Global Information Assurance Certification Paper

Building a Secure OpenBSD Mail System on a Small Budget

Jesse Trucks
GCUX Practical v.1.9
Securing UNIX Step-by-Step
Submitted 04JUL2003

Abstract: 3
1. Production Goal 4
2. System Design 4
2.1. Hardware 4
2.2. Software 4
2.3. Design Philosophy 5
3. Risk analysis 5
4. Building the server 8
4.1. Create the boot floppy. 8
4.2. OpenBSD installation 9
4.3. Initial System configuration. 18
4.4. OS Source and Patches 24
4.5. Ports Tree for Third party software 26
4.6. Installing Third-party applications 26
4.7. Alter system boot scripts 30
4.8. Configure the mail daemons 31
4.9. Configure SPAM filtering software 32
4.10. Create user management script 38
4.11. Enable mail daemons 39
4.12. Reboot system 40
4.13. Functional system testing 40
4.14. Final Security lockdown 41
5. Maintenance and upgrades 44
5.1. Operating System Patches 44
5.2. OpenBSD upgrades 45
5.3. Ports upgrades 45
5.4. Upgrading third-party software from source 46
5.5. System Maintenance 46
6. Pre-production configuration review 48
6.1. SMTP relay check 48
6.2. Network services check 50
6.3. Authorized shell accounts 51
6.4. Root accessibility and audit ability 52
6.5. Filesystem NOSUID & NODEV check 53
7. Reference: 54

Abstract:
This paper describes how to build an OpenBSD mail server on an Intel platform, with a small budget. This system is designed for a small or mid-sized organization or department within a larger organization. The primary focus is on security throughout the process. This server will handle both SMTP traffic for sending mail out, and client's using either POP or IMAP protocols for message retrieval. The system uses the built in OpenBSD POP mail server, the third-party University of Washington IMAP server, the default OpenBSD sendmail server, and the bogofilter SPAM filtering software. This tutorial describes the process for installing OpenBSD from an FTP site, though it is readily adaptable to other methods of accessing the install files. Guidance during the install process for how to secure a base system allows even people new to security to follow and understand why to perform certain steps in a particular manner. After system installation, this tutorial continues by explaining what system configuration changes to make for improved security. After the core system is operational, third-party software compilation, installation, and configuration completes the install process. Following installation are some custom scripts for user and system management, including a partially automated SPAM software management method. Afterwards, maintenance and upgrade procedures, including regular system checks for security and functionality, outline a strategy moving forward for system management. Finally, a review and testing of several critical security issues wraps up the server build.

1. Production Goal

This tutorial describes how to build a mail server suitable for a small organization or department on an Intel hardware platform using OpenBSD 3.3. The goal is to provide a comprehensive mail solution with modern features, such as SPAM filtering and IMAP, for a small to mid-size organization without compromising on security or manageability and without high hardware or software purchasing and licensing costs. The end product provides a winning solution for management, IT staff, and users alike. The resulting system will serve as a mail gateway in addition to serving mailbox contents to POP3 and IMAP clients, and it will provide SPAM email filtering. This allows for a fairly secure and relatively maintenance free mail solution, even for small budgets. For those installations requiring more robust hardware or redundancy, they can follow these steps on a fancier system with hot-swappable drives, RAID arrays, or any other hardware feature available, and have the same resultant functionality, with the caveat that any additional features require an increased workload to configure and manage. These features are beyond the scope of this document.

2. System Design

2.1. Hardware

The hardware is an Intel Celeron 1.7 GHz CPU with standard main board and peripherals, including 512Mb PC2100 RAM, a 40 GB IDE hard drive, and a single Ethernet interface. This hardware is far beyond minimal requirements necessary for a moderate mail server, but the hardware cost, not including a monitor for use as a console during installation, system upgrades, and emergencies, is below $500US, therefore, this is a good entry-level server package for nearly any budget.

2.2. Software

The Operating System is OpenBSD 3.3. The mail system uses the OpenBSD default Sendmail 8.12.9 for SMTP traffic, the OpenBSD system popa3d daemon for POP3 clients, and the University of Washington IMAP server v 2002.336 for IMAP services. The server uses the bogofilter package v.0.13.6.2 written by the author of Sendmail, Eric S. Raymond, for SPAM mail filtering. Backups and some aspects of system management are beyond the scope of this tutorial. Though it will cover some suggested approaches, integrating this server into your existing infrastructure should be a relatively trivial task, depending on your site configuration and

policies. A basic familiarity with UNIX systems is a requirement for understanding the configuration of the server and the above listed packages.

2.3. Design Philosophy

A major design philosophy behind this implementation is to use the most basic software possible with minimal security or maintenance difficulties. A server should utilize as many pieces of the default Operating System as possible in order to minimize both the introduction of new security problems and additional upgrade and configuration maintenance issues. If a server uses third-party applications, there are three ways security and configuration problems arise.

First is the typical commercial software situation where a pre-built package supplied by the vendor is the only installation option. These packages could contain flaws in operation or security which lead to a general system compromise or denial-of-service (DOS) situation. Only install pre-built packages from a highly trusted source, and only if you can do some level of security review first.

Second, compiling and installing third-party software opens more opportunities for bugs to slip in and affect either functiconal operation or security, especially when replacing built-in components of the OpenBSD Operating System in particular. OpenBSD, as one of the most secure Operating Systems under development, audits the code of any software shipping as part of the base system, such as client tools and system daemons, and OpenBSD has easy maintenance as a driving philosophy. Also, regardless of the quality and security of a particular software package, anytime upgrades require compiling additional software, the complexity of system maintenance dramatically increases.

Finally, each new software package added to the base Operating System install potentially increases system configuration and maintenance, especially at upgrade time. Each add-on package, whether vendor supplied or compiled locally, adds pieces to the boot sequence, in the case of system daemons, or alters the user-environment, in the case of client utilities. Careful documentation and planning can prevent serious problems at system upgrade time, but each upgrade is more complex with third-party applications of any kind.

3. Risk analysis

When the ARPANET hackers first started using email, the entire network was a relatively closed system of researchers, students, and government personnel; therefore, security was not a consideration. Anyone could send email to any server, and every server would route email for anyone who sent mail from anywhere, to anywhere. Nobody could envision the security requirements of today's network connected systems, and SPAM didn't exist. Despite the critical nature of email in the modern world, the email messages themselves are still not secure or a fully reliable means of communication, but the underlying infrastructure has vastly improved. Today's email servers can restrict who can send mail to the mailboxes it serves; they can block the now problematical SPAM messages, and restrict who can route messages from and to any address. The SMTP daemon handles routing, or relaying, mail messages, and security is a critical issue for stopping unauthorized use of the server for relaying messages to other locations. A favorite method used by SPAM companies is to use other mail servers to relay large amounts of mail via someone else's mail server, thereby avoiding the bandwidth and other infrastructure costs normally involved with sending email. The POP and IMAP daemons must allow authorized access to the mailboxes on the server, but they must not allow access to other resources on the machine or unauthorized access to the mailboxes.

The supporting infrastructure, the Operating System and physical hardware, must be secure in order to protect the organizational assets – the system use, bandwidth, and content of the email for users of the system. Direct access to the system is restricted to System Administrators, and access to the administration account, root, is restricted to local authenticated access, yet security must allow for remote administration from outside the building housing the server hardware. This minimizes downtime and eases maintenance by granting System Administrators the ability to access and alter the system while not physically in the office. This is an ideal situation for a small company hiring out the maintenance of this system to a consulting firm or utilizing a small IT staff. Reduced travel and maintenance time directly translates to a lower bill from the consultant, or smaller labor costs for in house IT staffers.

Also, for supporting the necessary third-party applications, a local compiler must exist. If the organization has the budget and resources for a pair of servers, one is for development and testing, and one is for production use, omitting the compiler on the production system. In this scenario, all software is packaged on the development machine for binary installation to the production server. This tutorial covers the development server half of that installation method. If an attacker gains access to the system and the compiler is readily available, they could quickly build and install their own custom tools to further compromise and damage the server. As a

result, security is even more critical than for a system without programming tools and source code locally available.

The primary concern is that any network service required for providing mail functionality and remote administration must answer on the network in some fashion in order to perform the intended function. In certain situations, limiting where connections to the SSH daemon used for remote administration can originate from can provide increased security. Though this is an ideal configuration, especially for a remote office, or consultant, to remotely maintain the server, it is not practical for a small organization with limited network resources available. There are some situations where this is not possible, and those situations require a more watchful eye on the server. This tutorial assumes the resources available preclude placing limitations on access to the SSH daemon. For the other daemons running on the mail server, they must accept a connection from nearly any address on the Internet in order to properly perform their duties. As a result, a locally based firewall will block access to any unused ports and prevent certain network based attack types from causing a penetration or denial-of-service situation.

One often overlooked aspect of security is the physical safety of the hardware the software runs on. There are various hardware platforms providing hardware password access, security modes, and other methods to limit the danger of physical access to the console or internals of a machine, but these are really only mild deterrents. If someone wants access to the data on your machine, and they gain access to the physical console and hardware for your server, they will gain access to everything on the system. This might mean they take the hard drives with them to examine elsewhere at their leisure. For the purposes of this tutorial, the assumption is that your hardware is in a relatively safe location, such as a secure data center, or a locked room with restricted access in your organizations office.

In summary, the security risks involve possible exploitation of bugs in the various packages running on the server or core server components like the kernel in order to gain elevated privileges and execute code as root, configuration errors allowing unauthorized access to any system resources or successful denial-of-service attacks which prevents authorized use of the system, and securing local account access to the System Administration accounts to prevent the elevation of privileges via an internal administrative account. Physical security is as crucial as the software security, but physical security is fairly simple. Putting it in a safe building, in a locked room, and guarding the key is often sufficient for smaller organizations. The author has successfully used this method of physical security for a number of years for smaller systems. For larger

installations or more sensitive servers, a secure data center facility is the recommended solution to physical security.

The only services required for operating this system as SSH, POP, IMAP, and SMTP. All other network services are disabled. The only user accounts on the system are for the mail accounts and the System Administration accounts. The mail users do not require a shell, and they will not have FTP access to the machine, either. The mail accounts will only have access to their own email, and they will have no other system privileges. Because this system is only hosting email and no proprietary organizational data, the security requirements are only moderate.

4. Building the server

4.1. Create the boot floppy.

To install OpenBSD, the system boots from a floppy disk or the distribution CD-ROM (available from the OpenBSD web site for a nominal fee). This tutorial uses the FTP install method, so the system boots from a floppy for installing the Operating System. The installation environment differs from many UNIX flavors in that it is not a complete and functional OpenBSD, even when booting from the CD-ROM. There are no network daemons, the network stack boots without configuration, and most system commands are missing from the directory tree. Due to the restricted nature of the install kernel and operating environment, a secure network-based install of OpenBSD is possible, as opposed to other UNIX flavors where a network-based install is fraught with perils of system and network compromise before securing the host after installation. The OpenBSD development team built the installation procedure securely to purposefully avoid the security pitfalls of other UNIX install methods, especially those of Red Hat Linux and Sun Solaris. This tutorial assumes there is a MS-Windows system available for generating the initial boot floppy, but another UNIX system should suffice. Follow the procedures for generating a floppy available in the OpenBSD online FAQ for creating a boot floppy on a UNIX host.

