



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

# **“Securing a Linux djbdns DNS Server”**

**By  
Matthew P. Brown**

**GIAC Practical Assignment Version 1.9  
August 27, 2002**

## Introduction

GIAC Enterprises is an e-commerce company that sells online fortune cookie sayings. Their business is done virtually minute by minute with each second either making them or costing them money. To insure maximum uptime the focus is on the three pillars of security, Confidentiality, Integrity, and Availability, which is the well know CIA model. Hosting a web service forces GIAC to directly deal with all aspects of the CIA model, in that the possibility of a vulnerability to the DNS software could cause either the data to be compromised losing its confidentiality and availability or for the service to be brought down entirely losing its availability. Understanding that these threats are a reality, the choice was made to go with the most stable DNS software available and then spend the effort to lock down my operating system.

This software is called djbdns and was written by D. J. Bernstein (<http://cr.yp.to/djbdns.html>, [www.djbdns.org](http://www.djbdns.org)). At first you see the name and it seems to carry no weight, but there is more to the software than its name. At first, BIND was going to be the DNS software that was implemented, but after researching the types of DNS software solutions the decision was made to go with djbdns. I did and this paper is about my experience implementing djbdns for GIAC, why you should use djbdns, and how to configure it correctly.

## Description of the System

The system will run Redhat Linux 7.2 on a Compaq Proliant server. Linux is free, the hardware is inexpensive and the staff has experience in running this version of Linux. Having experience running an operating system will only help with its implementation and administration.

### Hardware:

The hardware for the project is being recycled from a server that previously ran a Window's application. It is about two years old and still has the horsepower to provide the DNS service. Compaq has been the hardware of choice for years at GIAC and has proven itself over time. The hardware details are as follows:

<b>Server Type:</b>	Compaq Proliant 1850R
<b>Processor Speed:</b>	Pentium II 400 Mhz
<b>Memory:</b>	192 MB RAM
<b>Hard Drive(s):</b>	2 mirrored 4.3 GB hard drives

### Software:

Redhat Linux 7.2 is the host operating system for the DNS server. While Redhat 7.3 has been recently released, management desires not be on the bleeding edge so it will have to wait. OpenSSH is a version of SSH maintained by some of the OpenBSD project members, [www.openssh.org](http://www.openssh.org). This is the

version that is widely implemented in the Linux community and as such will be used for communicated to and from the DNS server. Djbdns is the DNS software that will be used to run the DNS service while daemontools is the set of tools used to monitor the DNS service. The following is an exact listing of the versions:

**Operating System:** Red Hat Linux 7.2  
**SSH:** OpenSSH 3.1p1-6  
**DNS Software:** djbdns 1.05  
**Monitoring Tools:** daemontools 0.76

At [www.compaq.com](http://www.compaq.com) there is a matrix that lists the certified version of Linux on each server platform. This might be a very important issue to your supervisor or manager. Redhat 7.2 is certified at kernel version 2.4.7-10 on the Compaq Proliant 1850R. Compaq's technical support states that these certifications are dependent on Compaq's set of monitoring tools and what version of the kernel they have been tested against. Thus, there are no fears then about patching the kernel when the time comes.

#### Purpose of Host on the Network:

Hosting web applications such as email or web services require a publicly accessible DNS server. There are two main reasons that GIAC wishes to now host the public DNS server locally. First, the benefit of hosting the DNS server locally is the convenience of making changes when and how they need to be made while not having to wait on the ISP to make the changes. This was usually at midnight of that day or the following day. GIAC will use the tinydns service of djbdns to answer name resolution queries from the public Internet.

Secondly, GIAC's internal network Window's based with Windows 2000 domain controllers implementing Active Directory. It is recommended by Microsoft that these Domain Controllers run DNS so that Dynamic DNS will function. GIAC has no desire to expose these DNS servers directly to the Internet. Instead, GIAC will use the dnscache service of djbdns to act as the DNS caching server for our internal DNS servers. This only causes a small configuration change for the Domain Controllers in that they will now forward requests to the new server in the DMZ instead of actually performing the queries themselves. No configuration changes will have to be made on the client side as they will still point to the Domain Controllers for their DNS needs.

### **Risk Analysis of the System**

GIAC Enterprise's management is firmly committed to providing the online selling of fortune cookie sayings. To do so they must provide public web servers and possibly other services in the form of FTP or SMTP. The securing of these services is not our concern for now. The main concern is the fact that the DNS server daemon will be exposed to the Public internet.

To fully understand what risks face our server we need to delve a little bit into the network architecture surrounding it. We see that a firewall stands in between the DNS server and the Internet. This firewall answers for the public IP address of the DNS server and will then forward the request on to it. This insures that only udp port 53 is exposed to the Internet.

Given the role of the host as described above, what are the key security concerns? What are the primary threats to the system? How “secure” does it need to be? All are good questions and all will be answered.

#### **4 Key Areas:**

- 1: DNS daemon
- 2: SSH daemon
- 3: Physical Access
- 4: Configuration Errors

#### **DNS daemon:**

The primary threat to the system lies in the daemon(s) that faces the public Internet. Tinydns is the Name Resolving service part of djbdns that will answer name queries from the Internet. The major risks of hosting a DNS server are that of bugs, vulnerabilities, and other security holes that would allow someone to gain access to the server either in the form of root access or a general user with the opportunity of gaining elevated privileges. They could then change the DNS records, DOS the service by just disabling it, or they could attempt to gain access to other parts of GIAC's network.

One often hears of cache poisoning that affects some DNS server where someone has been able to change a record(s). GIAC cannot have its information altered or corrupted in any way. It would be disastrous if someone was to go to [www.giac.com](http://www.giac.com) and end up at a competitor's site or another site that held somewhat contrary views to GIAC, such as one that said fortune cookie sayings were made in some European country and that it is all a farce. An example of this would be the misdirection of a pro Hillary Clinton website during the 200 campaign that took you to a website that was totally anti-Hillary. Something of this magnitude cannot be allowed to happen and the risk is there while the DNS daemon is answering queries on the Internet.

#### **SSH daemon:**

While the DNS service is the only daemon that is exposed to the public Internet, the SSH daemon is open through the firewall from the internal network. In the past few months there have been several vulnerabilities in OpenSSH (such as the recent Redhat Security Advisory RHSA-2002:127-18 which contained an integer validation error that could result in an integer overflow or privilege escalation) which is the brand of SSH that we are implementing. It is often quoted that, on average, about 70-80 percent of hacking attempts come from inside your network. Having a buggy version of OpenSSH would only give them

more of a chance to hack the DNS server. While the SSH daemon is enabled on the server, the firewall only allows access from a certain IP address range that has been set for administrators. Granted, people can spoof an IP address which would allow them access through the firewall but it does help us control access. Compared to using telnet or ftp the security has been greatly increased through the use of encryption. The configuration of SSH will be covered in full detail later in the report.

### **Physical Access:**

Physical access must be a main focus in the security policy. If someone was able to freely walk into a bank's vault, then they might just start taking some of the money. The same paranoia must exist when it comes to restricting physical access to the server or console room. Server and console rooms refer to **any** room that has physical access to a server or a console that is directly connected to a server. If someone is able to walk up to a console that is not password protected they might be able to perform a number of undesirable things including rebooting the server. This is possible in Windows and Linux if the system is not properly configured. Other areas of concern are the CD-Rom and floppy drives, the network cable, and the power cables, which all have the potential for disaster if not restricted properly. Restrict access to these rooms and others with keypads, magnetic card readers that are unique to each person, biometrics, and/or the old lock and key approach. Someone unplugging a cable is just as much of a threat to your business as running a vulnerable version of the DNS or SSH server.

### **Configuration Errors:**

Configuration errors are the most prominent threat to any system. Configuration errors can come in the form of a remote access service such as SSH that allows a user to directly connect as root. It has been done many times where in the installation process the password is made easy to remember so that it won't be forgotten and is never changed. If there is a recent patch to the SSH server that goes unapplied, then a crafty user may apply code that exploits an integer validation bug and they gain root access to the system, game over. Leaving non-key services running that access software that is not patched, not patching your key software for the daemons that runs on the Internet or on the network, and having no password policy are all ways of imposing major security threats to your system.

The question remains, "how secure does it need to be?" For the public DNS server the question should rather be "how secure **can** it be?" The proper approach to a server of this type is similar to the proper firewall rule base approach; deny everything except that which is explicitly permitted. A "default deny" approach would imply that the only services running are the ones that have to be running in order for the DNS service to function properly. This approach would even imply that SSH is turned off and the only access is via a console

cable or similar means. Of course, the extremity that one goes to depends on the need and thus SSH will be allowed to run as long as it is constantly monitored. The same is true of all the packages installed on the system, especially the DNS and SSH server packages.

If an administrator tells you he would rather have ftp and telnet turned on since he or she is used to using them and does not want to learn SSH, tell them tough, and point them to the O'Reilly book, "SSH, the Secure Shell: The Definitive Guide" by Daniel J. Barret and Richard E. Silverman where they can learn all they want about scp, sftp, and ssh.

## Step by Step Guide

The step by step guide to installing Red Hat Linux 7.2 will be broken up into four main segments:

- 1: Installing the operating system
- 2: Updating the packages
- 3: Installing and configuring the software
- 4: Locking down the system

The installation of a Red Hat Linux server has become so user friendly that any novice can easily glide through it using only the installation manual or any document such as this. If someone is unable to commercially purchase a copy of Redhat, then you can easily download the ISO images from Redhat or one of their mirror sites. At <http://www.redhat.com/download/mirror.html> is where the download mirror locations are listed. A favorite site of mine is the one hosted by Georgia Tech, <ftp-linux.cc.gatech.edu> where they offer a great download speed in the evenings. Redhat has instructions posted on their website at [http://www.redhat.com/download/howto\\_download.html](http://www.redhat.com/download/howto_download.html) which tell you all you need to know about downloading Redhat.

### Installing the Operating System:

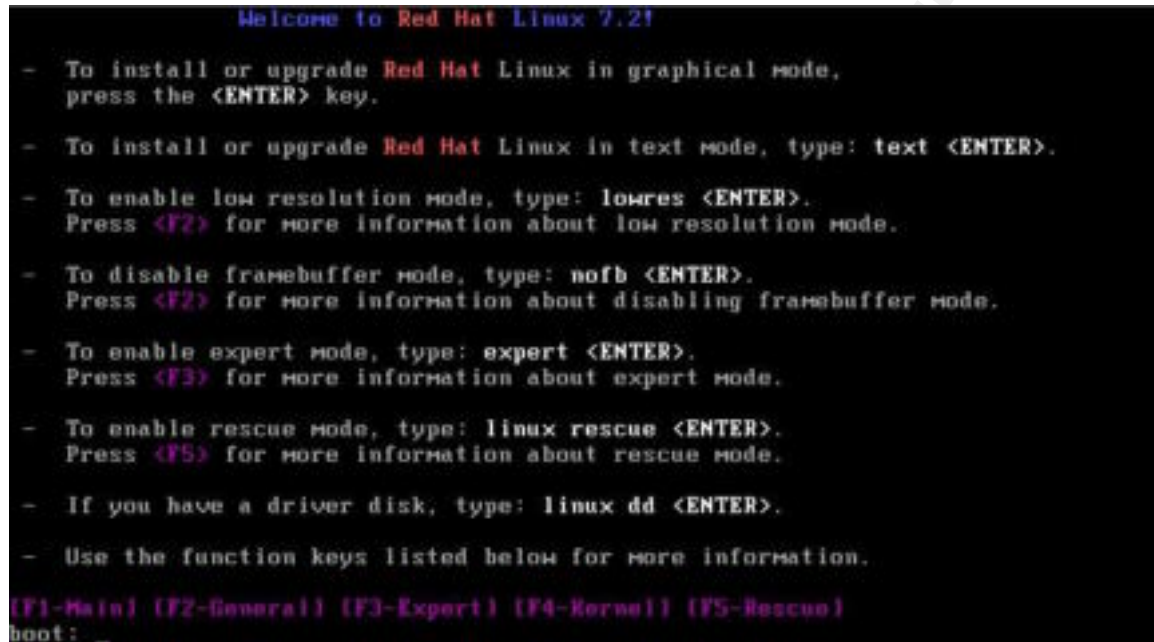
Hopefully you have a server that will boot off of the CD-Rom, so start out by configuring the BIOS to allow this. If you are unable to boot from the disk, you can make a boot disk by using Red Hat's utility, rawrite, and the Red Hat CD. Assuming you do this from a Windows machine, here are the steps taken from the Red Hat 7.2 Installation Guide and assuming that your CD-Rom is the D: drive:

```
C:\> d:
D:\> cd \dosutils
D:\dosutils> rawrite
Enter disk image source file name: ..\images\boot.img
Enter target diskette drive: a:
Please insert a formatted diskette into drive A: and
press --ENTER-- : [Enter]
D:\dosutils>
```

For more information check out the following link:

(<http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/install-guide/s1-steps-install-cdrom.html#S2-STEPS-MAKE-DISKS> – reference)

After booting off of the CD-Rom or off of the floppy you are presented with a menu with many options, you want to just press **Enter** here to proceed with a graphical installation.



```
Welcome to Red Hat Linux 7.2i

- To install or upgrade Red Hat Linux in graphical mode,
  press the <ENTER> key.

- To install or upgrade Red Hat Linux in text mode, type: text <ENTER>.

- To enable low resolution mode, type: lowres <ENTER>.
  Press <F2> for more information about low resolution mode.

- To disable framebuffer mode, type: nofb <ENTER>.
  Press <F2> for more information about disabling framebuffer mode.

- To enable expert mode, type: expert <ENTER>.
  Press <F3> for more information about expert mode.

- To enable rescue mode, type: linux rescue <ENTER>.
  Press <F5> for more information about rescue mode.

- If you have a driver disk, type: linux dd <ENTER>.

- Use the function keys listed below for more information.

(F1)-Main (F2)-General (F3)-Expert (F4)-Kernel (F5)-Rescue
boot: _
```

Note: If you have a system that does not have a good video card, for instance installing Red Hat in a vmware session, then you could use the **text** mode installation, but that is not as intuitive as the graphical one.

