Global Information Assurance Certification Paper

Securing Unix Step By Step: Building A Secure File Exchange Server

Matt Morton
Ball Corporation, Inc.

September 26, 2003

Table of Contents

Introduction:

This document describes the design and configuration of a Secure File Exchange server using FreeBSD 4-5STABLE and Secure Shell (SSH). My hope is that this server will provide a secure solution to a new and growing business need for dynamic, secure file exchange. Most companies today offer some sort of ftp service which can be used to allow the exchange of data amongst company employees and outside customers, partners, employees, and others. My company currently offers such a service in our anonymous ftp server. Since internet anonymous ftp servers are generally available to the whole world, this service is only suitable for the transfer of publicly available data. This service has historically been intended for small files which are only stored on the server temporarily for during transfer. FTP also, contains a rather major security problem; it passes passwords over the network in clear text! In addition to our "no security" anonymous ftp site, our company also hosts a highly secure file transfer site for very sensitive data. All data is and sessions to this server a highly controlled and encrypted. In order to use this site, partners must provide detailed information, sign contracts and meet many stringent requirements. All these requirements add up to a very slow service to implement which is not very flexible. These two options are the only file transfer options we offer today. Over the past few months it has been becoming increasingly clear that we need a solution capable of solving growing business needs which aren't addressed by the other two solutions. It is becoming more and more clear that we need a site with a moderate level of security, which allows us to respond quickly to a project's file transfer needs without compromising the security amongst the projects. This server will provide such a solution and allow for the sharing of project sensitive data, up to several gigabytes in size. This will not be an archive site, the data will be removed at regular intervals. This server will be explicitly offered for the sharing of data amongst project members (inside and outside of the company). We will use OpenSSH server software on our Server and a commercial SSH client (SSH for Workstations www.ssh.com) on the clients to securely enable simplistic, secure file transfer. The commercial SSH software includes support for sftp, including a GUI client for file transfer which is very similar to GUI ftp clients. Because of this familiarity, our users should be able to get up and running using this new service with very little training. In addition to the encryption of data and sessions provided by SSH, our installation will provide for multiple methods of authentications and authorization, including but not limited to the following:

Static IP address enforced by the firewall
Static IP address / DNS name enforced using TCPWrappers
Certificates via SSH
Name/password using SSH
Future support for Tokens (should it be required)

Users of the system will be provided project based group accounts with a single login and password. Outside users will also be required to provide a static IP addresses which will be enforced using TCPWrappers. Some of the other options above will be implemented as needed.

NOTE: The system being built for this paper is meant to be a representative model of the actual system. The actual system will be purchased and configured for production pending approval by a company security review based on this paper and the concepts presented. So the actual hardware used for this paper is only meant as an example, the production system will have much higher disk space, memory and CPU requirements. In addition, it will be company standard, H.P. hardware including typical requirements such as redundant disks, fans and power. In addition, the network environment during the build and production phases is meant to be as representative as possible of the production environments.

As always, the goal of system security is to provide the minimal services needed to fulfill the business objectives while making every effort to minimize risks. This guide provides the step-by-step instructions for removing extraneous software and unnecessary services and for properly configuring the operating system and applications required build a secure file exchange server which will meet the outlined business requirements.

System Configuration Overview:

Our mock system hardware consists of little more than a middle of the road Intel system which has an average amount of resources.

Hardware

System:	Mock System	Production System
System Type:	Gateway G6 350	H.P. Netserver
Processor:	Pentium II 350MHZ	Pentium III 1.4 GHZ
Memory	128MB	512MB
Storage	10GB (single disk)	72GB 36GB Mirrored (2 disks)
Video	3DFX Voodoo 3	HP SVGA
Ethernet	3Com 3C509CTXM 10/100 PCI	Dual 10/100 TX
Removable Media	Sony 24X CDROM	CDROM
Tape	HP SureStore DAT8	External HP Surestore DAT 24
Extras	N.A.	Redundant Power and fans

** NOTE: for detailed information about FreeBSD 4-5STABLE supported hardware, see:http://www.freebsd.org/releases/4.5R/hardware.html

Software

FreeBSD has been selected for the operating system because it is considered to be a very stable and secure operating system. It is also considered to have one of the most

advanced and efficient TCP/IP implementations available. It also has a very large and established developer base and wide availability of supported applications.

