Global Information Assurance Certification Paper

RECYCLING SUN HARDWARE FOR USE AS MAIL AND DNS SERVERS

Prepared for:

Global Information Assurance Certification
GCUX Practical Assignment: Securing Unix Step-by-Step v1.8

Prepared by:

Sean J. Schluntz

Table of Contents

1 Introduction

Due to the fast innovations in the computer industry, many companies are now finding their IT storage shelves filling with what some view as obsolete and useless equipment. You can now find servers and workstations that ten years ago were bleeding edge technology at electronic flea markets and online auction sites for less than a few hundred dollars.

These systems, when loaded with the standard install of the current supported commercial operating system (be it Windows XP, Solaris 8, AIX, MacOS), can seem too slow to be useful.

What this document demonstrates is that this hardware can still provide useful services for companies. Older equipment, when loaded with trimmed down freely available operating systems (like the BSD UNIX variants: FreeBSD, NetBSD and OpenBSD, as well as the UNIX-like operating system Linux) can easily handle the burdens of day-to-day business needs, at a fraction of the cost of "modern" equipment.

1.1 Overview

This document is designed to walk beginning to intermediate computer users through the setup, installation and hardening of a sealed mail server using the NetBSD operating system on end of life Sun hardware. A sealed mail server differs from a standard mail server in that users must interact with the server remotely; they do not have local login accounts on the system.

Each part of the system is explained (from hardware setup through the major software packages used) before the install process is begun. It is recommended that you read the entire document before beginning the build of your own system.

Though this document is specifically about Sun hardware, the installation instructions here can be used to build a mail server on the variety of different platforms that are supported by NetBSD. Besides the initial boot commands used for each platform, the install and configuration changes little between different platforms.

1.2 Assumptions

This paper was written with the following assumptions:

1. There is a publicly routable static IP address available for use by the server. (For other mail servers to deliver mail to your server it must have a static IP). For the examples in this paper the server will have the IP address of 10.0.0.2.
2. The server will, when told to by the document, be connected to either an unprotected (i.e. not firewalled) network segment or on a DMZ. The configuration

listed in this document will work equally as well behind the protection of a firewall as it does on the open Internet. This document errs on the side of caution so that you will have a server that is as secure as possible.

3. While connected to the network the computer will have access to a gateway to the Internet (listed as 10.0.0.1 in this document) and a name server (listed as 10.0.0.3).
4. You have a known network you want to accept mail from, in this case we will be using 10.0.0.0/24.
5. The person reading this paper has a basic understanding of UNIX (ie, how to change directories, how to find out what directory you are in, how to copy, move, rename and delete files).
6. The person reading this paper has a basic understanding of the vi or ed editor[1].
7. The person reading this paper has access to an IMAP enabled mail client.

1.3 The Build Process Outlined in This Paper

It is the belief of the author that all system builds, whether it be for single tasked servers or general-purpose client systems, can be started from the same point. There is a single minimal level of configuration that can be used no matter what you plan to do with a system. With that in mind, the build process outlined in this paper is broken down into steps.

First, you build the basic system and you harden it (build steps one and two). You now have a system, which can be configured for any use. The following steps detail how to complete the configuration for the systems task; in the case of this paper, it is to build a mail server. If you wanted to build a web server, a file server, or a LDAP server, the first two-steps would always be the same.

1.4 Conventions Used in this Paper

- Internet addresses (URLs) and system commands are specified in a mono-space font, for example: `www.netbsd.org` or `dmesg`.
- Any command that requires a space will be included in single quotes, or on a line by itself in a mono-spaced font, for example: `` `ls –la` ``

[1] Understanding the vi or ed editors is important for all system administrators. Many times if a system crashes or is newly installed they may be the only editors available to you. If you are unfamiliar with the vi or ed editors there are two good O'Reilly books on the subject:
"Learning the vi Editor" 2nd Edition by Linda Lamb and Arnold Robbins
"vi Editor Pocket Reference" by Arnold Robbins

- If a URL is longer than a single line it will be broken at a special character (like dash or hash). This means that both lines should be used as a single line without a newline or any spaces. As an example, the following URL:

  ```
  http://www.netbsd.org/Ports/sparc/faq.html
  #boot-floppy
  ```

  When used it should be:

  ```
  http://www.netbsd.org/Ports/sparc/faq.html#boot-floppy
  ```

- Recommended user selections from menu options are underlined and capitalized as shown in the selection list. For example: "Choose Continue to activate the changes and return to the menu."
- Text from the terminal output is shown in a small mono-space font which is indented to separate it from the body of the document. For example:

  ```
  SPARCstation 5, No Keyboard
  ROM Rev. 2.15, 256 MB memory installed, Serial #7800338.
  Ethernet address 8:0:20:77:6:12, Host ID: 80770612.

  No boot device found.
  ok
  ```

- User interaction with terminal output is shown as ***bold italic*** text. For example:

  ```
  No boot device found.
  ok boot cdrom
  ```

- Code listings are mono-spaced, non-indented with a medium sized type face

  ```
  #!/bin/sh
  echo "Hello World!"
  ```

- Special key actions will be contained in brackets. For example:

  [ENTER] means to hit the enter key on your key pad, and
  [CTRL]-a, p means to hit the 'a' key while holding down the control key and then release both and press the p key.

2 Risk Assessment

Before beginning any system build, it is important that the system administrator be aware of the security issues that might be associated with the system. In this paper the author does not differentiate between internal (valid user) and external (invalid user) threats when reviewing security policy. It is the authors opinion that the same strict standards should be held to all.

This paper does not attempt to design the most secure server possible; this paper instead attempts to find balance between security and proper Internet courtesy. The firewall portion of the server could have more ports closed and still function, but it would not be polite (in the author's opinion) to lock the system down to the point where it makes more work for other sysadmins. As an example, in the case of allowing external pings to be responded to from the server, it is polite to allow an external sysadmin to ping your mail server so they can check connectivity in the event they can not make a mail connection to the server, but it is not required.

2.1 Physical Connectivity and Security

No system which will have inbound connectivity to external networks should be connected to a company's internal network or be able to open connections in to the company's internal network. This is to protect the internal network from direct attack in the event the server is compromised.

The server should be outside of the company's firewall, or if possible, on a separate network (known as a DMZ), which is not connected to the internal network but is still partially protected from the external networks.

The physical security of a system is up to the policy of the company it is to be used for, the abilities of that company and the importance of the services run on the system. The author recommends that the system be secured in the best manner feasible, but the specifics will not be covered in this document.

2.2 General Security Risks for Mail Servers

Mail servers are inherently open to risks. If the server is to be used for more than just local communications, it needs to be attached to the Internet. You must allow certain levels of user interaction with systems you do not control to allow mail to be delivered or picked up.

Malicious users can attempt to gain local access to a mail server through the network daemons running on the server or just cause the server to become overloaded and stop responding to requests (known as a denial of service or DOS).

Many mail servers also run local domain name servers. Name servers are a popular target of gaining unauthorized local access to servers. Clever users can even to force a name server to give the mail server bad information causing mail to be redirected to a site other than the one intended.

Many mail servers also allow local user login. This opens them up to a whole new class of attacks which are much harder to defend against because the users have shell access to the system and are past the first hurdle of taking over the system (having access).

If the mail server is not secured behind a firewall (and some times even if it is) the system can be fingerprinted. This is the process of discovering what services, and the versions of those services, are running on the system. This can give the attacker an inroad to finding attacks that will work against your system instead of having to try a list of attacks blindly.

2.3 Specific Security Risks Pertaining to the Mail Server Being Built

Intrusion by Login Account

The server described in this paper is a sealed mail server. This means that all but administration interaction is remote, be it the delivering, managing or retrieving of mail.

Because only administrators have local shell access on the server the number of accounts available for a malicious user to attempt to break in to is greatly reduced. As long as the admins use good passwords (or even better, public/private key pairs with pass phrases) the risk one of those accounts will be accessed is significantly lower than for a standard mail server.

This does not protect you from all attacks, if one of the admins is using a system that has a key stroke sniffer on it their password can still be captured. Also, if one of the admins is not careful with the storage of their public/private keypair that can also be a method of access for the unscrupulous.

The only solution to this is proactive administrators. The administrators have to be careful what passwords they use, and how they use them. The must also be careful how they store their key pairs and immediately have them revoked if they are lost.

Buffer Overflows and Bugs

A number of services are listening for connections on the mail server. Both BIND and sendmail have had numerous exploits, and though there are teams of people working on each product it is accepted that more exploits will be found for them in the future.

Any service listening for a connection is a possible avenue for attack, people are constantly reading the code and testing products for possible vulnerabilities and then posting the results.

The best solution to this problem is staying on top of the bug lists and the patches and updates available for the software that is run. When a new vulnerability is found the

developers of the (open source) application are often quick to supply fixes to problems when reported.

Denial of Service (DOS)

With the updates to the NetBSD network stack that are now in the system a DOS style of attack is more difficult than it had been previously. It is still possible to overload the sendmail, Cyrus and SSH services individually to the point where they can no longer accept new connections (or at least delay new connections to the point where the client reaches its timeout period and stops trying to contact them).

Possible solutions to individual service DOS attacks can be in the form up software updates. Another solution is the upgrading of the hardware of the system, or even the clustering of separate systems, to allow for a larger amount of traffic. In many cases DOS attacks can't be protected against because of the manner in which they are done.

System Fingerprinting and Network Mapping

The mail server is susceptible to system fingerprinting. With the allowance of ping response and the banners supplied by Cyrus and sendmail on their ports the SSH server a remote user can ascertain the OS version as well as a partial list of the services the system is running.

Because of the packet filtering a full discovery is blocked. In addition, because of the kernel settings made and the packet filter the system will not aid in the discovery of other systems on the network.

Possible solutions include removing the banners from the individual services as well as blocking all un-requested icmp traffic to the server.

Mail Relaying

The mail server is configured to only accept mail that matches one of two criteria; it must either be addressed to a local user of the system or the mail must be sent from a limited list of acceptable network addresses.

3 Hardware

This section covers the reasons for choosing Sun hardware, what kind of hardware you will need to build the systems described, and the hardware that was used to test and demo this paper.

3.1 Why Sun Hardware

Putting aside the great range of x86 based hardware, the next most available hardware is from Sun Microsystems. Sun Sparc 4, 5, 10 and 20 systems are in abundance in office back rooms, electronics flee markets and online auction sites and all make very capable servers. Slower Sun Sparc 1 and 2 systems along with the lunchbox Classic, IPC and LX systems are all great for individuals wanting to learn their way around UNIX.

Unlike most x86 systems, the Sun hardware listed above is very complete from a components standpoint. Sound, networking and SCSI controllers are all on the main board and are provided by Sun. You don't have to worry about what is compatible; Sun has strict requirements for components so you have a higher assurance they work together.

This does not mean that everything you might find on a Sun system is supported by the operating system out of box. Some add-on cards provided by third parties need special drivers (which they supply for SunOS/Solaris but usually not for other operating systems) and some video cards are not supported so you have to use a serial console. But all in all the Sun Sparc platform is an easy to use and reliable solution for most needs.

Sun has made many things easier for the UNIX enthusiast. There is a powerful hardware layer (known as the PROM) you can interact with, allowing you to test basic system functionality without loading an OS. The PROM also makes it easy to specify where you want to boot the system from, be it the local hard disk, CD or floppy or a remote network server.

3.2 Recommended Minimum Equipment

The minimum requirements for the mail server depends on the load you are expecting it to manage. The author has found that a Sparc LX system (50mhz MicroSparc CPU with 64megs of RAM) can handle the needs of 50 medium use mail clients (remote users of POP and SMTP who check their mail about every 30 minutes and do not leave their mail on the server with a total mail server volume of under 50mb a day).

With the cost of hardware as low as it has become, the recommended minimum would be 80mhz Sparc 4 or 5 system with at least 128mb of RAM with two 4gb SCSI hard disks as well as a CDROM drive.

Since mail servers are more concerned with the ability to open sockets, receive data and store it to the disk than any heavy computational requirement, the author recommends going for systems with higher RAM and CPU speeds instead of trying to get server class equipment. A 110mhz Sparc 5 with 256mb of RAM will out perform a 50mhz Sparc 20 with 256mb of RAM and cost you less as well. Some still view the Sparc 20 as a better system for all needs. At this time, none of the open source operating systems support multiple CPU's on Sun hardware so the ability for a Sparc 10 or 20 to have multiple CPU's is not valuable.

It is also recommended that you have access to a CD burner. Though there are other methods of doing the install, this document outlines an install from CD. If you do not have a CD burner, but the system does have a CD drive, then it is recommended that you purchase the NetBSD distribution[2] and then ftp the remaining files over to the system using a crossover cable between the system you are setting up and the system you will be using to test.

3.3 Description of Equipment Used in Examples

The hardware used for the server is a Sun Sparc 5 system. It has a 110mhz MicroSparc II CPU; built in SCSI controller, network ports (10BaseT & AUI), serial ports and a keyboard port. All extraneous hardware has been pulled (video card, expansion ports etc). The system holds 256mb of RAM and two 4gb SCA SCSI hard disks and an internal 4x CD.

The client used to configure the system (using minicom through the serial port) as well as run the external tests against the systems is an IBM ThinkPad T21 running RedHat Linux v7.1. CD's were burned using the IBM CDR drive for the ThinkPad T series laptops.

The other miscellaneous equipment used:

- NetGear 4 port 10BaseT hub (or Cat5 crossover cable)
- Home made null modem cable
- Cat 5 cable (x3)

[2] NetBSD can be purchased from the BSD Mall (http://www.bsdmall.com/opsys.html) or from Wasabi Systems (http://www.wasabisystems.com/products/products.html)

4 Software

The software listed in this section is the major component of the systems being built. Some come with the base operating system and some must be installed separately. All of these applications have been chosen because of direct experience the author has with them and because of peer acceptance of their abilities.

4.1 NetBSD - "Of course it runs NetBSD"[3]

After hardware the next most important piece of any server is the operating system it is running. It should be reliable and efficient as well as support the applications you choose to run on it.

When considering the Sun Sparc hardware, you have three basic choices for an operating system: Solaris, *BSD and Linux[4].

1) The Solaris Operating Environment (http://www.sun.com/solaris/) is the Sun recommended and supported operating system for Sun hardware. It is System V based and designed to run on the current line of Sun hardware. Modern Solaris installs tend to be large, even when the minimal install option is chosen. Sun does not include a working compiler with Solaris so this must be downloaded and installed separately (along with some support files) and can cause problems with some software builds.

2) Linux (sometimes referred to as GNU/Linux) is a UNIX like operating system modeled after the System V version of UNIX. There are many different distributions of Linux available (all based around the Linux Kernel). Many Linux distributions tend to be large and hard to pair down (with the notable exceptions of Slackware http://www.slackware.org and Debian http://www.debian.org) though they do include a development environment.

3) NetBSD (http://www.netbsd.org) is developed and maintained for more hardware platforms and CPU types than any other operating system available today. There are currently 51 ports of NetBSD supporting 17 different CPU types (see http://www.netbsd.org/Ports/), which run the gambit of computer systems from obsolete Sun2 and Atari systems to the new Power Macintosh and Sun Ultra servers.

NetBSD is a branch of the original 4.4BSD and 386BSD code released by the University of California, Berkeley. Its goal from the beginning has been to focus on multi-platform

[3] NetBSD motto, from http://www.netbsd.org

[4] There are actually many different distributions of Linux that are available. A partial list can be found at http://www.li.org/getlinux.php. For the purposes of this paper they can be considered a single base branch (being the Linux Kernel http://www.kernel.org) with different distributed software options.

support. A brief timeline of NetBSD and its cousins FreeBSD, OpenBSD and BSD/OS can be found at http://www.netbsd.org/Misc/history.html.

The author has long appreciated NetBSD's slimness and efficiency, especially when compared with recent offerings in the Linux world which make it difficult to strip out all of the unneeded applications during and after a system install.

