can automatically download and install any updates that are available for the packages that currently exist on the system. As a security measure, it can be configured to require the packages contain a valid GPG signature before installing them. The utility can be obtained via by anonymous FTP downloaded from ftp.kaybee.org/pub/redhat/RPMS/noarch. Note that the perl-libnet package is required and may be obtained from the same site. To install the rpm packages issue the following commands

```
[root@bastion]# rpm -ivh perl-libnet-1.0605-2.noarch.rpm
Preparing...                ########################################### [100%]
   1:perl-libnet            ########################################### [100%]
[root@bastion]# rpm -ivh autorpm-1.9.8.4-2.noarch.rpm
Preparing...                ########################################### [100%]
   1:autorpm                ########################################### [100%]
```

First, edit the `/etc/autorpm.d/autorpm.conf` file so that the AutoRPM site is not checked for updates by commenting out the following line:

BEFORE:

```
Config_File("/etc/autorpm.d/autorpm-updates.conf");
```

AFTER:

```
Config_File("/etc/autorpm.d/autorpm-updates.conf");
```

It is also a good idea to change the email address of for reports to one that is frequently checked, instead of the default, root@localhost, by modifying the line:

BEFORE:
```
Set_Var("ReportDest","root");
```
AFTER:
```
Set_Var("ReportDest","dave@mycompany.com");
```

Next, edit the `/etc/autorpm.d/redhat-updates.conf` configuration file to automatically install update packages if the GPG signature is valid.

BEFORE:
```
action (updated) {

    Install (Interactive);
    # If you want to check the PGP or GnuPG signature …
    #Install (Auto);
    #PGP_Require (Yes);

}
```

AFTER:
```
action (updated) {

    #Install (Interactive);
    # If you want to check the PGP or GnuPG signature …
    Install (Auto);
    PGP_Require (Yes);

}
```

Finally, edit the `/etc/autorpm.d/pools/redhat-updates` file to point to one or more close FTP mirror sites. Note that the file distributed with the autorpm package uses the {$RHVersion} variable defined in the `autorpm.conf` file and relies on the autorpm script itself to add the architecture. In order to get the package to work properly, it was necessary to hard code the full FTP path to the architecture directory in this file (including the trailing slash '/').

redhat-updates file contents:
```
ftp://mirror.cs.wisc.edu/pub/mirrors/linux/redhat/updates/7.0/en/os/i386/
```

Before testing the installation and updating all the packages, the glibc package must be updated. This is because the updated version of glibc (2.2-12) is broken into two packages (glibc and glibc-common). Once both packages have been downloaded, use the follow command to install them.

```
[root@bastion]# rpm -Uvh glibc-2.2-12.i386.rpm glibc-common-2.2-12.i386.rpm
Preparing...                ########################################### [100%]
   1:glibc                  ########################################### [100%]
   2:glibc-common           ########################################### [100%]
```

Now, test the autorpm installation by running it to update the rest of the packages that were installed from the installation CDROM. Enter the command:

```
[root@bastion]# /usr/sbin/autorpm –print
```

The "—print" flag will instruct autorpm to send output to STDOUT instead of composing an email file. Messages should appear that the ftp site is being contacted and a directory listing obtained, followed by notes about the packages being updated. A to-do list of packages to update can be found in /var/spool/autorpm during the run. Also, packages will be downloaded to that directory prior to their installation.

Note that the `redhat-updates.conf` file is configured by default so that any package beginning with "kernel-" is automatically excluded. The kernel will be manually updated as discussed in the next step.