Software	Version	Function	Bundled with OS?
FreeBSD	4.5-STABLE	Unix BSD based Operating System	Y
OpenSSH	3.1p1 www.openssh.org	Secure (encrypted) telnet, ftp and more, Server	N – version 2.9 is included
Secure Shell for Workstations (commercial)	3.1 www.ssh.com	Secure (encrypted) telnet, ftp and more, Client	N
Logcheck	1.1.1	Check log file for anomalies and alert	Y
CVSUP	16.1f	FreeBSD source updates	Y
Comconsole	0.1	Configure serial console	Y

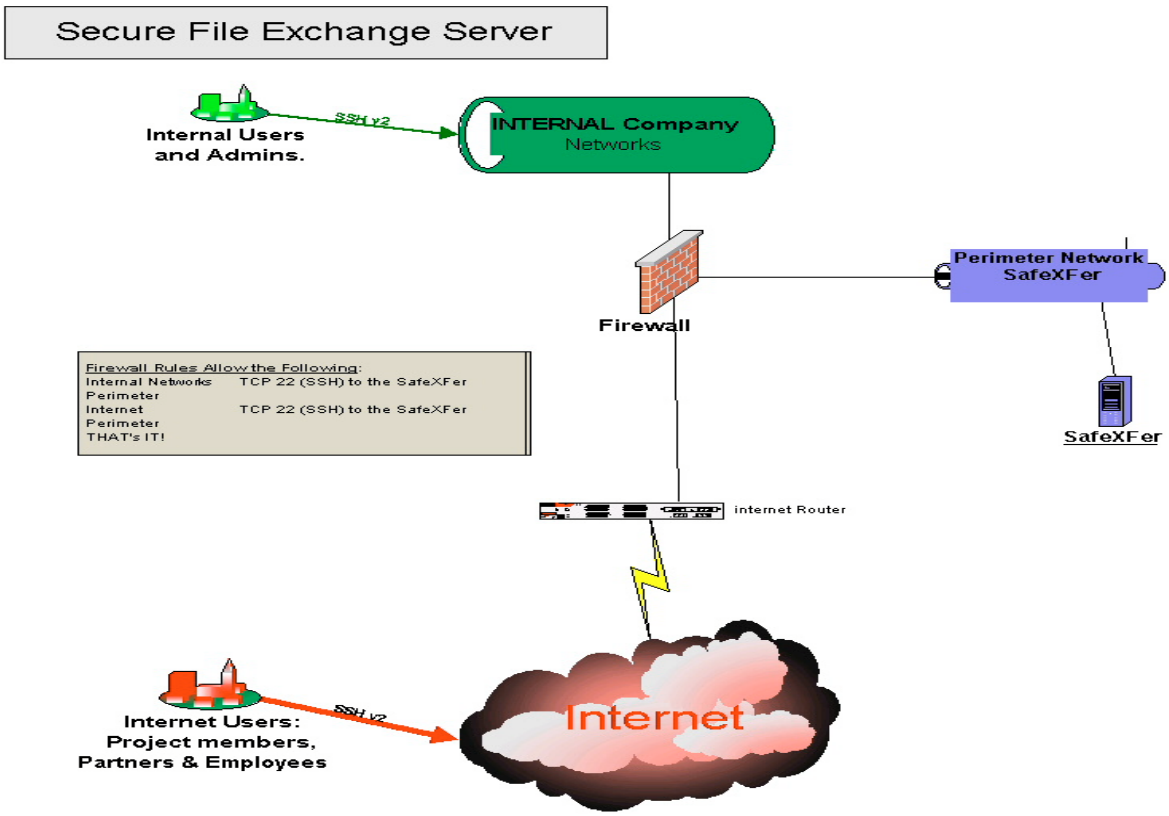
Physical Security

The physical security of the system is one of the most important aspects of overall security. On a Unix system access to the system console (and a little knowledge) are usually all that is required to obtain root access by force. The system we will be installing will be located in a central computer room which is locked at all times with key card readers for access. Access is strictly limited to network security personnel. All power to the room is conditioned and backed by a gas powered generator which takes over 10 minutes after a power failure. All systems are protected by a UPS with a minimum of 30 minutes of run time to cover the time needed for the generator to startup. The room is also equipped with monitoring equipment to track temperature and humidity. In addition, we will install Comconsole a program which will allow our server to run "headless" without a monitor or keyboard. Comconsole, allows us to use the PC's serial port for console access when needed. Since the configuration of a serial console is beyond the scope of this paper, it won't be covered in detail. For more information please see: http://www.freebsd.org/doc/en_US.ISO88591/books/handbook/serialconsole-setup.html