On the MS-Windows system, download the `floppy33.fs` file and the appropriate floppy imaging tool for your version of windows (`rawrite.exe` for Win95/98 systems, and `ntrw.exe` for WinNT/2k/XP systems). Find the closest download site to your location on the FTP mirror site listed on http://www.openbsd.org/ftp.html.

Format a standard 3½ floppy disk. The disk can't have bad blocks, because the disk image requires all available space on a floppy disk.

Open a command window (Click `Start` -> `Run` and enter `command.com` for Win95/98 or `cmd` for WinNT/2k/XP). Change to the directory containing the

downloaded floppy image and imaging tool, and insert the formatted floppy into the floppy drive.

If your system is Win95/98, follow these steps:

Run `rawrite.exe`

At the `Enter source filename:` prompt, enter `floppy33.fs`

At the `Enter destination drive:` prompt, enter the letter designation for the floppy drive where you inserted the floppy.

Press `ENTER` at the final prompt.

If your system is WinNT/2k/XP, run `ntrw.exe floppy33.fs a:` replacing `a` with the letter of your floppy drive.

4.2. OpenBSD installation

The OpenBSD installation process is relatively straight-forward, and it is significantly simpler than most UNIX install procedures in use today. Basically, the system boots from the either the install floppy or the CD-ROM, prompts for disk setup and partitioning, prompts for where to find the installation files and which of those files to install, and then installs the specified files. The installation itself does not contain much in the way of third-party applications or client tools, but it includes the usual Internet mail, web, and remote access tools. After installation, the server reboots from the installed system for configuration and operation.

Insert the boot floppy created in the **Create boot floppy** section into the new server floppy drive, and boot the system from the floppy.

Once the white on blue text starts scrolling, the OpenBSD kernel is booting from the floppy. Take a moment to hold SHIFT and use the PAGEUP and PAGEDOWN keys to scroll through the kernel device discovery output, which is the white text with a blue background. Mostly, look for entries ending in `not configured`, as the kernel may not have a driver for that device loaded. Because this is a floppy install, some of the sound drivers and other peripherals may not load, though some of those devices are not relevant to this server's application even if the full OpenBSD kernel does not support the hardware at this time. The test system used for this tutorial did not have the sound drivers recognized, for instance, but the hard drive, Ethernet interface, and other obviously critical components were recognized and configured correctly from the floppy boot.

After the kernel loads, the system prompts with `(I)nstall, (U)pgrade or (S)hell?` Enter `i` for install, as this is a new system. At the subsequent

prompt, accept the default terminal type, as this is usually sufficient. The test system used the default `vt220` without issues.

If you are using a non-US-English default keyboard encoding table, choose `Y` at the `Do you wish to select a keyboard encoding table?` prompt. Otherwise, choose the default `n`.

Because this is a new system, and there is no data to save from the hard disk, enter `y` at the `Proceed with install?` prompt.

Choose the default disk `wd0` that appears for the root disk. Because this will be a dedicated single OS server running OpenBSD exclusively, answer `yes` to the `Do you want to use *all* of wd0 for OpenBSD?` prompt.

The install script will run the `fdisk` commands for you in order to set the active MBR partition to type `A6` for OpenBSD. You may see an error in white-on-blue text saying, `wd0: no disk label`. This is normal because it is a new disk with no disk labels.

The install script runs the disk label editor for you to manually generate the disk partitions. Start with a clean disk label by removing the default `a` partition which spans the whole disk. Enter `d a` to delete the `a` partition.

To view the current disk label, enter `p m` to print the label with megabyte sizes listed, instead of the default blocks, which more confusing than helpful. Never remove or alter the `c` partition. It is a required partition that spans the whole disk.

For this installation as a mail server, here is how to partition the disk space:

```
/             256Mb
swap          512Mb
/tmp          512Mb
/var         1024Mb
/usr         1024Mb
/usr/local   2048Mb
/home         512Mb
/var/www      512Mb
/var/mail   31747Mb
```

The root partition must be partition `a`, and the swap space must be partition `b`. The rest of the partitions can have any designation up to the

letter `p`. We will add them in the order listed, taking the default letters assigned except for `/` and `swap`.

Enter `a a` to add the `/` partition:

```
> a a
offset: [63]
size: [78124032] 256m
Rounding to nearest cylinder: 524097
FS type: [4.2BSD]
mount point: [none] /
```

Enter `a b` to add the `swap` partition:

```
> a b
offset: [524160]
size: [77599935] 512m
Rounding to nearest cylinder: 1048320
FS type: [swap]
```

Enter `a` to add the `/tmp` partition

```
> a
partition: [d]
offset: [1572480]
size: [76551615] 512m
Rounding to nearest cylinder: 1048320
FS type: [4.2BSD]
mount point: [none] /tmp
```

Enter `a` to add the `/var` partition.

```
> a
partition: [e]
offset: [2620800]
size: [75503295] 1024m
Rounding to nearest cylinder: 2097648
FS type: [4.2BSD]
mount point: [none] /var
```

Enter `a` to add the `/usr` partition.

```
> a
partition: [f]
offset: [4718448]
size: [73405647] 1024m
Rounding to nearest cylinder: 2097648
FS type: [4.2BSD]
mount point: [none] /usr
```

Enter `a` to add the `/usr/local` partition.

```
> a
partition: [g]
offset: [6816096]
size: [71307999] 2048m
Rounding to nearest cylinder: 4194288
FS type: [4.2BSD]
mount point: [none] /usr/local
```

Enter `a` to add the `/home` partition.

```
> a
partition: [h]
offset: [11010384]
size: [67113711] 512m
Rounding to nearest cylinder: 1048320
FS type: [4.2BSD]
mount point: [none] /home
```

Enter `a` to add the `/var/www` partition.

```
> a
partition: [i]
offset: [12058704]
size: [66065391] 512m
Rounding to nearest cylinder: 1048320
FS type: [4.2BSD]
mount point: [none] /var/www
```

Enter `a` to add the `/var/mail` partition. Take the default for `size` as this partition will contain the most data on the server.

```
> a
partition: [j]
offset: [13107024]
size: [65017071]
FS type: [4.2BSD]
mount point: [none] /var/mail
```

Enter `p m` to view your partition assignments. Below the disk geometry data, you will see:

```
16 partitions:
#      size  offset fstype [fsize bsize cpg]
  a:   255.9M    0.0M 4.2BSD   1024  8192  16  # /
  b:   511.9M  255.9M 4.2BSD
  c: 38147.0M    0.0M unused  0     0
  d:   511.9M  767.8M 4.2BSD   1024  8192  16  # /tmp
  e:  1024.2M 1279.7M 4.2BSD   1024  8192  16  # /var
  f:  1024.2M 2303.9M 4.2BSD   1024  8192  16  # /usr
  g:  2048.0M 3328.2M 4.2BSD   1024  8192  16  # /usr/local
  h:   511.9M 5376.2M 4.2BSD   1024  8192  16  # /home
  i:   511.9M 5888.0M 4.2BSD   1024  8192  16  # /var/www
  j: 31746.6M 6399.9M 4.2BSD   1024  8192  16  # /var/mail
```

Enter `q` to quit out of the disk label editor, and press `ENTER` at the `Write new label?` prompt, to commit the changes to disk. You will see:

```
The root filesystem will be mounted on wd0a.
wd0b will be used for swap space.
```

The install script verifies mount points for each file system after generating the disk label. At the `Mount point for wd0d` prompt, it should list `/tmp` as the default, so press `ENTER`.

At the Mount point for `wd0e` prompt, it should list `/var` as the default, so press `ENTER`.

At the `Mount point for wd0f` prompt, it should list `/usr` as the default, so press `ENTER`.

At the `Mount point for wd0g` prompt, it should list `/usr/local` as the default, so press `ENTER`.

At the `Mount point for wd0h` prompt, it should list `/home` as the default, so press `ENTER`.

At the `Mount point for wd0i` prompt, it should list `/var/www` as the default, so press `ENTER`.

At the `Mount point for wd0j` prompt, it should list `/var/mail` as the default, so press `ENTER`.

You will see the `Mount point for wd0d` prompt listing `/tmp` as the default again. Type `done` to exit mount point verification.

The screen will display `Done - no available disks found`. This is normal because the system only has a single disk, and the install script

goes through the partitioning process one disk at a time.

The screen displays:

```
You have configured the following partitions and mount points:

wd0a  /
wd0d  /tmp
wd0e  /var
wd0f  /usr
wd0g  /usr/local
wd0h  /home
wd0i  /var/www
wd0j  /var/mail

The next step creates a filesystem on each partition, ERASING
existing data.
Are you really sure that you're ready to proceed?
```

Enter `y` at the prompt.
The system now creates the filesystems specified in the above disk partitioning steps. Ignore any warnings for unallocated sectors in the last cylinder of the disk. Due to disk geometry, there will usually be a few unused sectors.

After a few minutes, the install script prompts for the system hostname. For this tutorial, the hostname is `gcux`, but be sure to enter the host name chosen for your site. Enter the host name to continue.

Press `ENTER` at the `Configure the network?` prompt to start configuring the network for an FTP install.

There should be one available interface on this system (the test system uses `vr0`), and it is the default option for the `Which one do you wish to initialize?` prompt. Press `ENTER` to configure this Ethernet interface for your network, and press `ENTER` again to choose the default hostname from the previous step for the system hostname for the `Symbolic (host) name` for your interface.

The script responds with:

```
The default media for vr0 is
      media: Ethernet autoselect (10baseT)
Do you want to change the default media?
```

Press `ENTER` to accept this default.

Enter the IP address for the test system on your private network. This tutorial uses `192.168.0.100`.

Enter the netmask for the network the host is on at the `Netmask?` prompt. The default setting of `255.255.255.0` is usually sufficient for a small local network, but check the network settings or with the network IT staff to verify. Ignore the `Done – no available interfaces found.` message, as this is normal after the install script configures the one interface available.

Enter the domain name for the host at the `DNS domain name?` prompt. This tutorial uses `domain.com` rather than a real domain name.

At the `DNS nameserver?` prompt, enter the IP address of your primary nameserver. Most likely the ISP provides the nameserver IP addresses unless the organization runs a name server of its own.

As this is an FTP install, press `ENTER` at the `Use the nameserver now?` prompt.

Enter the gateway address for the network at the `Default route?` prompt. The tutorial network uses `192.168.0.1` for the router.

Press `ENTER` when asked if to `Edit hosts with ed?` This option is necessary when there is not a secure Internet gateway available and the name to IP translations needs manual entry for the install process. The editor invoked is `ed`, and it is very difficult to use. Avoid using it if at all possible. Use IP addresses rather than host names to bypass using `ed`.

Press `ENTER` to skip making manual network changes at the `Do you want to do any manual network configuration?` prompt.