The next few screens ask you some basic questions:

**What language would you like to use during the installation process?**

- English

**What model keyboard is attached to the computer?**

Model:

- Generic 105-key PC (or whatever is the default detection)

Layout:

- U.S. English

Dead Keys:

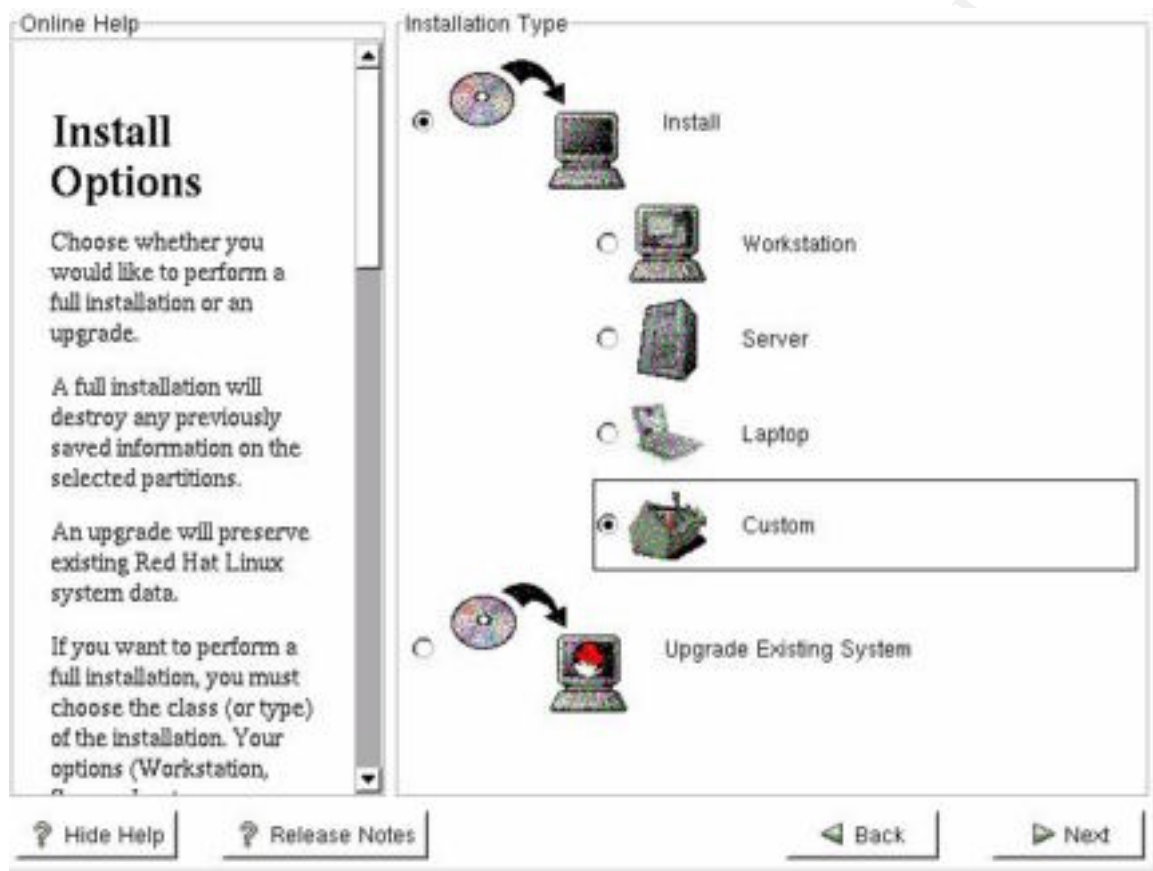
- Enable dead keys

**What model mouse is attached to the computer?**



- Generic 3 button mouse (PS/2)

So far the decisions have been pretty intuitive and most have been the default selections. Now we come to a welcome screen which is where our blood pressure begins to rise as our excitement mounts. Proceed forward to the next screen where you are faced with our first big decision, **which installation type** to choose?



There will not be many graphics but this is an important one, because you do not want to pick the wrong one here.

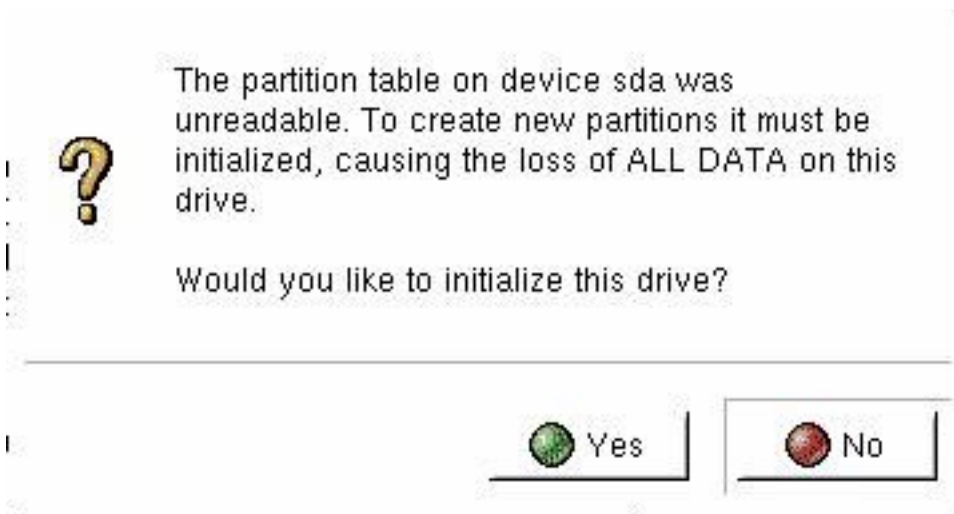
### Installation Type:

- Custom

Pick the **Custom** installation method so that you can control how the disk is partitioned and what packages are initially installed.

### Disk Partitioning Setup:

- Manually partition with Disk Druid



You do want to **initialize this drive** to insure that our installation will be clean. It is recommended to keep partitioning simple and throw most everything into root (/) instead of attempting a more complex one where you could later shoot yourself in the foot by running out of space on any specific partition.

#### Disk Setup (in Megabytes):

/boot	50
/var	400
swap	384
/	3450 (approximately),

For / chose the **fill to maximum available size option** instead of specifying it exactly. This setup is a little unique and is the one you can get from Lance Spitzner, ([www.enteract.com/~lspitz/linux.html](http://www.enteract.com/~lspitz/linux.html)). He suggests creating a /var partition of around 400 Megs for logs so if for some reason your logs go out of control then it will not cause a Denial of Service on your whole system. We shall use the same philosophy on disk partitioning.

We then come to the boot loader configuration.

#### Boot Loader Configuration:

- Use GRUB as the boot loader

Install the boot loader on:

- Master Boot Record (MRB)

Default boot image:

- Red Hat Linux

Choose to **password protect** the boot loader. Here it would be easy to either enter a user password or the root password, don't do this! Even though the password will be encrypted when it is stored on your disk, it is strongly recommended to use a third password here.

### Boot Loader Password Configuration:

- check **Yes** and enter your GRUB password

Now we come to the next big decision which is the network configuration. We are only using one Ethernet connection, **eth0**, since that is all the server has and this is all we will need.

### Network Configuration:

Uncheck **Configure Using DHCP** since it will be statically assigned. The network connection should become active at boot so leave **Activate on Boot** checked. This is necessary since it is desired to have the DNS service startup in case of a reboot.

IP Address	192.168.0.7
Netmask	255.255.255.0
Network	192.168.0.0
Broadcast	192.168.0.255
Hostname	green.giac.com
Gateway	192.168.0.1

Red Hat attempts to fill in as much of the information as they can automatically when the ip address is first entered. Be careful and always check what they do especially for the **Gateway** and **Primary DNS**, which is sometimes incorrect.

### Firewall Configuration:

Select **No Firewall** since we will manually be configuring the firewall later on.

### Additional Language Support:

Choose the default language for this system:

- English

Choose additional languages you would like to use on this system:

- English (USA)

If you have other languages you would like to support this is the place to add them.

### Time Zone Selection:

America/New\_York, Eastern Time

Do not use the **UTC** feature as we want our clock to reflect our own time zone.

## Account Configuration:

The **root** account is the built in system account similar to **Administrator** on Windows. It should be a standard practice never to let that be your only account on any system, so you need to add at least one other account.

User Name: mbrown  
Full Name: Matthew Brown  
Password: \*\*\*\*\*  
Confirm: \*\*\*\*\*

## Authentication Configuration:

Click **Next** to accept the defaults which have MD5 and shadow passwords enabled by default and does not have NIS, LDAP, Kerberos or SMB.

Now we come to perhaps the most critical component of the system, the package group selection. For those with much experience and know each package and its impact and need on the system then you can check the box at the bottom of the page for **Select individual packages**, but the novice administrator may find the experience one more of breaking the system by not putting on something necessary that is not fully understood yet.

The biggest problem with the package selection process is that you want to have all the features that you are used to so you can manage your server, locally and remotely. The other idea is to install ONLY that which is necessary for the basic functioning of the server to limit the impact of vulnerabilities on your system. For example, people like to use ftp to move packages back and forth from a server and on Linux it is easy to install such a FTP server, but if a new security hole comes out in the wu-ftpd ftp daemon package that you are in a bad situation. Whereas if you give in to the knowledge curve and use SCP, the tool that comes with most version of SSH, you no longer need the ftp server as your file transfer can take place through SSH.

This is part of the reason we chose a **Custom** installation method, we did not want to have any server packages installed without our knowledge. They could be outdated and need to be patched. There are threats with installing the **Software Development**, the **Utilities**, and the **Network Managed Workstation** packages, but it has been decided that it is a necessary risk in maintaining the server.