4.2 Sendmail

Sendmail (http://www.sendmail.org) is maintained by Sendmail, Inc ., (http://www.sendmail.com) the corporate group created to market the commercial version of sendmail.

Use of sendmail has been dropping steadily[5] on the Internet as other, easier to use and configure SMTP servers have been released. You should be aware of its history with security problems[6] and keep an eye on patches and vulnerabilities that may pop up literally over night.

We will use sendmail for two main reasons. The first is that it integrates with the Cyrus IMAP server very well providing us with our closed mail server (no user accounts) and second, if properly maintained, it is the most powerful of all the SMTP options.

4.3 Cyrus IMAP

The Cyrus IMAP server (http://asg.web.cmu.edu/cyrus/imapd/) is a powerful IMAP, POP and KPOP server designed run on sealed servers. A sealed server is one where the mail users do not have local accounts on the system, but instead rely on POP or IMAP clients to work with their mail remotely.

The author believes that it is better to have your services separated. This helps keep your services spread out to give you better reliability as well as making it more difficult for a malicious individual or group to gain access to all of your systems. If your users need a shell server to access their mail it can be built as a separate system with a network connection to the proper ports for mail retrieval.

The following statement is from the Cyrus web site: "The private mailbox database design gives the server large advantages in efficiency, scalability, and administratabiliy. Multiple concurrent read/write connections to the same mailbox are permitted. The server supports access control lists on mailboxes and storage quotas on mailbox hierarchies."

[5] See the Internet Surveys available at http://cr.yp.to/surveys.html for more information on the SMTP servers used on the Internet.
[6] Visit the Carnegie Mellon CERT Coordination Center at http://www.cert.org and search for sendmail to see a list of security problems with current and past distributions.

4.4 BIND

Most mail servers need to perform large numbers of DNS look-ups to find where mail should be delivered. To speed the process many systems administrators choose to run a local DNS server on the same system as the mail server. It does not have to be authorative for any domains, but having something keep a local cache of previous DNS entries can help reduce network usage and speed up mail delivery.

Maintained by the Internet Software Consortium (ISC), BIND (http://www.isc.org/products/BIND/) is the original domain name server. Freely available and actively maintained it is the backbone of the domain naming system of the Internet. Other companies, groups and individuals have released competing products, but none have gained the general acceptance of BIND. BIND has a long history of security problems[7] and you should keep a diligent eye on the security lists to make sure you are up to date with all available patches.

4.5 OpenSSH and OpenSSL

OpenSSH (http://www.openssh.com) is a free and open source implementation of the SSH protocol[8] for secure communications. It is a replacement for the telnet, ftp and UNIX r-commands (rcp, rlogin, rsh etc), which adds improved user authentication, session encryption and secure tunneling.

The SSH protocol is becoming widely accepted across the Internet and within companies as a secure method of communication. This has been helped by the expiration of some RSA patents[9] which made free SSH implementations, like OpenSSH, legal for use in the United States as well as in several other countries.

OpenSSH relies on the libraries provided by OpenSSL (http://www.openssl.org), the free and open source Secure Sockets Layer (SSL) implementation.

4.6 Integrit

An important security measure of any system (server or client) is the ability to verify that the operating system and its applications have not been tampered with. Historically, the tool Tripwire[10] is used to generate a database of information on the file system which can then be used to verify that no unexpected changes have occurred.

[7] The Carnegie Mellon CERT Coordination Center is good for watching the security issues of many applications, see footnote above.
[8] See: http://www.snailbook.com/protocols.html
[9] See: http://www.cnn.com/2000/TECH/computing/09/07/security.patent.release.idg/
[10] See: http://www.tripwire.com

There are three types of Tripwire currently available: a commercial product which is maintained by the Tripwire corporation, an open source version for Linux[11], and the Academic Source Release (ASR)[12].

In at attempt to keep the cost of these servers down we will not be using the commercial version of Tripwire. The open source version of Tripwire is currently only available for Linux. If this is the only platform you are using on your network it is a good application, but the author prefers to use a common tool across all of the server types he maintains. The third option is the Tripwire ASR. Many people use the ASR release of Tripwire, but a review of its license[13] shows that it is only legal for a person to install Tripwire ASR on a single system on their network. This license is much too restrictive for any corporate environment.

In searching for a replacement for Tripwire, the author came across Integrit (http://integrit.sourceforge.net). Integrit is a small and easy to understand file integrity checker with a number of useful options, including the ability to output its reports in XML or human readable format. Integrit has been favorably reviewed as a replacement for Tripwire in a SysAdmin web exclusive article[14] and works well on all UNIX platforms. Integrit has all the appearances of being a good replacement for Tripwire.

4.7 Other System Software

The following software packages are useful to have on the system. Some are required by other software and some are just useful as sysadmin tools.

4.7.1 NetBSD Package System

Keeping your system software up to date can be a full time job if you have a number of servers all running different applications. You will see notices on security lists when there is a problem with a package, but you may not know when there is a regular update (which might head off security problems in the future). The NetBSD packages system[15] is a powerful resource for keeping your system up to date. It has the ability to keep itself synchronized with the NetBSD distribution site and notify you of any upgrades that are available to you. It can also be used to streamline the installation of software packages you might need.

[11] See: http://www.tripwire.com/products/linux/
[12] See: http://www.tripwire.com/products/tripwire_ASR/
[13] Available in the distribution of Tripwire ASR but also condensed in questions 1 and 2 of their FAQ available at http://www.tripwire.com/downloads/tripwire_asr/asr_eula_faq.cfm
[14] See: http://www.samag.com/documents/s=1147/sam0108l/0108l.htm
[15] For more information on the NetBSD packages system see http://www.netbsd.org/Documentation/softwrae/packages.html

4.7.2 Perl

Perl is a very powerful programming language used for many tasks. It can be used as a scripting language (platform independent), converted into byte code much like java (again, platform independent) or compiled into an architecture dependent stand-alone executable[16]. The two main sites for Perl information are The Comprehensive Perl Archive Network (also known as CPAN http://www.cpan.org) and the O'Reilly Perl site (http://www.perl.com).

Perl is not directly used in this paper. All scripting is done in Bourne/Korn Shell, but it is a required tool for building and configuring some of the other software used.

4.8 Testing Software

A sysadmin shouldn't just trust that the systems built work the way they were planned; they must be constantly tested. The tools listed in this section are a start. They will help you make sure that your system is working in a manner that you expect.

It is recommended that you only use what is listed in this paper as a starting point for verifying your systems. There are many other tools available, with new ones being developed ever day, and if you build or manage systems, you should keep up to date with the tools available.

4.8.1 Nmap

Nmap is a useful tool for mapping networks, finding which ports are open on hosts and probing out what type of operating system the host is running. It is available from Insecurity.Org (http://www.insecure.org/nmap/) and is free of charge.

4.8.2 Nessus

Nessus is a powerful security-scanning tool available from the Nessus Project Group (http://www.nessus.org). Whereas nmap will only report which ports are currently listening on a given host, Nessus will try to determine what service is running on the open ports. Testing for the true service is a very good thing for a security tool. You don't know when someone may open a telnet server on port 22 (the ftp port) to provide a back door into the system. Nessus also has the power to try to attack known vulnerabilities in tools to see if the system you are testing is susceptible to that form of attack.

[16] See: http://www.oreilly.com/catalog/pperl3/chapter/ch18.html
http://www.the-labs.com/Perl/#perl_compiler or the man page for perlcc at:
http://www.justanotherperlhacker.org/manual/utils/perlcc.html

5 Preparation

Preparation is the key to build systems in a safe manner. If you take the time to prepare everything you need before you start then you will have a higher chance of success.

5.1 Starting With a Safe Environment

The safest environment for any system is one where it is not connected to a terminal or a network. Unfortunately this makes the system unusable. In the case of this document it is strongly recommended that the system not be attached to any network, even an internal network you may feel is secure, until the document specifically requests that you do.

The reason for this is that there are a number of points during a system install where the system is very vulnerable to the network, so the longer it is kept from the network the safer your install will be.

5.2 Choosing Your Install Method

There are a number of different methods available for the installation of NetBSD and the other software we will be using, from standard CD and floppy based installs to network based installs.

This document covers installing with a CD. If you don't have access to a CD drive you can use the NetBSD documentation[17] for instructions on how to install the operating system and refer back to this document for the recommended settings.

5.3 The Software Packages You Will Need

This is a list of the software packages you will need before the install starts, along with where to download them and which version this document uses. The author recommends that you always use the newest version of the software available, even if it is not the version listed in this paper.

The software not listed here will be installed using the NetBSD packages system. For many applications the packages system is the best solution to keeping your system up to date. Unfortunately some of the packages we need are not in the packages system yet and some others you need to do the update by hand to be sure they are done as soon as patches are available.

- NetBSD v1.5.2 ISO Image (72MB)
  ISO Image: ftp://ftp.netbsd.org/pub/NetBSD/iso/1.5.2/sparccd.iso

[17] The install documents for NetBSD v1.5.2 can be found at ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-1.5.2/sparc/install.html

- NetBSD v1.5.2 Kernel Source (18MB)
  ftp://ftp.netbsd.org/pub/NetBSD/NetBSD-1.5.2/source/sets/syssrc.tgz
- OpenSSH v3.1p1 Source (8MB)
  ftp://ftp3.usa.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-3.1p1.tar.gz
- Integrit 3.00.05-beta Source (210KB)
  http://prdownloads.sourceforge.net/integrit/integrit-3.00.05-beta.tar.gz

6 The Build: Step 1 – Generic Build

The first step of the build process gets the system configured for general use. When you are done with this section the system will be ready to be customized for any task needed.

6.1 Installing the Base Operating System

After assembling all of the pieces and attaching your terminal (be it another computer or a dumb terminal) to serial port 0 (or A) of the Sun system, you can power it on. It will first identify itself and then do a test of system RAM.

The next thing the system does depends on whether it has a bootable hard disk. If it does, the system will begin to boot from that disk. You will need to send a break signal to the system to return it to the PROM prompt.

Most dumb terminals have a [BREAK] key, if you are using Minicom you type [CTRL]-a, f to send a break. With HyperTerminal, it's [CTRL]-[BREAK] (with the Windows 95 version of HyperTerminal it's [CTRL]-[F6]-[BREAK]). If you are using a different terminal program you should refer to its documentation for how to send a break command.

The startup screen of a Sun Sparc after breaking out of a hard disk boot should look something like this:

```
SPARCstation 5, No Keyboard
ROM Rev. 2.15, 256 MB memory installed, Serial #7800338.
Ethernet address 8:0:20:77:6:12, Host ID: 80770612.


Boot device: /iommu/sbus/espdma@5,8400000/esp@5,8800000/sd@3,0  File and args:
>> NetBSD/sparc Secondary Boot, Revision 1.9
>> (toor@proxima, Tue Aug 21 23:18:13 CST 2001)
Booting netbsd
1100596+79400[BREAK]
Type  'go' to resume
Type  help  for more information
ok
```

You are now ready to begin the install of NetBSD. Place the NetBSD CD you created (or purchased) in the tray of the Sun system, then type `boot cdrom` at the ok prompt and hit enter. This will boot the system from the CD drive.

The NetBSD kernel will start and probe your system, then the microroot setup utility will begin, welcome you to the setup and then offer you three choices for install: 1) cdrom, 2) tape or 3) floppy. We want to choose 1 for cdrom.

You will then be prompted for the CD device, hitting [ENTER] to choose the default which is usually the correct option.

During a CD install, you might occasionally see an error message like the following:

```
cd0(esp0:6:0): soft error (corrected), info = 3100 (decimal), data = 00 00 00 00
```

You can safely ignore this message. On rare occasions, you will see this message many (10+) times in quick succession and then the install will fail. This usually means that there was an error on the CD media, and that you should burn a new copy and start over.

You will next be asked for your terminal type. In most cases, vt100 is the correct answer, most terminal programs support vt100 commands. Above this line you will see what the control characters are: [CTRL]-h is erase/backspace, [CTRL]-w is word erase, [CTRL]-u is kill and [CTRL]-c is interrupt.

"erase/backspace" is the most important to know. Many terminal programs send [CTRL]-? when you hit backspace, so if you need to change a setting you will need to manually enter the control character instead of hitting the backspace key.

The next question will be whether you want to do an (I)nstall, (U)pgrade, (H)alt, (S)hell or (X) for the experimental sysinst tool. Choose X for the new tool; it makes installs much easer.

The new sysinst tool can be used either with your arrow keys to select entries or by typing the letter that corresponds to the entry you want to choose.

You will be welcomed to the new installer and offered a number of choices; the screen will look like this:

```
Welcome to sysinst, the NetBSD-1.5.2 system installation tool.  This
menu-driven tool is designed to help you install NetBSD to a hard disk, or
upgrade an existing NetBSD system, with a minimum of work.  In the following
menus, you may change the current selection by either typing the reference
letter (a, b, c, ...).  Arrow keys may also work.  You activate the current
selection from the menu by typing the enter key.

If you booted from a floppy, you may now remove the disk.

Thank you for using NetBSD!

             +*****************************************+
             * NetBSD-1.5.2 Install System             *
             *                                         *
             *>a: Install NetBSD to hard disk          *
             * b: Upgrade NetBSD on a hard disk        *
             * c: Re-install sets or install additional sets *
             * d: Reboot the computer                  *
             * e: Utility menu                         *
             * x: Exit Install System                  *
             +*****************************************+
```

Choose a to continue with a standard install. You will then be warned that if there is any data on the system it will be lost during the install. You can choose to stop the install at this point to make sure you have backed the system up. If you are ready to continue, choose Yes to continue the install.

Next, a list of recognized disks is displayed; you are prompted to choose one to be your root disk. If you have two disks installed they should be displayed as sd0 and sd1. We will be using sd1 as the root disk. The reason we will use sd1 is that in the Sparc 5 chassis sd0 is the top drive and sd1 is the bottom drive making sd0 easer to reach and upgrade if needed in the future (be it to increase system storage space or replace a worn drive).

The installer will then ask you if you want to do a Standard Install, Standard Plus X or Custom Install. Choose Custom. You are then prompted for how you would like sizes displayed. Choose Megabytes (that is how we will refer to things throughout the document).

It is now time to setup your partitions on the boot disk. First, you are prompted for the size of your root partition. If you do not have a 4gig hard disk for your boot disk you will want to adjust these numbers up or down accordingly. Enter 500 for the size of the root partition.

You will next be asked about swap space. You will want to choose about 512mb. Next is the /usr partition; give that 1500mb.

The installer has now finished with the 'standard' partitions; it will now prompt you to use up the remainder of the disk space. Next we will create a temp partition, so assign 500mb and tell it that it will be /tmp. The last partition we are going to create is the /var partition. We will make this large so it will have room for multiple purposes. Assign the rest of the space to it (you should have just over 1gig left) and then tell the installer it is for /var.

Your partition layout is now displayed to you. If it is set up the way you wanted, select Partitions are ok. If not select Change a partition and you will be given a chance to change your options. Select Partitions are ok to continue with the install.

NetBSD wants every disk to be named. This name is not important so you can accept the default of mydisk or choose another name.

You are now prompted to continue. This is your last chance to stop the install process before the disk is erased and the install begins. Select Yes to continue.

The installer will then spend some time writing file systems to the partitions you selected. Depending on the size of your disk and the speed of the system this can take a while. On the test system, it took about 2 minutes.

Once the file systems have been created, the system prompts you to hit [ENTER] to continue.

You can now choose between a full or custom install. Choose Custom installation so we will have some control over what is loaded.

NetBSD is distributed as a group of gzipped tar balls. If you decide that you need features in the future that were not installed now, it is as simple as mounting the install CD and unpacking the part of the distribution you are missing into the root of your system.

The only packages we want are: Kernel, Base, System, Compiler, Online Manual Pages, Miscellaneous and Text Processing Tools. Select the other packages one at a time to turn them off (all packages are installed by default, you toggle the package install on or off for each package by selecting them). When you have the install paired down, select Exit to continue.