Step 9 – Update and recompile the kernel

There are at least three reasons to compile a customized Linux kernel for bastion. First, to improve security, performance, and functionality, it is beneficial to have the latest stable version of the kernel running on a system. The kernel that is installed during the default installation of Red Hat 7.0 is version 2.2.16, and, as of the time of writing this document, the updates directory had a version of 2.2.17. Both are considerable older than the latest stable version at www.kerenl.org is 2.4.2. The second reason to compile a customized kernel is that the kernel built during Red Hat installation includes the symmetric multi-processor option (SMP), even if the machine has only one CPU. If this option is not needed, better performance will result if it is not included. Finally, the installed kernel is built with loadable module support. There is at least one known Trojan kernel module for Linux (knark) and, therefore, since loadable module support is not required for bastion to fulfill its mission requirements, it would enhance security to disable it.

The task of compiling a kernel is a bit daunting, but made drastically simpler by obtaining the *autoconfigure* and *buildkernel* packages. The first package detects what hardware is present on the system and creates a kernel configuration file with the appropriate variables. The second package leads the user through each step required to build the kernel including downloading, extracting and patching the kernel source, running *make menuconfig*, running *make*, editing the `lilo.conf` file, and moving the compiled image to the `/boot` directory.

First obtain the RPMs from http://sourceforge.net/projects/kautoconfigure and http://users.dhp.com/~whisper/buildkernel/index.html, respectively. Since these are not FTP sites, download them to a separate system that has a web browser installed and copy them over to bastion. Create the `.config` file that will be used to compile the kernel with the following commands:

```
[root@bastion]# mkdir /usr/src/linux
```

```
[root@bastion]# ./autoconfigure.sh > /usr/src/linux/.config
```

Next make the following edits to the configuration file `/usr/src/linux/.config`. The text after the # on each line (if present) is a comment to give information about the option. It is not necessary to include.

```
Add these lines
CONFIG_SMP=n            # Disable SMP support
CONFIG_X86_UP_APIC=y    # APIC support on uniprocessors
CONFIG_X86_UP_IOAPIC=y # IO-APIC support on uniprocessors
CONFIG_PARPORT=y        # Enable Parallel Port Support
CONFIG_PARPORT_PC=y     # Parallel Port Support for PC-style hardware
CONFIG_PNP=n            # Disable Plug and Play support
CONFIG_NFS_FS=y         # Enable NFS client support
CONFIG_NFS_V3=y
CONFIG_NFSD=n           # Disable NFS server support
CONFIG_VGA_CONSOLE=y    # Enable VGA text console
CONFIG_VIDEO_SELECT=y   # Enable Video mode selection support
CONFIG_MODULES=n        # Enable Loadable Module Support

Confirm or add these lines for the Promise Ultra 100 Controller
CONFIG_BLK_DEV_IDE=y          # The remaining lines in this
CONFIG_BLK_DEV_IDEDISK=y      # section configure the IDE
CONFIG_BLK_DEV_IDECD=y        # and ATA devices to work at
CONFIG_BLK_DEV_IDEFLOPPY=y    # peak performance.  See [1]
CONFIG_BLK_DEV_IDESCSI=y      # Some of are already in
CONFIG_BLK_DEV_IDEPCI=y       # the file from autoconfigure.
CONFIG_IDEPCI_SHARE_IRQ=y
CONFIG_BLK_DEV_IDEDMA_PCI=y
CONFIG_IDEDMA_PCI_AUTO=y
CONFIG_BLK_DEV_IDEDMA=y
CONFIG_IDEDMA_PCI_WIP=y
CONFIG_BLK_DEV_PDC202XX=y
CONFIG_BLK_DEV_VIA82CXXX=y
CONFIG_IDEDMA_AUTO=y
CONFIG_IDEDMA_IVB=y
CONFIG_BLK_DEV_IDE_MODES=y

Remove this line
CONFIG_FB_ATY128=y      # Disable Frame Buffer Support
```

Before invoking buildkernel, retrieve the latest kernel patches by anonymous FTP to ftp.kernel.org/pub/linux/kernel/people/alan/2.4. As of the time of writing, the latest patch was patch-2.4.2-ac24.gz. Unzip the patch, rename the file, and move it to where the buildkernel script expects to find it with the following commands:

```
[root@bastion]# mkdir /usr/src/kpatches-2.4.2
[root@bastion]# gunzip patch-2.4.2-ac24.gz
[root@bastion]# mv patch-2.4.2-ac24 /usr/src/kpatches-2.4.2/01
```