## Package Group Selection:

X-Window System  
GNOME  
Network Support

Network Managed Workstation  
Router/Firewall  
Utilities  
Software Development  
Kernel Development

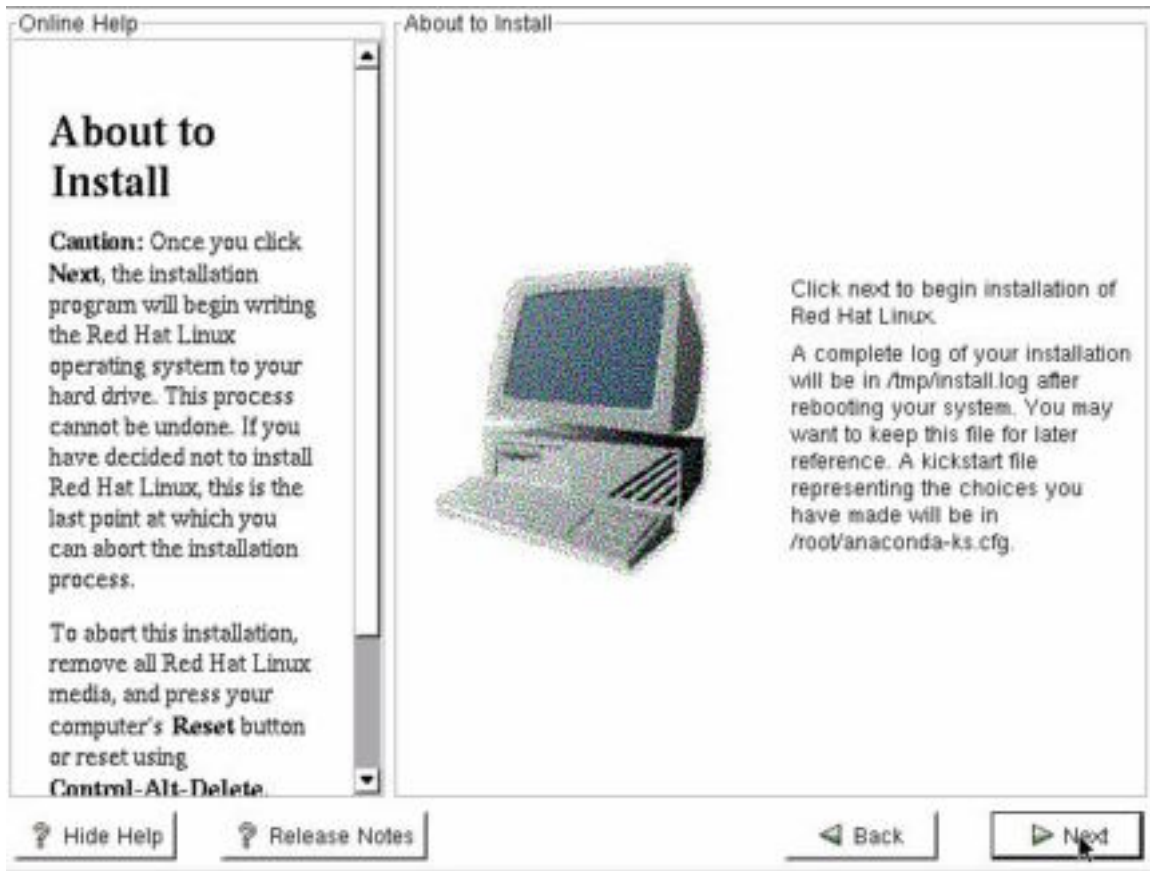
There is no need for printer support, sound support and anything that does not have to do with accessing the network or running our DNS service. As mentioned earlier we could have gone through and selected only the packages from each of the above groups but it is easier, and this ease is necessary, and simpler just to install these groups. GNOME has been installed for local management and if you are not a wizard at the Linux command line then it is necessary for you to install a Window manager like GNOME, KDE, or Windowmaker.

### Graphical Interface (X) Configuration:

ATI Mach64 3D Rage II is the video card for my Compaq server. Select the appropriate one for yours.

Video Card RAM:  
2 MB

Here we are at the end of the installation but it is really just the beginning of the work to secure the DNS server. For a moment let's sit back and take a break and let the computer do some work for us. The next screen shows the **About to Install** option, click on **Next** and let it go to work. This is the perfect moment to get up and walk around, take a break, or whatever you want to do. You will have around 10 to 15 minutes before you have to put in disk 2, but don't worry it will tell you when you have to. The installation will begin by formatting the hard drive and then will transfer the installation image to the disk from the CD-Rom, and then away it will go.



I hope that you a good break. After the installation completes you will have the option to create a boot disk or not.

### Boot Disk:

Check the box creating a boot disk and insert a brand new floppy disk into the floppy drive. This is often overlooked but can save the day if your MBR becomes corrupt.

### Monitor Configuration:

Select the appropriate monitor and the right horizontal and vertical syncs. My monitor is a HP Ergo 1280 with a 17-inch display.

### Graphic Configuration:

This part depends somewhat on preference and somewhat on the power of your video card. To keep it simple and as general as possible chose **High Color (16 bit)**, with a resolution of **800x600**, and the login type as **text**. It is important to not have the graphical login on this type of server so as to not start X on startup. It is also recommended that you do not test these settings as it has been known to crash the installation if it is an unsuccessful test, you will find out soon enough

when you reboot. When you login all you have to do is type **startx** and the command line and this will start GNOME.

Red Hat Linux 7.2 has now been installed, reboot and get logged in. Verify that GNOME starts up and once it has open a gnome terminal by both right-clicking and selecting New Terminal or by clicking on the icon that looks like a computer screen with the GNOME foot on the corner. The terminal or shell is where we will be doing the majority of our work.

### **Updating the Packages:**

Updating packages in any operating system can be done by downloading the specific update and then manually applying the package and by doing so have total control over what is installed or updated. The other way is by using an automated service such as Windows Update or Red Hat's up2date. For the novice admin this is where she or he should start. If you purchase a copy of Red Hat instead of downloading the iso images then also have purchased an entrance to Red Hat Network, a system which allows the graphical updating and record keeping on your server(s). There is a DMZ off of the firewall which is trusted, in that nothing else is plugged into it. I plugged the network interface of the DNS server into this DMZ and configure the firewall to allow **http** and **ftp** access outbound to the internet from the new server.

Once you are able to access the Internet from your new Red Hat server, you must first go login at <https://rhn.redhat.com> and entitle your new server. Once that is complete you will have access to Red Hat's up2date ftp server through a GUI client on your machine. Since we installed the groups of packages, i.e. Network Managed Workstation, we probably got some packages installed that we do not need and for the novice admin going through and selecting the packages individually will be tedious and possibly disastrous.

We will thus take the risk of possibly updating packages that we won't need for the advantage of the ease of use. The method is available and easy to use. You want to be as careful as you possibly can, that is why there is an alternative, and you can do this the manual way if it is preferred. It is the way that the Center for Internet Security recommends ([www.cisecurity.org](http://www.cisecurity.org)). Run the following commands to update the packages manually:

```
mkdir /usr/local/updates
```

```
cd /usr/local/updates
```

```
wget ftp://updates.redhat.com/7.2/en/os/i386/*.rpm
```

```
wget ftp://updates.redhat.com/7.2/en/os/noarch/*.rpm
```

**(Linux Benchmark v1.0.0, p.4)**

Once you have downloaded all of the available updated rpm packages, you want to install the ones you need. Some people will simply run **rpm -ivh \*.rpm** or **rpm -Uvh \*.rpm** and let it get everything, but what we want to do is run

**rpm -F \***

which will only “**freshen**” the rpm’s that are already installed and will not “accidentally” install something we do not know about? Again, I recommend this only for experienced administrators.

Once you have updated all of your packages you have to think of a daily/weekly/monthly scheme to keep everything updated. If you use the Red Hat Network then you can give them your email address and they will send you alerts when there are package updates. This is a good but is also a good idea to check their website and others on a regular basis for any security releases. When you are done installing the packages you can remove the whole directory since you will not need it now and also to conserve disk space.

**cd ../rm -fr updates**

**-f, --force**

**ignore nonexistent files, never prompt**

**-r, --recursive**

**remove the contents of directories recursively**

This is one way of moving up a directory and then deleting the entire updates directory without it prompting you for each file.

### **Installing and configuring the software:**

#### **DNS:**

If you paid close attention to the security news lately or to the package list that you downloaded then you probably noticed the recent Denial of Service vulnerability in the most well known and used DNS software, BIND. Here is what Red Hat listed on their errata details:

BIND (Berkeley Internet Name Domain) is an implementation of the DNS (Domain Name System) protocols. Versions of BIND 9 prior to 9.2.1 have a bug that causes certain requests to the BIND name server (named) to fail an internal consistency check, causing the name server to stop responding to requests. This can be used by a remote attacker to cause a denial of service (DOS) attack against name servers.

Red Hat Linux 7.1, 7.2 and 7.3 shipped with versions of BIND vulnerable to this issue. All users of BIND are advised to upgrade to the errata packages containing BIND 9.2.1 which is not vulnerable to this issue



([https://rhn.redhat.com/network/errata/errata\\_details.pxt?eid=1085](https://rhn.redhat.com/network/errata/errata_details.pxt?eid=1085))

As you can see, if we had installed any version of BIND previous to 9.2.1 then we would have been faced with a very serious situation. This is just one example of a recent BIND vulnerability, there is and have been many others. This section will go into detail as to why your fears should be at ease while using djbdns.

At <http://cr.yp.to/djbdns.html> we find Dan Bernstein's own website for djbdns with many documents and description of his tools. I learned how to install djbdns from reading this website and so it makes sense that the next few steps come straight from his website and I take no credit for them. To install and run djbdns, in particular the **dnscache** and **tinydns** services, you first need to install Dan's set of tools called **daemontools**. This set of tools will start, monitor, and control the dnscache and tinydns service on your machine. Stop, start, kill, restart, you name it and daemontools will perform it and it will do it for any service, not just djbdns. More information about daemontools can be found at <http://cr.yp.to/daemontools.html>. So here is how we install these wonderful tools.

### **Installation of Daemontools:**

Create a /package directory

```
mkdir -p /package
chmod 1755 /package
cd /package
```

*Why?* You should notice that the permissions on this directory are world readable but only owner writeable, and the owner is root.

Download daemontools from <http://cr.yp.to/daemontools/daemontools-0.76.tar.gz> into **/package**.

```
gunzip daemontools-0.76.tar.gz
tar -xpf daemontools-0.76.tar
rm daemontools-0.76.tar.gz
cd admin/daemontools-0.76
```

Then run the following command to compile and install daemontools.

```
package/install
```

Now you have daemontools installed on your system and are ready to begin the installation of dnscache and tinydns.

*Why?* It is very important that you create the directory exactly as was shown here to insure that the proper permissions are set for the dnscache users to be able to operate properly.

The previous installation steps were adapted from <http://cr.yp.to/daemontools/install.html>.

### **Installation of djbdns:**

Download djbdns from <http://cr.yp.to/djbdns/djbdns-1.05.tar.gz> into a download directory that you normally use and perform these simple commands.

```
gunzip djbdns-1.05.tar.gz
tar -xf djbdns-1.05.tar
cd djbdns-1.05
```

Now compile the program with the following command:

```
make
```

Become root and run

```
make setup check
```

and you installed djbdns version 1.05.

*Why?* The installation of djbdns needs no specific directory permissions like daemontools did.

The previous installation steps were adapted from <http://cr.yp.to/djbdns/install.html>.

### **Configuration of dnscache:**

Now that we have daemontools and djbdns installed we are ready to begin the main part of our whole mission, install a DNS server that acts as a primary name server for the giac.com domain and a DNS caching server for GIAC's internal network. We will take this in two parts, first the configuration of dnscache and second the configuration of tinydns.

#### **dnscache**

There are two ways to implement dnscache

- 1) local cache
- 2) external cache

The local cache would run on the local host address of 127.0.0.1 and only cache DNS queries for that machine, while the external cache would run on the 192.168.0.7 ip address and handle DNS queries from any network that you

permit. This is useful if you only need a DNS server for that one machine or for your own for that matter. Instead, we will be installing an **external cache** to handle DNS queries from GIAC's internal network.

To run dnscache you first need some user accounts on your system to run these services chrooted as that user. Basically we are going to create some users whose only right will be to run that particular service and that is it. So, create the users **dnscache** and **dnslog**.

```
/usr/sbin/useradd -s /bin/false dnscache  
/usr/sbin/useradd -s /bin/false dnslog
```

*Why?* You should notice that the users have the shell **/bin/false** which will not allow them to perform any function that would be dangerous. They will only be allowed to run their particular service and do nothing else.

Then begin configuring dnscache with the built-in dnscache-conf program:

```
/usr/local/bin/dnscache-conf dnscache dnslog /etc/dnscachex 192.168.0.7
```

The following command line calls **dnscache-conf** and passes it the user who will run the dnscache service, **dnscache**, the user who will run the logging service, **dnslog**, it also creates a directory called **/etc/dnscachex** and defines the ip address **192.168.0.7** for which the dnscache service will listen on the network.

Now that we have created your directory for which dnscache will run under we need to create a symbolic link from our directory to the **/service** directory so daemontools will know about it and take care of it for us.

```
ln -s /etc/dnscachex /service
```

Once we have done this, svscan (another utility in daemontools) will start the dnscache service within 5 seconds.

For more information on daemontools such as svscan, consult <http://cr.yp.to/daemontools.html>.

Now we have a dns caching service up and running but if you point your machine to it you will find that it does not answer. For dnscache to answer queries for any specific ip or network you must add that ip/network in the **/service/dnscachex/root/ip** directory with the following command(s):

```
touch /service/dnscachex/root/ip 10.10
```

this will allow the 10.10.0.0/16 network to access the dns caching service. Adding or removing networks is as easy as adding this type of file or removing

this type of file with no restart of the service needed. If GIAC was to have any other servers in their DMZ that they wanted to use this service then they would simply add the line:

**touch /service/dnscachex/root/ip 192.168.0**

This tells dnscache to accept queries from the 192.168.0.0/24 network. If you wanted to permit only specific hosts then you would “touch” a file with that machines ip address.

The log files for dnscache are kept in the **/service/dnscachex/log/main** directory if you need to check on any recent queries. Most administrators that I have learned from use the **tail -f LOGFILE** way of looking at logs as they happen and this is not a bad way, but daemontools comes with two tools, **tai64n** and **tai64nlocal**.