You will now be asked whether or not you would like to see the file names as they are extracted. I would recommend against this as it slows down the install process (the file list takes a lot of time to display at 9600 baud). So, select the default of no.

You are now asked to choose which install medium you wish to use. Choose cdrom. Then you are asked for the location of the files. If you used the ISO image from the NetBSD site you can accept the default of /sparc/binary/sets by choosing Continue. If not, you will need to enter the new path here by selecting Directory. This document does not cover installing from a raw device.

This part of the install can take anywhere from 5 to 90 minutes depending on the speed of your CD and your system. Remember that it is ok to see the corrected soft errors as mentioned earlier in this document.

When the extraction is done it will prompt you to continue. When you hit [ENTER] the device files are created. The installer then tells you that the selected sets have been installed, and that you will have a chance to configure some system settings. Choose ok to continue.

You are now shown a list of time zones which you can choose from. The arrow keys will move you up and down, > (greater than) will page down and < (less than) will page up. Select the time zone you want to use and hit [ENTER] to accept, then press x and then [ENTER] to save the change.

Next, you are asked if you would like to set the root password. Choose yes. Set your root password. This is the doorway to your system. It should be rememberable to you but hard for others to guess. Enter it a second time to verify it was entered correctly.

You have completed the basic install. Hit [ENTER] when prompted to be returned to the install menu. At the main menu choose Exit Install System. This will stop the installer and halt the system returning you to the PROM ok prompt.

Type `boot` and hit [ENTER] to restart the system and have it boot from the hard disk.

Congratulations, you are now running NetBSD.

6.2 Post Install Cleanup

Log in to your newly installed system as root using the root password we set during the install. By default the root shell for NetBSD is csh. There are a couple of settings we need to make so that your text editor (vi or ed, no others have been loaded at this point) can understand your terminal.

First we need to set your terminal type. By default NetBSD will assume you are on a Sun console (terminal type “sun”). If you are not on a Sun console you will need to update that setting. If you are not sure what kind of terminal emulation your program handles a safe guess is vt100. It is a standard that almost all terminal programs can understand. You set that with the following command:

```
setenv TERM vt100
```

This line is case sensitive so it is important that you type it exactly as shown above. Next you will need to tell the system the dimensions of your screen. In many cases your screen is 80 characters wide by 25 characters tall. The author is using minicom which uses the bottom line to show status, so the working screen size is 80 by 24 (instead of 25). To tell the system your screen size you use the `stty` command:

```
stty cols 80 rows 24
```

Now we want to look at `netstat` to see if there are any processes currently listening for network connections (by default NetBSD ships with telnet listening). Run `` `netstat –an` `` to see if there are any listening Internet ports (note that there is no Active Internet connections section).

```
# netstat –an

Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
f0896000 dgram       0      0 f533d518        0        0        0 /var/run/log
```

The above output from `netstat` shows that the Unix domain sockets are in use. These are local to the system and not available on the network interfaces. This is how NetBSD installs by default, with no services listening for network connections.

Now it is time to make the first system change. Turn off the `inetd` deamon as it is not in use (if you decide to use it in the future you can turn it back on, but the author prefers not to use if it possible). Change to the /etc directory and edit the rc.conf file. It should look like this:

```
#       $NetBSD: rc.conf,v 1.85.2.9 2001/04/24 22:42:44 he Exp $
#
# see rc.conf(5) for more information.
#
# Use program=YES to enable program, NO to disable it. program_flags are
# passed to the program on the command line.
#

# These can be overridden below.
#
if [ -r /etc/defaults/rc.conf ]; then
        . /etc/defaults/rc.conf
fi

# If this is not set to YES, the system will drop into single-user mode.
#
rc_configured=YES

# Add local overrides below
#
```

At the end of the file, after the “Add local overrides below” section, add the following line:

```
inetd=NO
```

This will tell the system not to start the `inetd` daemon when the system starts. If in the future you decide you need this daemon you can remove this line, but for the basic server build it is not needed. Save the file and exit the text editor.

Now find the `inetd` deamon by using the following command:

```
ps –auxwww | grep inetd
```

You should see either one or two lines (the second line would be the `grep` you just performed). It should looks something like this:

```
root 180  0.0  0.1  96 128 ?? Is    4:49PM 0:00.03 /usr/sbin/inetd -l
```

The PID, or process ID, is the second column, in this case it's 180. Kill that process with the `kill` command and the PID.

```
kill 180
```

Another change we need to make to the system is to add the CD (or floppy if you do not have a CD) to the /etc/fstab file. Open the fstab file in an editor. If you followed this document's suggestion for the file system it should look like this:

```
/dev/sd1a / type ffs rw 1 1
/dev/sd1b none swap sw 0 0
/dev/sd1d /tmp ffs rw 1 2
/dev/sd1e /var ffs rw 1 2
/dev/sd1g /usr ffs rw 1 2
/kern /kern kernfs rw
```

Remove the /kern line (it is not needed for the system) and replace it with the following line if you have a single CD:

```
/dev/cd0a /cdrom cd9660 ro,nosuid,noauto 0 0
```

If you have a floppy drive, you will want to add the following line (even if you also have a CD):

```
/dev/fd0a /floppy msdos rw,nosuid,noauto 0 0
```

The use of msdos for the floppy file system is recommended so that you can modify the floppy on almost any platform (Linux, *BSD, Windows and MacOS all support the reading of DOS formatted floppy disks).

We now want to turn on Soft Dependencies[18] (also known as Soft Updates). This is a method of accessing the file system that improves system performance while also improving the stability of the file system.

To make the file systems use Soft Dependences add "softdep" to the mount options of each mounted file system (not including the swap partition) so that "rw" becomes "rw,softdep". The /etc/fstab file would look like this after the changes (assuming the system had a cdrom but no floppy):

```
/dev/sd1a / type ffs rw,softdep 1 1
/dev/sd1b none swap sw 0 0
/dev/sd1d /tmp ffs rw,softdep 1 2
/dev/sd1e /var ffs rw,softdep 1 2
/dev/sd1g /usr ffs rw,softdep 1 2
/dev/cd0a /cdrom cd9660 ro,nosuid,noauto 0 0
```

[18] More information can be found on Soft Dependencies at: http://www.mckusick.com/softdep/

It is argued by a growing number of people that a file system using Soft Dependencies is actually faster[19] than file systems using Journaling. The author has not tested to see if this is true, but does know that it improves performance and reboot after system crashes.

Save the file and exit from the editor. Check to make sure the directories exist (/cdrom and /floppy if defined in /etc/fstab), if they don't then create them. Also, unmount the kernel file system (with the `umount /kern` command) and then remove that directory as it is no longer needed.

Next you should open the /etc/ttys file in your editor and change the three entries that end in `secure` to `insecure`. By making this change the system will require the root password before someone at the console can boot into single user mode.

Now it is time to create a user account. To do this use the following command:

`useradd –d` *`pathtouserhome`* `–m –c` *`"User Full Name" username`*

This will add the user *username* with a home directory of *pathtouserhome* and populate that directory with the standard startup files from the skel directory. For more information on the `useradd` please view its man page with `man useradd`.

Once the user has been added use the `passwd` command to set a password for the user. By default the `useradd` command locks the account but you should not rely on this behavior. At this point you must give an account the ability to become root (using the su command). To do this you need to add the users name to the wheel line of the /etc/group file. This is a very important step because by changing the console entries in the /etc/ttys file to insecure the system will no longer allow the root user to log directly into the system. If you have no users in the wheel group you have locked yourself out of the system.

To add a user in the /etc/group file open it in your editor. The wheel group is the first group listed. It should look like this:

```
wheel:*:0:root
```

The user names are comma delimited, so to add a user to the wheel group you would add a *,username* to the end of the line. It should then look like:

```
wheel:*:0:root,username
```

Save the file and exit from the editor.

Finally, you need to make a couple of changes to the system. First copy the `/sbin/nologin` script to `/sbin/mailaccount` with the following command:

```
cp /sbin/nologin /sbin/mailaccount
```

[19] See the paper titled "Journaling Versus Soft Updates: Asyncronous Meta-data Protection in File Systems" by Margo I. Seltzer, Gregory R. Ganger, M. Kirk McKusick, Keith A. Smith, Craig A. N. Soules and Christopher A. Stein available at the USENIX web site: http://www.usenix.org/publications/library/proceedings/usenix2000/general/full_papers/seltzer/seltzer_html/index.html

Open /etc/mailaccount in your text editor. Change the part of the line that reads "This account is currently not available." to "This is not a login account.". Save and exit from that file.

Next add the /sbin/mailaccount script to the /etc/shells file with the following command:

```
echo "/sbin/mailaccount" >> /etc/shells
```

NOTE: It is very important that you use two greater than symbols (>>) instead of just one, as a single greater than symbol will replace the file with the echo statement instead of appending it to the end.

The `mailaccount` script will be used for your user accounts which shouldn't be login accounts, but need passwords (this is to have them not report as errors in your security script).

6.3 Configuring Your Kernel

In order to remove unneeded code and add support for some of the software we wish to use, we must configure a custom kernel for the system. The author recommends that you never run a server with a default, or generic, kernel; you should always customize it to your needs and the hardware[20].

Place the data CD you created in the CD of the system and mount it with the `mount /cdrom` command. Change to the / directory and unpack the kernel source archive you downloaded with the following command:

```
tar –zxvf /cdrom/syssrc.tgz
```

This will extract the source code for the kernel into the /usr/src/sys tree in your file system. NetBSD stores the configuration file for its kernel in /usr/src/sys/arch/*ARCH*/conf (where you replace the *ARCH* with your system architecture). We are working on the Sparc platform, so change to the directory /usr/src/sys/arch/sparc/conf.

NetBSD provides a template called GENERIC which we can use to start configuring the kernel for your system. Copy that file to another name. The author recommends that you name it after the revision of the kernel (in this case we will name it netbsd_sparc5_1.5.2). This paper is not going to cover all of the parts of the kernel configuration file, only those that should be changed for the system. For detailed information on the kernel options please see: http://www.tac.eu.org/cgi-bin/man-cgi?options+$+NetBSD-current or do a `man options` at the prompt of your NetBSD system.

Before editing the file it is important to know what devices are on the system. To do this you run the `dmesg` command, the output should look similar to this:

[20] This paper does not attempt to give a full overview of the kernel build process. For more detailed information please see http://www.netbsd.org/Documentation/kernel/#how_to_build_a_kernel

```
# dmesg
NetBSD 1.5.2 (GENERIC) #0: Wed Aug 22 04:33:09 CST 2001
    toor@proxima:/usr/src/sys/arch/sparc/compile/GENERIC
total memory = 255 MB
avail memory = 233 MB
using 896 buffers containing 13188 KB of memory
bootpath: /iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/esp@5,8800000/sd@3,0
mainbus0 (root): SUNW,SPARCstation-5
cpu0 at mainbus0: MB86904 @ 110 MHz, on-chip FPU
cpu0: 16K instruction (32 b/l), 8K data (16 b/l): cache enabled
obio0 at mainbus0
clock0 at obio0 slot 0 offset 0x200000: mk48t08 (eeprom)
timer0 at obio0 slot 0 offset 0xd00000 delay constant 52
zs0 at obio0 slot 0 offset 0x100000 level 12 softpri 6
zstty0 at zs0 channel 0 (console i/o)
zstty1 at zs0 channel 1
zs1 at obio0 slot 0 offset 0x0 level 12 softpri 6
kbd0 at zs1 channel 0
ms0 at zs1 channel 1
slavioconfig at obio0 slot 0 offset 0x800000 not configured
auxreg0 at obio0 slot 0 offset 0x900000
power0 at obio0 slot 0 offset 0x910000 level 2
fdc0 at obio0 slot 0 offset 0x400000 level 11: no drives attached
iommu0 at mainbus0 addr 0x10000000: version 0x4/0x0, page-size 4096, range 64MB
sbus0 at iommu0: clock = 22 MHz
dma0 at sbus0 slot 5 offset 0x8400000: rev 2
esp0 at dma0 slot 5 offset 0x8800000 level 4: ESP200, 40MHz, SCSI ID 7
scsibus0 at esp0: 8 targets, 8 luns per target
bpp0 at sbus0 slot 5 offset 0xc800000 level 2 (ipl 3): rev 2
ledma0 at sbus0 slot 5 offset 0x8400010: rev 2
le0 at ledma0 slot 5 offset 0x8c00000 level 6: address 08:00:20:77:06:12
le0: 8 receive buffers, 2 transmit buffers
audiocs0 at sbus0 slot 4 offset 0xc000000 level 9: CS4231A
audio0 at audiocs0: full duplex
power-management at sbus0 slot 4 offset 0xa000000 not configured
cgsix0 at sbus0 slot 3 offset 0x0 level 9: SUNW,501-2325, 1152 x 900, rev 11
cgsix0: attached to /dev/fb
scsibus0: waiting 2 seconds for devices to settle...
probe(esp0:1:0): max sync rate 10.00MB/s
sd0 at scsibus0 target 1 lun 0: <SEAGATE, ST34502LCSUN4.2G, 0828> SCSI2 0/direct
fixed
sd0: 4094 MB, 3882 cyl, 16 head, 135 sec, 512 bytes/sect x 8385121 sectors
probe(esp0:3:0): max sync rate 10.00MB/s
sd1 at scsibus0 target 3 lun 0: <SEAGATE, ST34502LCSUN4.2G, 0828> SCSI2 0/direct
fixed
sd1: 4094 MB, 3882 cyl, 16 head, 135 sec, 512 bytes/sect x 8385121 sectors
probe(esp0:6:0): max sync rate 4.23MB/s
cd0 at scsibus0 target 6 lun 0: <TOSHIBA, XM-4101TASUNSLCD, 3424> SCSI2 5/cdrom
removable
root on sd1a dumps on sd1
root file system type: ffs
```

Because its output is greater than one screen you might want to pipe that output through the `more` utility so that it will pause after every screen.

Take note of the devices found by the system. You will need to know this information when configuring the kernel. The devices that will need to be configured are any line that start with "xxxxx at yyyyy". The word "at" signifies that the device xxxxx has been found attached to device yyyyy,. Write down each device xxxxx and the device yyyyy it is attached to on something that can be referenced during the kernel configuration.

Now open the copy of the configuration file you created in your text editor. The beginning of the file should look like this:

```
#       $NetBSD: GENERIC,v 1.107.2.5 2001/05/06 14:04:07 he Exp $

include "arch/sparc/conf/std.sparc"

#ident          "GENERIC-$Revision: 1.107.2.5 $"

maxusers        32

## System kernel configuration.  See options(4) for more detail.


# Options for variants of the Sun SPARC architecture.
# We currently support three architecture types; at least one is required.
options         SUN4            # sun4/100, sun4/200, sun4/300
options         SUN4C           # sun4c - SS1, 1+, 2, ELC, SLC, IPC, IPX, etc.
options         SUN4M           # sun4m - SS10, SS20, Classic, etc.

options         SUN4_MMU3L      # sun4/400 3-level MMU


## System options specific to the sparc machine type

# Blink the power LED on some machines to indicate the system load.
#options        BLINK
```

Any line in the configuration file that begins with a hash symbol (#) is a comment and is ignored by the system. These comments are just there to help you understand the configuration options.

The "include" entry is to add some Sparc platform required settings. This line should not be changed. The maxusers line is set to a good baseline. Depending on the use for the server it is usually safe to leave it at the default setting.

The first section we will look at is the "Sun SPARC architecture" section. The available architectures are SUN4, SUN4C, SUN4M and SUN4_MMU3L (if you are working on a SUNU system, also known as an UltraSparc, you should be using the Sparc64 distribution instead of the Sparc distribution. The Sparc64 distribution is not covered in this paper). If you are using a Sparc 5, then the selection you want to keep is SUN4M. Place a hash mark in front of the other lines (if you do not have a Sparc 5 you should select the line that is appropriate for your system).