Enter a secure root password at the root password prompt. Make sure the password is eight characters or more, and it contains letters, numbers, case changes, and special characters. Make sure the password has a mnemonic device associated with it so you don't forget what it is, but don't make it something easily deduced.

Choose the FTP installation method at the `Where are the install sets?` prompt by entering `f`. If the site requires an FTP proxy, enter the URL in ftp://proxyhost.domain.com:8000 format. Make sure to get the port number correct! Because the tutorial site does not use an FTP proxy, it used the default by pressing `ENTER`.

If known, enter the URL of the OpenBSD FTP mirror site, or press `ENTER` at the `Do you want to see a list of potential ftp servers?` prompt to see a list of available servers. The list scrolls one screen at a time, advancing with the space bar. Look at the physical location listed

in the far right column to locate the closest mirror. Make note of the number in the first left-hand column for the chosen site. The test system for this tutorial is in the Midwest United States, so the number 66 mirror, the site listed in Lake in the Hills, IL, USA is nearby.

At the `Server IP address, hostname, or list#?` prompt, enter the number for the server you chose from the list. Verify that the correct host name is the default, which is `rt.fm` for the chosen tutorial mirror, and press `ENTER` if it is correct.

If the installation uses a standard mirror from the list, press `ENTER` to choose `yes` to the `Does the server support passive mode ftp?` question, unless the network uses an FTP proxy that does not support passive mode transfers. If unknown, press `ENTER` as most FTP sites do support passive transfers.

If using a standard mirror, press `ENTER` at the `Server directory?` prompt to accept the default (the chosen mirror for this test system listed `pub/OpenBSD/3.3/i386` as the server directory, which is a fairly standard location). If using a local FTP server, enter the path to the distribution files at this prompt.

At the `Login?` prompt, press `ENTER` to choose the `anonymous` user if using a standard FTP site. If installing from a private server, enter the username required to access the files on the FTP server, in which case the script prompts for the password for that account.

The server will now attempt to contact the FTP server and obtain a list of available installation files. If everything works correctly, it displays some instructions and a vertical file list like this:

```
       [X] bsd
       [ ] bsd.rd
       [X] base33.tgz
       [X] etc33.tgz
       [X] misc33.tgz
       [X] comp33.tgz
       [X] man33.tgz
       [X] game33.tgz
       [ ] xbase33.tgz
       [ ] xshare33.tgz
       [ ] xfont33.tgz
       [ ] xserv33.tgz
File name? (or 'done') [bsd.rd]
```

The files marked with an `X` will install to the system. The game package is unnecessary on this server. To remove it enter `-game33.tgz` at the prompt.

The list displays without the `game33.tgz` package marked. Next, remove the `misc33.tgz` file, as this contains the system dictionary files and some documents ready for printing, such as the System Managers Manual. Though these are important documents, they are not necessary for this server's operation. To remove the file, enter `-misc33.tgz` at the prompt.

This system will never run an X server, so do not add the four X server component sets, `xbase33.tgz`, `xshare33.tgz`, `xfont33.tgz`, and `xfont33.tgz`.
The final list displayed looks like this:

```
       [X] bsd
       [ ] bsd.rd
       [X] base33.tgz
       [X] etc33.tgz
       [ ] misc33.tgz
       [X] comp33.tgz
       [X] man33.tgz
       [ ] game33.tgz
       [ ] xbase33.tgz
       [ ] xshare33.tgz
       [ ] xfont33.tgz
       [ ] xserv33.tgz
File name? (or 'done') [bsd.rd]
```

Because this system is self-supporting, it will have programming tools onboard. These tools are in the `comp33.tgz` package. The bsd file is the bootable kernel, `base33.tgz` holds the barebones system, `etc33.tgz` contains the configuration files (which are in a separate package for upgrading purposes), and `man33.tgz` contains the man pages. It is important to note here a feature for replicating systems and upgrades in the future. If a file called site33.tgz was on the FTP server, it appears in the list as well as the others. This file can contain any files required for a customizing the local install. If more than one system is required, the first system's packages and customized configuration files are places into the site33.tgz file with fully qualified paths used in the tar wrapper. Those files install in the same manner as the core Operating System components. Use this method to save time when deploying future systems with similar configurations.

To continue with the install, enter `done` at the prompt. The install script displays, `Ready to install sets?` Enter `y` to begin.

The script will download the files in the order listed above, and there is a progress bar listed for each file. After it downloads the files, it will uncompress them onto the hard drive partitioned earlier in this

process. This will take some time depending on network traffic and your connection speed, as it transfers several dozen megabytes of data.

After the files transfer and install, the `Sets can be located on a ...` prompt returns again. Enter `done` to continue.

Next the install script asks, `Do you expect to run the X Window System?` Answer `no`, because, as mentioned previously, this system will not run X windows.

The install script generates the `host.random` file and asks what time zone to use. The tutorial system is in the Midwest USA using Central Standard Time (CST), and the time zone file matching CST is `US/Central`. If the correct file entry to use is unknown, enter `?` for a list of available entries. Enter the appropriate entry for the local time zone.

The install script completes the install process by creating the devices in `/dev` and installing a boot block to the boot disk.

Finally, a shell prompt appears. To verify the system boots correctly, use a full power cycle to make sure nothing goes wrong on the first system boot.

Enter `halt` at the prompt, and the system will halt and sync the disks. Then it will display:

```
The operating system has halted.
Please press any key to reboot.
```

Turn off the power for the server and remove the boot floppy to complete the installation process.

4.3. Initial System configuration.

Before deploying the server into production, some default settings require altering for security and functionality, and some additional software needs installing to support the intended function as a mail server with SPAM filtering. The OpenBSD system installs with many features considered best practices for security, such as system daemons disabled, file permissions set correctly, and a minimal number of applications to reduce risk and complexity.

Begin configuring the system by powering on the system.

Assuming all went well with the boot sequence from the newly installed Operating System, log into the console as root, and accept the default terminal type, `vt220`, by pressing `ENTER`.

The only network services answering on boot for new OpenBSD systems are SSH, time services, and `ident`. The `comsat` services, used by local mailers to determine when new mail arrives for a user using an interactive shell, only answer to `localhost`, but the IPv6 entry is not needed. First, disable root SSH access, then add an administrative user, and comment out the unnecessary network services before putting together the rest of the system.

Start by editing the `/etc/ssh/sshd_config` file and uncomment and change the line

```
#PermitRootLogin yes
```

to read

```
PermitRootLogin no
```

Restart the `sshd` daemon by running

```
kill -1 `cat /var/run/sshd.pid`
```

You should see a message appear on the console:

```
sshd[#####]: Received SIGHUP; restarting.
```

Where the `#####` is the PID of the sshd process. There may be one or two messages about the ports `sshd` is listening on, as well.

Next, create an administrator account. System Administrators should not directly log into the `root` account. Instead, they should log into the machine using their own account, which means each System Admin has their own private account, and gain `root` privileges using `sudo`, or `su` in extreme cases where `sudo` is broken. This allows for an auditable record of `root` access and the commands used by each administrator. Not only does this provide accountability for each person's actions, it provides for change management and faster root cause discovery above and beyond the obvious security implications

Choose a unique name for this account. If there are other administrators, add those accounts later, just before the server goes to production. Choose a cryptographically sound password, with case changes, letter, number, special character combinations. The maximum password length is 128 characters, so use the longest password you can remember. To add the account, enter `adduser` at the prompt.

Before adding the user, the `adduser` program will prompt for some system defaults. First, it requests the default shell. Choose either `csh` or `ksh` depending on your chosen environment. I come from a `sh`/`ksh`/`bash` background, so I choose `ksh` (though I generally install `bash` on my systems and use that instead, this system won't have bash installed as part of this tutorial).

For the default user home directory on this system, accept the programmed default, `/home`, as there is a separate partition allocated for this directory tree.

Press `ENTER` to copy the dotfiles from the system default location of `/etc/skel` and press `ENTER` to set the new user message to `no`. This system won't have many users, and it more than likely will not need a new user message. If your site policies are different, customize the new user message and change this setting after the system enters production.

Press `ENTER` to have `adduser` prompt for passwords when creating new accounts. Then, press `ENTER` to accept `blowfish` as the default password hashing algorithm. Blowfish is more secure than the other options available, and it is a fairly efficient encryption scheme. Using `blowfish` will keep the passwords safe for years to come.

With the new user defaults set, `adduser` now prompts for the new username. Enter the chosen username at the `Enter username [a-z0-9_-]:` prompt. For the tutorial system, the account is just `admin` though remember to create accounts for each administrator with their own name, user id, and other site specific conventions.

Enter the administrator's name, as it should appear in the GECOS field of the password file. Keep in mind that this entry is the default name on outbound email sent from the system.

Press `ENTER` to accept the default shell (unless for some reason this account should have a different shell than chosen earlier as the system default).

Press `ENTER` to accept the default UID (which should be 1000), and enter `staff` for the default login group. Enter `wheel` at the `Invite admin into other groups` prompt. This allows the account to `su` to the `root` account and gain other elevated privileges you need for administrating the server. It is vital that only authorized System Administrators have accounts in the `wheel` group, as those accounts have full `root` access

via the sudo command. The sudo configuration occurs later in this process.

Enter a good password at the password prompts, heeding the earlier stated advice on choosing passwords.

The adduser script displays the entries made and prompts to accept the new account as entered with OK? Press ENTER if everything looks good (no typos, etc), otherwise enter n to re-enter the data correctly.

The adduser script creates the user and copies the default environment files to the user's home directory. It, also, prompts to add another user. Enter n to exit the script.

To allow root access to the new admin account with an audit trail, setup sudo to allow the account to run any command as root. The advantage in using sudo over just su is three-fold. First, no user uses the root password for root level access; they type their own password, instead. This reduces the chances of someone discovering the root password by access to the server, the network, or the client connecting to the server to zero. Second, sudo keeps a log of all commands run via sudo, so all administrator level activity is auditable. The one instance where this logging becomes useless is when an administrator opens an interactive root shell. Unless you create a wrapper for the shell which logs all shell activity by user of origin, the commands entered via the root shell are not traceable to a single person. Having timestamp or username data in the root history is another approach, but the author has yet to figure out a reliable method of doing so. Whoever gains root privileges can bypass any method to create an audit trail for root access, so don't rely on a single method. Generating the habit of using sudo for every command, or using scripts via sudo to run a serious of commands, is the best approach. Lastly, using sudo allows granular control over privileged access to any command as any account, not just root. This means access to applications or resources owned by their own account, such as group data repositories or application administration accounts like www, can be controlled with an audit trail, too. Any user on the system who requires access to a special account uses sudo instead of learning that special account password. You can even lock the password on most special accounts and allow access via sudo only.

Run visudo and, uncomment the line:

```
# %wheel    ALL=(ALL)   ALL
```

to allow the accounts in group wheel to perform all commands as root via sudo. Save the file and exit.

Test the sudo-based root access by running su -l admin to get a prompt as the admin user (substituting the administrative user created earlier in this process for the username admin). As the administrative user, run sudo -l to get a list of commands available via sudo for this user. The sudo command will prompt for the user's password, not the root password, as su requires. Enter the account's password at the prompt.