The tai64n utility reads lines from stdin and prints a precise timestamp for each line and the tai64nlocal utility does the same thing but prints each line in a human readable form. An example of tai64n would be the following:

**tail -15 /service/dnscachex/log/main/current | tai64n**

```
[root@green main]# tail -5 /service/dnscachex/log/main/current | tai64n
@400000003d3826e21b8fa80c @400000003d3826c32ab48884 sent 93015 113
@400000003d3826e21b8fc74c @400000003d3826c416c235c4 query 93016
0a0a1e2a:0f1d:3880 1 moneycentral.msn.com.
@400000003d3826e21b8fe2a4 @400000003d3826c416c260bc cached cname
moneycentral.msn.com. moneycentral.com.
@400000003d3826e21b8ffa14 @400000003d3826c416c27ffc cached 1
moneycentral.com.
@400000003d3826e21b901184 @400000003d3826c416c2976c sent 93016 113
```

while an example of tai64nlocal would be

**tail -15 /service/dnscachex/log/main/current | tai64nlocal**

```
[root@green main]# tail -5 /service/dnscachex/log/main/current | tai64nlocal
2002-07-19 10:48:25.716474500 sent 93015 113
2002-07-19 10:48:26.381826500 query 93016 0a0a1e2a:0f1d:3880 1
moneycentral.msn.com.
2002-07-19 10:48:26.381837500 cached cname moneycentral.msn.com.
moneycentral.com.
2002-07-19 10:48:26.381845500 cached 1 moneycentral.com.
2002-07-19 10:48:26.381851500 sent 93016 113
```

I always use the **tai64nlocal** utility since it outputs the logs in a human readable format.

Now that we have dnscache up and running and are able to look at the logs to monitor how it is running we can proceed to configuring our name server using **tinydns**.

### **IP Address Problem:**

Before we begin our configuration of **tinydns** we must first deal with a problem that we now have, the fact that tinydns and dnscache will not run on the same IP address.

**PROBLEM:** Dnscache and tinydns will not run on the same ip address.

**SOLUTION:** We will add a second IP address to our single Ethernet NIC.

There are several philosophies when it comes to running a DNS caching and name resolution server on the same machine. One of them is to run them on separate machines, but since we only have the one server to work with we will have to provide an alternative solution. The method we will use is to add a second IP address to our server's NIC. One way to do this is outlined in a paper by Anton Chuvakin.

<http://www.tldp.org/HOWTO/ISP-Setup-RedHat-HOWTO.html>

In section 4.6 of his paper "“Pocket” ISP based on Redhat Linux HOWTO,” Anton describes how to add “virtual” IP address to one network interface card. The first thing is to verify that your kernel version supports multiple IP addresses, and since we are using kernel 2.4.7-10 we are ok. If it is not supported you can look at his paper since he goes into some detail about how to turn it on in the kernel source code. I will let you read that if you feel it necessary since Red Hat 7.2 has it already built in. We will add a second address of 192.168.0.9 in the following command:

**/sbin/ifconfig eth0:0 192.168.0.9**

Now, when we run ifconfig we get the following output:

```
[root@green mbrown]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:B0:88:8B:85
          inet addr:192.168.0.7  Bcast:192.168.0.7  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3242730 errors:0 dropped:0 overruns:0 frame:0
          TX packets:252686 errors:0 dropped:0 overruns:0 carrier:0
          collisions:90 txqueuelen:100
          RX bytes:850035038 (810.6 Mb)  TX bytes:14797474 (14.1 Mb)
          Interrupt:9 Base address:0xece0

eth0:0    Link encap:Ethernet  HWaddr 00:60:B0:88:8B:85
          inet addr:192.168.0.9  Bcast:192.168.0.255  Mask:255.255.255.0
```

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
Interrupt:9 Base address:0xece0

lo Link encap:Local Loopback  
inet addr:127.0.0.1 Mask:255.0.0.0  
UP LOOPBACK RUNNING MTU:16436 Metric:1  
RX packets:300 errors:0 dropped:0 overruns:0 frame:0  
TX packets:300 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0 txqueuelen:0  
RX bytes:27360 (26.7 Kb) TX bytes:27360 (26.7 Kb)

We need to add these changes to the bottom of **/etc/rc.d/rc.local** which is a script that runs at startup.

Now that we have a second IP address to work with we can configure our DNS server called tinydns.

### **Configuration of tinydns**

Tinydns is a DNS server that answers queries about the particular domain(s) that you are hosting. It only listens on UDP port 53, so no zone transfers here. If you are interested or are required to perform zone transfers then you need to check out <http://cr.yip.to/djbdns/fag/tinydns.html#tcp> and look for the section answering TCP queries. Dan has a tool called axferdns which will take care of this for you if you have to do it.

As with dnscache we will need to add the users who will be responsible for running tinydns. We will need two, tinydns and dnslog, and we have already created the dnslog user. As before we will add them with the **/bin/false** shell so that they will be limited in what they can do.

### **/usr/sbin/useradd -s /bin/false tinydns**

*Why?* Once again I will reiterate that if someone would miraculously compromise the tinydns user account then they would simply be stuck since tinydns has no valid shell. This is very important since most serious vulnerabilities are dangerous only due to the fact that it allows you to compromise a service that is being run by a user with elevated privileges or root. If they get root, then game over, if they get a regular user with a valid shell then they can attempt to become root through various ways.

Configure tinydns by running the **tinydns-conf** program.

### **tinydns-conf tinydns dnslog /etc/tinydns 192.168.0.9**

Now we can create the **/service** directory associated with tinydns so that daemontools will monitor it for us.

## **ln -s /etc/tinydns /service**

Tinydns will start within a few seconds, and you can check it by using the **svstat** command,

## **svstat /service/tinydns**

The commands above are performed below:

```
[root@green etc]# tinydns-conf tinydns dnslg /etc/tinydns 192.168.0.9
[root@green etc]# ln -s /etc/tinydns/ /service/
[root@green etc]# svstat /service/tinydns/
/service/tinydns/: up (pid 12097) 10 seconds
[root@green etc]#
```

Now that we have the tinydns service up and running we need to configure the data file and provide the necessary name to IP resolutions that tinydns will answer with. While most zone files would be larger or more complex, the following will be a sample of what you can do and proficient for GIAC's needs.

```
[root@green etc]# cd /service/tinydns/root/
[root@green root]# ls
add-alias add-childns add-host add-mx add-ns data Makefile
[root@green root]#
```

There are five tools that help you with your zone configuration; we will be using three of them, **add-ns**, **add-mx**, and **add-host**. The following IP addresses have been made up and are in no way reflective of the actual name or IP.

Add the domain name record for giac.com:

```
[root@green root]# ./add-ns giac.com 1.1.1.1
[root@green root]# ./add-ns 1.1.1.in-addr.arpa 1.1.1.1
[root@green root]# make
/usr/local/bin/tinydns-data
[root@green root]# cat data
.giac.com:1.1.1.1:a:259200
.1.1.1.in-addr.arpa:1.1.1.1:a:259200
[root@green root]#
```

Now, add any host records that you have, for instance, your web server and your ftp server:

```
[root@green root]# ./add-host www.giac.com 1.1.1.2
[root@green root]# ./add-host ftp.giac.com 1.1.1.3
[root@green root]# make
/usr/local/bin/tinydns-data
```

```
[root@green root]# cat data
.giac.com:1.1.1.1:a:259200
.1.1.1.in-addr.arpa:1.1.1.1:a:259200
=www.giac.com:1.1.1.2:86400
=ftp.giac.com:1.1.1.3:86400
[root@green root]#
```

I prefer colors to name my servers as is my recommended scheme, but for public servers you must have names that other people would expect and be able to understand, so you can add alias. You add an alias in the same way that you add a host record but with the **add-alias** command. So, we add the proper names to our servers.

```
[root@green root]# ./add-alias red.giac.com 1.1.1.2
[root@green root]# ./add-alias blue.giac.com 1.1.1.3
[root@green root]# make
/usr/local/bin/tinydns-data
[root@green root]# cat data
.giac.com:1.1.1.1:a:259200
.1.1.1.in-addr.arpa:1.1.1.1:a:259200
=www.giac.com:1.1.1.2:86400
=ftp.giac.com:1.1.1.3:86400
+red.giac.com:1.1.1.2:86400
+blue.giac.com:1.1.1.3:86400
[root@green root]#
```

We almost forgot to add the name record for the name server itself, and of course its alias.

```
[root@green root]# ./add-host ns1.giac.com 1.1.1.1
[root@green root]# ./add-alias green.giac.com 1.1.1.1
[root@green root]# make
/usr/local/bin/tinydns-data
[root@green root]# cat data
.giac.com:1.1.1.1:a:259200
.1.1.1.in-addr.arpa:1.1.1.1:a:259200
=red.giac.com:1.1.1.2:86400
=blue.giac.com:1.1.1.3:86400
+www.giac.com:1.1.1.2:86400
+ftp.giac.com:1.1.1.3:86400
=ns1.giac.com:1.1.1.1:86400
+green.giac.com:1.1.1.1:86400
[root@green root]#
```

You may also host an email server so you will need to publish mx records which are done in virtually the same way with the **add-mx** command. More information can be found at <http://cr.yp.to/djbdns/faq/tinydns.html#add-mx> on creating mail exchanger records. The page <http://cr.yp.to/djbdns/faq/tinydns.html#config> has information on all types of DNS configurations.



The configuration of a DNS server should not be confusing or overbearing as with BIND. A table that compares the ease of use of djbdns versus BIND is at the following link:

<http://cr.yp.to/djbdns/blurb/easeofuse.html>. If you are sold on BIND, take a look you might like what you see.

## **SSH:**

The SSH configuration files are located in `/etc/ssh/` with the server configuration file called **sshd\_config** and the client configuration file called **ssh\_config**.

These are the files where we will set some specifications that help secure our system. In particular we want to specify the SSH protocol version, disable X11 forwarding, block root logins, and display the security banner message. We also want to specify the IP address that sshd listens on. In the `sshd_config` file uncomment the following lines to reflect these changes.

Specify the version of SSH that you will accept:  
**Protocol 2**

Block root login access:  
**PermitRootLogin no**

Disable X11 Forwarding:  
**X11Forwarding no**  
**#X11Forwarding yes**

Display security banner:  
**Banner /etc/motd**

Set IP Address that SSHD listens on  
**ListenAddress 192.168.0.7**

SSH is the tool by which we will perform all administrative functions. In the next section we will write our host-based firewall rules to only allow incoming DNS and SSH connections.

## **IPTABLES:**

Having a perimeter firewall is essential in today's Internet, but we should not stop there. Redhat Linux comes with the iptables firewall, which can be configured as a network or a host-based firewall. We will write a simple script that runs on startup to configure a host based firewall for our public djbdns server.

I have worked with Checkpoint firewalls before so delving into iptables has been an adventure, and one not without much help. I started out by reading "Linux Firewalls: Second Edition" by Robert L. Zeigler to gain a broader understanding of the theory and the syntax. I referenced the firewall examples

by Brian Melcher and Mark Oram in their papers "Securing a Red Hat Linux 7.2 Anonymous FTP Server with Security Support syslog Server" ([http://www.giac.org/practical/Brian\\_Melcher\\_GCUX.doc](http://www.giac.org/practical/Brian_Melcher_GCUX.doc)) and "Using and Securing a Red Hat Linux 7.3 server as a DNS, Mail, and Web server" ([http://www.giac.org/practical/Mark\\_Oram\\_GCUX.doc](http://www.giac.org/practical/Mark_Oram_GCUX.doc)), respectively, to help me get started. The following script is called iptables\_config\_green and will be put in /etc/rc.d/rc.local to be run at startup.

```
#!/bin/sh
```

```
#The next three lines give a description of the firewall script  
# iptables_config_green  
# iptables configuration script to only permit incoming DNS and SSH and  
# reject all others.
```

```
#Resets the path in case it has been tampered with  
PATH=/sbin:/bin
```

```
echo "Flush existing tables"
```

```
#Removes any preexisting rules or policies  
#We are echoing these lines after each command to help us troubleshoot in case there  
#is a problem.  
iptables --flush  
iptables --flush -t nat
```

```
echo "Set up the default deny policy"
```

```
#Sets the default deny  
iptables -P INPUT DROP  
iptables -P OUTPUT DROP  
iptables -P FORWARD DROP
```

```
echo "Set no restriction on the loopback interface"
```

```
#As the echo states, this sets no restriction on the loopback  
iptables -A OUTPUT -j ACCEPT -o lo  
iptables -A INPUT -j ACCEPT -i lo
```

```
echo "Permit outbound packets "
```

```
#Permits all outbound connections  
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
```

```
echo "Permit established incoming packets"
```

```
#This command tells iptables to accept incoming established packets  
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
echo "Allow access to the SSH and DNS daemons"
```

```
#The next two lines set up the rules to allow the incoming DNS and SSH connections  
iptables -A INPUT -p udp --dport 53 -j ACCEPT  
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

