

John Holbrook
Step by Step Installation of a Secure Linux Web, DNS and Mail Server
Feb 10, 2004
GIAC GSEC Practical – Version 1.4b, Option 1

Table of Contents

	Abstract.....	
Introduction.....		4
Current Setup.....		4
Reasons for new install		4
Sudo.....		5
Security Comparison of Redhat 9.0 and Openna 1.0.....		7
Default Installed Services.....		7
Configuration Notes.....		8
The New Setup.....		8
Layers of Protection.....		9
Verifying Integrity of Downloaded Files.....		9
RPMs.....		9
Md5sums.....		10
PGP/GPG Keys.....		11
A Word About Passwords.....		12
Openna Linux 1.0 Installation.....		12
Adding a User		13
OpenSSH Configuration.....		14
MySQL Installation.....		16
Securing MySQL		18
BIND Installation.....		20
Chroot Jailing BIND.....		23
Qmail Installation.....		24
Vpopmail Install.....		32
Apache Installation.....		34
Mod_security Installation.....		39
Mod_Dosevasive Installation.....		40
PHP Installation.....		41
Giptables Installation.....		43
Giptables Customization.....		45
Snort Installation.....		46
MySQL Snort Configuration		48
ACID Installation.....		49
ADODB.....		49
PHPLOT.....		50
JPGRAPH.....		50
ACID Installation.....		50
Authenticated access to the acid pages.....		52
Time Synchronization.....		53
AIDE.....		53
Final Cleanup.....		56
Chattr of key files.....		56

Remove Development RPMs.....56
Removal of Downloaded Files.....56
Autoupdate56
Mailing Lists and other sources of information.....57
Appendix A BIND Configuration File – named.conf.....59
Appendix B – named Initialization Script.....61
Appendix C – Apache configuration options.....63
Appendix D Apache Initialization File65
Appendix E – Apache Initialization file.....67

Abstract

This paper will show how the author configured a Linux based web and e-mail server for a small company. This server is co-located at a local ISP.

Because of budget limitations, the company can only locate one physical box at the ISP which limits what security measures that can be installed. The author will seek to explain the choices made. The paper will include instructions on how to build a secure web and e-mail server with an emphasis on two key security areas:

- 1) Keeping crackers out
- 2) Detecting any signs of cracker activity and limiting the changes a cracker can make

This document expects the reader to have a good understanding of installing Linux and the various tools included for text editing, configuration etc.

Introduction

Current Setup

The currently configured server is a Red Hat 7.2 box running several externally available services:

- Apache 1.3.x Web Server (hosting approximately 10 domains)
- Bind 9.x
- qmail
- Openssh

Reasons for new install

The current server has been in service for approximately 30 months. When it was originally configured the author's knowledge of securing Linux was somewhat limited. Specifically the following items were not installed on the server or configured correctly:

- 1) Firewall
- 2) Intrusion Detection System
- 3) Bind was not configured in a chroot jail

The author has since set up several Linux servers and has standardized on locations for configuration files, etc which make it easier to administer. This wasn't done on the existing server and has caused several problems over the last year or so when updating software.

Another reason for an upgrade is Red Hat has announced the end of life for Red Hat 7.2

as of December 31, 2003 and is discontinuing their freely available download distribution in favor of a commercially packaged version.

Their new free version is now called the “Fedora Project” (<http://fedora.redhat.com/>) but this version is intended for 'bleeding edge' type development, not for a stable, secure web server.

The author looked at several Linux distributions including Mandrake (www.mandrake.com) SuSE (www.suse.com), Debian (www.debian.org) and Openna (www.openna.com). After comparing these distributions, the decision was made to use Openna Linux 1.0 which is available as a free download or can be purchased in a retail package.

Why the author chose Openna Linux:

- Secure distribution. What isn't needed isn't installed by default. With Red Hat the author usually spends several hours disabling unneeded services and removing unnecessary packages.
- All software packages for Openna Linux are compiled for the i686 processor which gives us better performance on newer CPUs
- Prior experience with the creator of Openna Linux – Gerhard Mourani. Gerhard has written several books on securing and optimizing RedHat Linux and Openna Linux which the author has used in the past.

Sudo

Instead of using 'su' (super user) to gain root access Openna uses Sudo.

"Sudo (superuser do) allows a system administrator to give certain users (or groups of users) the ability to run some (or all) commands as root or another user while logging the commands and arguments."¹

Here's an example of how you can fine tune Sudo. I have a user named “bob” who I want to allow to start and stop Apache and make changes to the Apache configuration files under /etc/httpd. Normally, I would have to give “bob” root access by making him a member of the 'wheel' group, give him the root password, and trust that he does not do anything beyond administering Apache. With sudo here's what I can do:

```
# visudo
```

visudo is the administration tool for the sudo configuration file - /etc/sudoers.