Next install the buildkernel package and invoke the command with the options to download the source code for the 2.4.2 kernel.

```
[root@bastion]# rpm –ivh buildkernel-1.03.noarch.rpm
Preparing...                ########################################### [100%]
1:buildkernel               ########################################### [100%]
[root@bastion]# buildkernel 2.4.2 –BKOPENFRESH=YES
(many, many lines … )
```

The output from buildkernel is fast and furious and may run off the screen before it can be read. It is copied to /tmp/Bklog## while the application is running, and can be reviewed there. It will connect to ftp.kernel.org and download the zipped kernel source files. They will be copied to the `/usr/src/source/linux-2.4.2` directory on the system. The source tree will be extracted to the `/usr/src/linux-2.4.2` directory and the patch should be applied. The existing `/usr/src/linux` will be moved to `/usr/src/linux.old` and a symbolic link named `/usr/src/linux` will be created pointing to `/usr/src/linux-2.4.2`. The `.config` file will be moved to `/usr/src/configs/.config.2.4.2` and used later for menuconfig. The first opportunity for interaction will be to edit the Makefile. No changes are required, so simply quit the vi editor. Next the menuconfig screens will be presented. The current settings can be reviewed, but no changes should be required since the configuration file was manually edited earlier. Exit and save the configuration file. The compiling will begin. Upon successful completion, the kernel image, named `bzImage-2.4.2-1`, will be copied to the `/boot` directory and the `/etc/lilo.conf` file will be brought into the editor. No changes are required, however, the following contents is sufficient.

```
boot=/dev/hde1                    # Make sure this is the boot device
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
lba32                             # This line is important.
default=linux

image=/boot/vmlinuz-2.2.16-22smp  # The back-up kernel
    label=linux.old
    read-only
    root=/dev/hde1
    append="ide2=0x1088, 0x1096 ide3=0x1080, 0x1092"
image=/boot/vmlinuz               # The new kernel
    label=linux
    root=/dev/hde1
    read-only      # Passing IDE memory locations are no longer required
```

For the above `lilo.conf` file to work properly, some rearranging of the `/boot` directory is required. Perform the following commands:

```
[root@bastion]# cd /boot ; rm –f *2.2.16-22
[root@bastion]# ln –sf bzImage-2.4.2-1 vmlinuz
[root@bastion]# ln –sf System.map-2.4.2-1 System.map
```

Reboot the system; enter linux at the LILO prompt (or just press enter to accept the default) and the new kernel should start. Once the new kernel has successfully booted, some clean up should be performed. First, move the `/etc/modules.conf` file to

`/etc/modules.conf.old` so the system will not attempt to load kernel modules while booting. Since they are no longer supported, the attempt to load them will fail anyway, but by moving this file, it will reduce extraneous error messages. Second edit the /etc/cron.d/kmod file and put a # sign in front of (*i.e.* comment out) the line that removes kernel modules that are no longer in use from the running kernel. This step will prevent crond error messages from being emailed to the root user about kernel module removal failure.

For optional additional cleanup the directories source, linux.old (which should be empty), kpatches-2.4.2, and the symbolic link linux.ac in the `/usr/src` directory may be removed.

Step 10 – Setup TCP Wrappers
The tcpwrappers application uses the /etc/hosts.allow and /etc/hosts.deny file to control access to services in offered by the inetd daemon. No such services are or will be offered on bastion. However, as will be discussed in the next section, SSH can also be configured to use these files to control access. As such, the files should be edited to have the following contents (again, text after the # are comments and need not be included):

```
/etc/hosts.deny:
    ALL:ALL                     # This will deny everything not explicitly allowed

/etc/hosts.allow:
    sshd: 10.1.1.0/32 10.1.2.0/32  # This allows SSH clients from our network
    sshdfwd-3306: 127.0.0.1        # This allows port forwarding – see MySQL section
```

It may also be desirable to allow SSH clients to connect from somewhere on the Internet so that someone is not forced to come to the local office to access the web server in an emergency. An IP address (address block) can be added at the end of the sshd line to enable this access.