If you don't see the lines below, double check your editing of the sudoers file in the steps above:

```
User admin may run the following commands on this host:
      (ALL) ALL
```

Exit the administrative account.

Next, test remote ssh access to the administrative account.

From any other system on the network, ssh to the new server. The ssh client will prompt to accept the host key and continue the connection. Accept the host key to log into the gcux server. You should successfully get a prompt on the gcux host.

You can now safely finish the rest of the configuration via the SSH connection, but the console is still a viable option. Using the console limits the ability to copy and paste, capture output, and log messages clutter the primary console, so it can be more cumbersome to work with. Because the system is on a secure network, with no external access to the gcux system, and it is not running any server daemons (as already stated, most everything dangerous is off by default on an OpenBSD server), ssh is a very safe method to use.

If using the console for the reminder of system setup and configuration, log out of the ssh connection. If using the ssh session, log out of the console.

Correct the system time and date:

```
sudo date CCYYMMDDHHMM.SS
```

where CC=Century (presumably 20), YY=Year, MM=month, DD=day, HH=hour, MM=minute, and SS=second. The following command corrected the date for the test system:

```
sudo date 200306291520.50
```

and date showed:

```
Sun Jun 29 15:20:50 CDT 2003
```

for confirmation.

Add additional DNS server IP addresses to the `/etc/resolv.conf` file, in case the primary DNS server becomes unavailable at any time. Most likely there is more than one DNS server available for the network. Add at least one more to the end of the `/etc/resolv.conf` file by adding an entry like:

```
nameserver 192.168.0.222
```

substituting the correct IP address of the DNS server for the `192.168.0.222` address listed.

Next, verify the mount points and mount options. Partitions not containing system binary executables should not allow SUID execution, and only the `root` filesystem, which contains the system device tree, should allow device files. Otherwise, if someone gains elevated code execution privileges, they can install a SUID binary or add device files which allow back door to `root`-level privileges. Correctly mounting the various parities and restricting access to the `root` partition prevents this activity. The mount settings are correct by default, and should look the same as shown in the `/etc/fstab` file. Here are the contents of the `/etc/fstab` file and the mount command for the test system:

`/etc/fstab`:

```
/dev/wd0a / ffs rw 1 1
/dev/wd0h /home ffs rw,nodev,nosuid 1 2
/dev/wd0d /tmp ffs rw,nodev,nosuid 1 2
/dev/wd0f /usr ffs rw,nodev 1 2
/dev/wd0g /usr/local ffs rw,nodev 1 2
/dev/wd0e /var ffs rw,nodev,nosuid 1 2
/dev/wd0j /var/mail ffs rw,nodev,nosuid 1 2
/dev/wd0i /var/www ffs rw,nodev,nosuid 1 2
```

mount:

```
/dev/wd0a on / type ffs (local)
/dev/wd0h on /home type ffs (local, nodev, nosuid)
/dev/wd0d on /tmp type ffs (local, nodev, nosuid)
/dev/wd0f on /usr type ffs (local, nodev)
/dev/wd0g on /usr/local type ffs (local, nodev)
/dev/wd0e on /var type ffs (local, nodev, nosuid)
/dev/wd0j on /var/mail type ffs (local, nodev, nosuid)
/dev/wd0i on /var/www type ffs (local, nodev, nosuid)
```

Though OpenBSD boots out of the box with very little in the way of network services enabled, there are still some services which the paranoid System Administrator ought to disable. To disable the `ident` and `time` services, edit the `/etc/inetd.conf` file to stop these services from answering on the network. The IPv6 `comsat` entry answering on the IPv6 network is not necessary, though the IPv4 entry answering on `127.0.0.1` is fine to leave enabled. Comment out the following lines:

```
ident           stream  tcp     nowait  _identd usr/libexec/identd     identd -el
ident           stream  tcp6    nowait  _identd /usr/libexec/identd     identd -el
[::1]:comsat  dgram   udp6    wait    root    /usr/libexec/comsat     comsat
daytime         stream  tcp     nowait  root    internal
daytime         stream  tcp6    nowait  root    internal
time            stream  tcp     nowait  root    internal
time            stream  tcp6    nowait  root    internal
```

Restart inetd by sending the process a HUP signal to enable these changes. Do this by running:

```
sudo kill -HUP `cat /var/run/inetd.pid `
```

At this point, the server is completely setup for proper administration and use on a network. For any server built on OpenBSD as a network server of some kind, follow all the above steps in the same manner for a base platform to start with. Though many of the following operations are applicable to any server setup, the output will show custom settings for this server's intended purpose. If adapting this tutorial for other server functions, follow the remaining steps by changing the third-party software installed and he network services enabled to match the intended use.

4.4. OS Source and Patches

There is a large debate in the security community regarding whether to keep source code and compilers on production systems. The real issue is whether a particular implementation requires more security or more manageability, and in some cases, as with the system this tutorial describes, increased manageability or self-supporting systems require local source code and compilers. All packaged Operating System

distributions release patch updates to fix various critical security and functional bugs discovered after releasing a particular version of the Operating System. OpenBSD releases patches as source code updates. To use them, the OS source must be available to generate the resultant patched versions of the updated files. For networks with more than one installation of the same Operating System version, use one system to generate the updated files for packaging and installing to the other same version systems. For smaller operations, such as outlined in this tutorial, there is only one stand-alone server, so it does contain the compilers and maintains the Operating System source for patching purposes.

OpenBSD uses a Concurrent Versions System (CVS) source system for obtaining all OS and ported software source files and patches. To obtain the source code for the 3.3-release version of OpenBSD used on this system, start by opening a `root` shell with `sudo su -l` for a full login shell (this correctly reads `root's` profile data). For those not familiar with `csh`, run a `ksh` manually after gaining a `root` shell in order to use familiar commands. For `csh` users, adjust the environment commands below for `csh` syntax, but `ksh` users can copy and paste from the examples if using the same CVS servers.

On this server, place the source files in the `/usr/local` partition due to space and to keep `/usr` clean. Therefore, create `/usr/local/src-trees` and `/usr/local/src-trees/src`, then link `/usr/src` by running:

```
ln -s /usr/local/src-trees/src /usr/src
```

Find the nearest anonymous CVS server to you from the master OpenBSD list found on their site (at the time of writing it was located at http://www.openbsd.org/anoncvs.html#CVSROOT). Then, set the CVSROOT variable (the suggested method is setting it in your .profile unless the server accessed changes often). With the test system being in the Midwestern United States, the University of Michigan based system, `anoncvs6.usa.openbsd.org` is nearby. To use this server, enter:

```
CVSROOT=anoncvs@anoncvs6.usa.openbsd.org:/cvs;export CVSROOT
```

Change the current directory to `/usr`. Then, download the OpenBSD 3.3 sources from the CVS server chosen by running:

```
cvs -q get -rOPENBSD_3_3 -P src
```

Because the CVS system uses `ssh`, the first time it connects to a remote system it will prompt you to accept the remote host key. Enter `yes` to continue.

CVS now downloads all the OpenBSD source files for the 3.3 release version. This took over 3 hours on a fairly fast connection for the test system, so schedule time accordingly.

Normally any patches available for the current release (which is 3.3 at the time of this writing) are available for download from the OpenBSD web site at http://www.openbsd.org/errata.html, but there are no critical patches available for the 3.3-stable release at this time. Therefore, the system is now ready for future patches as they come with minimal time needed for downloading, building, and applying the patches, yet requires no patching at this time.

4.5. Ports Tree for Third party software

For the additional functionality required to support the required services on this system, some third party software comes from the OpenBSD ports tree system. The ports system is a source-based package building and managing tool for third-party software. It is equivalent to Red Hat's RPM system or the Solaris package management system. It uses a combination of building software from source with patches to smoothly integrate the software into the OpenBSD system and a binary package management system for installing and maintaining those packages. Obtain the files for the ports system via the same CVS system used for the base Operating System source code.

Traditionally, the ports system installs in `/usr/ports` though this location is not required. This system maintains source code under `/usr/local`, so create `/usr/local/src-trees/ports` and link `/usr/ports` to it by running:

```
ln -s /usr/local/src-trees/ports /usr/ports
```

Change the current directory to `/usr`, and obtain the 3.3-release branch of the ports system by running:

```
cvs -q get -rOPENBSD_3_3 -P ports
```

This process completes fairly quickly compared to the `src` CVS retrieval. It took less than 20 minutes to download the ports files on the same connection that took over 3 hours for the full `src` download.

4.6. Installing Third-party applications

Many third-party applications are available via the ports system, including several security tools used later in tutorial and some application related software for running the mail server. Generally, installing software from the ports collection is straight-forward. Move into the directory containing the software package desired, configure, compile, and install the software in three easy steps. Unless some instance requires a newer version of a particular package available in the ports collection or a package is not available in the ports collection, use the ports collection to install third-party applications. This method avoids many maintenance troubles at patch and upgrade time in the future.

First, start with the UW-IMAP package. Move into the `/usr/ports/mail/imap-uw` directory.

Type `make` to download, patch, and build the source. This package builds very quickly.

To install, type `make install`. It will build a package for the UW-IMAP files and install the package. The installation script displays some inetd.conf entries for the various pieces to the package. Copy only the imap entry, which should look like this:

```
imap    stream  tcp     nowait  root    /usr/local/libexec/imapd    imapd
```

Edit the `/etc/inetd.conf` file, and add this line commented out right after the `pop3` entry. Technically, the entry can go anywhere in the file, but it is very helpful to keep similar services together. The UW-IMAP software install is complete. Just before taking the server to production, the actual POP and IMAP daemons will be enabled, but they should remain turned off for the rest of the system configuration and installation steps of this operation.

The `procmail` package supports mail filtering with configurable rules. It is the software which calls the SPAM filtering software, `bogofilter`. For the purposes of this server, little understanding of how `procmail` works under the hood is required. In fact, installing `procmail` and copying a configuration file, called `.procmailrc`, for each user created is the extent of its use for this server. Advanced use of `procmail` is beyond the scope of this text.

To install `procmail`, move into `/usr/ports/mail/procmail`, and type `make` to download and compile the software. This compiles very quickly, so don't leave.

The text output strongly suggests running `procmail` SUID, but this is not required at all. The author strongly suggests not doing so. It works

fine without being SUID, and there is better security as a result. Type `make install` to build and install the package for procmail.

Before compiling the SPAM filtering package, the supporting software bogofilter requires needs to reside on the system. The bogofilter package requires the version 4 or higher of the Berkeley DB database management package. The OpenBSD 3.3 ports collection contains only a version 3 of this package, so build it from source. The Berkeley-DB v4 package requires use of the libtool package, which is available in the ports collection. Install libtool first, then the Berkeley DB software, then build bogofilter last.

To install libtool, change to the `/usr/ports/devel/libtool` directory, type `make` to download and compile the software, and then type `make install` to build and install the software package.

For compiling software not available in the ports collection or in the core OpenBSD system, create the `/usr/local/src-trees/compile` directory for building from raw source outside the `src` and ports distribution systems.