```
echo "Start the logs"
```

```
# Tell iptables to log dropped connections and gives an explanation about the certain  
commands  
# here the log-prefix option merely tells you what each line will begin with  
# in the logs
```

```
iptables -A INPUT -j LOG -m limit --limit 1/second \
--log-prefix "iptables drop:"
```

The script is called `iptables_config_green` and is located in `/usr/local/etc`. The `/etc/rc.d/rc.local` file now looks like:

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
echo "Authorized uses only. All activity may be monitored and reported." > /etc/issue
/sbin/ifconfig eth0:0 192.168.0.9
/usr/local/etc/iptables_config_green
```

This is where we call our iptables configuration file on startup. And this is what it looks like when it is run:

```
[root@green mbrown]# ./iptables_config_green
Flush existing tables
Set up the default deny policy
Set no restriction on the loopback interface
Permit outbound packets
Permit established incoming packets
Allow access to the SSH and DNS daemons
Start the logs
```

Next, run **iptables -L -v -n** to will list the rules in a verbose, numbered format so nothing will be left to the imagination.

```
[root@green mbrown]# iptables -L -v -n
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
    0    0 ACCEPT    all  --  lo      *       0.0.0.0/0  0.0.0.0/0
    0    0 ACCEPT    all  --  *       *       0.0.0.0/0  0.0.0.0/0    state
RELATED,ESTABLISHED
    0    0 ACCEPT    udp  --  *       *       0.0.0.0/0  0.0.0.0/0    udp dpt:53
    0    0 ACCEPT    tcp  --  *       *       0.0.0.0/0  0.0.0.0/0    tcp dpt:22
    0    0 LOG       all  --  *       *       0.0.0.0/0  0.0.0.0/0    limit: avg 1/sec
burst 5 LOG flags 0 level 4 prefix `iptables drop: '
```

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
```

```
Chain OUTPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
    0    0 ACCEPT    all  --  *       lo      0.0.0.0/0  0.0.0.0/0
    0    0 ACCEPT    all  --  *       *       0.0.0.0/0  0.0.0.0/0    state
NEW,RELATED,ESTABLISHED
```

Again, our host-based firewall is only one piece of our puzzle. One should never completely rely on a network or host-based firewall; you should utilize both if at all possible. Iptables is just one of the added benefits that come with Linux.

### **Locking Down the System:**

The most essential piece to our puzzle is configuring our server with the appropriate permissions on files and folders and verifying available services. We will use several documents to help us in our task and the most important will be the Linux Benchmark test from CISecurity ([www.cisecurity.org](http://www.cisecurity.org)). To give you a brief introduction on CIS visit the following link, [www.cisecurity.org/charter.html](http://www.cisecurity.org/charter.html), we find the following mission statement:

**The Center for Internet Security** mission is to help organizations around the world effectively manage the risks related to information security. CIS provides methods and tools to improve, measure, monitor, and compare the security status of your Internet-connected systems and appliances, plus those of your business partners.

CIS is not tied to any proprietary product or service. It manages a **consensus process** whereby members identify security threats of greatest concern, then participate in development of practical methods to reduce the threats. This consensus process is already in use and has proved viable in creating Internet security benchmarks available for widespread adoption.

I was introduced to the CIS security team at the SANS conference and their suite of tools, in particular was the one for Linux. If you are running Redhat or Mandrake then you can use their Linux tool to help in the locking down of your system. Download the Linux Security Benchmark and Scoring tool and follow these instructions to install it.

```
[mbrown@green downloads]$ ls
cis-linux.tar.gz
[mbrown@green downloads]$ tar xzf cis-linux.tar.gz
[mbrown@green downloads]$ cd cis
[mbrown@green cis]$ ls
CISscan-1.2.0-1.2.i386.rpm LinuxBenchmark.pdf README
[mbrown@green cis]$
Now su to root and install the CIS scan package,
```

```
[root@green cis]# rpm -ivh CISscan-1.2.0-1.2.i386.rpm
Preparing... ##### [100%]
 1:CISscan      ##### [100%]
```

Now change to /usr/local/CIS.

```
[root@green cis]# cd /usr/local/CIS/
[root@green CIS]# ls
cis_ruler_sgid_programs_mandrake_7.1 cis_ruler_suid_programs_mandrake_8.0
cis_ruler_sgid_programs_mandrake_7.2 cis_ruler_suid_programs_mandrake_8.1
cis_ruler_sgid_programs_mandrake_8.0 cis_ruler_suid_programs_redhat_6.1
```

```
cis_ruler_sgid_programs_mandrake_8.1 cis_ruler_suid_programs_redhat_6.2
cis_ruler_sgid_programs_redhat_6.1 cis_ruler_suid_programs_redhat_7.0
cis_ruler_sgid_programs_redhat_6.2 cis_ruler_suid_programs_redhat_7.1
cis_ruler_sgid_programs_redhat_7.0 cis_ruler_suid_programs_redhat_7.2
cis_ruler_sgid_programs_redhat_7.1 CISscan
cis_ruler_sgid_programs_redhat_7.2 README
cis_ruler_suid_programs_mandrake_7.1 tester.sub
cis_ruler_suid_programs_mandrake_7.2
[root@green CIS]#
```

The CIS scan tool is a read only tool and will make no changes, those are up to us to make. Once you run the tool it will produce a log file with its findings. This log file will have a score ranging from one to ten in it that you can use to “grade” your system. Without making any changes our server received a 5.89.

**Preliminary rating given at time: Mon Aug 5 15:30:00 2002**

**Preliminary rating = 5.89 / 10.00**

**Negative: 6.5 Non-standard SUID program /usr/bin/sperl5.6.1**  
**Ending run at time: Mon Aug 5 15:31:29 2002**

**Final rating = 5.89 / 10.00**

This log file goes hand in hand with a PDF document with information and sections of code that you can run if need be to fix any security hole that the scan finds. For example,

**Positive: 3.2 NFS Server script nfs is deactivated.**

**Negative: 3.3 NFS script nfslock not deactivated.**

The number following the Positive or Negative directs you to a section of the corresponding document. Our log file is as follows:

\*\*\* CIS Ruler Run \*\*\*

Starting at time 20020805-15:29:58

Positive: 1.1 System appears to have been patched within the last month.

**Negative: 2.2 No Authorized Only banner for telnet in file**  
**/etc/xinetd.d/telnet.**

**Negative: 2.2 No Authorized Only banner for login in file**  
**/etc/xinetd.d/rlogin.**

Positive: 2.3 telnet is deactivated.

Positive: 2.4 ftp is deactivated.

Positive: 2.5 rsh, rcp and rlogin are deactivated.

Positive: 2.6 tftp is deactivated.

**Negative: 2.7 xinetd either requires global 'only-from' statement or one for each service.**

**Negative: 3.1 apmd not deactivated.**

**Negative: 3.1 gpm not deactivated.**

**Negative: 3.1 isdn not deactivated.**

Positive: 3.2 NFS Server script nfs is deactivated.

**Negative: 3.3 NFS script nfslock not deactivated.**

**Negative: 3.3 NFS script autofs not deactivated.**

Positive: 3.4 NIS Client processes are deactivated.

Positive: 3.5 NIS Server processes are deactivated.

**Negative: 3.6 portmapper not deactivated.**

Positive: 3.7 samba windows filesharing daemons are deactivated.

**Negative: 3.8 netfs rc script not deactivated.**

**Negative: 3.9 lpd (line printer daemon) not deactivated.**

Positive: 3.10 Graphical login is deactivated.

**Negative: 3.11 Mail daemon is on and collecting mail from the network.**

Positive: 3.12 Web server is deactivated.

Positive: 3.13 snmp daemon is deactivated.

Positive: 3.14 DNS server is deactivated.

Positive: 3.15 postgresql (SQL) database server is deactivated.

Positive: 3.16 routing daemons are deactivated.

Positive: 3.17 Webmin GUI-based system administration daemon deactivated.

Positive: 3.18 Squid web cache daemon deactivated.

**Negative: 3.19 xinetd is still active.**

Positive: 3.20 Found a good daemon umask.

**Negative: 4.1 Coredumps aren't deactivated.**

Positive: 4.2 /etc/exports is empty or doesn't exist, so it doesn't need to be tuned for privports.

**Negative: 4.3 /proc/sys/net/ipv4/tcp\_max\_syn\_backlog should be at least 4096 to handle SYN floods.**

Positive: 4.4 All 'additional' network parameters set correctly.

Positive: 5.1 syslog captures auth and authpriv messages.

**Negative: 6.1 Removable filesystem /mnt/floppy is not mounted nosuid.**

**Negative: 6.2 PAM allows users to mount CD-ROMS.**

(/etc/security/console.perms)

**Negative: 6.2 PAM allows users to mount floppies.**

(/etc/security/console.perms)

Positive: 6.3 password and group files have right permissions and owners.

Positive: 6.4 all temporary directories have sticky bits set.

**Negative: 7.1 rhosts authentication not deactivated in /etc/pam.d/rlogin.**

**Negative: 7.1 rhosts authentication not deactivated in /etc/pam.d/rsh.**

Positive: 7.2 /etc/hosts.equiv file not present or has size zero.

**Negative: 7.3 /etc/ftpusers doesn't exist**

**Negative: 7.4 Couldn't open cron.allow**

**Negative: 7.4 Couldn't open at.allow**

**Negative: 7.5 The permissions on /etc/crontab are not sufficiently restrictive.**

**Negative: 7.6 No Authorized Only message in /etc/motd.**

**Negative: 7.6 No Authorized Only message in /etc/issue.**

**Negative: 7.7 /etc/securetty has a non tty1-12 line: tty10.**

**Negative: 7.8 GRUB isn't password-protected.**

**Negative: 8.1 news has a valid shell of /bin/sh. Remember, an empty shell field in /etc/passwd signifies /bin/sh.**

Positive: 8.2 There were no +: entries in passwd, shadow or group maps.

Positive: 8.3 All users have passwords

Positive: 8.4 Only one UID 0 account AND it is named root.

Positive: 8.5 root's PATH is clean of group/world writable directories or the current-directory link.

Positive: 8.6 root account has no dangerous rhosts, shosts, or netrc files.

Positive: 8.7 No user's home directory is world or group writable.

Positive: 8.8 No group or world-writable dotfiles!

Positive: 8.9 No user has a .netrc or .rhosts file.

**Negative: 8.10 Default umask may not block world-writable. Check /etc/profile.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/profile.**

**Negative: 8.10 Default umask may not block world-writable. Check /etc/csh.login.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/csh.login.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/csh.cshrc.**

Positive: 9.1 System is running sshd.

**Negative: 9.2 This machine isn't synced with ntp.**

Preliminary rating given at time: Mon Aug 5 15:30:00 2002

Preliminary rating = 5.89 / 10.00

Negative: 6.5 Non-standard SUID program /usr/bin/sperl5.6.1

Ending run at time: Mon Aug 5 15:31:29 2002

Final rating = 5.89 / 10.00

As you can see we have our work cut out for us. We will start at the top and work our way through.

### **Summary of Steps Taken to Lock Down System:**

**Negative: 2.2 No Authorized Only banner for telnet in file /etc/xinetd.d/telnet.**

**Negative: 2.2 No Authorized Only banner for login in file /etc/xinetd.d/rlogin.**

#### **Action: Configure Logon Banners**

It is always recommended that you have logon banners to your system to help with any legal matters that may arise and it is necessary that you consult your

legal department (if you have one) to get the most proper working. As root perform the following action:

```
mkdir /etc/banners
cd /etc/banners
if [ -e /usr/doc/tcp_wrappers-7.6/Banners.Makefile ]
then
cp /usr/doc/tcp_wrappers-7.6/Banners.Makefile Makefile
else
cp /usr/share/doc/tcp_wrappers-7.6/Banners.Makefile \
Makefile
fi
echo "Authorized uses only. All activity may be \
monitored and reported." > prototype
make
```

*Why?* The question can be asked here, “why put login banners on our system if we are not using the services that the banners are for?” The best practice is to perform every possible action to secure the server. These services may not be in use now, but possibly in the future so go ahead and do what you can now to make your job easier in the future.

Once you have created your banners you have to tell your services to point use them when called. Modify the /etc/xinetd.d/telnet and rlogin files to include the following line:

For rlogin:

```
vi /etc/xinetd.d/rlogin
banner          = /etc/banners/in.rlogind
```

For telnet:

```
vi /etc/xinetd.d/telnet
banner          = /etc/banners/in.telnetd
```

Now when run the scan again it will see the appropriate login banners.

**Negative: 2.7 xinetd either requires global 'only-from' statement or one for each service.**

**Action: Disable xinetd completely**

We will take care of this when we disable xinetd completely later on.

**Negative: 3.1 apmd not deactivated.**

**Negative: 3.1 gpm not deactivated.**

**Negative: 3.1 isdn not deactivated.**

**Negative: 3.3 NFS script nfslock not deactivated.**

**Negative: 3.3 NFS script autofs not deactivated.**



**Negative: 3.6 portmapper not deactivated.**

**Negative: 3.8 netfs rc script not deactivated.**

**Negative: 3.9 lpd (line printer daemon) not deactivated**

**Action: Disable Services Not Needed**

We must disable any system daemon that does not have a specific purpose for our DNS server. We will disable all of the suggested services except for **gpm**, since it is required for mouse activity when logged into GNOME. As root perform the following actions:

```
[root@green banners]# chkconfig apmd off
[root@green banners]# chkconfig isdn off
[root@green banners]# chkconfig nfslock off
[root@green banners]# chkconfig autofs off
[root@green banners]# chkconfig portmap off
[root@green banners]# for user in rpc rpcuser ; do
> /usr/sbin/usermod -L -s /dev/null $user
> done
[root@green banners]# chkconfig netfs off
[root@green banners]# chkconfig lpd off
[root@green banners]#
```

Chkconfig is a Redhat native utility which allows you to turn services on or off at different run levels. The command **chkconfig #service\_name# off** will disable any service you list completely. In the same way you could turn any service on. The previous commands disabled the apmd, isdn, nfslock, autofs, portmap, netfs, and lpd services while giving the rpcuser and rpc user account an invalid shell to prevent any unwanted access through them if ever hacked.

**Negative: 3.11 Mail daemon is on and collecting mail from the network.**

**Action: Disable mail daemon and configure mail to be processed every 15 minutes if not sooner**

The mail daemon should only be running if this server is going to be acting as a mail server, i.e. other machines will relay mail through it. Since this is not the case we will disable the sendmail daemon and set a queue time. Run these commands as root:

```
[root@green banners]# cd /etc/sysconfig/
[root@green sysconfig]# cat <<END_ENTRIES > sendmail
> DAEMON=no
> QUEUE=15m
> END_ENTRIES
[root@green sysconfig]# chown root:root sendmail
[root@green sysconfig]# chmod 644 sendmail
```

**Negative: 3.19 xinetd is still active.**

**Action: Disable xinetd**

Xinetd is the service that runs and looks for any connections attempts being made to services like ftp, telnet, and tftp. If an attempt is made and that service is enabled, then xinetd will starts the particular daemon and it will answer on the network. Since we will not be running these services we can completely turn off xinetd. Run these commands as root:

```
[root@green sysconfig]# chkconfig xinetd off
[root@green sysconfig]# chkconfig --list xinetd
xinetd          0:off  1:off  2:off  3:off  4:off  5:off  6:off
```

I ran the **chkconfig --list xinetd** command to verify and show that xinetd was turned off at every level.

**Negative: 4.1 Coredumps aren't deactivated.**

**Action: Disable Coredumps**

Coredumps should only be turned on if you have developers using the system and will need to run any type of debugging. To do this run the next commands as root.

```
[root@green sysconfig]# cat <<END_ENTRIES >>/etc/security/limits.conf
> * soft core 0
> * hard core 0
> END_ENTRIES
```

**Negative: 4.3 /proc/sys/net/ipv4/tcp\_max\_syn\_backlog should be at least 4096 to handle SYN floods.**

**Action: Adjust network parameters to streamline the systems' response to unnatural network traffic.**

The next few commands will add some tweaks to the system which will help it handle strange traffic such as syn\_floods.

```
[root@green sysconfig]# cat <<END_SCRIPT >> /etc/sysctl.conf
> net.ipv4.ip_forward = 0
> net.ipv4.conf.all.accept_source_route = 0
> net.ipv4.tcp_max_syn_backlog = 4096
> net.ipv4.conf.all.rp_filter = 1
> END_SCRIPT
[root@green sysconfig]# chown root:root /etc/sysctl.conf
[root@green sysconfig]# chmod 0600 /etc/sysctl.conf
```

If you would like some additional information about these settings then check out <http://www.linuxhq.com/kernel/v2.4/doc/networking/ip-sysctl.txt.html> or <http://www.linuxhq.com/kernel/v2.2/doc/networking/ip-sysctl.txt.html>.

**Negative: 6.1 Removable filesystem /mnt/floppy is not mounted nosuid.**

**Action: Change the mount permission on the floppy drive or any other specified device.**

To change the mount permission on the floppy drive simply add **nosuid** to the fourth field of **/etc/fstab**.

```
/dev/fd0          /mnt/floppy      auto  noauto,owner,kudzu,nosuid 0 0
```

*Why?* Often the choice must be made between convenience and security. Above where we are only allowing root to mount the floppy could inconvenience an administrator, but it also protects the server from having a malicious floppy mounted on the server by a local user.

**Negative: 6.2 PAM allows users to mount CD-ROMS.**

**(/etc/security/console.perms)**

**Negative: 6.2 PAM allows users to mount floppies.**

**(/etc/security/console.perms)**

**Action: Disallow anyone besides root to mount CD-ROMS and floppy drives.**

The PAM modules in Redhat allow someone at the console to have additional privileges, such as mounting CD-ROM's and floppy drives. By making the following changes you can disable those rights and leave only root to be able to perform them.

```
[root@green security]# egrep -v '(floppy|cdrom)' console.perms > console.perms.new
[root@green security]# mv console.perms.new console.perms
mv: overwrite `console.perms'? y
[root@green security]# grep -v supermount /etc/fstab > /etc/fstab.new
[root@green security]# chmod 0644 /etc/fstab
[root@green security]# mv /etc/fstab.new /etc/fstab
mv: overwrite `/etc/fstab'? y
[root@green security]# chown root:root console.perms /etc/fstab
[root@green security]# chmod 0644 /etc/fstab
```

This is also a good exercise at using egrep.

**Negative: 7.1 rhosts authentication not deactivated in /etc/pam.d/rlogin.**

**Negative: 7.1 rhosts authentication not deactivated in /etc/pam.d/rsh.**

**Action: Disable rhosts authentication completely.**

It is not necessary whatsoever to have rhosts authentication turned on. SSH has all the functionality plus encryption, so use it. Disable the .rhosts file with the following commands:

```
[root@green pam.d]# grep -v rhosts_auth rlogin > rlogin.new
[root@green pam.d]# mv rlogin.new rlogin
mv: overwrite `rlogin'? y
```

```
[root@green pam.d]# chown root:root rlogin
[root@green pam.d]# chmod 644 rlogin
[root@green pam.d]# grep -v rhosts_auth rsh > rsh.new
[root@green pam.d]# mv rsh.new rsh
mv: overwrite `rsh'? y
[root@green pam.d]# chown root:root rsh
[root@green pam.d]# chmod 644 rsh
```

**Negative: 7.3 /etc/ftputers doesn't exist**

**Action: Create an ftputers file that prohibits those users from logging in with ftp.**

This would be a more important item had we not already disabled xinetd which now prohibits ftp from running at all. But, for the sake of discussion assume you have to have an ftp server running on this box as well, then this is how you would restrict certain users from logging in via ftp, like the news user for example.

```
[root@green pam.d]# for name in `cut -d: -f1 /etc/passwd`; do
> if [ `id -u $name` -lt 500 ]
> then
> echo $name >> /etc/ftputers
> fi
> done
[root@green pam.d]# chown root:root /etc/ftputers
[root@green pam.d]# chmod 600 /etc/ftputers
```

**Negative: 7.4 Couldn't open cron.allow**

**Negative: 7.4 Couldn't open at.allow**

**Action: Create a cron.allow and an at.allow to restrict access to these functions.**

The cron.allow and at.allow files contain a list of users who are able to run these functions; it is a good idea to restrict this to specific administrators. Run the following command to setup these files with restricted access only to root.

```
[root@green etc]# rm -f cron.deny at.deny
[root@green etc]# echo root>cron.allow
[root@green etc]# echo root>at.allow
[root@green etc]# chown root:root cron.allow at.allow
[root@green etc]# chmod 400 cron.allow at.allow
```

**Negative: 7.5 The permissions on /etc/crontab are not sufficiently restrictive.**

**Action: Set appropriate permission on crontab.**

We want to limit who can modify cron files so that no one will possible be able to elevate their privileges with it. Run the next few commands as root.

```
[root@green etc]# chown root:root /etc/crontab
[root@green etc]# chmod 400 /etc/crontab
[root@green etc]# chown -R root:root /var/spool/cron
[root@green etc]# chmod -R go-rwx /var/spool/cron
[root@green etc]# chown -R root:root /etc/cron.*
[root@green etc]# chmod -R go-rwx /etc/cron.*
```

*Why?* This is just an example of proper file permissions that help in the overall desire to make the server as secure as possible.

**Negative: 7.6 No Authorized Only message in /etc/motd.**

**Negative: 7.6 No Authorized Only message in /etc/issue.**

**Action: Create security warning messages.**

"It is a widely held belief that presenting some sort of statutory warning message at login time will assist the prosecution of trespassers on the computer system. Changing some of these login banners also has the side effect of hiding OS version information and other detailed system information, which an attacker might find useful when targeting their attacks (though there are other mechanisms available for acquiring this information). Clearly, the content of all such statutory warnings should be reviewed by the organization's local legal counsel before the above modifications are made."

**The Center for Internet Security, Linux Benchmark v1.0.0**

I believe that they said it best in this case, so run the next few commands to create security warning messages on your system.

```
echo "Authorized uses only. All activity may be monitored and reported." >>/etc/motd
chown root:root /etc/motd
chmod 644 /etc/motd
cat <<END >> /etc/rc.d/rc.local
echo "Authorized uses only. All activity may be monitored and reported." >> /etc/issue
echo "Authorized uses only. All activity may be \ monitored and reported." >>
/etc/issue.net
END
```

**Negative: 7.7 /etc/securetty has a non tty1-12 line: tty10.**

**Action: Restrict anonymous root logins.**

The only time you should consider allowing root to logon anonymously to your system is at the console and only in an emergency. A user should always first use a normal user account then **su** to root for any privilege escalation that they need.

```
cat <<END_FILE >/etc/securetty
tty1
tty2
tty3
```

```
tty4
tty5
tty6
END_FILE
chown root:root /etc/securetty
chmod 400 /etc/securetty
```

*Why?* Root should never be allowed to directly connect remotely to the server. Always first connect as a basic user then become root.

**Negative: 7.8 GRUB isn't password-protected.**

**Action: Password protect the GRUB boot prompt.**

We could have done this during the installation phase but I wanted to hold off until now so that it might have more effect. This will not require a password to boot the system but it will require one to be able to modify the boot parameters of your system. You would never want these to be changed unless you knew about it first.

Since we are using grub we will add the following line to the grub.conf file located in /etc. Make sure you use a hard to guess password here and not your user or root password.

```
password <hard-to-guess>
```

Set the permissions on the file

```
chown root:root /etc/grub.conf
chmod 600 /etc/grub.conf
```

**Negative: 8.1 news has a valid shell of /bin/sh. Remember, an empty shell field in /etc/passwd signifies /bin/sh.**

**Action: Give any non-essential users an invalid shell.**

If we are not using the news service then the news user does not need to be able to do anything on our system, so give it an invalid shell.

```
/usr/sbin/usermod -L -s /dev/null news
```

*Why?* The only two users on the system with a valid shell should be root and any local administrators.

**Negative: 8.10 Default umask may not block world-writable. Check /etc/profile.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/profile.**

**Negative: 8.10 Default umask may not block world-writable. Check /etc/csh.login.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/csh.login.**

**Negative: 8.10 Default umask may not block group-writable. Check /etc/csh.cshrc**

**Action: Set a default umask for users at login time.**

Having too much power without knowing it could be a bad thing. Allowing users to create world or group writable files is asking for trouble so execute the following script to add a default umask.