Step 11 – Setup an SSH daemon and client
(SSH v2.4.0 from SSH Communications Security)
All remote access to bastion, aside from the visitors to its web site, will be encrypted using secure shell. This includes remote logins and communication from the database client to the server inside the corporate network. To obtain SSH, go to www.sssh.com/products/ssh/download.html and select an appropriate mirror site (ftp.cis.fed.gov/pub/ssh in this example). Download the tarball archive of the latest version by anonymous FTP. To compile and install the software with TCP wrapper support, without rsh fallback and without SUID perform the following commands (system responses are left out for clarity):

```
[root@bastion]# tar –zxf ssh-2.4.0.tar.gz
[root@bastion]# cd ssh-2.4.0
[root@bastion]# ./configure –with-libwrap –without-rsh –disable-suid-ssh
[root@bastion]# make
```

```
[root@bastion]# make install
[root@bastion]# cp sshd2.startup /etc/rc.d/init.d/sshd
[root@bastion]# chckconfig –add sshd
[root@bastion]# /etc/rc.d/init.d/sshd start
```

The second and third to last lines configure the SSH daemon to start automatically when the system boots using the chkconfig command and the last line starts the server immediately. Verify that the server is running and that you can connect to it by the following commands:

```
[root@bastion]# ps -ef | grep sshd
root              321     1  014:19 ?    00:00:00 /usr/local/bin/sshd
[root@bastion]# ssh bastion
root's paasword:
```

The configuration files for SSH will be located in the `/etc/ssh2` directory. Edit sshd2_config to disallow remote root logins via ssh by changing the following line:

```
BEFORE
PermitRootLogin  yes
AFTER
PermitRootLogin  yes
```

Note that since the "—with-libwrap" option was specified, access to the SSH daemon will also be controlled by TCP wrappers. See section x.x below for details.

STEP 12 – Obtain a web server certificate

Since bastion will be sending sensitive data to Internet users, SSL encryption is required for the web server. In addition, to guard against spoofing, bastion will have an X.509 digital server certificate signed by a trusted third party (like VeriSign or Thawte). The gory details of filing for this type of certificate depend on the vendor are beyond the scope of this document. However, regardless of the vendor, the steps are:

1. Generate a certificate-signing request (*e.g.* `mycert.csr`) using *openssl* (see http://www.thawte.com/certs/server/keygen/mod_ssl.html for an excellent step-by-step procedure)
2. Send it to the vendor with payment and proof of identification.
3. The vendor will confirm your identity; digitally sign your certificate, and return the signed copy (mycert.crt).

The rest of the instructions assume these steps have already been completed and the returned file (along with mycert.key) have been copied to the `/etc/httpd/conf` directory.

STEP 13 – Setup an SSL enabled web server
(modssl v2.8.2 and apache server v1.3.19)