Network Environment

This system, if approved, will be installed in accordance with corporate standards for internet accessible hosts. The system will be isolated on a perimeter network protected by the company firewall. This perimeter network consists of a firewall routed VLAN. The only traffic allowed to reach this host will be TCP port 22 SSH. Although, both SSH version 1 and version 2 use TCP port 22, our server will be configured to only use SSH version 2. Version 1, is very old and has numerous, well exploited, vulnerabilities. It is the general security community's consensus that SSH version should no longer be

used. For more information see: http://www.ssh.com/products/ssh/advisories/deprecation.cfm . This is not to say that version 2 is free from vulnerabilities, many recent examples exist and this will be our most important application to keep patched and up-to-date. The firewall will be configured to allow SSH traffic from the internet to the perimeter network and from the internal networks to the perimeter network. No other traffic to the secure file exchange server will be allowed or accepted. SSH is considered a secure protocol which works similar to a virtual private network (VPN) to encrypt data and session information as it flows across the network. In addition to encrypted traffic, the system will run TCPWrappers with a default deny policy. This means that for any host that will need access to our server, we will need to add the static IP address of the system to the host.allow. Only hosts who's IP Addresses are contained in the host.allow file will be allowed access for SSH. TCPWrappers provides another layer of security and allows us to only accept that which is explicitly enabled. Outside users will only need to provide a static IP Address and project approval for access to this system. For those users that don't have a static IP Address, we will use client certificates with SSH, to authenticate their workstations. For more information on setting up ssh with public key authentication, see: http://www.ssh.com/products/ssh/win-administrator31/ssh2win-adminguide.pdf In addition to IP address, users connecting to the server will have to provide a user name and password. Since separate projects will need to share our server, we will configure group (project) accounts and use file system level security to make sure that each project only has access to its secured home directory and nothing else. Most people are familiar with GUI ftp clients and the SSH commercial client looks and feels very similar, so training should be minimal. The following is a network diagram depicting the production environment:



Risk Analysis:

During the build phase, our server will also be isolated on a firewall protected network. During this process we will use a secondary system with which to obtain software and patches and relay them to our server. This is the only way to ensure that our system isn't compromised during its most vulnerable periods before hardening. Once the server is built and secured, it will be located on an isolated perimeter network secured from all other networks by a company firewall. As I have stated before, the firewall will be configured to allow only TCP port 22 (SSH) from the internal network and the internet to our server. SSH version 1 will not be allowed on our server. Clients will be forced to use the newer and more secure version 2. Even though we are using version 2, we will need to be diligent at keeping up with patches to our system, most importantly, to fix any vulnerabilities in SSH. That is not to say we won't be concerned about securing the rest of the applications and services, these are also very important and will be addressed individually. But the SSH daemon is the only service which will be exposed to the internet, as such it should be our most likely avenues of attack. Any system which is accessible to the internet must be hardened even if it will also be protected by a firewall. Remember, security works best in layers so that if one defense mechanism should fail you still have others in place to protect the system. Just keeping up with patches and vulnerabilities in SSH is enough to keep anyone busy, a sampling of some of the recent CERT Advisories for SSH follows:

www.cert.org

Date Updated	Cert ID#	Summary
03/08/2002	VU#408419	**This bug can be exploited locally by an authenticated user logging into a vulnerable OpenSSH server or by a malicious SSH server attacking a vulnerable OpenSSH client**
03/05/2002	VU#786900	**SSH host key authentication can be bypassed when DNS is used to resolve localhost**
03/05/2002	VU#945216	**SSH CRC32 attack detection code contains remote integer overflow**
03/05/2002	VU#118892	**Older SSH clients do not allow users to disable X11 forwarding**
03/05/2002	VU#315308	**Weak CRC allows last block of IDEA-encrypted SSH packet to be changed without notice**
03/05/2002	**VU#665372**	**SSH connections using RC4 and password authentication can be replayed**

Because SSH is going to be central to the security of our system, we must be sure to download and build the most current stable release available. This requires obtaining the software directly from the www.openssh.org website in source format and compiling and installing it on the system. We can't risk a binary version which could have been tampered with before install. After install, we will need to be sure and keep up with the security advisories. Some good sources of information to help include:

The Cert Advisory mailing list:
http://www.cert.org/contact_cert/certmaillist.html
The Security Focus (formerly Bugtraq) mailing list:
http://www.securityfocus.com/cgi-bin/subscribe.pl

Summary of the Most Critical External Threats:

The primary external threats for this secure file exchange server are:
1. The sshd daemon that is exposed to the internet
2. Viruses and Trojans in transferred data
3. Denial of Service (DOS) attacks

1.) The sshd daemon could contain some programming problems or errors which might allow an outside attacker to find a hole into the system using a buffer overflow or similar attack. Because the SSH daemon runs as the root user, this is particularly dangerous as a successful exploit could mean full superuser access to our system. Keeping up on vulnerabilities and patches for SSH will provide the best defense against these types of exploits.

2.) All internal systems are running virus scanning software and all updates centrally pushed out as they become available. In addition, all internal file servers also contain antivirus scanning software. Outside users of this site must sign a release form stating that they will use antivirus software at all times and will strive to keep it current. Should this become a problem, virus scanning can be added to the server itself using a product that supports FreeBSD.

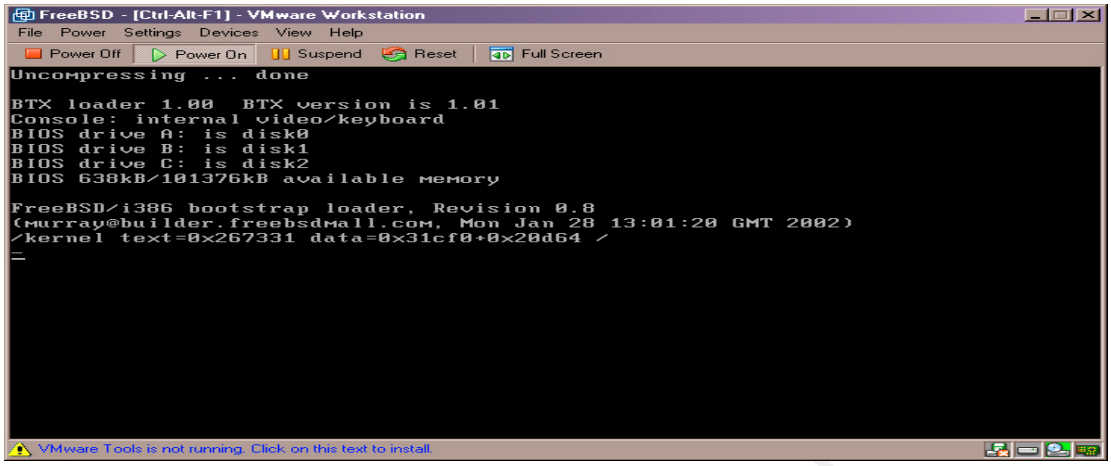
3.) While DOS attacks could be used to render our server useless, they are do not normally lead to a system compromise. Our border routers have been configured with countermeasures to decrease the success of these types of attacks. For more information, see: http://www.cisco.com/warp/public/707/newsflash.html

FreeBSD 4.5-STABLE Installation:

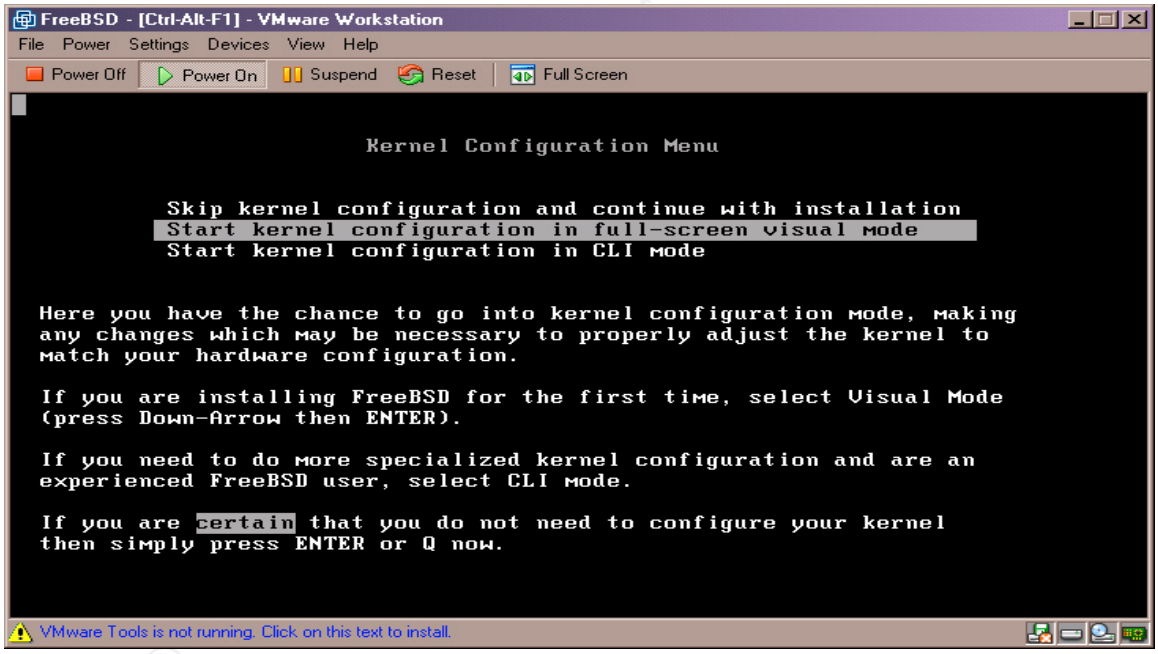
To begin the installation of FreeBSD 4.5-STABLE you must first obtain the software. 4.5 is the latest release of the operating system and the STABLE branch is the only one which should ever be used for a production server. CURRENT is the cutting edge software and should only be used for developers or those who are experts with FreeBSD. As with most opensource software, FreeBSD can be obtained in a number of different ways, see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mirrors.html. For this installation, I downloaded the ".iso" images (CDROM Images approx. 600MB each) and burned them to CDROMs using a CDRW drive in my computer.
The four CDs contain the following:

CD #	Contents	Notes:
1- install	Installation files	This is the only CD required for installation
2	Live file system Recovery CD	Critical for recovery when things go wrong
3	Extra software for FreeBSD	
4	**Extra software for FreeBSD**	

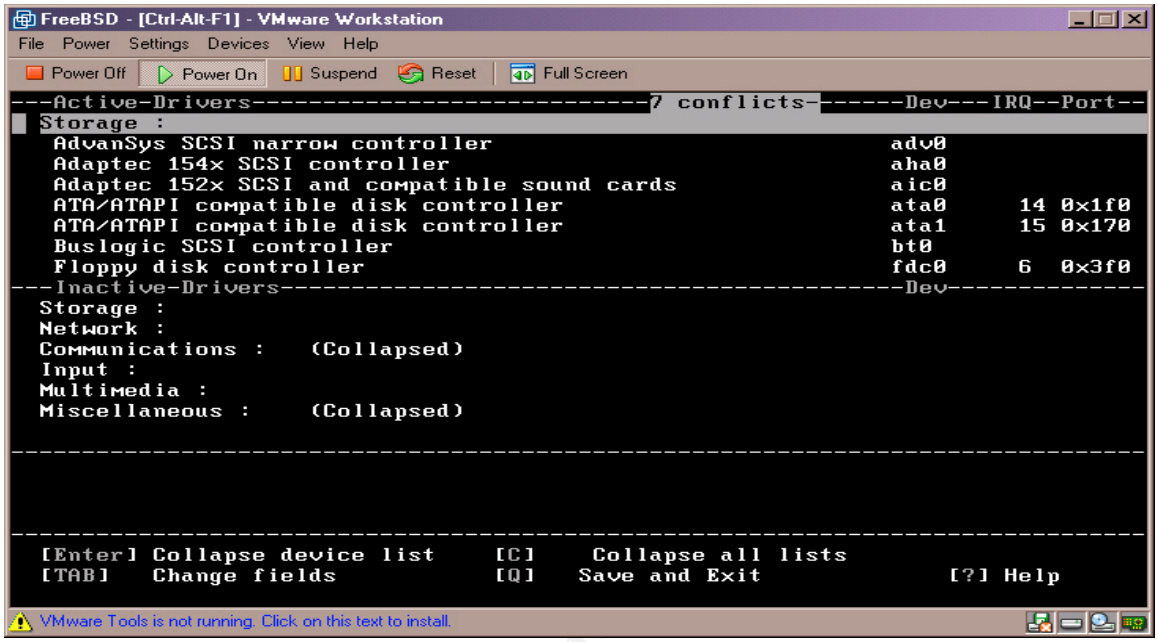
Once you have obtained the software, insert the install CD into your CDROM and boot your computer. You should see the following screen as the installation starts up. If your system doesn't automatically boot from the cdrom, you may need to check your computer's BIOS settings to ensure that the CDROM is configured as a boot option. If your system still won't boot off the CDROM (some older CDROM drives do not support autoboot) then you will need to create installation diskettes, see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-pre.html section 2.2.6.