```
cd /etc
for file in profile csh.login csh.cshrc bashrc
do
if [ `grep -c umask $file` -eq 0 ];
then
echo "umask 022" >> $file
fi
chown root:root $file
chmod 444 $file
done
```

**Negative: 9.2 This machine isn't synced with ntp.**

**Action: Configure your system to use NTP.**

NTP is one of the easiest things to set up and one of the most beneficial to use. It does you no help when you are comparing logs from different systems if there times are all different. Consider a court case where three different witnesses said they saw the perpetrator at the same time and in different places. The same thing could happen if your machines are not synced with NTP. Create your NTP configuration files with the following commands:

```
cat <<END_CONFIG >/etc/ntp.conf
driftfile /etc/ntp.drift
restrict default nomodify
server <10.10.10.1>
server <10.10.10.2>
server <10.10.10.3>
END_CONFIG
```

This assumes that GIAC has three NTP servers at the IP addresses listed in the ntp.conf file.

Now that we have done our best to lock down our system in the most feasible, secure way, reboot your machine then run the CIS scan again.

So, after a reboot I ran the scan and received the following score:

Final rating = 9.64 / 10.00

```
[root@green CIS]# egrep "^Negative" ./cis-ruler-log.20020810-12\03\13.3587
Negative: 3.1 gpm not deactivated.
Negative: 6.5 Non-standard SUID program /usr/bin/sperl5.6.1
```

Is 9.64 good enough for your system? In our case we were able to do almost every thing recommended by CIS. It makes sense since all we are running is DNS and SSH and we can turn off everything else. The score is relative based on what your system is doing and what is acceptable to you and your management. Does it cover everything that you can do? No, but it covers the essentials. For now we are happy with it and will move on to the next phase of our Project. The only word of caution is that over time people always need additional software installed on their systems. If such actions are taken make sure to rescan the system and make the appropriate modifications.

## Ongoing Maintenance

What would the appropriate strategy be for the ongoing maintenance of our system? Installing a secured system is great but it is only the beginning. What should our backup strategy be, how often should we patch the system, what is the password policy, when and how should we scan it, and what is the logging policy for our server? These are all questions we will answer.

### Backup Strategy:

Backups are an integral part of your maintenance plan if you want to quickly recover in case of a severe accident or system crash. Since our server will only have static data on it, i.e., DNS zone file and SSH server configuration, we only need to perform a full backup each week. Our log files will be sent to the logging server so we won't have to worry about the /var partition. We will run the following SHELL script at 6am on Sunday mornings:

```
#!/bin/bash
```

```
date=`date '+%Y%m%d'`
```

```
outfile="/backups/green_full_"$date".tgz"
```

```
tar zcvf $outfile /command /etc /home /package /root /service /usr
```

This file will then be copied to an internal server where it will be backed up as part of GIAC's backup strategy which is beyond the scope of this document. The question with backups is always what files to backup? What type of media will you use? How will you test your backups?

### What files to backup?



For our server the question of which files to backup is answered by what files have we changed since the installation? The /command, /package, and /service directories are all modified and have to do with djbdns and daemontools. The /etc directory contains all configurations files and all of our binaries that we have added live in /usr. We could almost just backup /usr/local since that is where all of the djbdns related binary files live, but we will backup all of /usr just to be sure. We will also backup /root and /home to insure that the distinctive home areas remain intact.

### **What type of media to use?**

We will be using the media of the server that we copy our tar.gz file to. This question is answered only by knowing that whatever your internal network's backup media will be the same for this public server since we are merely copying our file to it.

### **How will you test your backups?**

While browsing <http://www.linux-backup.net/issue.gwif.html> you see the statement *If you can recreate your entire system on another disk...you've selected the "right directories to backup."* A sufficient disaster recovery type test would be to get similar hardware and build it with the same instructions and then restore the backup. If you get the same system as desired then you have selected the appropriate files to backup. I have tested deleting my home area, then restoring it from backup and all was fine. Depending on your company's backup/restore strategy you should test it as often as prescribed, though once a quarter should be sufficient.

### **Software Updates and Patches:**

For software updates and patches, a huge part will be played on how we are notified of the updates. There are several mailing lists on the Internet that you can be a member, but for us the ones that are most important are the Redhat notifications, the djbdns mailing list, and the infamous Bugtraq.

If you have subscribed to Redhat network then you can set it up to email you whenever there are updates that affect your system, which is helpful but not at all conclusive. At <http://www.redhat.com/mailling-lists/> you can subscribe to any of the Redhat mailing list, of interest are the redhat-announce-list and the Redhat-watch-list. From these two you will be notified about almost everything concerning Redhat.

The djbdns mailing list is where I had all of my questions concerning its usage answered. These are a great bunch of people who give their time to help people like me to learn djbdns. To subscribe to this list send an email to [dns-subscribe@list.cr.yo.to](mailto:dns-subscribe@list.cr.yo.to) and follow the received instructions.

The last list is perhaps the most well known, it is bugtraq. It is not for the faint of heart, for you will receive many emails here, many of which are posts from vendors themselves and the rest are user posts. To learn more about bugtraq

and many other mailing lists out there visit <http://online.securityfocus.com/archive> and to subscribe to any of them go to <http://online.securityfocus.com/cgi-bin/sfonline/subscribe.pl>. You get a lot of emails from bugtraq, and this is good since you won't be left out of anything!

Now that we have taken care of begin notified of any bugs/vulnerabilities and their corresponding patches, we need to address how we will apply the updates. There are two cases with updates, those that cause the server to go down, i.e. be rebooted, and those that do not.

### **Downtime updates:**

GIAC has a biweekly downtime schedule where late on Monday night of that week we are allotted a 7pm to 7am window to request for any downtime. If there is a patch that needs to be applied that needs the server to be rebooted, then that Monday night is the time.

### **Non-downtime updates:**

These are patches that affect services such as SSH, where it is only being used for administration and not for production. These patches can be applied at the administrator's discretion.

All updates for Redhat can be found at <ftp://updates.redhat.com/7.2/en/os/i386/> for any that are released.

Here you will find updates to all packages we have used except for daemontools and djbdns itself. If any updates or bugs are released in them you will find out from the mailing list or by visiting [www.djbdns.org](http://www.djbdns.org) or <http://cr.yp.to/djbdns.html> and <http://cr.yp.to/daemontools.html>.

### **Password Policy:**

The password policy though small on paper reigns large in application. An effective password policy should contain the following guidelines:

*All root passwords should be changed monthly or sooner if so desired.*

*All user passwords must change on a quarterly basis if not sooner.*

*All passwords must be eight characters in length and contain at least two letters, two numbers, and two special characters.*

If properly followed and enforced the password policy will help eliminate security breaches through password guessing or social engineering. Periodic audits with

tools such as LC (L0phtCrack) or the UNIX tool crack can help you access whether the policy is being followed.

### Scanning Schedule:

Scanning can be done as often as needed depending on the type of scan. A simple port scan to see what services have daemons listening on the network is quick and easy to perform. Running the CIS scan can be done in moments with not impact to your system. A weekly and/or daily port scan can be set up using nmap with the results being compared against an initial scan. If something new pops up then you can be sent an email to go check it out. The types of scans to run will be described in the last section of the report where the configuration is checked.

### Logging Scheme:

All logs will be sent to an internal syslog server. The logs will then be reviewed for any interesting events and further action taken if necessary. The syslog configuration file in use by our server is shown here.

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none    @loghost

# The authpriv file has restricted access.
authpriv.*                                   @loghost

# Log all the mail messages in one place.
mail.*                                       /var/log/maillog

# Log cron stuff
cron.*                                       @loghost

# Everybody gets emergency messages
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                             /var/log/spooler

# Save boot messages also to boot.log
local7.*                                    /var/log/boot.log
~
```

There is an entry in our **/etc/hosts** for **loghost** so that we only have to touch the host file if we ever need to change syslog hosts. A sample of my **/var/log/messages** log file in which I become root via su is shown below.

Aug 11 21:43:40 green su(pam\_unix)[1388]: session closed for user root  
Aug 11 21:43:44 green su(pam\_unix)[1568]: session opened for user root by mbrown(uid=500)

Aug 11 21:46:24 green su(pam\_unix)[1601]: authentication failure; logname=mbrown uid=500 euid=0 tty= ruser= rhost= user=root

If you wanted to be alerted anytime anyone attempted to become root either successfully or unsuccessfully, then you could run the following script in cron every hour and have yourself notified if you see anything of interest.

```
#!/bin/bash
```

```
echo "Someone has attempted to login and has failed!" > /var/log/su_alert  
echo "Below are the details:" >> /var/log/su_alert  
echo >> /var/log/su_alert  
grep "authentication failure" /var/log/messages >> /var/log/su_alert
```

```
mail -s Login_Alert linuxadmin@giac.com < /var/log/su_alert
```

If you are not familiar with cron, then as root you would run this command:

**crontab -e**

Enter the following line:

**0,30 \* \* \* \* /home/mbrown/log\_script**

This runs the script at the top and bottom of every hour, every day of the week, every week of the month, and every month of the year.

This script is very basic and is only meant to be a proof of concept, its only purpose is to illustrate an example of what you could do to check your logs and how you can set it up to run via cron.

If you would rather use a tool that someone else has written for you (this is what I do) you can get logsentry from <http://www.psionic.com/downloads/logsentry-1.1.1.tar.gz>. In Securing Linux Step-By-Step Version 2.0 pages 74-75 by Lee E. Brotzman and Jay Beale, a detailed description of how to install and configure is given.

## Check Your Configuration

Now that we have installed and configured our system we need to verify that it has been done correctly. Every administrator should get into the habit of testing

his or her system(s) to verify that they are functioning properly, and we are no exception.

## **Port Scan:**

**Nmap** ([www.insecure.org/nmap/index.html](http://www.insecure.org/nmap/index.html))

Nmap (Network Mapper) is the classic network and port mapping tool that every security person should know and use. I have claimed that only two ports are open to inbound connections on our system, SSH (tcp/22) and DNS (udp/53). This test will be done with the firewall turned on so we can verify that the rules are working properly. The iptables firewall script that is running on our box only allows these two ports for inbound connections and we will first verify that. To further prove that we have indeed locked down our box to the extent that has been claimed, we will turn off the iptables firewall and perform an additional scan to verify that only the two ports are open.

We will use a TCP SYN scan to see what TCP services are open.

The flags tell nmap the following:

- v** -- specifies verbose output, you can add more **-v's**, i.e. **-vvv**, to make the output more verbose
- sS** -- tells nmap to perform a TCP SYN stealth scan
- P0** -- do not ping the host before scanning it
- p--** specifies the port range
- o filename** -- tells nmap to write its result to a file name *filename*

```
# nmap (V. 2.54BETA31) scan initiated Tue Aug 20 19:43:37 2002 as: nmap -v -sS -P0 -o
nmap_tcp.out 192.168.0.7,9
Interesting ports on (192.168.0.7):
(The 1553 ports scanned but not shown below are in state: filtered)
Port      State      Service
22/tcp    open       ssh
```

```
Interesting ports on (192.168.0.9):
(The 1553 ports scanned but not shown below are in state: filtered)
Port      State      Service
22/tcp    closed     ssh
```

```
# Nmap run completed at Tue Aug 20 19:59:17 2002 -- 2 IP addresses (2 hosts up) scanned in
940 seconds
```

We see that the only tcp port open on our system is SSH (tcp/22), which is what we expected. We also see that it is only actively listening on the 192.168.0.7 address. This is the address that all internal connections will be directed to from the firewall.

**NOTE:** If an administrator is logged onto the console locally and has a window manager running, then you would see that TCP 6000 is open, but don't let this alarm you.

Now let's perform a scan of the UDP ports.

```
# nmap (V. 2.54BETA31) scan initiated Tue Aug 20 21:32:49 2002 as: nmap -vvv -sU -P0 -o
nmap_udp.out 192.168.0.7,9
Interesting ports on (192.168.0.7):
(The 1457 ports scanned but not shown below are in state: filtered)
Port      State      Service
53/udp    open       domain
```

```
Interesting ports on (192.168.0.9):
(The 1457 ports scanned but not shown below are in state: filtered)
Port      State      Service
53/udp    open       domain
```

```
# Nmap run completed at Tue Aug 20 21:33:13 2002 -- 2 IP addresses (2 hosts up) scanned in
24 seconds
```

You can see that our firewall rules are holding up as planned. Only tcp/22 and udp/53 are open on either IP address with SSH only actively listening on 192.168.0.7 while DNS is actively listening on both addresses.

Now that we have verified our firewall rules we can shut off iptables and scan again to see what is really listening on our system.

Run `/etc/rc.d/init.d/iptables stop` to stop the current firewall rules and establish a default accept policy. We will run the scan of TCP ports again and see what open that our firewall might have been blocking is.

```
# nmap (V. 2.54BETA31) scan initiated Tue Aug 20 21:39:20 2002 as: nmap -v -sS -P0 -o
nmap_tcp_nofw.out 192.168.0.7,9
Interesting ports on (192.168.0.7):
(The 1551 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open       ssh
53/tcp    open       domain
6000/tcp  open       X11
```

```
Interesting ports on (192.168.0.9):
(The 1553 ports scanned but not shown below are in state: closed)
Port      State      Service
6000/tcp  open       X11
```

```
# Nmap run completed at Tue Aug 20 21:39:22 2002 -- 2 IP addresses (2 hosts up) scanned in 2
seconds
```

We see that ssh is really only listening on 192.168.0.7 which was initially expected. We see that X11 is open, which is due to the fact that I am logged into the console right now in GNOME. Interestingly we see that tcp/53 is open. Why is that? Dnscache listens on udp/53 for all caching requests, but if the reply is too large to fit in a UDP datagram then dnscache will set the truncation bit in the UDP response and the client will retry the query over TCP. (Taken from Paul Jarc on the djbdns mailing list) This raises the question of whether we should open inbound tcp/53 on the 192.168.0.7 interface. The decision is up to you. We block queries on tcp/53 through the firewall from the Internal network to the caching server's IP address so the server will never receive a request on that port. If this becomes a problem it is easily changed.