The mod_ssl package enables apache to use SSL encryption. The version of mod_ssl must be compatible with the version of apache in order for the application to properly run. In order to successfully install an SSL enabled web server using the latest version of mod_ssl and apache, first visit www.modssl.org and determine what the latest supported apache version is (as of the time of this writing it was 1.3.19). Retrieve the tarball archive of mod_ssl (mod_ssl-2.8.2-1.3.19.tar.gz) by anonymous FTP to ftp.modssl.org/source. To obtain the tarball archive of apache (apache-1.3.19.tar.gz), first visit http://www.apache.org/dyn/closer.cgi to identify a convenient mirror site, and then retrieve it by anonymous FTP to the site (in this case ftp.twoguys.org/pub/apache/dist/httpd). For better performance, it is recommended that the shared memory library be used. This can be downloaded by anonymous FTP to ftp.engelschall.com/sw/mm. To compile the apache server, use the following commands (detailed in the INSTALL file of the mod_ssl source distribution):

```
[root@bastion]# tar –zxf mm-1.1.3.tar.gz
[root@bastion]# tar –zxf mod_ssl-2.8.2-1.3.19.tar.gz
[root@bastion]# tar –zvf apache-1.3.19.tar.gz
[root@bastion]# cd mm-1.1.3 ; ./configure –disable-shared ; make
[root@bastion]# cd ../mod_ssl-2.8.2-1.3.19
[root@bastion]# ./configure –with-apache=../apache_1.3.19 –with-mm=../mm-1.1.3 –prefix=/usr
[root@bastion]# cd ../apache_1.3.19
[root@bastion]# SSL_BASE=SYSTEM ./configure –with-layout=RedHat –enable-module=ssl
[root@bastion]# make
[root@bastion]# make certificate TYPE=existing CRT=/etc/httpd/conf/mycert.crt
KEY=/etc/httpd/conf/mycert.key
[root@bastion]# make install
```

Before attempting to start the web server, edit its configuration file `/etc/http/conf/httpd.conf`. The table below summarizes the changes to be made.

Find the lines with	Change them to	Purpose
`Server Root=…`	`Server Root = "/etc/httpd"`	
`User nobody` `Group nobody`	`User www` `Group www`	The web server should run as the www user, not the nobody user.
`ServerAdmin …`	`ServerAdmin email@mycompany.com`	Configure where emails directed to the server admin should go.
`ServerName …`	`ServerName bastion.mycompany.com`	

`<Directory>`
`Options FollowSymLinks`
`AllowOverride None`
`</Directory>`
	`<Directory>`
	`Options None`
	`AllowOverride None`
	`order deny, allow`
	`deny from all`
	`</Directory>`
		Explicitly deny all access to the root of the file system and disallow following of symbolic links

`<Directory "/home/httpd/html">`
 	
`(Everything in between)`
 	
`</Directory>`
	`<Directory "/home/httpd/html">`
	`Options None`
	`AllowOveride None`
	`Order allow,deny`
	`Allow from all`
	`</Directory>`
		The only thing in the document root of the non-SSL server is a file directing users to the SSL home.

`ServerSignature …`	`ServerSignature Off`	Disable printing of server version on error pages

<VirtualHost _default_:443> (and all lines in between) ServerAdmin…	<VirtualHost _default_:443> (and all lines in between) ServerAdmin email@mycompany.com DocumentRoot="/home/httpd/htlm_ssl" <Directory "/home/httpd/html_ssl"> Options ExecCGI AllowOveride None Order allow,deny Allow from all </Directory>	Configure the SSL portion of the server with a different document root.

Test the configuration by entering the following command:

```
[root@bastion]# apachectl startssl
/usr/sbin/apachectl startssl: httpd started
```

Attempt to connect to the server with a web browser from a remote host. If successful, the default apache pages will be loaded. Next remove the default pages and place a single HTML file named `index.html` in the `/home/httpd/html` instructing users that all content is served from via https. Create the `/home/httpd/html_ssl` directory and copy the web content that bastion will serve to that directory.