The next sections are Sparc specific options. The first is the BLINK option; this is not required but can be a nice feature to have. It causes the power light on the Sparc to flash at a rate determined by the system load. It is commented out by default, but if you would like to use it you can remove the hash to activate it.

Next are the console drivers. This is only needed if you have a graphical interface attached to the system. If you are controlling it from the serial port then these can be commented out to turn them off.

The sections that follow handle the kernel's memory and the System V compatible IPC system. The default settings are the best way to go so you don't need to make any changes there.

The next section is the loadable kernel module support section; it has a single line for LKM (Loadable Kernel Module). The author recommends that this be commented out.

Virus writers are beginning to create kernel modules that can bypass most security software packages[21]. Since we are building single use servers we have no need for loadable modules. Everything will be statically compiled in to the kernel.

Unless you will be netbooting this server you can comment out the NFS_BOOT_BOOTPARAM line in the NFS boot options section.

Following the NFS boot section is a large section on debugging information. This is commented out by default and it is best to leave it that way. Most debugging options cause kernel bloat and can hinder system performance. It should only be enabled if you are attempting to debug kernel activity.

Now skip down to the "Options for compatibility" section. Comment out the two SunOS compatibility sections: COMPAT_SUNOS and COMPAT_SVR4.

The next section is "File Systems". In this section you can comment out: NFS, KERNFS, NULLFS, OVERLAY, MFS, FDESC, UMAPFS, LFS, PORTAL, PROCFS, UNION and CODA. This only leaves: FFS, CD9660 and MSDOSFS. (If in the future you find that you need one of these other options you can uncomment them and recompile the kernel). Continuing with the file system options you can comment out NFSSERVER. (You can also comment out quotas, but you may want to enable those in the future).

In the "Network protocol support" you should comment out every line except for: INET, IPFILTER_LOG and NTP. (If you are going to need IPv6 you should also leave INET6 uncommented).

The next section of the configuration file is the devices section. As you look through this section refer to the notes you took earlier which list the devices and what they were attached to.

It is important that you pay close attention to device association as you go through the device section. There are sometimes more than one choice for each device and it is important that you choose the correct one. An example of this is the sbus0 device. You can choose from "sbus0 at mainbus0" or "sbus0 at iommu0". With the Sparc 5 hardware the correct choice should be "sbus0 at iommu0", the other options will generate a kernel that does not boot your computer.

Comment out any device (ignoring pseudo-device entries) that is listed in the configuration file that is not also listed in your `dmesg` output. A couple of exceptions you may wish to make are for SCSI devices like tape drives (st*) and cdrom drives (cd*) as well as the floppy device (fd*). Even if these devices don't show up in the `dmesg` output, you may want to add these devices in the future and it does not hurt to leave the support for them in the kernel.

[21] A guide to LKM's is available at Packet Storm Security's website http://packetstormsecurity.nl/docs/hack/LKM_HACKING.html

Looking through the entire configuration file, the author recommends you turn off the following pseudo-device entries (unless you have specific need for them): vnd, ccd, sl, ppp, tun, gre, gif, vlan and vcoda.

Finally, you need to add four option entries to the end of the file. The author recommends that you add a comment as well so you will remember why you added them in the future. The option entries should be added as listed below:

```
# IPFORWARDING=0 turns off IP packet forwarding
options IPFORWARDING=0

# IPFORWSRCRT=0 turns off support for source routed packets
options IPFORWSRCRT=0

# IPFILTER_DEFAULT_BLOCK tells IPFilter to block all packets
# by default until set by the firewall script (more secure
# than default of allow all until blocked).
options IPFILTER_DEFAULT_BLOCK
```

Now that you have completed the changes to the kernel configuration file, saved it and exited from the editor, you need to run the `config` command to ready the kernel for compiling. To do this you launch configure with the name of the file you created with your kernel configuration. In our example you would do the following:

```
# /usr/sbin/config netbsd_sparc5_1.5.2
Don't forget to run "make depend"
```

If the configuration was successful, you will see the above message. If there is a problem with the configuration it will report the error and stop. Now you need to change to the compile directory, which is /usr/src/sys/arch/*ARCH*/compile/*config_file_name* replacing *ARCH* with your architecture (sparc) and *config_file_name* with the name of your kernel configuration file.

```
cd /usr/src/sys/arch/sparc/compile/netbsd_sparc5_1.5.2
```

Once you are in that directory you can run a `make depend` and a `make` to build the kernel.

```
make depend && make
```

If the compilation runs without any problems you can then do a `make install` to have the system move the new kernel into place and ready the system to use it. If there is an error during the install it most likely caused by an error in the configuration file. Go back and check the file to make sure there are no typos in the settings you have entered and that all of the notes in the file have been read. If you continue to have a problem you should contact the NetBSD port-sparc[22] mailing list for help.

Reboot your system with the `reboot` command. If your system boots to the login prompt you are now running the new kernel. The author recommends that you keep the

[22] You can send a message directly to the port-sparc@netbsd.org address to ask questions about the Sparc port of NetBSD. You can also read the archives of the port-sparc mailing list at http://mail-index.netbsd.org/port/sparc/

onetbsd file in case you add new hardware to the system at some point in the future and need to probe it out with the generic kernel.

If your system freezes or crashes while trying to boot the new kernel you can revert to the old one by following these directions[23]:

1. Return to the `ok>` prompt by performing a [BREAK]
2. Type `` `boot onetbsd –s` `` at the ok prompt and wait for the system to boot into the old kernel in single user mode.
3. Check the file system by entering `` `fsck /` `` at the shell prompt.
4. Remount the root file system in read/write mode with `` `mount /` ``.
5. Copy the old kernel back in to place with `` `cp /onetbsd /netbsd` `` so the old kernel will be the default boot kernel at next reboot.
6. Continue to full multi-user state by entering `` `exit` `` at the shell prompt.

6.4 Configuring Basic System Settings

Open the /etc/rc.conf file in your text editor. Go to the end of the file where the `inetd=NO` was added before and insert the following text in to the file.

```
hostname=""
defaultroute=""
savecore=NO
update_motd=NO
securelevel="1"
ipfilter=YES
ipmon=YES
sshd=YES          sshd_flags=""
```

Enter the fully qualified host name for this system in the quotes after `hostname` and the IP address for your network's router in the quotes after `defaultroute`. Neither of these have to be the entries you will be using in the future, they just need to work long enough to place this system on your network to install and setup the packages system listed later in this paper (at this time the system should still be disconnected from any network).

The `savecore` command tells the system not to save core files. Unless this is going to be a development box it is best not to leave core files as they take up space and can contain sensitive information.

`update_motd` tells the system not to write over the message of the day file (/etc/motd) on boot. This way you can customize the file (mentioned later in this paper) and not lose your changes.

[23] These instructions are adapted from the ones on the NetBSD site at http://www.netbsd.org/Documentation/kernel/#how_to_build_a_kernel.

The `securelevel` setting tells the system what security level the kernel should be set at after it finishes initializing the system. Level 2 is the most secure you can set the system to, zero is the least secure[24] (there is also a -1, but that tells the system to always run at 0). Setting it to 1 now will give us some added security while still allowing us to configure the system. Once the configuration is done it should be set to 2.

`ipfilter=YES` tells the system to start the IPFilter firewall using the settings in /etc/ipf.conf and `ipmon=YES` tells the system that syslog should collect log information from IPFilter.

`sshd=YES` tells the system that it should start the Secure Shell daemon when the computer boots. Use a tab between any switch and its options. In this case there should be a tab between `sshd=YES` and `sshd_flags=""`.

Once these changes have been made, save the file and exit from the editor. Next you will want to open the /etc/hosts file in the editor and add an entry for your system's IP address and the IP address of your gateway. If your system's name was foo.bar.org and your gateways name was gateway.bar.org your entries should look like this (tabs, not spaces):

```
10.0.0.2    foo.bar.org foo
10.0.0.1    gateway.bar.org         gateway
```

Save the hosts file and exit from the editor. Next you need to create the configuration file for your network interface which contains the IP address as well as the subnet mask for the interface. If you don't know your network interfaces name you can do an `` `ifconfig –a` `` to list all network interfaces. Ignoring the entry for lo0 (the loopback device) your network interfaces are the ones listed in the output. In the case of the Sparc 5 the default network interface is le0. Continuing with the example above for the /etc/hosts file if you want to configure the le0 interface with the IP address of 10.0.0.2. In a class C network space (small net of 255 addresses) you would use the following command to setup the interface configuration file.

```
# echo "10.0.0.2 netmask 255.255.255.0" > /etc/ifconfig.le0
```

The format of the file is *IP address* (space) netmask *netmask*. For more information on the entries that can be placed in the interface configuration file, please review the man page for the `ifconfig` command.

Now you have to tell the system about a DNS server it can use for host name resolution. To do this edit the /etc/resolv.conf file (it is not part of the default install so you will have to create it).

Now you will want to enter two lines in the file. The first tells the system what its domain name is, the second what server it should use for name resolution (you can have as many entries of the second line as you like, one for each different server you want the system to use). The file should look like this when you are done:

[24] View the man page for init(8) by entering `` `man init` `` or by going to the web site http://www.tac.eu.org/cgi-bin/man-cgi?init+8 for more information on what the security levels mean.

```
domain bar.org
nameserver 10.0.0.3
```

Save this file and exit from the editor.

6.5 Turning on IPFilter[25]

This section is not meant to teach you how to configure or use the IPFilter firewall system. It is only meant to show you how to set your system up in a more secure way than having the system un-firewalled. For more information on configuring the IPFirewall please see the "NetBSD Security Processes and Services: Configuring IPFILTER" web page at http://www.netbsd.org/Documentation/network/nsps/config_ipf.html.

For the system to be able to use the network now that it is running with a new kernel (one that defaults to dropping all packets in and outbound) a configuration file needs to be created to tell IPFilter what to do.

The configuration file is /etc/ipf.conf. Create that file with your text editor and enter the following information (This information assumes you are using a Sparc 5 with the le0 network interface and the local IP address of 10.0.0.2. Change this as needed for your system). Remember that lines ending with a back slash "\" should count as a single line as the one that follows, removing the backslash.

```
# Block all traffic if it is not allowed below
block in on le0 all

# Block traffic to or from loopback on the real network interface
block in quick on le0 from 127.0.0.0/8 to any
block in quick on le0 from any to 127.0.0.0/8

# Allow the system ping other hosts
pass out quick on le0 proto icmp from 10.0.0.2/32 to any icmp-type echo
pass in quick on le0 proto icmp from any to 10.0.0.2/32 icmp-type \
      echorep

# Allow other systems to ping this host
pass in quick on le0 proto icmp from any to 10.0.0.2/32 icmp-type echo
pass out quick on le0 proto icmp from 10.0.0.2/32 to any icmp-type \
      echorep

# Allow systems to warn you if something is unreachable
pass in quick on le0 proto icmp from any to 10.0.0.2/32 icmp-type \
      unreach

# Temporary allow outbound tcp for ftp use
pass out quick on le0 proto tcp from 10.0.0.2/32 to any keep state

# Allow system to communicate with itself
```

[25] The information in the "Turning on IPFilter" section is taken from the Configuring IPFILTER web page maintained by the NetBSD group at http://www.netbsd.org/Documentation/network/nsps/config_ipf.html.

```
pass in quick on lo0 all
pass out quick on lo0 all

# Allow outbound sup connections (for pkg system)
pass out quick on le0 proto tcp from 10.0.0.2/32 to any port = 871 \
      keep state

# Allow outbound name service requests
pass out quick on le0 proto tcp from 10.0.0.2/32 to any port = 53 \
      keep state
pass out quick on le0 proto udp from 10.0.0.2/32 to any port = 53 \
      keep state

# Allow inbound ssh connections
pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 22 \
      keep state
```

You will notice that the first rule is to block all packets. IPFilter rulesets use the last match found in the ruleset for its action, not the first (the quick command changes this behavior). The rules are commented so you see what each section is for. This rule set is based on only allowing into and out of the system what is required, and no more than that. For a more detailed look at IPFilter and its configuration, please see http://www.obfuscation.org/ipf/.

One entry worth noting is the "Temporary allow". This is to allow all outbound tcp connections to work. This rule is needed so that passive ftp transfers will function for the software installs from the package tree. This entry will be commented out at the end of this paper.

Save the file and exit from the editor.

6.6 The NetBSD Ports System

We will be using the sup system to keep the packages synchronized with the NetBSD master sites. Before sup can be used you need to configure it. Start by creating two directories: /etc/supfiles and /usr/sup. You next need to choose the server you are going to synchronize with. In NetBSD v1.5.2 there are four main servers that you can choose from: sup.au.netbsd.org (Australia), sup.jp.netbsd.org (Japan), sup.netbsd.org (US) and sup2.fr.netbsd.org (France). Each site has a file located in /usr/share/examples/supfiles,. Copy the site you want to /etc/supfiles/coll.list (as shown below).

```
cp /usr/share/examples/supfiles/sup.netbsd.org /etc/supfiles/coll.list
```

Now open the /etc/supfiles/coll.list file with a text editor and comment out (using a hash symbol, #) all of the lines except for the group of two that start with "`current release=pkgsrc`". Save and exit from the file.

Now you can run the sup command to download the pkgsrc tree. The beginning of it will look something like this:

```
# sup -s -v
SUP 8.26 (4.3 BSD) for system software at Mar 22 15:26:37
```

```
SUP Upgrade of current-pkgsrc at Fri Mar 22 15:26:38 2002
SUP Fileserver 8.13 (4.3 BSD) 26409 on sup.netbsd.org at 15:26:38
SUP Requesting changes since Dev 31 16:00:00 1969
SUP Using compressed file transfer
SUP Created directory pkgsrc for pkgsrc/CVS/Entries
SUP Created directory pkgsrc/CVS for pkgsrc/CVS/Entries
SUP Receiving file pkgsrc/CVS/Entries
SUP Receiving file pkgsrc/CVS/Repository
```

Once the pkgsrc tree has been built we can configure the packages system for use. Change to the /etc direcotory and edit the mk.conf file (this file does not exist in a default install so we will be creating it). Add the following entries to the file:

```
MKCRYPTO=YES
PASSIVE_FETCH=YES
ACCEPTABLE_LICENSES+=no-profit
ACCEPTABLE_LICENSES+=fee-based-commercial-use
```

Setting MKRYPTO to YES tells the system to fetch and install cryptographic packages if asked. PASSIVE_FETCH tells the system to use passive ftp, instead of active ftp. This helps work around problems with many firewalls.

The ACCEPTABLE_LICENSES line tells the system that in addition to the standard "free" licenses the Sendmail license is also accepted. (The restriction in the sendmail license is that you may not build a commercial sendmail product and sell it. There are no restrictions on commercial use of the product). The second ACCEPTABLE_LICENSES line says that it is ok to install software that requires a fee for commercial use. This is for the OpenSSL package which uses some algorithms that are still patented. (The RSA patent has been released, but the IDEA[26] patent is still in effect, though most software can be built to not use the IDEA algorithm).

Save and exit from the mk.conf file. The system packages system is now ready for use. The first software we will configure is the `pkglint` package; this is the software that will report packages changes and updates.

To install any package under NetBSD change to the directory of the package in the /usr/pkgsrc tree. In the case of the `pkglint` program we will be going to /usr/pkgsrc/pkgtools/pkglint. Next you will tell the system to fetch the required software. You do this with the `` `make fetch-list | sh` `` command. `` `make fetch-list` `` generates a shell script which will download all of the files you need, piping this in to a shell automatically performs the operation.

Once all of the files have been downloaded you should run `` `make depend` `` to have the system build all of the dependencies (this includes other programs this application requires to run correctly). When that successfully completes you can run `make` to build the software package.

Once the build has completed `` `make install` `` will install the software to the system for use, and then `` `make distclean` `` will remove the build files from the /usr/pkgsrc

[26] More information about IDEA can be found at the Ascom site http://www.ascom.ch.

tree to save space (once the software has been built and installed you don't need the build files any more).