Next, go to http://www.sleepycat.com/download to download the latest release of the Berkeley DB v4.x source code. As of this writing, v4.1.25 with a single patch is the latest release. The fastest way to down the source via HTTP is with `lynx`. Copy the URL for the actual gzipped file (which was http://www.sleepycat.com/update/snapshot/db-4.1.25.tar.gz at the time of this writing) and run `lynx $URL` For example:

```
lynx http://www.sleepycat.com/update/snapshot/db-4.1.25.tar.gz
```

Once `lynx` loads the URL, enter `d` to download the file. After the download completes, press `ENTER` to save the file, and press `ENTER` again to save the file with the same filename as the remote file. Press `q` to exit `lynx`, and enter `y` when asked to quit out of `lynx`.

After obtaining the Berkeley DB file, `unzip` and `untar` the file contents into the `/usr/local/src-trees/compile` directory:

```
cd /usr/local/src-trees/compile ; tar zxvpf $PATHTODOWNLOADS/db-4.1.25.tgz
```

Download the patch file in the same manner used in obtaining the Berkeley DB source file, and place a copy of the patch file into the top level of the package's software tree, which is:

```
/usr/local/src-trees/compile/db-4.1.25
```

for the test system, if there is one for the currently available version. The current version, 4.1.25, does have a patch available. A copy is in the file named `4.1.25-patch.1` in the directory shown above.

From the top-level of the Berkeley DB package source files, install the patch by running:

```
patch < 4.1.25-patch.1
```

You may see output similar to:

```
Hmm...  Looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|*** fileops/fop_util.c 8 Jan 2003 05:01:56 -0000       1.57
|--- fileops/fop_util.c 12 Jan 2003 19:44:29 -0000      1.58
--------------------------
Patching file fileops/fop_util.c using Plan A...
Hunk #1 succeeded at 40.
Hmm...  Ignoring the trailing garbage.
done
```

Now, change to the `build_unix` directory contained in the Berkeley DB source directory, and configure the software by entering:

```
../dist/configure
```

Type `make` to compile the software, and if the compile completes without issues, type `make install` to install the software to the default location of `/usr/local/BerkeleyDB.4.1`. Take note of the install directory for the libraries. You will need this information for building the `bogofilter` package.

Finally, the installation of the support software packages for the `bogofilter` SPAM filtering package is complete.

The `bogofilter` software is not in the ports tree, either, so obtain the latest release from http://bogofilter.sourceforge.net/. Follow the SourceForge mirrors link, and find the copy in gzip format under the bogofilter-stable heading. The latest stable release at the time of this writing is 0.13.6.2. Follow the link and choose a download site nearby to save time and network traffic. Copying the shortcut direct to the file

and using `lynx` again will save some extra steps. Refer to the description of this method used in downloading the Berkeley DB files.

Open the archive into the `/usr/local/src-trees/compile` directory:

```
tar zxvpf $PATHTODOWNLOADS/bogofilter-0.13.6.2.tar.gz
```

Enter the new `bogofilter` directory created by the last step. To configure `bogofilter` for compiling run the following command to direct the configure script to the correct location of the Berkeley DB library files, substituting the directory path to the Berkeley DB installation if installed to a different location:

```
./configure --with-db=/usr/local/BerkeleyDB.4.1
```

Again, following the standard procedure, type `make` to compile the `bogofilter` software, and then, if no errors prevent the operation from completing, type `make install` to install the `bogofilter` software to the `/usr/local` directory structure.

This completes the installation of all application related third-party software and the support infrastructure required to maintain those packages.

4.7. Alter system boot scripts

To run sendmail on boot requires a change in `/etc/rc.conf`, but upgrades can easily overwrite this file. The recommended approach is to create the `/etc/rc.conf.local` file and make the boot changes to that file.

Copy the `/etc/rc.conf` file to `/etc/rc.conf.local`. Then, comment out or remove the lines in `/etc/rc.conf.local` at the end calling the `/etc/rc.conf.local` file:

```
local_rcconf="/etc/rc.conf.local"

[ -f ${local_rcconf} ] && . ${local_rcconf} # Do not edit this line
```

This will prevent a recursive loop on boot. With the /etc/rc.conf.local file sourcing itself endlessly.

Next, change the `sendmail` flags entry from:

```
sendmail_flags="-L sm-mta -C/etc/mail/localhost.cf -bd -q30m"
```

so it uses the `/etc/mail/sendmail.cf` file instead of the `/etc/mail/localhost.cf` configuration file. The new line should like:

```
sendmail_flags="-L sm-mta -C/etc/mail/localhost.cf -bd -q30m"
```

The `sendmail` daemon will correctly run on the next system boot, though the daemon won't answer on the network interface until that time (or until a manual restart of the daemon with the above flags).

4.8. Configure the mail daemons

The POP and IMAP daemons need no configuration, as they authenticate against the UNIX accounts automatically, and they read and write the same mailbox format and location as the default `sendmail` system. Though the POP and IMAP daemons should remain disabled until testing time for security purposes, they remain disabled during the final system configuration phases.

The `sendmail` system requires correct configuration to operate securely in the chosen environment. Because this mail server only handles one domain name, minimal configuration is required.

Edit the `/etc/mail/sendmail.cf` file to stop the banner displayed to any connecting server or client from advertising the `sendmail` server version in use. Change the `SmtpGreetingMessage` line:

```
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

to read:

```
O SmtpGreetingMessage=""
```

Also, because this is a single domain system, force sendmail to accept mail for the domain it should answer mail for by changing the Dj macro entry:

```
#Dj$w.Foo.COM
```

to read:

```
Djdomain.com
```

Remember to replace the `domain.com` domain with the correct domain for the local server.

The best approach is to comment out the original line and make a copy to edit so the default is clearly marked (and auditable). This method is useful for any text-based configuration file, but the `sendmail.cf` file is fairly complex, so this method is especially helpful here.

4.9. Configure SPAM filtering software

An important feature of modern mail servers is the ability to filter out SPAM from the users' mailboxes. There are two ways to accomplish this. First, have the mail server filter the SPAM out and file it away without the user interacting with it or even being aware of its presence. In this scenario, the user simply deletes the relatively few SPAM received in their email inbox, and is not a part of the filtering process in any way. Though initially attractive from both the users' and the server administrator's perspective, there are particular management issues involved that could cause problems in this case. If any false positives occur, that is real email being labeled as SPAM and filed away or deleted, the user never sees that email, and the original sender must resend the message and either the message must be re-written or somehow caught before it gets to the SPAM filter. The other option is for the server administrator to regularly review the SPAM messages filed away and make a manual determination whether any particular message is SPAM or real email. This requires a significant amount of manual overhead on the part of the administrator and is really an undesirable situation in which to be.

The second method is to filter the SPAM before it gets to the users' mailboxes, but file the messages flagged as SPAM into an IMAP accessible location for the user to check for false positives themselves. Also, if each user has a part in the SPAM filtering process, they can tune the filtering software for what they think is SPAM, not just what the System Administrator deems is SPAM. Also, instead of merely deleting false negatives, that is SPAM that slips through to the user's inbox, the user saves out the message to a folder designated as the SPAM drop box, which the server regularly clears out and uses to add to the SPAM database kept for that user. For many reasons, the second approach is

preferable, though it does take a slightly larger amount of disk storage, which could have consequences for larger installations. For the smaller system designed for this document, disk space for the low number of users is not a factor.

In accordance with the above reasoning, `bogofilter` will filter each user's mail separately. If possible, for the several days or weeks before installing this system, save several hundred or thousand SPAM as it comes into the current mail server or a personal email account. The author started using `bogofilter` with fewer than 500 SPAM messages saved, but that number increased to over 14,000 within four months. `bogofilter` captured most of those messages automatically.

If saving a notable amount of SPAM is not possible or is cumbersome, download a file full of SPAM from one of the links listed on Paul Graham's site at http://www.paulgraham.com/spamarchives.html. This tutorial will use the file found at http://www.dornbos.com/spam10k2.shtml.

First, enable `bogofilter` by creating the configuration file. The default file is sufficient out of the box for reliable functionality. Therefore, copy `/usr/local/etc/bogofilter.cf.example` to `/usr/local/etc/bogofilter.cf` for a complete configuration file to start out.

Create the `/usr/local/data` directory and create a template for the `procmail` package used to execute `bogofilter` on incoming mail. Name the file `procmailrc` and paste the contents below into the file:

```
# how to call procmail from your .forward (change jesse to your username)
### "|IFS=' ' && exec /usr/local/bin/procmail -f- || exit 75 #USER"

# global variables for procmail
MAILDIR=$HOME/mail
LOGFILE=$HOME/.Procmail.log

### run the mail through bogofilter, embed the Bogosity header
### this DOES NOT actually file spam to a separate file yet.
:0fw
| bogofilter -u -e -p

# -p passthrough, returns  useable exit code by procmail
# -e embed, puts the Bogosity header in

### check for an errorcode
:0e
{ EXITCODE=75 HOST }

### file the mail to spam-bogofilter if it's spam, checking for the
Bogosity setting.  You can omit this part and just sort/filter using your
mail client
:0:
* ^X-Bogosity: Yes, tests=bogofilter
spam-bogofilter
```

The comments are brief, but help to explain what each section does. Though elaborating on `procmail` rulesets is beyond the scope of this tutorial, this file tells `procmail` to run `bogofilter` with the message as input, and `bogofilter` returns the contents of the message, with an extra header inserted, back to `procmail` for further processing. Then `procmail` checks if the new header, `X-Bogosity`, flags the message as SPAM by pattern matching on the word `Yes`, and files the message to a the `spam-bogofilter` mailbox in the user's mail subdirectory if the header matches. If a message does not match the last rule, `procmail` files the message to the end of the user's inbox as a normally delivered message.

Copy the /usr/local/data/procmailrc file into the `admin` account home directory, name it .procmailrc, and change the `USER` text in the second line to `admin` so it looks like this:

```
### "|IFS=' ' && exec /usr/local/bin/procmail -f- || exit 75 #admin"
```

Though changing the `USER` to `admin` has no effect on the operation of procmail, the comments are confusing without this change to the text.

Later a wrapper for creating new users will utilize this file for generating each user's `.procmailrc` file.

`bogofilter` requires priming with at least one message of either known SPAM or known good email, which all `bogofilter` documentation refers to as HAM, which is why saving or downloading SPAM messages is required. Though the optimum is training `bogofilter` with an equal amount of both SPAM and HAM messages, as long as the numbers of messages used isn't too imbalanced, the daily operation of `bogofilter` is adequate.

For new users, generate a pre-built pair of database files for storing in the `/usr/local/data` directory. The `newuser` script created later in this tutorial quickly copies these files to the new user's home directory.

To save a couple steps, first, generate the `bogofilter` databases for the `admin` account. Assuming the downloaded and unarchived SPAM files from the URL listed above are located in `/usr/local/data/spamfiles`, run the following `bogofilter` command as `admin` from the `/usr/local/data/spamfiles` directory:

```
$ for FILENAME in `ls`;do bogofilter -s -M -I $FILENAME;done
```

There is no output from this command, but it creates `$HOME/.bogofilter` and generates the `bogofilter` databases, populating the `spamlist.db` file with the token data from the SPAM files used.

If there are any saved HAM messages, copy the mail in standard UNIX mailbox format onto the `gcux` system and run the following command for each file, substituting `FILENAME` for the actual file name:

```
$ bogofilter -s -M -I FILENAME
```

Generate the template for the `.forward` files used to invoke `procmail` for calling `bogofilter` on incoming messages. Create `/usr/local/data/forward` and paste in the following text:

```
"|IFS=' ' && exec /usr/local/bin/procmail -f- || exit 75 #USER"
```

Copy this file to `~admin/.forward`, and change `USER` to `admin`. The result looks like:

```
"|IFS=' ' && exec /usr/local/bin/procmail -f- || exit 75 #admin"
```

Change the ownership of the `~admin/.forward` file to `admin`.

Finally, copy the `~admin/.bogofilter/*.db` files to `/usr/local/data` to simplify setting up new users. Change the ownership of the files to `root`.

Next, create the two scripts called by each user's `crontab` file to clear their SPAM folder of saved false negatives for fixing the `bogofilter` databases and clearing their `spam-bogofilter` mail folder so they don't have hundreds or thousands of old messages to wade through each week when checking for false positives.

Create the `/usr/local/bin/bogospamcopy` script for clearing false negatives from the users' `mail/SPAM` folders and running the text through `bogofilter` to correct the databases for the error. Paste the following text into the script:

```
#!/bin/ksh

PATH=/usr/local/bin:/usr/bin:/bin

if [ "${USER}" == "" ]; then
        USER=`whoami`
fi

if [ "${HOME}" == "" ]; then
        HOME=`awk -F: '$1 ~ username {print $6}'  username=${USER} /etc/passwd`
fi

mkdir -p ${HOME}/mail
mkdir -p ${HOME}/SPAM

mv ${HOME}/mail/SPAM ${HOME}/SPAM/new.SPAM
touch ${HOME}/mail/SPAM
chmod 600 ${HOME}/mail/SPAM

bogofilter -Ns -M -I ${HOME}/SPAM/new.SPAM

cat ${HOME}/SPAM/new.SPAM >> ${HOME}/SPAM/spambox.${USER}
rm ${HOME}/SPAM/new.SPAM
```

The above script needs porting to a more efficient language, such as `perl` or `C`, but for a small server it is more than adequate.

Create the `/usr/local/bin/bogospamboxclear` script for moving the messages `bogofilter` files away as SPAM in the `mail/spam-bogofilter`

mailbox for each user. Paste the following text into the `bogospamboxclear` file:

```
#!/bin/ksh

PATH=/bin:/usr/bin

if [ "${USER}" == "" ]; then
        USER=`whoami`
fi

if [ "${HOME}" == "" ]; then
        HOME=`awk -F: '$1 ~ username {print $6}'  username=${USER} /etc/passwd`
fi

mkdir -p ${HOME}/mail
mkdir -p ${HOME}/SPAM

mv ${HOME}/mail/spam-bogofilter ${HOME}/SPAM/bogospambox
touch ${HOME}/mail/spam-bogofilter
chmod 600 ${HOME}/mail/spam-bogofilter

cat ${HOME}/SPAM/bogospambox >> ${HOME}/SPAM/spambox.${USER}
rm ${HOME}/SPAM/bogospambox
```

Again, this script is not as efficient as possible, but is more than adequate for a small server.

Each user needs a `crontab` file to run these commands automatically at specified intervals. The author recommends running the `bogospamcopy` script at midnight each day and the `bogospamboxclear` script once a week.

First, create the `admin` account `crontab`, by running `crontab -e -u admin` as `root`, or just `crontab -e` as `admin`. Paste these two lines into the editor and save the file:

```
@midnight /usr/local/bin/bogospamcopy
@weekly /usr/local/bin/bogospamboxclear
```

Next, change the permissions on the two new scripts so they are executable by everyone:

```
chmod 555 /usr/local/bin/bogospamcopy
chmod 555 /usr/local/bin/bogospamboxclear
```

The `bogofilter` system is completely ready for production for any user with the `.forward` file calling their `.procmailrc` file as shown for the `admin`

user, and a `crontab` to manage their SPAM folders. The new user creation script will copy the appropriate files into place for each new user, so `bogofilter` management is minimal for the System Administrator, though each user will spend a small amount of time managing their own SPAM and HAM filtering as noted above.

4.10. Create user management script

The system `adduser` script and the `useradd` programs create new user accounts, sets up their home directory, and allow for entering a new password for the user account, but with `bogofilter` installed as it is, each user needs a `.bogofilter` directory with their own database files, a `.procmailrc` file, and a `.forward` file to call `bogofilter` on incoming messages. As a result, a wrapper script needs to perform all the above functions for each new user account creation in order to reduce errors and omissions during any step required.

To create a secure new user account, set the shell to `/usr/bin/false` and choose a good password. Then, set up the correct files and file permissions to enable `bogofilter` for each new account. The best way to set some of the defaults in new accounts is by editing the `/etc/adduser.conf` file to force secure account defaults. Add the `/usr/bin/false` command as the first shell in the list, by changing the `shellpref` variable to look like:

```
shellpref = ('false', 'csh', 'sh', 'bash', 'tcsh', 'ksh', 'nologin')
```

Then, set the `defaultshell` variable to `false` like:

```
defaultshell = "false"
```

Lastly, set the `defaultgroup` variable to `users` like:

```
defaultgroup = users
```

Save the `/etc/adduser.conf` file. These settings are only defaults. Any of these settings are configurable per account through the batch or interactive modes of the `adduser` script.

The following script accomplishes all necessary steps for creating a new account on this server. It requires a manual password entry at the password prompts for security. Specify the username and person's full

name (in quotes), as it should appear in the `/etc/passwd` GECOS field, on the command line as shown in the usage statement. Paste the following into a new file called `/usr/local/sbin/newuser`:

```
#!/bin/ksh
#
# Create a new user, configuring bogofilter and procmail.
#

PATH=/usr/bin:/usr/sbin:/bin

if [ $# != 2 ]; then
        echo "usage: newuser username \"full name in quotes\"";
        exit 1
fi

USERNAME=$1
FULLNAME="$2"

echo "\nCreating user ${USERNAME} for ${FULLNAME}..."

adduser -group users -batch ${USERNAME} users "${FULLNAME}"
passwd ${USERNAME}


HOME=`awk -F: '$1 ~ username {print $6}'  username=${USERNAME} /etc/passwd`

echo "\nCopying SPAM filtering files..."

mkdir -p ${HOME}/mail
sed "s/USER/${USERNAME}/" /usr/local/data/forward > ${HOME}/.forward
sed "s/USER/${USERNAME}/" /usr/local/data/procmailrc > ${HOME}/.procmailrc
mkdir -p ${HOME}/.bogofilter
cp /usr/local/data/bogofilter/* ${HOME}/.bogofilter

chmod 644 ${HOME}/.forward ${HOME}/.procmailrc
chmod 600 ${HOME}/.bogofilter ${HOME}/mail

chown -R ${USERNAME} ${HOME}

echo "\nCreating user ${USERNAME}'s crontab file..."
crontab -u ${USERNAME} /usr/local/data/crontab
```

Change the permissions on the file for read and execute only by root:

```
Chmod 500 /usr/local/bin/newuser
```

4.11. Enable mail daemons

The last step in configuring the server for production readiness is enabling the POP and IMAP daemons. After enabling those two services, perform a

full system reboot by power cycling the server. On boot, the server should have POP, IMAP, SMTP, and SSH answering on the network, but no other services responding. Also, procmail and bogofilter should process mail sent to the admin account, and outgoing mail should work.

To enable the POP and IMAP services, uncomment those two entries in the /etc/inetd.conf file so they look like this:

```
pop3            stream  tcp     nowait  root    /usr/sbin/popa3d        popa3d
imap            stream  tcp     nowait  root    /usr/local/libexec/imapd    imapd
```

4.12. Reboot system

Shutdown the system cleanly with the shutdown command:

```
shutdown -h now
```

The system halts immediately.

Cycle the power on the hardware, and watch the console as the system boots. If the system configuration is correct, the mail services should start on boot, and the server is ready for mail testing and security review.

4.13. Functional system testing

When the server completes the boot cycle, conduct a review of the network services running and processes in the process table. Then, send and receive email to and from the `admin` user account to verify mail functionality, including `procmail` and `bogofilter` processing.

Start by logging into the `admin` account. A process display of all processes shows only 15 processes in the list. Three of those processes were for the `admin` account SSH connection, shell, and `ps` command itself. The output of `ps -auxw` should look like this display (the sendmail line wraps in the output below):

```
USER        PID %CPU %MEM   VSZ   RSS TT   STAT STARTED       TIME COMMAND
admin      8892  0.0  0.0   264   160 p0   R+      2:24AM    0:00.00 ps -auxw
root      27355  0.0  0.1   104   380 ??   Is      2:02AM    0:00.05 syslogd -a /var/empty/dev/log
root      11664  0.0  0.1    64   360 ??   Is      2:02AM    0:00.01 inetd
root       3988  0.0  0.2   356   872 ??   Is      2:02AM    0:00.27 /usr/sbin/sshd
root       7731  0.0  0.1   236   460 ??   Ss      2:02AM    0:00.01 cron
root      16418  0.0  0.1    48   408 C0   Is+     2:02AM    0:00.01 /usr/libexec/getty Pc ttyC0
root      14219  0.0  0.1    48   408 C1   Is+     2:02AM    0:00.01 /usr/libexec/getty Pc ttyC1
root      25339  0.0  0.1    48   408 C2   Is+     2:02AM    0:00.00 /usr/libexec/getty Pc ttyC2
root       3354  0.0  0.1    48   408 C3   Is+     2:02AM    0:00.01 /usr/libexec/getty Pc ttyC3
root      11318  0.0  0.1    48   408 C5   Is+     2:02AM    0:00.01 /usr/libexec/getty Pc ttyC5
root      14161  0.0  0.3   452  1248 ??   Is      2:10AM    0:00.05 sshd: admin [priv] (sshd)
admin     21599  0.0  0.2   404   988 ??   S       2:10AM    0:00.29 sshd: admin@ttyp0 (sshd)
admin     29318  0.0  0.1   396   316 p0   Ss      2:10AM    0:00.03 -ksh (ksh)
root      29810  0.0  0.2   888   956 ??   Ss      2:15AM    0:00.04 sendmail: accepting connections
(sendmail)
root          1  0.0  0.0   340   200 ??   Is      2:02AM    0:00.02 /sbin/init
```

Send mail from the `admin` account to any verifiable email address on a different server, and verify the message arrives in a few moments by checking that address. Next, send email from that account back to the `admin` account on the new server. Make sure the reply message has real text in it, or `bogofilter` will catch it and drop the message into the user's `mail/spam-bogofilter` file. Check its arrival by looking in the `/var/mail` directory to find a file named for the user and look at its contents. The test email should be in that file, and it should have an `X-Bogosity` header near the bottom of the headers.

If all these checks work as expected, the mail server is now functional.

4.14. Final Security lockdown

There are some final steps to securing the server before the installation and configuration process closes. OpenBSD, like most BSD systems, runs nightly system configuration and security audits automatically, so some of the security work is easier to deal with. The nightly, weekly, and monthly scripts run from `cron` and send their results to `root's` mail. If time permits, its best to leave the system running over night and allow the scripts to run at their normal time. This confirms `root` can receive mail and tests the built-in system checking scripts. The daily `security` script checks for changes in key configuration and system files, and shows those changes, checks consistency of the password and group files, and file permissions for critical files, including changes, additions, or deletions to SUID and SGID files. The `daily` script checks the mounted partition space usage, displays the mail queue, and shows network statistics for the various network interfaces. The first security report shows all the new devices, as there is no base to compare the results. The second day's results are more telling, and show any inconsistencies in the file permissions, configuration changes, and other critical security issues.

In the case of the mail server, the `/usr/src` directory is a link, though normally it is a directory, and the group ownership shows this directory should be `wsrc`, not `wheel`. To change the group permissions, enter