To configure the web server to start automatically when bastion boots up, edit the /etc/rc.local file and add the following lines:

```
# The next line starts up the secure web server
/usr/sbin/apachectl startssl
```

STEP 14 – Setup the web application access to the MySQL database

There is no reason that the database must be present on bastion. In fact, it greatly enhances security of the database server and the database contents to have the system on the internal network, not in the DMZ. The web-based applications on bastion will be written in C and use the MySQL API to connect to the database. The MySQL client software is not required, but is convenient for testing the set-up. If desired, go to www.mysql.org/downloads/mirrors.html and find a convenient mirror site. Obtain the latest MySQL client RPM by anonymous FTP (in this case to mirror.sit.wisc.edu/mirrors/mysql/Downloads/MySQL-3.23) Install the RPM with the following command:

```
[root@bastion]# rpm –ivh MySQL-client-3.23.33-1.i386.rpm
Preparing...                ########################################### [100%]
1:MySQL-client              ########################################### [100%]
```

In order for bastion to access the MySQL server in a secure way, the queries will be sent along an encrypted tunnel thorugh the firewall, set up by ssh. There are several actions required to allow the www user on bastion to access the database in this way. Some of these actions must be taken on the MySQL server and some on bastion. The steps are as follows:

1. On the MySQL server (*mysqlserver.com*), log into the database as the administrative user and issue the following command to grant the www user

privileges to select, insert, update and delete from all the tables in database db. Note that the www user is granted these privileges as if he were accessing the database from the MySQL server host itself. This seems strange but is a result of the SSH tunneling explained later. Also note the password is included at the end of the query.

```
mysql > grant select, insert, update, delete on db.* to www@mysqlserver.com identified by 'TiaRhp2c';
Query OK, 0 rows affected (0.01 sec)
```

2. On the MySQL server, add user account named tunnel with group users to use when bastion is setting up the encrypted tunnel. This user needs a home directory (`/home/tunnel`) but the account may be disabled, as follows. Set the users login shell to /bin/false and put an illegal character (like a *) in the password field for the user in `/etc/shadow`. Also, create a directory `/home/tunnel/.ssh2` to hold the public key of the user on bastion that will be logging in. Make sure the home directory and the .ssh2 directory are accessible only to the tunnel user using the commands:

```
[root@mysqlserver]# chown –R tunnel:users /home/tunnel
[root@mysqlserver]# chmod –R 700 /home/tunnel
```

3. On bastion, add user account named tunnel with group users. This user needs a home directory (`/home/tunnel`) and a login shell (/bin/bash) but password login may be disabled by putting an illegal character (like a *) in the password field for the user in `/etc/shadow`. Create a directory `/home/tunnel/.ssh2` to hold the public and private keys of the user. Make sure the home directory and the .ssh2 directory are accessible only to the tunnel user using the commands:

```
[root@mysqlserver]# chown –R tunnel:users /home/tunnel
[root@mysqlserver]# chmod –R 700 /home/tunnel
```

4. On bastion, generate key pair for user tunnel on client with no passphrase. Copy the private key to a file named `identification` on bastion using the following commands:

```
[root@bastion]# su - tunnel
[tunnel@bastion]$ ssh-keygen -P
[tunnel@bastion]$ echo "IdKey id_dsa_1024_a" > /home/tunnel/.ssh2/identification
```

5. On the MySQL server, copy the public key created in the previous step to the `/home/tunnel/.ssh2` directory and create the file named `authorization` with the following commands:

```
[root@mysqlserver]# cd ~tunnel/.ssh2
[root@mysqlserver]# scp www@bastion:/home/tunnel/.ssh2/id_dsa_1024_a.pub .
[root@mysqlserver]# echo "Key id_dsa_1024_a.pub" > authorization
[root@mysqlserver]# chown tunnel:users * ; chmod 400 *
```

6. On bastion, initialize the encrypted tunnel by issuing the following command which instructs bastion to fork a background process that forwards requests to local port 3306 (the MySQL default) to remote port 3306 on remote host mysqlserver.com:

   [root@bastion]# su - tunnel
   [tunnel@bastion]$ ssh –f –L 3306:mysqlserver.com:3306 mysqlserver.com

7. A user on bastion should now be able to connect to the MySQL database with the following command using the password for www defined in step 1:

   [root@bastion]# mysql –h 127.0.0.1 –u www –p

   If the database connection fails, check that the "`sshdfwd-3306: 127.0.0.1`" appears in the "`/etc/hosts.allow`" file on bastion, as discussed in the tcpwrappers section earlier.