Note: Never directly edit /etc/sudoers. Always use 'visudo'.

This is what my /etc/sudoers file will look like on Openna:

```

# /etc/sudoers: OpenNa, Inc.
# This file MUST be edited with the 'visudo' command as root.

# User alias specification
User_Alias    APACHE_ADMINS = bob

# Cmnd alias specification
Cmnd_Alias    HTTP = /etc/init.d/httpd, /bin/vi /etc/httpd/*

# User privilege specification
# Super-user root can run anything as any user.
root          ALL=(ALL) ALL

# Every users member of the group wheel will be allowed
# to run all commands as super-user root.
%wheel        ALL=(ALL) ALL

# Apache admins may administrate httpd
APACHE_ADMINS ALL = HTTP

```

Now to test this I secure shell into the server as user 'bob' and do the following:

```

$ sudo /etc/init.d/httpd restart

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these two things:

#1) Respect the privacy of others.
#2) Think before you type.

Password:
Shutting down httpd:          [ OK ]
Starting httpd:               [ OK ]

```

In /var/log/messages we see the following:

```

Feb  2 13:54:53 server sudo:    bob : TTY=pts/0 ; PWD=/home/bob ; USER=root ;
COMMAND=/etc/init.d/httpd restart

```

Now this is what happens if bob now tries to restart 'sshd' which he is not authorized for:

```

$ sudo /etc/init.d/ssh restart
Sorry, user bob is not allowed to execute '/etc/init.d/ssh restart' as root on
server.domain.com.

```

This unauthorized access is also logged in /var/log/messages:

```
Feb 2 13:59:17 server sudo:    bob : command not allowed ; TTY=pts/0 ;  
PWD=/home/bob ; USER=root ; COMMAND=/etc/init.d/ssh restart
```

Sudo is installed by default on Openna Linux and will allow the sysadmin to fine tune access for other users to administer the server. This is unlike plain 'su' which is an all or nothing proposition.

Sudo can be installed on any Linux distribution and would be highly recommended by the author.

Security Comparison of Redhat 9.0 and Openna 1.0

As mentioned earlier, Red Hat Linux installs quite a few services and packages by default which need to be disabled or removed to heighten security.

What follows is a quick security comparison of base installs of RedHat to Openna Linux.

Default Installed Services

Here's a view of the ports open on a base Red Hat 9.0 Server install with the only packages selected being development and ??

```
# netstat -natp  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local AddressForeign Address  State  PID/Program name  
tcp    0    0 0.0.0.0:32768    0.0.0.0:*      LISTEN 1572/  
tcp    0    0 127.0.0.1:32769 0.0.0.0:*      LISTEN 1702/xinetd  
tcp    0    0 0.0.0.0:111     0.0.0.0:*      LISTEN 1553/  
tcp    0    0 0.0.0.0:22      0.0.0.0:*      LISTEN 1688/sshd  
tcp    0    0 127.0.0.1:631   0.0.0.0:*      LISTEN 1770/cupsd  
tcp    0    0 127.0.0.1:25    0.0.0.0:*      LISTEN 1731/
```

Here's what it looks like on a base Openna install:

```
# netstat -natp  
Proto Recv-Q Send-Q Local Address Foreign Address  State  PID/Program name  
tcp    0    0 192.168.0.50:22 0.0.0.0:*      LISTEN 13527/sshd
```

Only 1 port open on Openna vs. 6 on RedHat 9.0.

This is not to say that Red Hat Linux can't be a highly secure distribution. It just takes more time and work to lock down the base install versus doing the same thing with Openna Linux.

Configuration Notes

For this paper I will be using the following IP Addresses:

192.168.0.50 – The new server

192.168.0.5 – Primary DNS for the local ISP

192.168.0.6 – Secondary DNS for the local ISP

Once all testing and installation is completed then these addresses will have to be changed to the correct internet routable addresses.

The new server will be called server.domain.com. The local ISP is isp.net.

I will also be downloading all source files to /usr/tmp unless mentioned otherwise. At the end of the install, and before the server is connected to the internet, all the packages under /usr/tmp will be removed.