6.7 Installing OpenSSL

To install OpenSSL change to the /usr/pkgsrc/security/openssl directory. As you did to install the pkglint software, you now do a `make fetch-list | sh` to download the needed software and then a `make depend` to have the system build all of the dependencies (if there are any. If nothing happens then there are no dependences) then `make` to build the software package.

Once the build has completed `make install` will install the software to the system for use, and then `make distclean` to remove the work files (and conserve disk space).

If there is another text editor you would prefer to be using this is a good time to install it from the packages system. Editors can be found in the /usr/pkgsrc/editors tree. You use the same commands to build and install your editor as you did above.

NOTE: Once you have completed the installation of the programs you want from the packages tree it is a good idea to unplug your network connection. It is not needed until later in the build so it is safer to be disconnected from the network.

6.8 Installing and Configuring OpenSSH

Enter the /usr/src directory again and unpack the OpenSSH distribution with the tar command `tar –zxvf /cdrom/openssh-3.1p1.tar.gz`.Change directory to /usr/src/openssh-3.1p1 and run the following command:

```
./configure --prefix=/usr --sysconfdir=/etc/ssh \
      --with-ssl-dir=/usr/pkg
```

Notice that though the OpenSSL build process used the config script, OpenSSH uses configure.

Once the script has completed configuring OpenSSH for your system you can run the `make` command to build the software and then `make install`. Once the software is installed you need to run the following command to remove the old Secure Shell configuration files so as not to confuse later administrators.

```
rm /etc/ssh?*
```

Next open the /etc/rc.d/sshd file in your editor and change every occurrence of /etc to /etc/ssh (in vi this can be accomplished with the command `[ESC] :11,$s/\/etc\//\/etc\/ssh\//g`). This will let the startup script know where the new configuration directory should be. You should also change the entry which reads "`/etc/${name}.conf`" to "`/etc/ssh/${name}_config`" so the script will be able to find the new configuration file.

Change to the /etc/ssh directory and open the sshd_config file in your editor. Find the line "`#Protocol 2,1`" and remove the hash mark and the "`,1`" so that the line now reads "`Protocol 2`". This change tells the Secure Shell daemon not to allow use of the older 1.x protocols and protects you from many of the security issues with the Secure Shell tools.

Find the line "`#PermitRootLogin yes`" and change that to "`PermitRootLogin no`". This will make Secure Shell act like the rest of the system requiring someone to login as themselves and then su to root instead of connecting directly as root.

Next find the line "`#PrintMotd yes`" and change it to "`PrintMotd no`". This will keep the message of the day from being printed, but that can cause problems with some remote administrative commands.

Finally, find the line "`#Banner /some/path`" and change it to "`Banner /etc/issue.net`" to have Secure Shell show the network banner when a client connects (more on this later in the document).

6.9 Installing Integrit

Return to the /usr/src directory and unpack the Integrit distribution as you did with the OpenSSH package. Change to the integrit-3.00.beta directory and run the following command:

```
# ./configure && make
```

This will configure Integrit for your system and the build the software. Once the build is done you can test it by doing a `make test` and if that reports that the code is good you can install it using the `make install` command. If you have any problems building or installing Integrit check the documentation on the Integrit website http://integrit.sourceforge.net/texinfo/integrit.html.

7 The Build: Step 2 – First Round Hardening

This section takes your generic system and makes the changes necessary to reduce the vulnerabilities the system may have, or stave off possible vulnerabilities in the future.

Why should you harden the system twice? (Once for the generic setup, once for the server specific setup). You can't be sure what every application you are going to load on your system will do unless you are good at following the code and making files associated with it. Hardening the system at this point will give you a checkpoint that you can use to review any changes made by the customization process. Before you start this section you should restart the system. This will let you review what software is started by the boot process.

7.1 What Really Needs to be Running?

It is important to make sure that you know what software is running on your system, and only have the software that is required actually running.

Assuming that you have logged into the system on the console using your standard user account, and then `su -` to become root you should see something very similar to this output when you run the `ps –auxwww` command.

```
# ps –auxwww
USER     PID %CPU %MEM VSZ RSS TT STAT STARTED    TIME COMMAND
root       0  0.0  0.3   0 764 ?? DLs  11:15PM 0:00.01 (swapper)
root     177  0.0  0.1 552 152 a  S    11:58PM 0:00.17 –csh
juser    152  0.0  0.1 572 160 a  Is   11:15PM 0:00.21 –ksh
root     150  0.0  0.0 248 100 ?? Is   11:15PM 0:00.05 /usr/sbin/cron
root     144  0.0  0.0 912  76 ?? Is   11:15PM 0:00.07 /usr/sbin/sshd
root      78  0.0  0.1 584 212 a- S    11:15PM 0:00.31 /usr/sbin/ipmon –sn
root      73  0.0  0.1 112 140 ?? Ss   11:15PM 0:00.18 /usr/sbin/syslogd –s
root       4  0.0  0.3   0 764 ?? DL   11:15PM 0:00.19 (ioflush)
root       3  0.0  0.3   0 764 ?? DL   11:15PM 0:00.05 (reaper)
root       2  0.0  0.3   0 764 ?? DL   11:15PM 0:00.00 (pagedaemon)
root       1  0.0  0.0 364  92 ?? Is   11:15PM 0:00.03 init
root     184  0.0  0.0 448  80 a  R+   11:59PM 0:00.00 ps -auxwww
```

The above listing shows what you should be running on your server at this point in the install. The two shells (`-csh` and `–ksh`) are of the user (`-ksh`) and the root (`-csh`) accounts that are currently logged in (the user account `su`'d to the root account in this case). You can also see the `ps` command that was run.

Kernel processes are shown in parentheses. These handle RAM and the file system. In NetBSD v1.5.2 there should be four: `swapper`, `ioflush`, `reaper` and `pagedeamon`.

The `init` process is the main control daemon of the system. It handles the initialization and shutdown of all base processes.

`cron` handles the execution of scheduled commands. `sshd` is the Secure Shell daemon that was installed earlier in the paper. `syslogd` is the system logging service and `ipmon` is the IPFilter monitoring tool.

7.2 Unneeded Accounts

NetBSD ships with a number of user accounts that are there either for historical reasons or because they might be needed by services the user could add later. The author believes that no accounts that are not currently in use should be on the system. If it is found later that an account is needed it can be added back to the system.

First, write down the list of users in the /etc/passwd file. We will need to search for files owned by these names to make sure they are not being used.

The command you can use to find if a user name is attached to any files is the following:

```
find / -user username –exec ls –la {} \;
```

The above command uses the find command to search the entire file system for any file whose user is set to *username*. When it finds a file that matches it then passes the file to the `` `ls –la` `` command (the braces {} tell find to give the file name found to the command your are executing). It is important to put the `\;` at the end of the command. This is required for the command to be executed correctly.

There are some users that should not be deleted, so there is no need to search for them on the system. These users are: root, daemon, bin, nobody and any user accounts you created.

There are also two accounts that are not needed and we can remove them without testing to see if they own any files. These users are: toor and falken. The toor user is a traditional back door, shipped disabled by default it has the same UID (0) as root and can be activated to give you a second password to access the system in case the first stops working or is lost. The falken user is just a historical account put in for fun, and is not required for anything.

You may find that an account owns just a few files. Depending on the account it may be ok to reassign those files to another user, or to remove them completely. This should be done with caution as you may not know if another program depends on these files. An example of this is the games account. The games account owns a tree of directories under /var called /var/games as well as a dir under /usr called /usr/games. If you unpacked the games.tgz file in the original distribution you should leave these files in place, but if you followed the recommended installation then there are no games installed on the system and it is safe to remove this directory tree.

In addition to the known unused accounts, the author recommends removal of the following accounts: operator, news, games, postfix, ingres. It is recommended that you run your own test to verify these findings as the user account requirements may change from distribution to distribution.

To remove the user perform the following commands:

```
cd /etc
tar –zcvf passwd_backup.tgz passwd master.passwd group
chmod 0000 passwd_backup.tgz
vipw
```

The `vipw` command opens the password file in the default editor (usually vi) and allows you to directly edit the file. Once you save and exit from the editor it recreates the shadow and password files for you.

NOTE: you should be very careful when working with the password files. A single mistake can lock you out of your system.

After you finish cleaning out your password file the next file to clean is the groups file, /etc/group. You can check this file the same way you checked the passwords file with find (replacing the –user with –group) and remove the groups that are unused. You backed up the /etc/group file at the same time you backed up the password files so you don't need to do that now.

Groups the author recommends keeping are: wheel, daemon, kmem, sys, tty, operator, mail, bin, wsrc, maildrop, staff, nobody, utmp, users, dialer and nogroup.

7.3 Checking Existing Automation

NetBSD ships with some default automation using the cron tools. `cron` handles the execution of scheduled commands (either recurring events scheduled through the crontab or single instance events scheduled through the at command).

In the default setup of NetBSD there is a single crontab setup. This is for the root user and can be viewed with the following command:

```
crontab -u root –l
```

The cron files have a specific format that you can learn more about by reading the man page for crontab(5)[27]. In the default crontab listing you will see that there are five events listed (one of which is disabled with the hash mark). Those five events are the `atrun` command, the `newsyslog` command and the `daily`, `weekly` and `monthly` scripts.

The `atrun` script handles the execution of at jobs. The `newsyslog` script handles the rollover of system log files (this will be covered later in the paper).

[27] The crontab(5) man page can also be viewed online at: http://www.tac.eu.org/cgi-bin/man-cgi?crontab+5+NetBSD-current.

8 The Build: Step 3 – Server Type Specific

In build step three we will customize the generic system into a specific server type. This is where you do your software specific configuration.

8.1 Mounting the Second Hard Disk

Because this system will be used as a mail server for sendmail and the Cyrus IMAP package, we will need to use the additional disk for two partitions: /var/spool/mqueue, the sendmail pending queue and /var/spool/imap, the Cyrus IMAP user directory.

The current system has two hard disks, sd0 and sd1. The sd1 drive has already been set up as the boot disk with the system on it. Now it is time to use the `disklabel` command to configure the second disk. Use the following command to launch `disklabel`:

```
disklabel –i –I sd0
```

The lowercase 'i' tells the system not to warn you if no disklabel exists on the drive, and the uppercase 'I' tells it that you want it to run in interactive mode.

Use the P (uppercase P) command to view the current layout of the disk. Remove all of the partition definitions that exist except for the "c:" partition. This one is required for the system.

To remove a partition, type the partition name and hit enter and accept the default for the first question (what the file system type will be, 4.2BSD). The second question is what the start offset will be. Set it to 0 (zero). The third question will be what the partition size will be. Set that to 0 as well. Now if you use the P command you will see that the partition is no longer listed (see the output below for an example).

```
# disklabel -i -I sd0
partition> P
8 partitions:
#        size   offset     fstype   [fsize bsize cpg/sgs]
  a:  1026000        0     4.2BSD     1024  8192    16   # (Cyl.    0 - 474)
  b:  1049760  1026000       swap                        # (Cyl.  475 - 960)
  c:  8380800        0     unused        0     0         # (Cyl.    0 - 3879)
  d:  1026000  5149440     4.2BSD     1024  8192    16   # (Cyl. 2384 - 2858)
  e:  2205360  6175440     4.2BSD     1024  8192    16   # (Cyl. 2859 - 3879)
  g:  3073680  2075760     4.2BSD     1024  8192    16   # (Cyl.  961 - 2383)
partition> a
Filesystem type [?] [4.2BSD]: [ENTER]
Start offset [0c, 0s, 0M]: 0
Partition size ('$' for all remaining) [475c, 1026000s, 500.977M]: 0
partition> P
8 partitions:
#        size   offset     fstype   [fsize bsize cpg/sgs]
  b:  1049760  1026000       swap                        # (Cyl.  475 - 960)
  c:  8380800        0     unused        0     0         # (Cyl.    0 - 3879)
  d:  1026000  5149440     4.2BSD     1024  8192    16   # (Cyl. 2384 - 2858)
  e:  2205360  6175440     4.2BSD     1024  8192    16   # (Cyl. 2859 - 3879)
  g:  3073680  2075760     4.2BSD     1024  8192    16   # (Cyl.  961 - 2383)
partition>
```

Once you have removed all of the old partitions (again, leaving the "c:" partition) you now need to create the two new partitions we will need for the system. We will call these two partitions "f" and "g" (the author has found from past system recoveries that if you stick to a standard scheme for setting up your disks you will have an easer time of figuring them out in the future. In that line of reasoning the author reserves the first two slices of the disk "a" and "b" for root and swap, leaving the rest open for other uses. When building a disk for special needs, like in this case, the author sets the partitions at the lower end of the scale as done in this case).

To create the first partition, type "`f`" at the "`partition>`" prompt. The system will prompt you for the filesystem type. Tell it 4.2BSD. For the start offset, tell it 0 (zero) for the beginning of the disk, and for partition size tell it 750M (for 750 megabytes). This partition will be for the sendmail outbound mail queue.

Now use the P command to see how the partition you setup was created. At the end of the output you should see some information in parentheses that look like this "`(Cyl. 0 – 711*)`". This tells you that the partition you created starts at the first cylinder of the hard disk (0) and ends on the 711th cylinder. You will need this information for the next partition you create.

To create the second partition type "`g`" at the "`partition>`" prompt. The system will prompt you for the filesystem type, tell it 4.2BSD. For the start offset tell it the end of the last partition plus one followed by a "c" for cylinder (in the case of the test machine the answer would be 712c), and for partition size tell it "$" (dollar sign) which means use the rest of the available space. This partition will be for the Cyrus IMAP servers mail storage.

Check your partitions again with the P command to make sure they are the way you want them, then save the current setup with the W command (it will ask you if you are sure you want to label the disk, answer y for yes). and then Q to quit the disklabel command.

You now need to create a file system on the two partitions you created. You do this with the `newfs` command. Use the command once for each partition you created as shown below:

```
newfs /dev/sd0f
```

Be very careful with this command. It does not ask if you are sure about building a file system, it just makes it.

Once both file systems have been created, change to the /var/spool directory. The mqueue directory will already exist, but you will need to create the imap directory with the mkdir command. Then mount both of the new partitions to the file system as shown below:

```
cd /var/spool
mkdir imap
mount /dev/sd0f /var/spool/mqueue
mount /dev/sd0g /var/spool/imap
```

Now you need to add the entries to the end of the /etc/fstab file so these file systems will be mounted automatically on system boot. Using your editor, add the following lines to your /etc/fstab file.

```
/dev/sd0f /var/spool/mqueue ffs rw,softdep 1 2
/dev/sd0g /var/spool/imap ffs rw,softdep 1 2
```

Then save and exit from the file.

8.2 Installing BIND

We will be using the –current tree for the name server to make sure that we have all current security fixes in place. As covered earlier in the paper, to install this package under NetBSD first change to its directory /usr/pkgsrc/net/bind9-current. Next you will tell the system to fetch the required software with `make fetch-list | sh`.

Once all of the files have been downloaded you should run `make` to build the software package, and at successful completion of the build you should run `make install` to have the system copy the file to their correct locations. Clean up the BIND package and its dependences with the `make distclean` command.

8.3 Configuring BIND

Before BIND can be started some directories need to be made, so run the following command:

```
mkdir –p /var/named/master /var/named/slave
```

The –p option to `mkdir` tells it to create any parts of the tree which do not currently exist. Now add the user and group that BIND will be using to the password file with the following command:

```
groupadd –g 42 bind
useradd –g 42 –c "BIND Psudo-User" –d /var/named –s /sbin/nologin \
      –u 42 bind
```

The `useradd` command will lock the password by default. Now change the ownership of the file tree you created above with the following two commands:

```
chown –R root:wheel /var/named
chown bind:bind /var/named/slave
```

The first command verifies that the tree is owned by root, the second gives the BIND tool the ownership of the /var/named/slave directory (where it will store temporary files).