Once the ability to establish the tunnel has been verified, configure bastion to start up automatically at boot by adding the following lines at the end of the /etc/rc.local file:

```
echo "Establishing the encrypted tunnel for MySQL"
/bin/su – tunnel –c "ssh –f –L 3306:mysqlserver.com:3306 mysqlserver.com"
```

This setup has many security advantages. Firstly, all database traffic between bastion and mysqlserver is encrypted. Furthermore, it travels through the firewall with a destination port of 22 (sshd), which is probably already open. Lastly, the tunnel is set up using user accounts on both systems that cannot be logged into with passwords. The main risk is that anyone who obtains the private key belonging to the tunnel user on bastion can create his or her own tunnel through the firewall (a shell login would be impossible since the shell for tunnel is /bin/false on mysqlserver). Therefore, it is important to protect the directory in which it resides, as noted.

For the web application to use this encrypted tunnel, make sure that the API calls are made with the hostname 127.0.0.1, user www and the appropriate password.

Step 15 – Miscellaneous Operating System Changes

15.1 Configure `/etc/inittab`

Modify settings in `/etc/inittab` to disable Ctrl-Alt-Del reboot and to require a password to enter single user mode.

Before:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

After:

```
#ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Add this line after si::sysint…:

```
~~:S:wait:/sbin/sulogin
```

15.2 Remove unnecessary users and groups.

Before making any edits, copy the files to backup versions:

```
[root@bastion]# cp /etc/passwd /etc/passwd.orig
[root@bastion]# cp /etc/shadow /etc/shadow.orig
[root@bastion]# cp /etc/group /etc/group.orig
```

Edit the `passwd` and `shadow` files and remove the lines for the following users: *lp*, *news*, *uucp*, *operator*, *games*, *gopher*, *ftp*. Also, disable login for users *nobody* and *mail* by setting each one's login shells to /bin/false and putting an "!" as the first character of each one's password string. Edit the group file and remove the lines for the groups *popusers*, *pppusers*, *slipusers*, *ftp*, *gopher*, *dip*, *games*, *news*. Also remove the username of any deleted user from the remaining groups. To verify that no mistakes exist that will prevent logging in, run the following commands:

```
[root@bastion]# pwconv
[root@bastion]# pwck
[root@bastion]# grpck
```

15.3 Configure logging and log rotation on bastion.

The `/etc/syslog.conf` file controls to what level and where logging occurs. The following lines should be added to capture additional kernel and auth messages and to configure realtime logging to VTY 7 and 8 (note that spaces must be <TAB> characters:

```
kern.*              /var/log/kernel
auth.*              /var/log/log.auth
kern.*              /dev/tty7
auth.*              /dev/tty7
authpriv.*          /dev/tty7
*.warn              /dev/tty8
```

Once these lines have been added, create the new files and send the syslogd process a HANGUP signal by issuing the following commands:

```
[root@bastion]# touch /var/log/kernel /var/log/log.auth
[root@bastion]# chmod 600 /var/log/kernel /var/log/log.auth
[root@bastion]# killall –HUP syslogd
```

Rotation of log files is handled automatically by `logrotate` on Red Hat systems. Add the following stanzas to the /etc/logrotate.d/syslog file so that the newly created logs will also be rotated:

```
/var/log/kernel {
    postrotate
      /usr/bin/killall -9 klogd
```

```
        /usr/sbin/klogd &
    endscript
}

/var/log/log.auth {
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

15.4 Improve File System Security
Section 4 of [6] outlines the following steps that should be conducted:

- Modify Default File Permissions
  Use the script in Appendix A of [6] to tighten the overly permissive file permissions in the default Red Hat installation. The script changes many directory permissions to remove read and execute access to everyone but the root user (i.e. mode 700)
- Do Not Allow SUID and Devices in User File Systems
  Help to prevent SUID exploits by disallowing SUID program execution on devices where it is not required, like the mount point of `/home`, for example.
- Remove SUID where necessary
  Default Red Hat installations have files like mount with SUID privilege so any user can mount a CDROM and play it, for example. On bastion, this is not necessary and creates an added security risk. The SUID bit should be removed on these files (see [6] for a list).