The New Setup

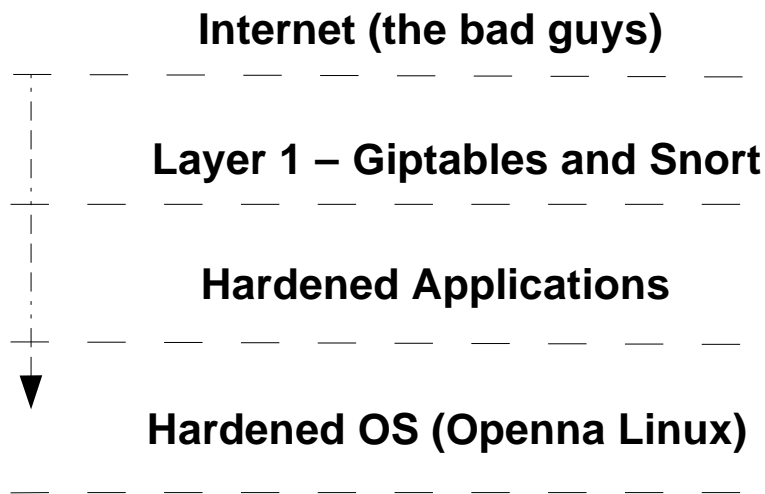
A base install of Openna 1.0 will be done on a clone whitebox PC. All core packages will be upgraded in the future via RPMs.

The following packages will be installed from source format instead of RPMs. The reason for source installs is twofold – ease of customization and new versions of software come out in source much quicker than in RPM/binary format. (The web developer for the server also comes out with some very strange requirements for the server which often mean special compile options)

- Apache 2.0.48 with modsecurity and dosevasive modules
- MySQL
- qmail with vpopmail
- Bind
- Giptables firewall scripts

One thing that you will quickly notice whenever you deal with security is that it is always a balancing act. You have to balance security with functionality. Is it possible to make a system 100% secure? Sure. Take the system, put it back in its original box and lock it in Fort Knox. That isn't a realistic option as we'll have high security and no functionality or usability. The decisions on how to install and what to install in this paper are always made with this in mind. However, security is always going to be a higher priority for the author over functionality.

Layers of Protection



With security, one talks about layers of protection. We never want to have only one layer of protection on our systems. We want multiple layers.

On this installation I will be working with what I perceive as three layers:

- 1) The outer layer which consists of giptables firewall scripts and snort intrusion detection
- 2) The middle layer which consists of 'hardening' the internet accessible applications to increase security
- 3) The lowest layer which consists of securing the core operating system, removing unneeded services and software packages.

Verifying Integrity of Downloaded Files

One very important item, from a security perspective, is to ensure is the integrity of any downloaded source files or RPMs. How can you be sure that the file you download has not been changed by a cracker and had a trojan installed on it?

The rest of this paper will assume that each downloaded file's integrity will be verified using one of the following procedures.

RPMs

RPM (RedHat Package Manager supports) pgp keys. Download the key from the website you download RPMs from. For example, the rpm gpg key for openna is available from <http://www.openna.com/downloads/RPM-GPG-KEY>. Download the file:

The file looks like this (this has been shortened substantially to save on space):

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.3 (GNU/Linux)

```
mQGiBD+7t5sRBADUKbPUlwYUihS1xbPyTCUS7v+TcCFi/uK1uosV/86Ql34Dq06h
9c87HGf6nSDikyUEEC6IIXMKF/dcxCL53L1cgUSf3YJLOS019cxfxkFyN75jJbm
KlviZtL2D2W9TePODKl0z4ziExCXULAUY/d+JMjjDH376PvIv9ojo9lJ0ic9OohJ
BBgRAgAJBQI/u7eeAhsMAAoJEMYYPAr6T2PHamAAn2NuEsVZq1qx+4ZYad4ivWUb
PDX4AJ9ZO+X0Akq6J8oRHu7LEl1kICY94w==
=wYvG
-----END PGP PUBLIC KEY BLOCK-----
```

As an example. First we import the key

```
# rpm --import RPM-GPG-KEY
```

Then to verify a signature on a file we can install with

```
# rpm -Uvh rpmfile-x.x.x.rpm
# rpm --checksig autoupdate-5.2.16-1.i686.rpm
autoupdate-5.2.16-1.i686.rpm: (sha1) dsa sha1 md5 gpg OK

# rpm -Uvh autoupdate-5.2.16-1.i686.rpm
Preparing...      ##### [100%]
1:autoupdate      ##### [100%]
```

If the key wasn't correct we'd see the following:

```
warning: autoupdate-5.2.16-1.i686.rpm: V3 DSA signature: NOKEY, key ID 4b9d15e6
```