Change to the /etc/rc.d directory and copy the new BIND startup script into the directory with the following command:

```
cp /usr/pkg/etc/rc.d/named9 .
```

Verify that it has execute permissions (you can set them by doing a `chmod 0555 named9`). Now edit the /etc/rc.d/named9 file and change the entry "/var/run/${name}.pid" to "/var/named/slave/${name}.pid" Finally you activate BIND in the /etc/rc.conf file by adding the following line:

```
named9=YES          named_flags="-c /var/named/named.conf –d 1 –n 1 \
      –u bind"
```

This tells the system to start BIND on boot with the command line flags you specified. The –c flag tells `named` where to find its configuration file, -d tells `named` what level of debug to report at (1 being very low), -n tells it how many processors your server has (if you tell it there is only one it does not have to check) and the –u tells BIND what user it should run as.

Change to the /var/named directory and move all of the files from /etc/namedb to the local dir with the command `mv /etc/namedb/* .` then remove the /etc/namedb directory with `rmdir /etc/namedb`.

To clean up the existing files run the following commands:

```
mv root.cache master/db.root
mv 127 master/db.127.0.0.0
mv localhost master/db.localhost
mv loopback.v6 master/db.localhost.v6
rm named.conf
```

You are now ready to configure the server. A secondary (or caching) name server running BIND requires at least two files, the configuration file and the root database. The following examples will setup the server to act as a caching name server. Open /var/named/named.conf with your editor and enter each section as listed below (each section will be explained).

```
options {
      directory "/var/named";
      query-source address * port 53;
      dump-file "slave/db.named_dump";
      pid-file "slave/named.pid"
      allow-query {
            127.0.0.1;
      };
};
```

The options section holds the general options for the entire server. The directory entry tells the daemon where its files can be found. The entry "query-source address * port 53" tells the daemon to listen to all network interfaces on port 53 (the standard DNS port) for queries. "dump-file" tells the daemon where to put the temporary files it creates while it is gathering information. The allow-query entry tells the system which IP addresses (or IP address ranges) can query the server for any host information; in this case it only allows queries from the local loopback adapter.

There are some attacks that are possible against name servers where the attacker queries a server for information on a domain name he controls. When the name server queries the

attacker's name server for the information the name server receives more information back than it requested (including information on domains the name server did not request), but it saves all of the information into its cache. The next time the users of the name server request information on one of the domains the attacker's server provided information on, the name server gives that bad information to the user instead of looking up the correct information. This attack can be used to redirect users to malicious sites.

This problem can be partially avoided by not allowing the name server to respond to requests from systems you don't control. In reality there is no reason for someone outside of your organization to make general queries against your name server, every ISP offers name servers of their own for their customers to use.

```
zone "." {
      type hint;
      file "master/db.root";

};
```

The zone marker tells the system that this is the beginning of the entry for the domain name "." or root. The type entry tells the server what kind of entry this record is for, in this case it is a hint entry. The file entry tells the name server which file (relative to the server path in the pervious section) holds the information.

The above entry tells the server where it can find its hint file. For a name server to be able to do name lookups it has to start with the root servers. These are the servers that can direct the name server to the correct server for its query. The hint file tells the name server how to find the root servers.

```
zone "0.0.127.IN-ADDR.ARPA" {
      type master;
      file "master/db.0.0.127.in-addr.arpa";
      allow-transfer {
            127.0.0.1;
      };
      allow-query {
            127.0.0.1;
            };
};
```

In the above entry we are creating a reference for the reverse lookup of the loopback adapter (which has an IP address of 127.0.0.1). In this record, type tells the name server that it is the master for this domain and file tells the server which file contains the information on the domain.

The allow-transfer entry tells the name server which IP addresses are allowed to do zone transfers. In zone transfers a name server or DNS tool requests a dump of all of the information on a domain. This should be restricted to just a list of name servers which are backing up this server as the information can be used to help map your network, as you can see from the above listing.

The allow-query entry tells the name server which hosts are allowed to query for individual records in this domain. For most domain names run by the server this entry

will be left off (you want people to be able to query your name server for domain your control) but for the loopback adapter only your local network, or just the local system, should be able to get information.

```
zone "localhost" {
      type master;
      file "master/db.loopback";
      allow-transfer {
            127.0.0.1;
      };
      allow-query {
            127.0.0.1;
            };
};
```

The above section is the forward lookup record for the loopback adapter. You can save the file and exit from your editor. Next open the /etc/resolv.conf file and add the loopback address (127.0.0.1) as the first "nameserver" on the list so the system will check the local system first for name lookups before checking your external server.

8.4 Installing Sendmail

We will be using the –current tree sendmail to make sure that we have all current security fixes in place. Add the user and group sendmail will be using to the password file with the following command:

```
groupadd –g 43 smmsp
useradd –g 43 –c "Sendmail Psudo-User" –d /var/mail –s \
      /sbin/nologin –u 43 smmsp
```

The `useradd` command will lock the password by default.

As covered earlier in the paper, to install this package under NetBSD first change to its directory /usr/pkgsrc/mail/sendmail-current. Next you will tell the system to fetch the required software with `` `make fetch-list | sh` ``

Once all of the files have been downloaded you should run `` `make` `` to build the software package.

If you see an error message that looks like:

```
# make
===> security issue found, build broken due to lack of useradd
```

This can be fixed be opining the file called "`Makefile`" and commenting out (with a hash mark) the line that starts "`IGNORE`".

At the successful completion of the build you should run `` `make install` `` to have the system copy the file to their correct locations. You will see a notice that if you want the new copy of sendmail to become your main transport agent you have to install a new mailer.conf file. Do that with the following command:

```
ln –fs /usr/pkg/etc/mailer.conf.sendmail /etc/mailer.conf
```

Clean up the sendmail package and its dependences with the `make distclean` command.

Now edit the /etc/rc.conf file and add the following line to the end of the file:

```
sendmail=YES            sendmail_flags="-bd –q30m"
```

8.5 Configuring Sendmail

The primary mail server must be configured to use the Cyrus IMAP server as its local delivery agent. Change to the /usr/pkg/share/sendmail/cf directory. In that directory copy the cyrusproto.mc file to mail_server.mc. We are going to create a mixture of this file and the one that can be found in /usr/share/sendmail/cf/netbsd-proto.mc (notice this directory is the main share directory, no the pkg/share directory). You should review both licenses listed at the top of those two files.

Open the mail_server.mc file you created in your text editor and make the additions listed below in bold (when you are done your configuration should look like what is listed here). The listing below omits the license for brevity, but both licenses can be found at the end of this document in appendix B.

```
divert(0)dnl
include(`../m4/cf.m4')
VERSIONID(`$Id: cyrusproto.mc,v 8.7 1999/09/07 14:57:10 ca Exp $')
OSTYPE(bsd4.4)dnl
define(`confBIND_OPTS',`-DNSRCH -DEFNAMES')
define(`confTO_IDENT', `0')
define(`CYRUS_MAILER_FLAGS',`SA5@!/')
define(`CYRUS_BB_MAILER_FLAGS',`S')
define(`MeToo',`m');
define(`CheckpointInterval',`5')
define(`confLOCAL_MAILER', `cyrus')
define(`confTRUSTED_USER', `curys')
FEATURE(genericstable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`genericstable')
FEATURE(mailertable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`mailertable')
FEATURE(virtusertable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`virtusertable')
FEATURE(domaintable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`domaintable')
FEATURE(access_db,DATABASE_MAP_TYPE` -T<TMPF> -o 'MAIL_SETTINGS_DIR`access')
FEATURE(`redirect')
FEATURE(`use_cw_file')
FEATURE(`nocanonify')
FEATURE(`always_add_domain')
MAILER(`local')
MAILER(`smtp')
MAILER(`cyrus')

MAILER_DEFINITIONS
Mcyrus,         P=[IPC], F=lsDFMnqA@/:|SmXz, E=\r\n,
                S=EnvFromL, R=EnvToL/HdrToL, T=DNS/RFC822/X-Unix,
                A=FILE /var/imap/socket/lmtp

LOCAL_RULE_0
Rbb + $+ < @ $=w . >   $#cyrusbb $: $1
```

NOTE: it is a tab, not spaces preceding the sections of the "MAILER_DEFINITIONS" and between the greater than symbol and the $#cyrusbb.

Once the file has been modified, save it and exit from the editor. You then need to generate the sendmail configuration file using the M4 macro language tool with the following command:

```
m4 mail_server.mc > mail_server.cf
```

It may generate an error which refers to the access_db line and a –T option. This can safely be ignored. Copy the mail_server.cf file to the /etc/mail directory and rename it to sendmail.cf, then change to that directory and create the number of files the configuration calls for as listed below:

```
cp mail_server.cf /etc/mail/sendmail.cf
cd /etc/mail
touch genericstable mailertable virtualusertable \
      domaintable access local-host-names
echo "bar.org" >> local-host-names
echo "foo.bar.org" >> local-host-names
```

The last two lines make sendmail aware that it should accept mail which is addressed to the domain "@bar.org" as well as to the to host "@foo.bar.org".

Next add the line "sendmail=YES" to the end of the /etc/rc.conf file with your editor.

Now you need to give the local system and your local network the ability to relay mail through the mail server. To do this open the /etc/mail/access file in your editor and add the following lines:

```
127.0.0.1          RELAY
10.0.0             RELAY
```

Save and exit from the file. The first entry tells sendmail to relay mail from the local loopback adapter (this is safe because the firewall will block packets coming from or going to the loopback adapter on the external network port). The second entry tells sendmail to relay mail for any system whos IP address starts with 10.0.0 (namely anything from 10.0.0.1 though 10.0.0.254).

Now you need to convert that text file in to a database for sendmail, you do that with the following command:

```
makemap hash /etc/mail/access < /etc/mail/access
```

This tells the `makemap` program to generate a hash database for the file /etc/mail/access (which it will name /etc/mail/access.db) out of the file /etc/mail/access.

8.6 Installing and Configuring Cyrus IMAP

As covered earlier in the paper, to install this package under NetBSD first change to its directory /usr/pkgsrc/mail/cyrus-imapd. Next you will tell the system to fetch the required software with `` `make fetch-list | sh` ``

Once all of the files have been downloaded, you should run `make` to build the software package. When the build completes you can run `make install` to install the software and then `make distclean` to remove the build files.

Now open the /etc/syslog.conf file in your editor. Add two lines to the bottom of the file that say the following:

```
local6.info                             /var/log/imapd.log
auth.info                               /var/log/imap-auth.log
```

These two lines will gather log data from the IMAP server and put it in to these files. If you have any difficulty getting the IMAP server running you might want to change the `.info` to `.debug` to get more information on what the server is trying to do.

Now we need to create the IMAP configuration and partition (mail spool) directories and set them to the correct permissions with the following commands (if you did not follow the adding a drive section you will also need to create the /var/spool/imap directory):

```
mkdir /var/imap
chown cyrus:mail /var/imap /var/spool/imap
chmod 0750 /var/imap /var/spool/imap
```

Change the password for the cyrus user with the `passwd` command. When the account was created by the install tool it was locked. You need a valid password to authenticate with to change system settings.

Next you have to become the cyrus user and run the `mkimap` tool to configure the system. Do so by following the steps below:

```
# su cyrus
$ cd /usr/pkg/cyrus/bin
$ ./mkimap
reading configure file...
i will configure directory /var/imap.
i saw partition /var/spool/imap.
you are storing sieve scripts in user's home directories.
done
creating /var/imap...
creating /var/spool/imap...
done
$ exit
#
```

Now open the /etc/services file in your editor and add the following lines (each should be placed in its correct position by its port number in the services file). Some of the entries will already exist and you only need to verify they exist.

```
pop3        110/tcp
imap        143/tcp
imsp        406/tcp
acap        674/tcp
imaps       993/tcp
pop3s       995/tcp
kpop        1109/tcp
sieve       2000/tcp
lmtp        2003/tcp
fud         4201/udp
```

Next edit the /etc/rc.conf file to add the line "sasl_pwcheck=YES" to the end. Change to the /etc/rc.d directory and copy the SASL startup script to it with the following command:

```
# cp /usr/pkg/etc/rc.d/sasl_pwcheck .
```

Verify that it has execute permissions (you can set them by doing a `chmod 0555 sasl_pwcheck`). Then add the entry "sasl_pwcheck=YES" to the end of the /etc/rc.conf file with your text editor.

Now use your editor to add the following lines just before the "echo '.'" line that ends the /etc/rc.local script.

```
echo –n " imapd"
/usr/pkg/cyrus/bin/master &
```

Save and exit from that file. Now open the /etc/group file and add the daemon user to the group mail (it should look like this when you are done "mail:*:6:daemon") as well as to the group smmsp. These changes are required so that sendmail can run the deliver tool to get mail in to the IMAP server.

Now that Cyrus has been installed, you can now test the `sendmail` deamon with the command (the sendmail configuration we setup requires the Cyrus tools to be installed for it to work):

```
# /usr/pkg/libexec/sendmail/sendmail –bd –q15m
```

If you are returned to a prompt with no errors, and you can see sendmail running in a `ps –aux` output your configuration is working. If you have any problems you should carefully check for typos in the sendmail configuration file you created. A common mistake is replacing a l (letter L) with a 1 (numeral one) in some of the options.

8.7 Configuring the Firewall for new Services

Now that the services have been installed the ports must be opened through the firewall. To do this open the /etc/ipf.conf file in your text editor and add the following lines to the end of the file (after the inbound ssh lines we already added).

```
# Allow inbound smtp connections
pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 25 \
      keep state

# Allow inbound DNS connections
#pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 53 \
      keep state
#pass in quick on le0 proto udp from any to 10.0.0.2/32 port = 53 \
      keep state

# Allow inbound pop3(s) connections
pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 110 \
      keep state
#pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 995 \
```

```
      keep state

# Allow inbound imap(s) connections
pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 143 \
      keep state
#pass in quick on le0 proto tcp from any to 10.0.0.2/32 port = 993 \
      keep state
```

The first section allows sendmail to accept inbound connections for the delivery of mail. The second (which is commented out) would allow BIND to accept DNS queries, which we would want if the server was acting as a name server for other hosts, but since it is just acting as a caching server for the mail server there is no reason to allow other systems to access it.

The third and fourth sections open the ports required for pop3 and imap. The entries for pop3s and imaps (SSL tunneled pop3 and imap) are included for future use, but are commented out now because the support has not been setup in the Cyrus server.

At this point you should restart the system and verify that `sendmail`, `master`, `pwcheck` and `named` show up in the process listing (they should all start automatically at this point).

8.8 Testing Mail Flow

Now that the services have been setup and the ports have been opened on the firewall, it is time to make sure mail can be accepted by the server, delivered to user mail boxes and is properly served by the IMAP server. We will do all of the testing for this on the local server.

8.8.1 Adding a User

As the user you defined as your administrator run the cyradm tool with the –u flag as follows to create a test mail box:

```
# su username
$ cyradm –u username localhost
IMAP Password: <username password>
localhost> cm user.testuser
localhost> quit
```

The “cm” command means create mailbox. You have to use the “user.” before the username as that is how the system expects the input. If you receive a “permission denied” message it means you are using an account that is not set as administrator in the /usr/pkg/etc/imapd.conf file.

Once that user has been created in the Cyrus server you have to create an account for authentication in the /etc/passwd file. Use the `useradd` command to create the user. The following command will work to create any user you want to have mail only access for:

```
useradd –c "User Full Name" –d /nologin –s /sbin/mailaccount \
      -p password username
```

You will see an error about the home directory not existing; that is ok. Under NetBSD if the home directory does not exist the account can't log in (and even if they are allowed to login they will be logged out by the `/sbin/mailaccount` shell). If you want your users to be able to use a .forward file to redirect mail you might want to set up up the accounts with this command:

```
useradd –m –c "User Full Name" –d /var/users/username –s \
      /sbin/mailaccount –p password username
```

This will create a home directory for the user under /var/users but the user will still be unable to login because of the `/sbin/mailaccount` shell. In either case be sure to set a password for the user (using the –p option shown above).