Step 16 – Remove the compiler and associated packages
Several packages should be removed now that web server and SSH applications have been installed. Use the following command to remove them:

```
[root@bastion]# rpm –e 6.gcc binutils cpp glibc-devel make kernel-headers dev86 patch
ncurses-devel
```

Step 17 – Test and verify the setup
Run the following commands and verify that no unexpected processes appear and no unexpected ports are open:

```
[root@bastion]# ps -ef
[root@bastion]# netstat -p
```

Run nmap from a remote machine and verify that only three ports are listening (22, 80, and 443).

Attempt to connect to bastion via ssh remotely from an IP address that is blocked by TCP wrappers. Does the connection get refused? Is it logged in `/var/log/log.auth`?

Open a web browser on a remote host and attempt to access a page in the `/home/http/html_ssl` directory with using https. Does the page load into the browser?

Open a packet sniffer on the mysqlserver.com while bastion is performing a query. Is the query readable or encrypted?

Step 18 – Backup the pristine configuration
Once all of the verification tests have been passed, backup the configuration several ways. First use a file integrity checker like Tripwire that will capture an MD5 hash, or equivalent, of all of the important files on the system. On bastion, that should be at least the files mounted on `/` and `/home` partitions. For step-by-step setup instructions for Tripwire on Red Hat see section 3 of [6].

Second, use the extra space on the hard disk to create a disk image back-up of the `/` (mounted on /dev/hde1) and `/home` (mounted on /dev/hde2) partitions. First create partitions of equal size (/dev/hdf6 - 1000 MB) and (/dev/hdf7 - 3000 MB), respectively. Use the following commands:

```
# init 1        # go to single user mode
# dd if=dev/hde1 of=/dev/hde6 bs=1k
# dd if=dev/hde2 of=/dev/hde7 bs=1k
# init 3        # return to multiuser mode
# fsck /dev/hde6
# fsck /dev/hde7
```

Finally, create a hard copy backup on tape. Again, good step-by-step instructions can be found in [6] section 8.2. The important points to remember are use the compare feature to verify the back-up and do a test restore on an empty partition. Back-up tapes are no good if the data is not captured correctly or if it cannot be retrieved. Also, remember to store the back-up tape in a safe and secure place, preferable at location where whatever natural disaster destroyed the files on the web server would not also claim the back-up tape, itself.

Step 19 – Turn the system loose
It is now time to deploy the system and expose it to the Internet. Following the steps outlined here will result in a very secure system, but vigilance is still required to maintain that level of security. New vulnerabilities are being discovered daily, and a system left alone will eventually be vulnerable. Monitor appropriate security news groups for relevant vulnerability announcements and keep the software up-to-date.

REFERENCES

[1] Aaron Cline, “Unofficial Asus A7V and Linux ATA100 Quasi-Mini-Howto” (http://www.geocities.com/ender7007)

[2] BIOS update for Intel OR840 Motherboard (http://appsr.intel.com/scripts-df/Product_Filter.asp?ProductID=104)

[3] “SSHSecure Shell for UNIX Servers Administrator’s Guide”, www.ssh.com/support/ssh/index.html

[4] “Securing Linux Step-By-Step”, v.1.0, SANS Institute, 2000.

[5] Acheson, Steve and Pomeranz, Hal, “Topics in UNIX Security”, SANS Security, New Orleans, January 2001.

[6] Gray, Michael “Build a Secure Web Server Using Red Hat Linux Version 6.2 Step By Step”, http://www.sans.org/y2k/practical/Michael_Gray_GCUX.doc