Md5sums

The MD5 message-digest algorithm "takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA."²

Here's an example of verifying a md5sum on the modsecurity source file
On the modsecurity website we can download mod_security-1.7.4.tar.gz.md5. The file contains the following:

```
629945812ca7aab4ef2f76ad00172444 *mod_security-1.7.4.tar.gz
```

To verify the integrity of mod_security-1.7.4.tar.gz

```
$ md5sum mod_security-1.7.4.tar.gz  
629945812ca7aab4ef2f76ad00172444 mod_security-1.7.4.tar.gz
```

We see that the two numbers match so we can feel comfortable that the file has not been tampered with. Another step to ensure integrity would be to download and verify the md5sum from several mirror sites (if available).

PGP/GPG Keys

As an example, we'll verify the integrity of our Apache web server source files.

Download the Apache PGP Key from <http://www.apache.org/dist/httpd/KEYS>

Note: On the apache.org website they recommend only downloading the KEYS file from their main ftp server NOT from a mirror.

I downloaded httpd-2.0.48.tar.gz and httpd-2.0.48.tar.gz.asc from one of the Apache mirrors.

First we import the digital key we've received from the website into our keyring.

```
# gpg --import KEYS
```

Now we verify the integrity of the downloaded file.

```
# gpg --verify httpd-2.0.48.tar.gz.asc httpd-2.0.48.tar.gz
```

If you would like further information on using gnupg, an excellent online resource can be found at <http://www.gentoo.org/doc/en/gnupg-user.xml>.

What if the author of the software being downloaded hasn't offered any way to verify the integrity of her software?

You have a few options. Download the software and hope it hasn't been manipulated. Not a really good idea from a security perspective. The author's recommendation (if you are not a programmer who can analyze the code thoroughly) is to download the software from several mirrors and compare the md5sums. If mirror sites are not available then download the software and hold on to it for several weeks before installing it. Keep a watch out for any notifications of security violations to the site on the various mailing lists from the end of this paper. This isn't the best security recommendation but sometimes its what you have to do.

A Word About Passwords

Passwords are often the last, and sometimes only, line of defense. In this paper I will be using rather simple passwords (test123,123test and testing). This is only for simplification of the documentation. Never use passwords like these. Always use good passwords.

"What is a good password?"³

A good password is:

private: it is used and known by one person only
secret: it does not appear in clear text in any file or program or on a piece of paper pinned to the terminal
easily remembered: so there is no need to write it down
not guessable by any program in a reasonable time, for instance less than one week.
week."

The best recommendation I can make is this don't use real words. Use the first letters from a sentence that you'll remember (something from a movie for example), change some characters to upper case and swap some numbers for letters.

As an example, let's take a quote from The Matrix. "What is real? How do you define real?" and we'll use the first letter from each word.

wirhdydr

We'll change every odd consonant to uppercase (we'll say y is a consonant).

WirHdYdR

We'll change the i to a 1 and add a space and a non alpha-numeric character.

W1r HdYdR#

Now we have a password that is private, secret, easily remembered and not easily guessable by any program.

Openna Linux 1.0 Installation

I will not go into a lot of details on the installation of the core Openna system. The installation is fairly self explanatory and there is excellent documentation at⁴. I always chose to manually partition the system and here's the new partition table:

```
dev/hda1 /boot - 50MB
```

```
/dev/hda5 <swap>  
/dev/hda6 / - 1024MB  
/dev/hda7 /usr - 2048MB  
/dev/hda8 /home - 25600MB  
/dev/hda9 /chroot - 512MB  
/dev/hda10 /var - 1024MB  
/dev/hda11 /tmp - 2048MB
```

During the install you are asked to enter a Grub (Bootloader password) This password is now required every time the machine is rebooted and must be entered locally from the console. From a security perspective, it is a great idea to have a Grub password. However, reality dictates that we must be able to remotely reboot the server since we do not have 24 hour a day access to the server and problems have a way of occurring in the middle of the night or on weekends. We'll remove the Grub password.

When the system has rebooted completely do the following:

```
# vi /boot/grub/menu.lst
```

You will see a line that reads something like:

```
password --md5 $1$7eJ380$uaA1zbekvQUclLKYpTVpT0
```

Comment it out by adding a '#' to the beginning of the line or you can remove the line completely.