8.8.2 Testing sendmail

To test sendmail you need to use the telnet command to connect to the sendmail port on the local system. You then generate a test message by hand for the system to deliver. Follow the example below to see if your system is accepting mail correctly. This example assumes you are using testuser as the recipient. If you created a different mail account use that instead.

```
# telnet localhost 25
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mail1.workofstone.net ESMTP Sendmail 8.12.0.Beta10/8.12.0.Beta10; Tue, 9 Apr
2002 17:43:56 -0700 (PDT)
helo localhost
250 mail1.workofstone.net Hello localhost [127.0.0.1], pleased to meet you
mail from:root@localhost
250 2.1.0 root@localhost... Sender ok
rcpt to:schluntz@localhost
250 2.1.5 schluntz@localhost... Recipient ok
data
354 Enter mail, end with "." on a line by itself
This is a test message

Nice, huh?
.
250 2.0.0 g3A0huaQ000802 Message accepted for delivery
quit
221 2.0.0 mail1.workofstone.net closing connection
Connection closed by foreign host.
```

As you can see from the above listing, after connecting to the server, you have to identify yourself (with the helo command). You then tell the system who is sending the message and who the message is to be delivered to. After that you tell the system to prepare to accept the message (with the data command) and then you enter the message itself.

Once you are done entering the message, stop data mode by putting a dot "." on a line by itself. The system will then tell you whether or not it was able to deliver the message. Then, to exit from the mail server connection, issue the quit command.

8.8.3 Testing Cyrus IMAP

To test and make sure Cyrus IMAP is working and authenticating for pop3 and imap connections you only need to follow a couple of steps.

First, to test pop3 functionality you would use the `telnet` command as follows (assuming you created the user "testuser", if you created another user use that name instead):

```
# telnet localhost 110
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK foo.bar.org Cyrus POP3 v2.0.16 server ready
user testuser
+OK Name is a valid mailbox
pass <echoed password>
+OK Maildrop locked and ready
```

Once you receive the final "+OK" you know that the account is working. Use the "quit" command to exit from the session.

You can use the `imtest` command to test the functionality of the imap server. Follow the example below to test your server.

```
# imtest -m login -a testuser localhost
C: C01 CAPABILITY
S: * OK foo.bar.org Cyrus IMAP4 v2.0.16 server ready
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS ID
NO_ATOMIC_RENAME UNSELECT MULTIAPPEND SORT THREAD=ORDEREDSUBJECT THREAD=REFERENCES
IDLE X-NETSCAPE
S: C01 OK Completed
Password: <not echoed password>
C: L01 LOGIN testuser {8}
+ go ahead
C: <omitted>
L01 OK User logged in
Authenticated.
Security strength factor: 0
```

The "L0l OK" and "Authenticated" messages report that the account is working. You can use the ". logout" command to exit from the session.

If you received any error messages and you should verify that the `passwd` server is running and that the user has been setup in the `cyradm` tool and `useradd` tools.

Now that you have verified that you can access your account you should connect the server to your test system with the crossover cable (or connect them both to the hub) and configure your test system to be on the same network as the server.

Now open your imap client and configure it for the server. You should see the message in your inbox that you sent in the previous step "Testing Sendmail". If not, you should review your sendmail setup and try to send another message.

8.9 Configuring Integrit

To use Integrit you need three files. The first is the configuration file which will be stored as /etc/integrit.conf. The second is the master database which will be created in /tmp but will be stored on a CD mounted on /cdrom. The third is the changes database which we will keep in /var/spool/Integrit.

Open /etc/integrit.conf in your text editor and enter the following information:

```
root=/
known=/cdrom/int_known.cdb
current=/var/spool/integret/int_current.dbc

/etc/spwd.db                    SIMC
/etc/pwd.db                     SIMC
=/etc/disklables                SIMC
/etc/integret.conf              SIMC

/dev                            S

=/usr/people                    S
=/usr/pkgsrc                    S
=/usr/src                       S

/usr/man/whatis.db              SIMC
/usr/pkg/man/whatis.db          SIMC
/usr/local/man/whatis.db        SIMC
/usr/share/man/whatis.db        SIMC
/usr/man/whatis                 SIMC

=/var/users                     S
=/var/spool                     SILMC
/var/log                        SIMC
=/var/tmp                       SILMC
/var/run                        SIMC
/var/run/utmp                   SIMC

=/tmp                           SILMC
```

The listing above is a short configuration file which can be made more detailed, but it will work for the system with out modification. Next create the directory /var/spool/integrit with the mkdir command.

Now generate the initial database as shown below:

```
# /usr/local/sbin/integrit –C /etc/integrit.conf –u
integrit: ---- integrit, version 3.00 -----------------
integrit:                      output : human-readable
integrit:                   conf file : /etc/integrit.conf
integrit:                     known db : /cdrom/int_known.cdb
integrit:                  current db : /var/spool/integrit/int_current.cdb
integrit:                        root : /
integrit:                    do check : no
integrit:                   do update : yes
integrit: current-state db md5sum --------------
integrit: 588bb65a4ab11f7e152a61bca958864d  /var/spool/integrit/int_current.cdb
```

Move the /var/spool/integrit/int_current.cdb off the system to where you have your CD burner. Rename it to int_known.cdb and make a CD with it in on the root.

Once you have burned the int_known.cdb file to the CD, place it in the cdrom drive of the mail server and mount it with the `mount /cdrom` command. Then open the /etc/fstab file in your editor and remove the ",noauto" portion of the cdrom entry (make sure there is no trailing comma). This change will make the cdrom mount automatically when the system boots. You should now have a /cdrom/int_known.cdb for the system to use as its master database.

If you want to use a floppy for the main database you will have to reduce the data set (perhaps by excluding the /usr/share tree) as the database generated from the above configuration is almost 3mb in size.

Now open the /etc/daily.local file in your text editor and add the following lines:

```
echo "Running file integrity checker."
/usr/local/sbin/integrit –C /etc/integrit.conf –c
```

This will cause Integrit to be run nightly during the `daily` script and have its output e-mailed to the root account. You can run the above command any time you want to check the system. A review of the documentation that comes with Integrit is strongly recommended so you can learn its features and better tune the configuration file to your needs.

9 The Build: Step 4 – Hardening and Automation

Now that the server is set up for its task you need to review it again to make sure everything is running as you would expect it to. It is also time to configure your automation tools to help you keep the system running smoothly.

9.1 General Cleanup

One change you can make at this time is to upgrade the "securitylevel" entry in the /etc/rc.conf file from 1 to 2. The recommended setting is 2, but this may be tos strict for many administrators. It locks all file systems so that none can be mounted or un-mounted (after the system has booted and the /etc/fstab has been processed) and it also locks the firewall ruleset so it can not be changed. If any changes need to be made to the system it must be rebooted into a lower security level. To some sysadmins the downtime is acceptable, to others it is more important to be able to keep the system running at all times. This choice is up to you.

Another change you should make at this time is to comment out the Temporary line in the /etc/ipf.conf file so that general tcp connections can no longer be made from the system. A good rule of security is that the firewall should only allow out specific ports that you expect to be used. General rules should not be left in on production systems.

9.2 What Needs to be Running Now?

It is important to make sure that you know what software is running on your system, and have only the software that is required to be actually running.

Assuming that you have logged into the system on the console using your standard user account, and then `su -` to become root, you should see something very similar to this output when you run the `ps –auxwww` command.

```
# ps –auxwww
USER     PID %CPU %MEM  VSZ RSS TT STAT STARTED    TIME COMMAND
root       0  0.0  0.3    0 776 ?? DLs   4:48PM 0:00.01 (swapper)
root     184  0.0  0.1  552 140 p0 S     4:50PM 0:00.24 –csh
schluntz 182  0.0  0.1  572 160 p0 Is    4:50PM 0:00.13 –ksh
root     172  0.0  0.0  248 100 ?? Is    4:49PM 0:00.04 /usr/sbin/cron
root     165  0.0  0.0   36  84 ?? Is    4:49PM 0:00.01 /usr/pkg/sbin/pwcheck
root     156  0.0  0.0 1040 112 ?? Ss    4:49PM 0:00.24 sendmail: accepting
connections
root     153  0.0  0.0  912  76 ?? Is    4:49PM 0:00.03 /usr/sbin/sshd
cyrus    143  0.0  0.1  172 204 a- I     4:49PM 0:00.26 /usr/pkg/cyrus/bin/master
bind      84  0.0  0.1  644 124 ?? Ss    4:49PM 0:00.37 /usr/pkg/sbin/named -
/var/named/named.conf -d 1 -n 1 -u bind
root      80  0.0  0.1  584 212 a- S     4:49PM 0:00.20 /usr/sbin/ipmon –sn
root      75  0.0  0.1  112 140 ?? Ss    4:49PM 0:00.19 /usr/sbin/syslogd –s
root       4  0.0  0.3    0 776 ?? DL    4:48PM 0:00.23 (ioflush)
root       3  0.0  0.3    0 776 ?? DL    4:48PM 0:00.06 (reaper)
root       2  0.0  0.3    0 776 ?? DL    4:48PM 0:00.00 (pagedaemon)
root       1  0.0  0.0  364  92 ?? Is    4:48PM 0:00.03 init
root     208  0.0  0.0  452  80 p0 R+    5:10PM 0:00.00 ps -auxwww
```

The two shells (`-csh` and `–ksh`) the kernel processes `swapper`, `ioflush`, `reaper` and `pagedeamon` as well as the `init`, `cron`, `sshd`, `syslogd` and `ipmon` daemons were all covered earlier in the paper and are ok to show up in the process list.

The other processes we see at this point are `pwcheck` and `master` (the Cyrus IMAP services), `named` (our BIND server) and `sendmail` (the mail server).

If you are connecting over the network port and not the console you will also see a `getty` process. This is the daemon waiting for a connection on the console and it is also an expected program.

If there are other programs running at this time you should research what they are and why they are running before continuing.

Once you know what software is running on your system it is important to verify what ports are being listed on the network. Below is the output from the `` `netstat –an | more` `` command now that we have services running (the configuration of the firewall does not affect this output).

```
# netstat -an | more
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        State
tcp        0      0  *.2000                 *.*                    LISTEN
tcp        0      0  *.995                  *.*                    LISTEN
tcp        0      0  *.110                  *.*                    LISTEN
tcp        0      0  *.993                  *.*                    LISTEN
tcp        0      0  *.143                  *.*                    LISTEN
tcp        0      0  *.587                  *.*                    LISTEN
tcp        0      0  *.25                   *.*                    LISTEN
tcp        0      0  *.22                   *.*                    LISTEN
tcp        0      0  127.0.0.1.53           *.*                    LISTEN
tcp        0      0  10.0.0.110.53          *.*                    LISTEN
udp        0      0  *.53                   *.*
udp        0      0  127.0.0.1.53           *.*
udp        0      0  10.0.0.110.53          *.*
```

The above output has been cut off at the end of the "Active Internet connections" part of the listing. It is quite long as there are a number of local system connections listed as well, but we are not concerned with those at this time.

The first five entries (200, 995, 110, 993 and 143) are all owned by the Cyrus IMAP server. We added these to the services file earlier in the document so we know that these are ok.

The next two are 587 and 25. Both are owned by the `sendmail` daemon, port 25 is the standard SMTP communications port, and 587 is listed as a mail message submission port.

Port 22 is the Secure Shell port, and is owned by the `sshd` deamon.

Finally you see that port 53 is open for TCP as well as UDP in a number of different ways. This is how BIND handles listening to connections and is correct.

At this point you should not see anything else listening for connections. If there is you should track down what is listening and determine why it is running.

A number of the ports that are open above are not allowed through the firewall. This is ok. We will only allow the ports through the firewall that we require for the server to perform its function, and nothing extra.

9.3 NetBSD Daily, Weekly, Monthly and Security Scripts

NetBSD comes with a set of scripts (all located in /etc) named `daily`, `weekly` and `monthly`. Each runs from `cron` at set times (`daily` runs daily at 3:15am, `weekly` runs weekly on Sunday at 4:30am and `monthly` runs on the first day of each month at 5:30am). By default the monthly script is disabled with a hash mark in the crontab file.

These scripts do system maintenance as well as some basic security checks and e-mail the output to the root account. These scripts are too long to review in this document, but it is recommended that you review them on your own so you will know what they do (one example is that there is an /etc/security script that is run to do security checks, but it is launched from the daily script and not from crontab).

Whenever possible you never want to modify the daily, weekly, monthly or security scripts. Any changes you make to these files will be overwritten in system updates. Unless you need to change the specific behavior of an existing test, the best way to add new tests is to create a *.local file. Each of the automated tests looks for a file of its own name with the .local prefix for local changes to the setup. (So the daily script looks for a `daily.local` file).

9.4 Temporary File Systems

Files can fill up your temporary directories and eventually cause a denial of service attack by not having enough space to create temporary files the system requires. The /etc/daily script has a method of cleaning out the temporary directories, but due to the way it was written it can be susceptible to what is called a race condition where someone could link to important files outside of the temp directory and delete important system files. With some small changes to the script you can alleviate these problems and allow them to run.

Open the /etc/daily.local file with your editor and enter the following listing:

```
echo ""
echo "Removing scratch and junk files:"
if [ -d /tmp -a ! -h /tmp ]; then
       cd /tmp && {
       find . -type f -atime +3 -exec rm -f -- {} \;
       find . ! -name . -type d -mtime +1 -exec rmdir -- {} \; \
           >/dev/null 2>&1; }
fi

if [ -d /var/tmp -a ! -h /var/tmp ]; then
       cd /var/tmp && {
       find . ! -name . -type f -atime +7 -exec rm -f -- {} \;
       find . ! -name . -type d -mtime +1 -exec rmdir -- {} \; \
```

```
            >/dev/null 2>&1; }
fi
```

The only change made (besides uncommenting the lines) from the commands in the /etc/daily script is in the second block where the string "`-type f`" is added. This makes both statements above only delete regular files and ignore all other types (including pipes and symlinks). Because of this you will want to periodically check the temp directories for symlinks and other out of place files and remove them by hand.

If there is a symlink inside of a directory, the commands above to remove the directories will not work. This is ok as you will catch this when you review the temp directories by hand (you don't want to force delete the directories because of a possible race condition inside).

9.5 Set User ID and Set Group ID Files

The NetBSD security script (which is run from the daily script) does a very good job of finding SUID/SGID files and reporting the addition and deletion of them in the daily report. For a system that does not allow user logins this (and the use of Integrit) is usually enough, though security conscious administrators should review the full listing of SUID/SGID files for ones that could be removed or have the SUID/SGID bit removed.

9.6 World Writeable Files and Directories

An important thing to check before your system goes live is that there are no world writeable files, and that none are created on the system. You can do this with the following command:

```
# find / -perm o+w  -print
```

You will want to run that check now to make sure the system does not have any world writeable files. Create the /etc/security.local file and enter the following listing:

```
# Display any changes in world-writeable files.
#
if checkyesno check_worldwrite; then
        > $ERR
        (find / -perm o+w –print | xargs -0 ls -ldgTq | \
        sort +9 > $LIST) 2> $OUTPUT

        # Display any errors that occurred during system file walk.
        if [ -s $OUTPUT ] ; then
                printf "World Writable find errors:\n" >> $ERR
                cat $OUTPUT >> $ERR
                printf "\n" >> $ERR
        fi

        # Display any changes in the WR file list.
        egrep -v '^[bc]' $LIST > $TMP1
        if [ -s $TMP1 ] ; then
```

```
                CUR=/var/backups/worldwrite.curren
                BACK=/var/backups/worldwrite.backup
                if [ -s $CUR ] ; then

                        if cmp -s $CUR $TMP1 ; then
                                :
                        else
                                > $TMP2
                                join -110 -210 -v2 $CUR $TMP1 > $OUTPUT
                                if [ -s $OUTPUT ] ; then
                                        printf "World Write \
additions:\n" >> $ERR
                                        tee -a $TMP2 < $OUTPUT >> $ERR
                                        printf "\n" >> $ERR
                                fi
                                join -110 -210 -v1 $CUR $TMP1 > $OUTPUT
                                if [ -s $OUTPUT ] ; then
                                        printf "World Write \
deletions:\n" >> $ERR
                                        tee -a $TMP2 < $OUTPUT >> $ERR
                                        printf "\n" >> $ERR
                                fi

                                sort -k10 $TMP2 $CUR $TMP1 | \
                                    sed -e 's/[  ][      ]*/ /g' | \
                                    uniq -u > $OUTPUT
                                if [ -s $OUTPUT ] ; then
                                        printf "World Write
changes:\n"\ >> $ERR
                                        column -t $OUTPUT >> $ERR
                                        printf "\n" >> $ERR
                                fi

                                cp $CUR $BACK
                                cp $TMP1 $CUR
                        fi
                else
                        printf "World Write additions:\n" >> $ERR
                        column -t $TMP1 >> $ERR
                        printf "\n" >> $ERR
                        cp $TMP1 $CUR
                fi
        fi
fi
```

The above listing is based off of the portion of the /etc/security script which checks for changes in SUID and SGID files (explained above) and will notify you any time a world writable file is added or removed from the system.