```
chgrp -R wsrc /usr/src /usr/local/src-trees
```

Unless another disk is added to the system, `/usr/src` will remain a link, not a directory. Therefore, ignore this error, though the difference should appear in documentation, or change the `mtree` entry for `/usr/src` to reflect the new local policy. Remember upgrades may overwrite any change of this nature, so document any change to base system configurations.

The `mtree` system is a utility for checking the file system against a known baseline, much like Tripwire or other similar tools. Because `mtree` comes built into the system, its use is preferred above third-party applications, as noted earlier in this document. The specifications `mtree` follows are in `/etc/mtree`. To change the above mentioned `/usr/src` link definition, edit the `/etc/mtree/special` file, and change the src entry from:

```
src   type=dir mode=0775 uname=root gname=wsrc ignore optional
```

to:

```
src   type=link mode=0775 uname=root gname=wsrc ignore optional
```

The security script runs a fairly simplistic tripwire-like integrity check on the files listed in the `/etc/mtree` maps every night. Keep watch on the output of these scripts, as they often show signs of trouble.

Make sure someone reads `root's` mail every day. A common method is to send `root's` mail to the current administrator on call, or the only administrator for a smaller organization. To do this, enter an alias for `root` to the email address for the administrator into the `/etc/mail/aliases` file:

```
root: admin@domain.com
```

After making changes to the `/etc/mail/aliases` file, run the `newaliases` command.

Disable the shell for the `uucp` system account by running `chsh uucp`. The command opens the user information in an editor. Change the shell line to read `/sbin/nologin`.

Change the /etc/motd file so it displays a warning:

```
Warning! Only authorized use of this system permitted.
```

Though OpenBSD comes with `tcpwrappers`, this server runs no services which currently require IP based access control, so this function is unused at this time. In the future, if a need arises to implement `tcpwrappers` controls, a quick addition to the `/etc/inetd.conf` entries for the POP and IMAP services is required. The details are found in the OpenBSD `inetd.conf` man page.

The usual danger of `.rhosts` file entries is missing due to the lack of `telnet` or r-command usage on this host.

The `sendmail` daemon has privilege separation and runs as a user other than `root` for most of its operation, so this implementation of `sendmail` is more secure than most other instances. This eliminates the need for active paranoia on the usual level when running `sendmail`.

This server runs no RPC services, so all the inherent security issues for those services are not an issue.

Check for world writable files and directories before turning the system over to production. Any directory, besides certain temporary file spaces such as `/tmp`, should not be world writable. Use find to discover any problem files:

```
find / -perm -0002
```

The resulting list is surprisingly long, though it contains many device files, whose permissions should remain as they are. The list below shows that the bogofilter directory and its files need the permissions changed:

```
/usr/local/src-trees/compile/bogofilter-0.13.6.2
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/dcdflib
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/dcdflib/src
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/dcdflib/doc
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/inputs
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/outputs
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/bogoutil
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/bogofilter
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/bogofilter/inputs
/usr/local/src-trees/compile/bogofilter-
0.13.6.2/src/tests/bogofilter/inputs/split.d
/usr/local/src-trees/compile/bogofilter-0.13.6.2/src/tests/bogofilter/outputs
/usr/local/src-trees/compile/bogofilter-0.13.6.2/doc
/usr/local/src-trees/compile/bogofilter-0.13.6.2/doc/programmer
/usr/local/src-trees/compile/bogofilter-0.13.6.2/contrib
/usr/local/src-trees/compile/bogofilter-0.13.6.2/tuning
```

Use chmod to fix the permissions:

```
chmod -R o-w /usr/local/src-trees/compile/bogofilter-0.13.6.2
```

This concludes the installation, configuration and securing of the OpenBSD mail server outlined in this tutorial.

5. Maintenance and upgrades

5.1. Operating System Patches

OpenBSD patches are fairly simple to install, and there usually are very few of them, so this portion of maintenance takes very little effort and time, though is one of the more critical pieces to keep up with. First, join at least the OpenBSD `security-announce` mailing list, (information found at http://www.openbsd.org/mail.html). This mailing list releases security information and URLs to obtain the latest patches. Whenever the list announces a patch, go download the patch file as soon as possible. After downloading the file, open it in the root directory of the system source code, `/usr/src`. For detailed instructions for installing patches, refer to the OpenBSD online FAQ section of patches at http://www.openbsd.org/faq/faq10.html#Patches. Below is a brief overview.

Once the patch files are unarchived in the /usr/src directory, run the patch command. The example assumes the file name is `999_sendmail.patch`:

```
patch -p0 < 999 sendmail.patch
```

This command updates any files necessary for the patch update. Next, change to the directory of the patched package, and run:

```
make obj && make depend && make
```

If everything appears to compile cleanly, run `make install` to commit the changes.

Do this with each patch as it comes along, and the system stays secure, manageable, and current.

5.2. OpenBSD upgrades

Upgrading an OpenBSD is very straight-forward. Create a boot floppy or boot from a purchased OpenBSD release CD, and at the `(I)nstall, (U)pgrade or (S)hell?` Prompt, enter `u` to upgrade the system. The `upgrade` script follows the same procedure as the install described earlier in this document, but skips the disk configuration and does not install the `etc##.tgz` files. After the upgrade script completes, manually compare the files in `/etc` to see what changes to the new versions the system requires in order to function correctly. Use the `diff` command on each file one at a time, and manually edit the new versions before installing them, again one at a time, until further experience with the operation allows for better automation. There is little else to upgrading the system, though sometimes OpenBSD has a critical system change that requires more work than this. Always check the mailing list archives, FAQ, or new version announcement for details concerning any serious changes requiring more work than listed here for the current version. After any upgrade, the /usr/src tree needs updating with the newest version. Either copy the source from the CD or use CVS to obtain the latest stable base (replace the #_# in the command with the major and minor release numbers, such as 3_4 for OpenBSD v3.4):

```
cd /usr; cvs -q get -rOPENBSD_#_# -P src
```

5.3. Ports upgrades

The programs installed via the ports system are the UW-IMAP server, `procmail`, and the `libtool` library set. After updating the OpenBSD system, update the `/usr/ports` source files by copying the files from the

CD or updating via CVS. Obtain the latest from CVS with (substituting the #_# with the major and minor version of OpenBSD in use):

```
cd /usr/ports; cvs up -r OPENBSD_#_# -Pd
```

After the entire tree is updated, enter the directory for each package listed in the table below and run `make clean && make && make install`.

```
UW-IMAP           /usr/ports/mail/imap-uw
procmail          /usr/ports/mail/procmail
libtool           /usr/ports/devel/libtool
```

This process upgrades all three ports installed packages for the new Operating System.

5.4. Upgrading third-party software from source

The Berkeley DB and `bogofilter` packages installed from raw source need updating for each new OpenBSD release and periodically for feature and security updates. Before re-installing directly from source, check the ports system to see if the package is updated or added. This will eliminate the need for compiling from source outside the ports system. If the software is not available in the correct version or not available at all via ports, follow the same instructions above for downloading and compiling the packages from source used during the installation operation. Before installing, make backup copies of the configuration files and read the README and UPGRADE files included in those packages. Most often a simple `make && make install` after configuring the software is sufficient, with the added task of merging changes to the new `bogofilter` configuration file as needed. For some `bogofilter` releases, the developers recommend rebuilding the databases afresh from saved SPAM and HAM messages. Use the saved SPAM for each user and their saved HAM to provide the most accurate results for each account.

5.5. System Maintenance

Maintaining this system requires minimal effort. The server itself runs very few processes and contains few users with valid shells to monitor. In fact, many small organizations have only the one administrative user for a single System Administrator. Keeping a watchful eye on the server is still very important, even if it takes little time. Many of the checks listed below are fairly manual, but reviewing the system manually is more reliable than using automated tools. After gaining a strong understanding of how the system operates in normal situations, automated tools may be developed

or installed to assist with the process, but the occasional manual review is critical in case the tools are compromised by an attacker.

Review the `root` messages containing the `daily` and `security` script output each day. If anything stands out as anomalous, such as a file system filling up or an unexpected change to a system configuration file, investigate immediately. Hesitation or delays of any kind give an attacker more time to cover their tracks or cause system generated problems to worsen.

After checking the daily `root` messages, skim through the last day's `/var/log/maillog` file, which records all POP, IMAP, and SMTP messages, and look for any sendmail daemon restarts, newaliases commands, or other odd messages which seem out of the ordinary. Become very familiar with all the log formats for each daemon and type of message they produce during normal operation.

Next, skim through the `/var/log/messages` file for any system or hardware errors, kernel accounting changes (these are really bad), or file system messages. Again, gain a close familiarity with the formats of messages in this file. Most systems do not produce many log entries in `/var/log/messages`, so this should be an easy task.

Check the `/var/log/secure` log file for anomalies or unexpected sudo command entries. Misuse of the system often starts here.

Look through the `/var/log/authlog` file for failed login attempts or unusual sequences of logins or unknown source IP addresses. Looking in the log every day will allow a comfortable knowledge of what normal activity on the server looks like, and any odd entries will quickly stand out for investigation.

Check the process list to see what the server is running. The fully functioning server has only 19 processes with a single administrator logged in via SSH and using a `root` shell. If the mail traffic is high, this number will increase, but an idle system should look like this:

```
# ps -auxw
USER       PID %CPU %MEM   VSZ   RSS TT   STAT STARTED       TIME COMMAND
root     24978  0.0  0.0   268   164 p0   R+      9:15PM    0:00.00 ps -auxw
root     29538  0.0  0.1   104   384 ??   Is      6:23PM    0:00.07 syslogd -a
/var/empty/dev/log
root       744  0.0  0.1    64   356 ??   Is      6:23PM    0:00.00 inetd
root     30511  0.0  0.2   356   876 ??   Is      6:23PM    0:00.58 /usr/sbin/sshd
root     20412  0.0  0.1   236   448 ??   Ss      6:23PM    0:00.04 cron
root      8872  0.0  0.2   888   852 ??   Ss      6:23PM    0:00.31 sendmail: accepting
connections (sendmail)
root     29953  0.0  0.1    48   404 C0   Is+     6:23PM    0:00.01 /usr/libexec/getty Pc ttyC0
root      4100  0.0  0.1    44   400 C1   Is+     6:23PM    0:00.01 /usr/libexec/getty Pc ttyC1
root     13082  0.0  0.1    48   404 C2   Is+     6:23PM    0:00.01 /usr/libexec/getty Pc ttyC2
root      7436  0.0  0.1    48   404 C3   Is+     6:23PM    0:00.01 /usr/libexec/getty Pc ttyC3
root     16109  0.0  0.1    44   400 C5   Is+     6:23PM    0:00.01 /usr/libexec/getty Pc ttyC5
root      3225  0.0  0.3   452  1248 ??   Is      6:37PM    0:00.05 sshd: admin [priv] (sshd)
admin    27644  0.0  0.2   408   992 ??   S       6:37PM    0:01.10 sshd: admin@ttyp0 (sshd)
admin    12511  0.0  0.1   388   300 p0   Is      6:37PM    0:00.01 -ksh (ksh)
root      2839  0.0  0.1   348   272 p0   I       6:37PM    0:00.03 -csh (csh)
root     32122  0.0  0.1   396   312 p0   S       6:38PM    0:00.09 ksh -o ksh
root         1  0.0  0.0   340   200 ??   Is      6:23PM    0:00.02 /sbin/init
```

Investigate any suspicious processes with `fstat`, which is the OpenBSD equivalent to the popular `lsof` tool.

6. Pre-production configuration review

The final stage before adding the mail user accounts and turning the system over to production is a security and functionality review. The questions to consider for this system are:

- Does the SMTP server allow only relaying from the local network?
- Are only the core network services absolutely required to run the mail services and encrypted remote administration answering on the network?
- Are only authorized administration accounts allowed shell access to the system?
- Is root access only available locally and, except for the console, auditable?
- Is /usr the only file system mounted `NOSUID` except the / file system?

6.1. SMTP relay check

The sendmail server answering on the SMTP port should not relay for anyone outside the local network. To test this properly, connect to port 25 on the `gcux` server from a remote network and manually type some mail commands. Also, verify the banner displays no information about the server whatsoever.

First, send mail to a user on the gcux server to confirm proper incoming mail services function correctly.

```
$ telnet gcux 25
Trying 24.196.136.29...
Connected to gcux.
Escape character is '^]'.
220  ESMTP
EHLO remote.server.tld
250-domain.com Hello user@remote.server.tld [##.##.##.##], pleased
to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-DELIVERBY
250 HELP
MAIL From:<user@remote.server.tld>
250 2.1.0 <user@remote.server.tld>... Sender ok
RCPT To:<admin@domain.com>
250 2.1.5 <admin@domain.com>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Testing.
.
250 2.0.0 h652ZlWh001939 Message accepted for delivery
```

This shows local delivery to the `gcux` users' functions correctly.

Now test remotely to send from `gcux` to remote address.

```
$ telnet gcux 25
Trying 24.196.136.29...
Connected to gcux.
Escape character is '^]'.
220  ESMTP
EHLO remote.server.tld
250-domain.com Hello user@remote.server.tld [##.##.##.##], pleased
to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-DELIVERBY
250 HELP
MAIL From:<admin@domain.com>
250 2.1.0 <admin@domain.com>... Sender ok
RCPT To:<user@remote.server.tld>
550 5.7.1 <user@remote.server.tld>... Relaying denied
```

This proves sendmail disallows remote relaying from outside the local network to an external address.

6.2. Network services check

The only network connections allowed on the system are SSH, POP, IMAP, and SMTP. All over network services should not answer, as they are disabled. Test this with the NMAP package from a remote system. Run a standard `nmap` scan to discover the ports the server answers on:

```
b$ nmap gcux.unixstuff.info

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on gcux.domain.com (192.168.0.100):
(The 1596 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
110/tcp    open        pop-3
143/tcp    open        imap2
587/tcp    open        submission

Nmap run completed -- 1 IP address (1 host up) scanned in 16
seconds
```

The submission service answering on port 587 is part of the sendmail SMTP service and is normal to have answer on the network interface. The other services show only the list we want answering, so the network scan shows a secure system.

6.3. Authorized shell accounts

Check the accounts on the system for any accounts with unauthorized shell access. After adding at least one mail account via the `newuser` script, verify that only the administrative users have a valid interactive shell, and only those accounts are in the `sudoers` file and group `wheel`.

Review the /etc/passwd file for shells:

```
# cat /etc/passwd
root:*:0:0:Charlie &:/root:/bin/csh
daemon:*:1:1:The devil himself:/root:/sbin/nologin
operator:*:2:5:System &:/operator:/sbin/nologin
bin:*:3:7:Binaries Commands and Source,,,:/:/sbin/nologin
smmsp:*:25:25:Sendmail Message Submission Program:/nonexistent:/sbin/nologin
popa3d:*:26:26:POP3 server:/var/empty:/sbin/nologin
sshd:*:27:27:sshd privsep:/var/empty:/sbin/nologin
_portmap:*:28:28:portmap:/var/empty:/sbin/nologin
_identd:*:29:29:identd:/var/empty:/sbin/nologin
_rstatd:*:30:30:rpc.rstatd:/var/empty:/sbin/nologin
_rusersd:*:32:32:rpc.rusersd:/var/empty:/sbin/nologin
_fingerd:*:33:33:fingerd:/var/empty:/sbin/nologin
_x11:*:35:35:X server:/var/empty:/sbin/nologin
_spamd:*:62:62:Spam daemon:/var/empty:/sbin/nologin
uucp:*:66:1:UNIX-to-UNIX Copy:/var/spool/uucppublic:/sbin/nologin
www:*:67:67:HTTP server:/var/www:/sbin/nologin
named:*:70:70:BIND Name Service Daemon:/var/named:/sbin/nologin
proxy:*:71:71:Proxy Services:/nonexistent:/sbin/nologin
nobody:*:32767:32767:Unprivileged user:/nonexistent:/sbin/nologin
admin:*:1000:20:SysAdmin User:/home/admin:/bin/ksh
jdoe:*:1001:10:John Doe:/home/jdoe:/usr/bin/false
```

Notice that only `root` and `admin` have valid shells.

Check the `/etc/sudoers` file:

```
# cat /etc/sudoers
#       $OpenBSD: sudoers,v 1.4 2002/07/16 18:19:18 naddy Exp $
#
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

# Defaults specification

# User privilege specification
root    ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL

# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
```

Notice the only two uncommented lines are for `root` and the group `wheel`. Now check that only the `admin` account is in `wheel`.

```
# grep wheel /etc/group
wheel:*:0:root,admin
```

The user configuration and administrative access rights are secure.

6.4. Root accessibility and audit ability

Access to the `root` account must be restricted not only by which local accounts can access it, but access must be restricted from the network, allowing only a console login directly as `root`. Also, whenever another account access `root` privileges, there should be an audit trail in a log to see who accessed `root` at what time. On OpenBSD both access to `sudo` and `su` commands appear in the system logs.

Log into the administrative account on the `gcux` server.

First, enter a command via `sudo`, such as `sudo ls`. Now check the `/var/log/secure` log file for an entry for the command just run:

```
Jul  4 22:12:20 gcux sudo:    admin : TTY=ttyp0 ; PWD=/home/admin ; USER=root ; COMMAND=/bin/ls
```

This entry shows `sudo` logs `root` access correctly.

Next, access `root` directly via `su`, then check the `/var/log/authlog` for an entry showing the `su` command:

```
Jul  4 22:15:20 gcux su: admin to root on /dev/ttyp0
```

This shows an audit trail exists for all `root` access on the system.

6.5. Filesystem NOSUID & NODEV check

The only file systems not mounted NOSUID are `/usr/local`, `/usr`, and /. The only reason this system has `/usr/local` not mounted NOSUID is because he source tree exists here, and it is cumbersome to build with this option set. The only file system not mounted NODEV is `/`. To quickly check this, run the `mount` command and examine the output to verify the `mount` settings:

```
# mount
/dev/wd0a on / type ffs (local)
/dev/wd0h on /home type ffs (local, nodev, nosuid)
/dev/wd0d on /tmp type ffs (local, nodev, nosuid)
/dev/wd0f on /usr type ffs (local, nodev)
/dev/wd0g on /usr/local type ffs (local, nodev)
/dev/wd0e on /var type ffs (local, nodev, nosuid)
/dev/wd0j on /var/mail type ffs (local, nodev, nosuid)
/dev/wd0i on /var/www type ffs (local, nodev, nosuid)
```

This confirms out settings as correct. Next, verify that the `/etc/fstab` file mounts these file systems correctly on boot:

```
# cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/wd0h /home ffs rw,nodev,nosuid 1 2
/dev/wd0d /tmp ffs rw,nodev,nosuid 1 2
/dev/wd0f /usr ffs rw,nodev 1 2
/dev/wd0g /usr/local ffs rw,nodev 1 2
/dev/wd0e /var ffs rw,nodev,nosuid 1 2
/dev/wd0j /var/mail ffs rw,nodev,nosuid 1 2
/dev/wd0i /var/www ffs rw,nodev,nosuid 1 2
```

This file shows the `mount` options are correct on system startup.

This concludes the system configuration review. The entire system is hardened against external intrusion, and this server can function well even with a live Internet IP address. To further secure the system, review the OpenBSD documentation and mailing list archives for the `pf` firewall software configuration.

Add the rest of the mail users, and place the system into production.

7. Reference:

"4 – OpenBSD 3.3 Installation Guide." OpenBSD FAQ. v 1.142 2003/06/16 23:48:04. URL: http://www.openbsd.org/faq/faq4.html (June 27, 2003)

"10 – System Management." OpenBSD FAQ. v 1.425 2003/06/17 09:07:49. URL:http://www.openbsd.org/faq/faq10.html#Patches (July 4, 2003)

"Bogofilter Home Page." URL: http://bogofilter.sourceforge.net (June 28, 2003)

"OpenBSD AnonCVS." 1.187 2003/05/21 11:49:05. URL: http://www.openbsd.org/anoncvs.html#CVSROOT (July 2, 2003)

"OpenBSD 3.3 Errata." v 1.442 2003/05/08 08:38:20. URL: http://www.openbsd.org/errata.html (July 2, 2003)

"OpenBSD FTP." URL:http://www.openbsd.org/ftp.html (June 27, 2003)

"OpenBSD Mailing lists." v 1.63 2003/03/02 21:12:38. URL: http://www.openbsd.org/mail.html (July 3, 2003)

Robbins, Arnold. sed & awk: Pocket Reference. Sebastopol: O'Reilly & Associates, 2000. 43 – 45.

"Sleepycat Software: Download." URL: http://www.sleepycat.com/download/ (July 2, 2003)

"Spam." URL: http://www.dornbos.com/spam01.shtml (July 1, 2003)

"Spam Archives." Paul Graham. URL:http://www.paulgraham.com/spamarchives.html (July 1, 2003)