### **Vulnerability Scan:**

#### **Nessus ([www.nessus.org](http://www.nessus.org))**

Nessus is a free vulnerability scanner that checks your system for known security holes. We need to update the plugins for our system before we scan our new server, and once we are done we can test it to see if there are any vulnerabilities on our server. I will assume that you have already installed nessus and have configured it correctly. Before any round of scanning make sure that you run

#### **nessus-update-plugins**

This will give you the latest scripts to scan with. Now, after performing the update I launched nessus with most of the default scan options and specifying nessus to scan 192.168.0.7 and 192.168.0.9. The nessus report in html format is as follows:

---

#### **Nessus Scan Report**

---

*Number of hosts which were alive during the test : 2  
Number of security holes found : 2  
Number of security warnings found : 1  
Number of security notes found : 6*

List of the tested hosts :

- [192.168.0.7](#) (Security holes found)
- [192.168.0.9](#) (Security notes found)

---

[\[ Back to the top \]](#)

**192.168.0.7 :**

List of open ports :

- [general/tcp](#) (Security notes found)
- [ssh \(22/tcp\)](#) (Security hole found)
- [domain \(53/udp\)](#) (Security warnings found)

[\[ back to the list of ports \]](#)

#### Information found on port general/tcp

"Default scan" set. nmap will ignore the user specified port range and scan only the 1024 first ports and those declared in nmap-services

[\[ back to the list of ports \]](#)

#### Information found on port general/tcp

Nmap found that this host is running Linux Kernel 2.4.0 - 2.4.17 (X86)

[\[ back to the list of ports \]](#)

#### Vulnerability found on port ssh (22/tcp)

You are running a version of OpenSSH older than OpenSSH 3.2.1

A buffer overflow exists in the daemon if AFS is enabled on your system, or if the options KerberosTgtPassing or AFSTokenPassing are enabled. Even in this scenario, the vulnerability may be avoided by enabling UsePrivilegeSeparation.

Versions prior to 2.9.9 are vulnerable to a remote root exploit. Versions prior to 3.2.1 are vulnerable to a local root exploit.

Solution :  
Upgrade to the latest version of OpenSSH

Risk factor : High  
[CVE : CAN-2002-0575](#)

[\[ back to the list of ports \]](#)

#### Vulnerability found on port ssh (22/tcp)

You are running a version of OpenSSH which is older than 3.4

There is a flaw in this version that can be exploited remotely to give an attacker a shell on this host.

Solution : Upgrade to OpenSSH 3.4  
Risk factor : High

[\[ back to the list of ports \]](#)

#### Information found on port ssh (22/tcp)

a ssh server is running on this port

[\[ back to the list of ports \]](#)

#### Information found on port ssh (22/tcp)

Remote SSH version : SSH-2.0-OpenSSH\_3.1p1

[\[ back to the list of ports \]](#)



#### Information found on port ssh (22/tcp)

The remote SSH daemon supports the following versions of the SSH protocol :

- . 1.99
- . 2.0

[\[ back to the list of ports \]](#)

#### Warning found on port domain (53/udp)

The remote name server allows recursive queries to be performed by the host running nssd.

If this is your internal nameserver, then forget this warning.

If you are probing a remote nameserver, then it allows anyone to use it to resolve third parties names (such as www.nessus.org). This allows hackers to do cache poisoning attacks against this nameserver.

Solution : Restrict recursive queries to the hosts that should use this nameserver (such as those of the LAN connected to it). If you are using bind 8, you can do this by using the instruction 'allow-recursion' in the 'options' section of your named.conf

If you are using another name server, consult its documentation.

Risk factor : Serious  
[CVE : CVE-1999-0024](#)

---

[\[ Back to the top \]](#)

#### 192.168.0.9 :

List of open ports :

- [general/tcp](#) (Security notes found)

[\[ back to the list of ports \]](#)

#### Information found on port general/tcp

"Default scan" set. nmap will ignore the user specified port range and scan only the 1024 first ports and those declared in nmap-services

---

This file was generated by [Nessus](#), the open-sourced security scanner.

We see that there was a major warning on 192.168.0.7 concerning the version of OpenSSH. Here lies an interesting note, the version number that comes from the OpenSSH website would imply that you need to update to version 3.4 or later while if you are using Redhat's package of OpenSSH you know that we have already updated to the most recent package. It is just that Redhat release an rpm for the patched version of 3.1 instead of releasing one for the current version on OpenSSH's website.

We had a couple of security notes, one was that we were running a resolving DNS server on 192.168.0.7. This is fine (which is why it is a note) since we are meaning to run a resolving DNS server there.

We also see that NMAP correctly guessed the operating system version of our server!

Our scan with Nessus has told us that we have no unknown vulnerabilities and has confirmed our efforts to update and lock down our system.

### **Logging:**

In our /etc/syslog.conf file we direct the logs to an internal syslog server. We need to verify that if an event happens it populates to the internal loghost. There are many ways to produce a login failed message, we will be logged on locally as our basic administrator user, **mbrown**, and try to become root but type the password incorrectly.

```
[mbrown@green mbrown]$ su
Password:
su: incorrect password
[mbrown@green mbrown]$
```

At the same time on our internal host we were running the command `tail -f /var/log/messages` to see if an authentication failure message appeared.

```
[root@loghost mbrown]# tail -f /var/log/messages
Aug 20 16:31:04 green su(pam_unix)[2027]: authentication failure; logname=mbrown
uid=500 euid=0 tty= ruser= rhost= user=root
```

We see the entry showing a failed authentication attempt and conclude that our logging is working properly.

### **SSH:**

We do not permit a user to logon remotely via SSH directly as root. Our policy dictates that a user must first login as themselves and then **su** to root. We will test this by attempting to ssh to green as root.

```
[mbrown@mbrown mbrown]$ ssh -l root green.giac.com
Authorized uses only. All activity may be monitored and reported.
root@green.giac.com's password:
Permission denied, please try again.
root@green.giac.com's password:
Permission denied, please try again.
root@green.giac.com's password:
Permission denied (publickey,password,keyboard-interactive).
[mbrown@mbrown mbrown]$
```

I was unable to login directly as root and here are the logs that were captured from our syslog server:

```
[root@loghost mbrown]# tail -f /var/log/messages
Aug 20 16:37:49 green sshd[2031]: Failed keyboard-interactive for root from
12.6.100.194 port 12632 ssh2
Aug 20 16:37:53 green sshd[2031]: ROOT LOGIN REFUSED FROM a.b.c.d
Aug 20 16:37:53 green sshd[2031]: Failed password for root from a.b.c.d port 12632
ssh2
Aug 20 16:37:56 green sshd(pam_unix)[2031]: authentication failure; logname= uid=0
euid=0 tty=NODEVssh ruser= rhost=a.b.c.d user=root
Aug 20 16:37:58 green sshd[2031]: Failed password for root from a.b.c.d port 12632
ssh2
Aug 20 16:38:02 green sshd[2031]: ROOT LOGIN REFUSED FROM a.b.c.d
Aug 20 16:38:02 green sshd[2031]: Failed password for root from a.b.c.d port 12632
ssh2
Aug 20 16:38:02 green sshd[2031]: Connection closed by a.b.c.d
```

As you can see our policy was enforced correctly. The second part of our SSH policy is requiring SSH version 2. This was one of our configuration recommendations for the SSH server and to test it we will first telnet TCP port 22 to verify the it is listening for SSH.

```
[mbrown@mbrown mbrown]$ telnet green.giac.com 22
Trying ...A.B.C.D
Connected green.giac.com
Escape character is '^]'.
SSH-2.0-OpenSSH_3.1p1
```

We now have verified that it is offering the SSH service on that port. We can use a command line option with OpenSSH that will force the client to use protocol version 1 only. If we ssh to the host with the **-1** flag we will force protocol version 1.

```
[mbrown@mbrown mbrown]$ ssh green.giac.com -1
Protocol major versions differ: 1 vs. 2
[mbrown@mbrown mbrown]$
```

In our logs we see the following message stating that the SSH connection did not work properly.

```
Aug 20 16:54:10 green sshd[2277]: Did not receive identification string from a.b.c.d
```

We can safely conclude that our policies of not allowing a remote connection as root and forcing SSH protocol version 2 are successfully being implemented.

### **DJBDNS User Accounts:**

The main fear of running a daemon that faces the Public Internet is that of a compromise. When running certain DNS software implementations like BIND, you should be overly concerned since BIND runs as root. If a vulnerability was released in BIND that allows someone to take over the service, then you are in a bad situation. Now we ask ourselves the question, "What if someone is able to hack the tinydns user account, what would they be able to do then?" Remember, dnscache, dnslog, and tinydns all have an invalid shell, /bin/false, so they should not be able to even have a prompt in a shell. In the next example I am already root on our server and I try to become the user tinydns and dnslog.

```
[root@white mbrown]# su dnslog
[root@white mbrown]# su tinydns
[root@white mbrown]#
```

```
[root@white mbrown]# tail -f /var/log/messages
Aug 20 16:58:19 white su(pam_unix)[17639]: session opened for user dnslog by
mbrown(uid=0)
Aug 20 16:58:19 white su(pam_unix)[17639]: session closed for user dnslog
*****
Aug 20 16:59:01 white su(pam_unix)[17725]: session opened for user tinydns by
mbrown(uid=0)
Aug 20 16:59:01 white su(pam_unix)[17725]: session closed for user tinydns
****
```

Even as root you are unable to become a user with an invalid shell. With BIND you are able to configure it so that it runs chrooted, in that even if you do compromise BIND you cannot get anywhere. By default (which is the key here) djbdns comes with everything already set up in a chrooted fashion, no further modifications are needed to secure the DNS software. This is an extremely nice feature of djbdns, especially with all of the overworked system administrators out there.

The main threat of running an Internet exposed daemon is that your system could be compromised in some way. With djbdns the only way this can happen is through cracking the user account that the service is running under, i.e. tinydns, dnscache, or dnslog, or by cracking the root account on the box. In our situation both ways have been severely taken out of play in that the user account that our DNS services run under have no valid shell and no one is allowed to remotely connect to the box as root, they must first have a valid login account.

The necessary steps have been taken to make the DNS server as secure as possible. All that can be done now is to sit back and perform the daily/weekly/monthly maintenance and watch it like a hawk.

## References

Ziegler, Robert L. Linux Firewalls: Second Edition Indianapolis: New Riders Publishing, 2002

Brotzman, Lee E. and Beale, Jay. Securing Linux Step-By-Step: Second Edition The Sans Institute, 2002

Barret, Daniel J. and Silverman, Richard E. SSH, The Secure Shell: The Definitive Guide O'Reilly & Associates, Inc., 2001

Chuvakin, Anton " "Pocket" ISP based on RedHat Linux HOWTO" version 2.0.0. January 2001 <http://www.linux.org/docs/ldp/howto/ISP-Setup-RedHat-HOWTO.html>

Center for Internet Security. The Linux Benchmark v1.0.0" [http://www.cisecurity.org/bench\\_linux.html](http://www.cisecurity.org/bench_linux.html)

Bernstein, Dan "How to install djbdns" <http://cr.yp.to/djbdns.html>

Bernstein, Dan "How to install daemontools" <http://cr.yp.to/daemontools.html>

Bernstein, Dan "How do I configure an external cache" Frequently Asked Questions <http://cr.yp.to/djbdns/faq.html>

Bernstein, Dan "How do I configure a DNS server" Frequently Asked Questions <http://cr.yp.to/djbdns/faq.html>

Oram, Mark "Using and Securing a Red Hat Linux 7.3 server as a DNS, Mail, and Web server" May 2002 [http://www.giac.org/practical/Brian\\_Melcher\\_GCUX.doc](http://www.giac.org/practical/Brian_Melcher_GCUX.doc)

Melcher, Brian "Securing a Red Hat Linux 7.2 Anonymous FTP Server with Security Support syslog Server" April 2002 [http://www.giac.org/practical/Brian\\_Melcher\\_GCUX.doc](http://www.giac.org/practical/Brian_Melcher_GCUX.doc)

## Websites of Interest:

<http://www.djbdns.org>

<http://www.lifewithdjbdns.org>

<http://www.redhat.com>

<http://www.linux-backup.net/issue.gwif.html>