Additional RPMs which are required:

There are several RPMs that are needed for various packages we're going to install. Instead of installing them individually we'll do one big install.

```
# mount /mnt/cdrom  
# cd /mnt/cdrom/Openna/RPMS  
# rpm -Uvh gnupg-1.2.3-1.i686.rpm autoconf-2.57-1.i686.rpm automake-1.7.8-  
1.i686.rpm m4-1.4.1-1.i686.rpm libtool-1.5-1.i686.rpm openssl-devel-0.9.7c-1.i686.rpm  
freetype-2.1.5-1.i686.rpm freetype-devel-2.1.5-1.i686.rpm libjpeg-6b-1.i686.rpm libjpeg-  
devel-6b-1.i686.rpm libpcap-0.7.2-1.i686.rpm
```

Adding a User

To add a user do the following:

```
# groupadd john  
# useradd -g john -c "John Doe" -m -d /home/john -s /bin/bash john  
# passwd john
```

```
Changing password for john
Enter the new password (minimum of 8, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
Password changed.
```

To allow the user 'john' to sudo to root he needs to be a member of the 'wheel' group:

```
# usermod -G10 john
```

To allow john to log in locally add john to /etc/security/access.conf:

```
# vi +59 /etc/security/access.conf
```

Change

```
 -:ALL EXCEPT root users:ALL
```

to

```
 -:ALL EXCEPT root john:ALL
```

OpenSSH Configuration

In the past, sysadmins used unencrypted communication methods such as Telnet, rsync, and FTP to administer the server. This is unacceptable, when easy to use secure alternatives are available.

To enable encrypted communication, we will use OpenSSH which is already installed on our server but will make some changes to the configuration to secure it even more.

What is SSH?

“SSH (Secure Shell) is a program to log into another computer over a network, to execute commands on a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. It is intended as a replacement for rlogin, rsh, rcp, and rdist.”⁵

Instead of using password authentication, public key encryption will be used as an additional layer of security.

The steps to set up public key encryption with OpenSSH are:

- 1) Create a public/private key pair on our client PC
- 2) Copy the public key to the server and place it in ~/.ssh
- 3) Change the configuration on the server to use public key authentication instead of password authentication.

To generate a 2048 bit DSA key on our client PC:

```
$ ssh-keygen -t dsa -b 2048
Generating public/private dsa key pair.
Enter file in which to save the key (/home/john/.ssh/id_dsa):
Created directory '/home/john/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/john/.ssh/id_dsa.
Your public key has been saved in /home/john/.ssh/id_dsa.pub.
The key fingerprint is:
33:0f:97:4f:d8:f7:a6:44:84:05:a4:64:7a:88:e1:a3john@client.domain.com
```

For the password use a unique password. You do not want to use the same password as your login or root password.

This produces a pair of keys under /home/john/.ssh/ - id_dsa (private key) id_dsa.pub (public key). The private key stays on the client and must be kept secure. The public key needs to be copied to the server and placed under /home/john/.ssh.

```
$ cd ~/.ssh
$ scp id_dsa.pub john@192.168.0.50:~/
john@192.168.0.50's password:
```

Do not use the same password as your login password.

To configure the public key on the server:

```
$ cd ~
$ mkdir .ssh
$ chmod 700 .ssh
$ cat id_dsa.pub > .ssh/authorized_keys2
$ chmod 600 ~/.ssh/authorized_keys2
```

To change OpenSSH to use DSA key encryption instead of passwords on the server:

```
# vi /etc/ssh/sshd_config
```

```
Change:
Protocol          1,2
to
Protocol          2
```

```
Change:
ListenAddress     0.0.0.0
to
```

ListenAddress 192.168.0.50

Change:

RhostsRSAAuthentication yes
to
RhostsRSAAuthentication no

Change:

RSAAuthentication yes
to
RSAAuthentication no

Change:

PasswordAuthentication yes
to
PasswordAuthentication no

Restart OpenSSH

```
# /etc/init.d/sshd restart
```

Now when you ssh from your client to the server you will enter the password from your DSA key.

```
$ ssh server.domain.com  
Enter passphrase for key '/home/john/.ssh/id_dsa':
```

Public key encryption increases the level of security on the server because for somebody to secure shell into the box they must have two things – the password on the DSA key and a copy of the DSA private key.

If you try to secure shell into the server and don't have the correct information you will receive the following:

```
$ ssh server.domain.com  
Permission denied (publickey,keyboard-interactive).
```

Another way to protect our OpenSSH traffic would be to block access to the service to specific IP addresses using either tcpwrappers or our Giptables Firewall scripts which we install later. Unfortunately, this is not possible as the users who require access are using cable modem for internet access which do not offer static IP addresses.

MySQL Installation

Download mysql-max-4.0.17-pc-linux-i686.tar.gz from one of the mysql mirrors at