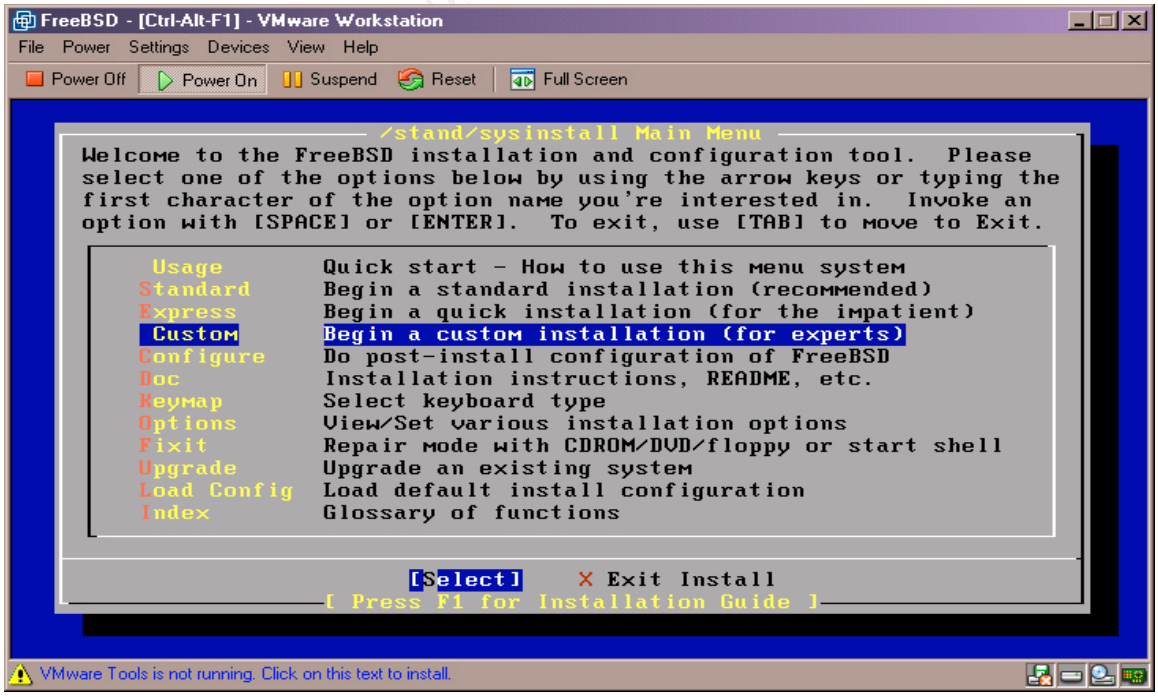
Once the installation gets going, you are presented with the kernel Configuration Menu, see below. Here you must decide if you need to customize the drivers to be included in your kernel. Since we are worried about all aspects of security, we don't want anything besides what is required to operate. Select, "Start kernel configuration in Full-screen visual mode" and press the enter key to continue.