Now add the entry “check_worldwrite=YES” to the /etc/security.conf file. This section will be executed during the run of the `security` script.

9.7 Keeping Your Applications Up-to-date

The NetBSD packages collection has a number of useful tools that can be used to automatically check for updates on your installed packages. Create the file /etc/weekly.local with your editor and enter the following listing:

```
echo “Updating package collection from network.”
/usr/sbin/sup –s

echo “Checking for updates to existing packages.”
/usr/pkg/bin/lintpkgsrc –i
```

The first command does a silent update from the packages collection on the NetBSD server to update any packages that are in the /usr/pkgsrc tree. The `lintpkgsrc` command checks the versions of installed packages against the version available in the /usr/pkgsrc tree and reports any differences.

The output from this check will be e-mailed to root during the weekly system audit. Some people may want to do this daily, but it is considered impolite to have `sup` updating the trees against the NetBSD main server every day. The author recommends that you let it run automatically once a week, and then run it by hand if you know of a reason to update sooner.

10 Day to Day Operation

Everything takes work, and maintaining servers is no exception to that rule. Besides keeping an eye on security issues and software patches you need to keep an eye on your systems themselves.

10.1 Manual Maintenance

Earlier in the document a script was created that cleaned all regular files and directories from /tmp and /var/tmp. This script skipped symlinks and directories that contain them. The sysadmin for the system will need to check these directories on a regular basis to make sure they are clean.

The sysadmin will also be responsible for keeping the Integrit database up to date. More information on that can be found in the documentation that comes with the Integrit source code.

10.2 Reading Your Reports

You will get between 8 and 9 e-mails a week from your server. This may not seem like a lot but after you have been reading these e-mails (which will seem to be the same from week to week) you might decide to ignore the output from time to time when you are busy with other projects.

It is important to review these reports every day when they come in. The author knows a number of admins who spend the first half an hour of the day (with their first cup of coffee) looking over the reports for changes.

10.3 Reviewing Your Logs

Reviewing your logs can be a time consuming and tedious job, but it has to be done. After some practice you will become good at it. A method used by some of the authors collegues is to copy the recent logs to a temporary location (copy, not move) and then delete the records already reviewed (if you know you last looked at the log file at 9:10am the day before, you would go to an entry at that time in the log file and delete everything before that point).

Next you would start removing messages that you know about, things like the Cyrus checkpoint of its file “ctl_mboxlist[204]: checkpointing mboxlist” is just a standard message and can be ignored.

Once you have removed the entries from the log file you know you can ignore or you know the cause of you are left with the entries that need your attention (which can be a long list, or it can be nothing).

Using this method, the logfiles are quickly reduced to a more manageable size and you will be less likely to miss something important that was hidden amongst a number of standard messages.

Another method is to use one of the many tools available. A few quick searches on a search engine like Google (http://www.google.com) will reveal commercial and open source log analysis tools that you can use to help ease the burden of your logfile review.

The most important thing is that you do it. If you don't review your log files you can miss important system events, which can hurt your server in both the long and short term.

11 Testing Your Systems

Before you put your system into production you should run some tests to make sure it is acting the way you expect it to.

11.1 Mail Relaying and Storage

You should use a mail client (or do it by hand using telnet as demonstrated earlier in the document) to test your ability to sendmail mail both to local use accounts as well as to remote, or relayed, accounts. These tests should be done both from the network the server is expecting to be able to relay mail as well as from a network/IP address that it is supposed to deny the relay to the remote user (but still accept mail to the local user.)

You should then use a few different mail clients to check and manage the delivered mail. Use both pop3 and imap to be sure the client side is also working as expected.

11.2 nmap

The network mapping tool nmap is of great use in helping to find what ports are open on a system (as well as helping you to map the operating system it is running). It can be downloaded from its site http://www.insecure.org/nmap/. This document does not describe the installation of the product, it just gives a quick overview of its use.

A standard way to use nmap is to do a TCP SYN stealth scan along with operating system fingerprinting. It looks like this:

```
# nmap -sS -O  10.0.0.2

Starting nmap V. 2.54BETA25 ( www.insecure.org/nmap/ )
Warning:  OS detection will be MUCH less reliable because we did not find at least
1 open and 1 closed TCP port
Interesting ports on  (10.0.0.2):
(The 1112 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh
25/tcp     open        smtp
53/tcp     open        domain
110/tcp    open        pop3
143/tcp    open        imap

Remote operating system guess: NetBSD 1.5_ALPHA i386
Uptime 0.010 days (since Sun Apr 14 23:16:39 2002)

Nmap run completed -- 1 IP address (1 host up) scanned in 134 seconds
```

As you can see, the results are as we expect them with the services: SSH, smtp (sendmail), domain (BIND), pop3 (Cyrus) and imap (Cyrus) listening for connections. Notice also that it correctly identified the system as NetBSD and was close on the revision (1.5_ALPHA instead of 1.5.2).

If you have the inbound ping blocked in the firewall rules you will need to add the –P0 (zero) flag to tell the system to scan the server even though it can't ping it.

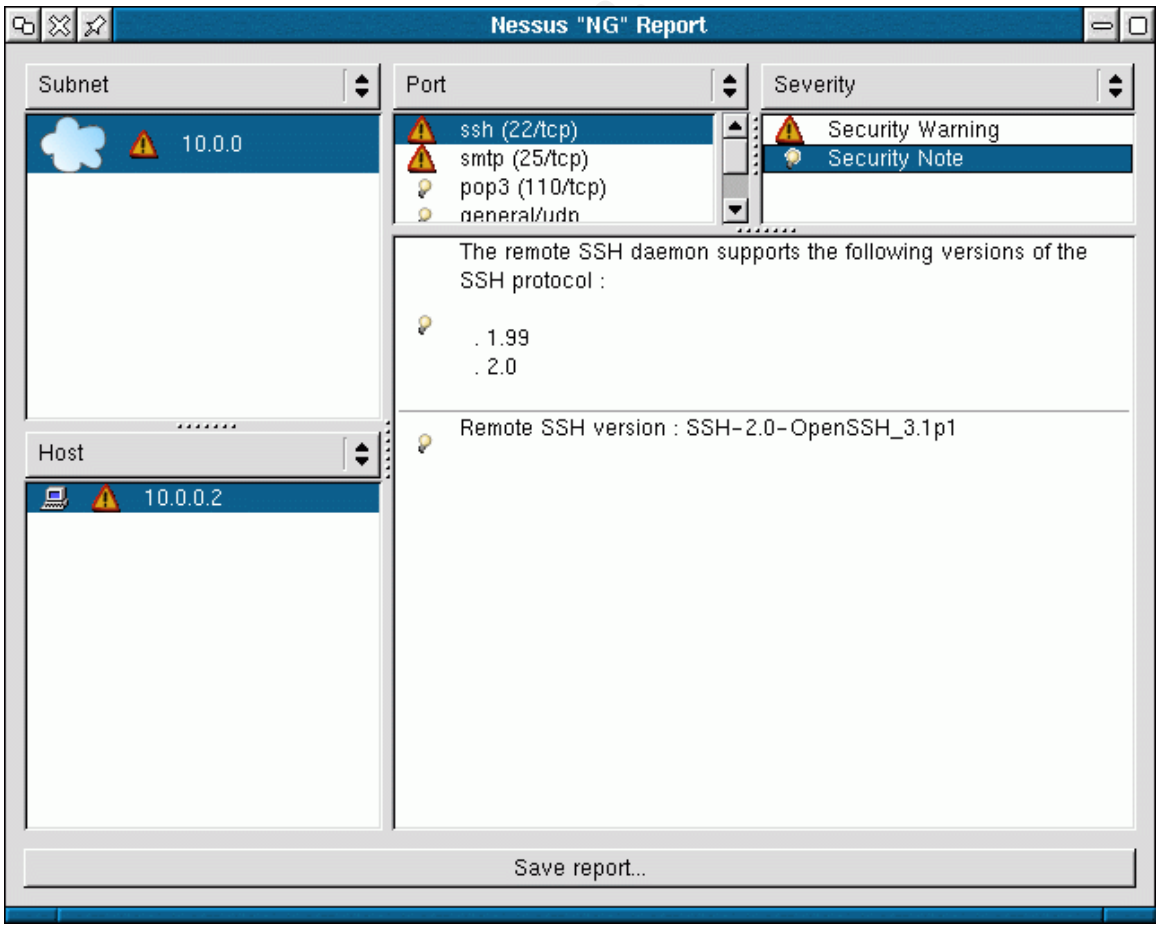
Running nmap by itself will list the options available.

11.3 Nessus

Nessus is a graphical remote security scanning tool which can be downloaded from http://www.nessus.org. It works with nmap to probe your system for open ports but goes even further. It can probe the ports to verify what service is running on them (some administrators run services in non-standard ports to try to fool attackers). It can also run known exploits against ports to see if you are vulnerable to that form of attack.

After the tool has been installed and you have started the server (with the `nessusd –D` command) you can launch the `nessus` client.

You will be presented with the Nessus client screen. Enter your user name and password (setup during the install process). You can configure the scan you would like. If you go to the "Target selection" tab and enter the IP address of the system and then click Scan the system will work for a while and present you with a screen similar to the one below.



For each of the services found on the system Nessus will report what was found about that service and what possible security issues there are with it. The author recommends that you check the Nessus scripts page located at http://www.nessus.org/scripts.html for new tests that can be used by Nessus against your systems.

The author strongly suggests the regular use of this tool to watch your systems.

12 Wrapping it up

Congratulations, you now have a working mail server. You should familiarize yourself with your new system's workings before placing it in service.

Now that the server has been built, you might want to review some additional topics that will assist you to extend the server to better suit your needs as well as some additional references you might find useful.

12.1 Additional Ideas to Explore

- PopAuther.pl: a script which allows to automatic management of relaying mail through sendmail from clients based on successful authentication of a pop3/imap session. A tool of this type is helpful if you have clients which need to be able to send and receive mail from off site while keeping the server from acting as an open relay. http://www.theonering.net/staff/corvar/software/sendmail.html
- Enabling SSL encrypted communications in Cyrus IMAP so that pop3 and imap authentication and communications are not in clear text. See the "SSL, TLS and OpenSSL" sections of the Cyrus IMAP documentation (located in the docs/install-configure.html file in the Cyrus archive).
- Mailing List Manager. Some sites require a list management tool for company and client communications. The author recommends looking into Mailman, the GNU Mailing List Manager. More information can be found at the web site http://www.gnu.org/software/mailman/mailman.html.
- Configuring the Network Time Protocol (NTP) so that your mail system's time will stay synchronized with a known good time source.

12.2 Good References to Investigate

- "Top 50 Security Tools". In mid 2000 the people who run Insecure.Org surveyed the 1200 members of their nmap-hackers[28] mailing list in order to find out what their favorite security tools were at that time. They have made the information available at their web site at http://www.insecure.org/tools.html.
- The NetBSD tech-security mailing list (signup at http://www.netbsd.org/cgi-bin/subscribe_list.pl?list=tech-security) to help stay up-to-date on security issues with NetBSD.

[28] Information on the Insecure.Org nmap-hackers mailing list can be found at http://lists.insecure.org/#nmap-hackers.

Appendix A: Bibliography

Albitz, P. & Liu, C. (2001). "DNS and BIND" (4th Edition). Sebastopol, CA.: O'Reilly & Associates

Strebe, M. & Perkins, C. (2002). "Firewalls 24seven" (2nd Edition). Alameda, CA: Sybex Inc

Mullet, D. & Mullet, K. (2000). "Managing IMAP". Sebastopol, CA: O'Reilly & Associates

Costales, B. & Allman, E. (1997) "sendmail" (2nd Edition). Sebastopol, CA: O'Reilly & Associate

Appendix B: NetBSD Sendmail Prototype File

/usr/share/sendmail/cf/netbsd-proto.mc

```
# $NetBSD: netbsd-proto.mc,v 1.4.2.3 2001/03/09 18:02:13 he Exp $

divert(-1)
#
# Copyright (c) 1994 Adam Glass
# Copyright (c) 1983 Eric P. Allman
# Copyright (c) 1988, 1993
#       The Regents of the University of California.  All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in the
#    documentation and/or other materials provided with the distribution.
# 3. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#       This product includes software developed by the University of
#       California, Berkeley and its contributors.
# 4. Neither the name of the University nor the names of its contributors
#    may be used to endorse or promote products derived from this software
#    without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
# SUCH DAMAGE.
#

#
#  This is the prototype file for a configuration that supports SMTP
#  connections via TCP and some commonly required features.
#

include(`../m4/cf.m4')
VERSIONID(`@(#)netbsd-proto.mc  $Revision: 1.4.2.3 $')
OSTYPE(bsd4.4)dnl
MAILER(local)dnl
MAILER(smtp)dnl
define(`confAUTO_REBUILD', True)dnl
FEATURE(genericstable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`genericstable')
FEATURE(mailertable,  DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`mailertable')
FEATURE(virtusertable,DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`virtusertable')
FEATURE(domaintable,  DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`domaintable')
FEATURE(access_db,    DATABASE_MAP_TYPE` -o 'MAIL_SETTINGS_DIR`access')
FEATURE(`redirect')

# Enable IPv6.  IPv6 is marked as optional so the configuration file
# can be used on IPV4-only kernel as well.
DAEMON_OPTIONS(`Family=inet, address=0.0.0.0, Name=MTA')dnl
DAEMON_OPTIONS(`Family=inet6, address=::, Name=MTA6, Modifiers=O')dnl
```

Appendix C: Cyrus Sendmail Prototype File

/usr/pkg/sendmail/cf/cyrusproto.mc

```
divert(-1)
#
#       (C) Copyright 1995 by Carnegie Mellon University
#
#                       All Rights Reserved
#
# Permission to use, copy, modify, and distribute this software and its
# documentation for any purpose and without fee is hereby granted,
# provided that the above copyright notice appear in all copies and that
# both that copyright notice and this permission notice appear in
# supporting documentation, and that the name of CMU not be
# used in advertising or publicity pertaining to distribution of the
# software without specific, written prior permission.
#
# CMU DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
# ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL
# CMU BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR
# ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
# WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
# ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
# SOFTWARE.

#
#       Contributed to Berkeley by John Gardiner Myers <jgm+@CMU.EDU>.
#
#       This sample mc file is for a site that uses the Cyrus IMAP server
#       exclusively for local mail.
#

divert(0)dnl
VERSIONID(`$Id: cyrusproto.mc,v 8.7 1999/09/07 14:57:10 ca Exp $')
define(`confBIND_OPTS',`-DNSRCH -DEFNAMES')
define(`confLOCAL_MAILER', `cyrus')
FEATURE(`nocanonify')
FEATURE(`always_add_domain')
MAILER(`local')
MAILER(`smtp')
MAILER(`cyrus')

LOCAL_RULE_0
Rbb + $+ < @ $=w . >   $#cyrusbb $: $1
```