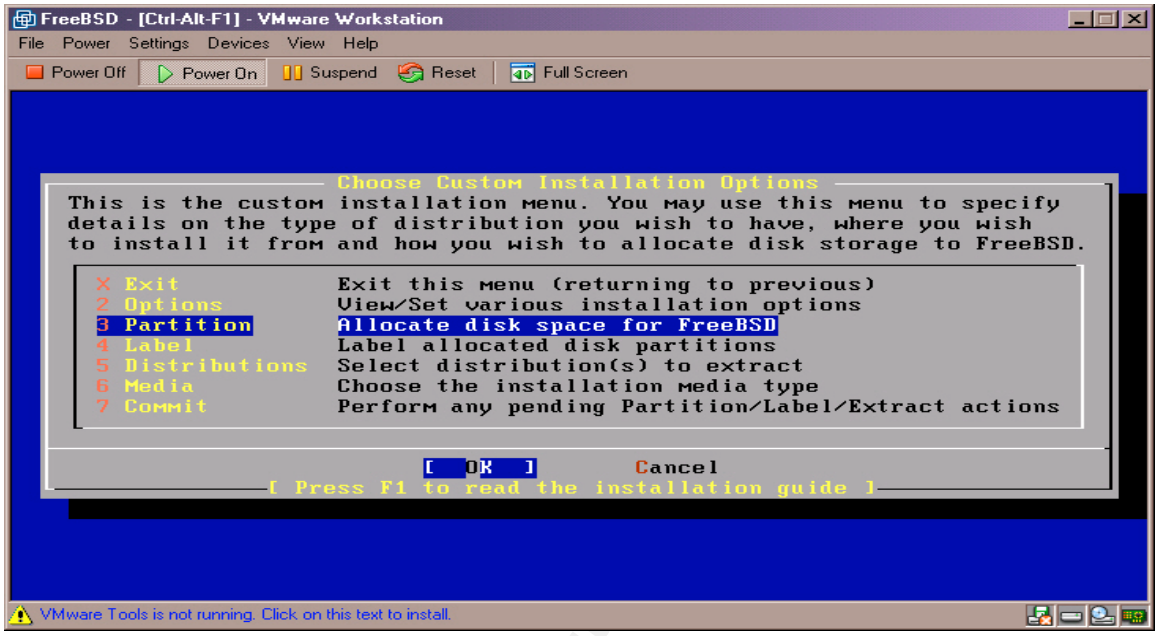
You should see the following screen appear listing the major categories of drivers which you can edit. Navigate the categories and use the delete key to remove un-needed drivers from the kernel configuration (those which don't represent the hardware in your system). When you are finished choose Q to save and continue.



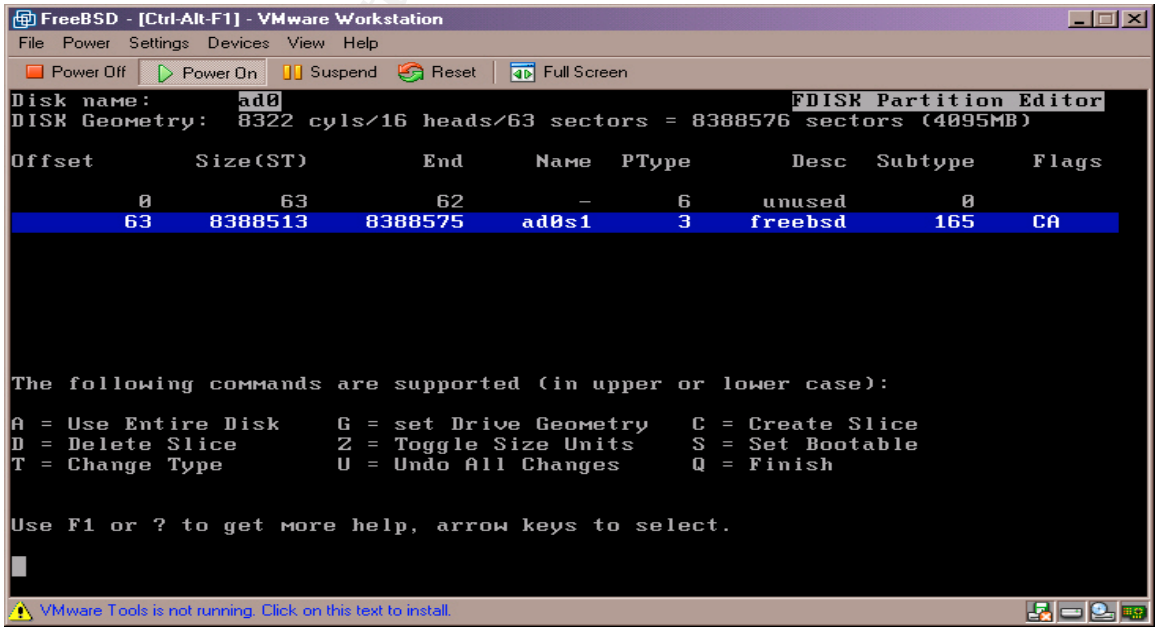
The main installation menu should appear listing all the categories of the FreeBSD installation program. The first choices in the listing are related to the type of install you want to perform. Since we want to control exactly what is installed on this system, we choose custom and then choose select to continue.



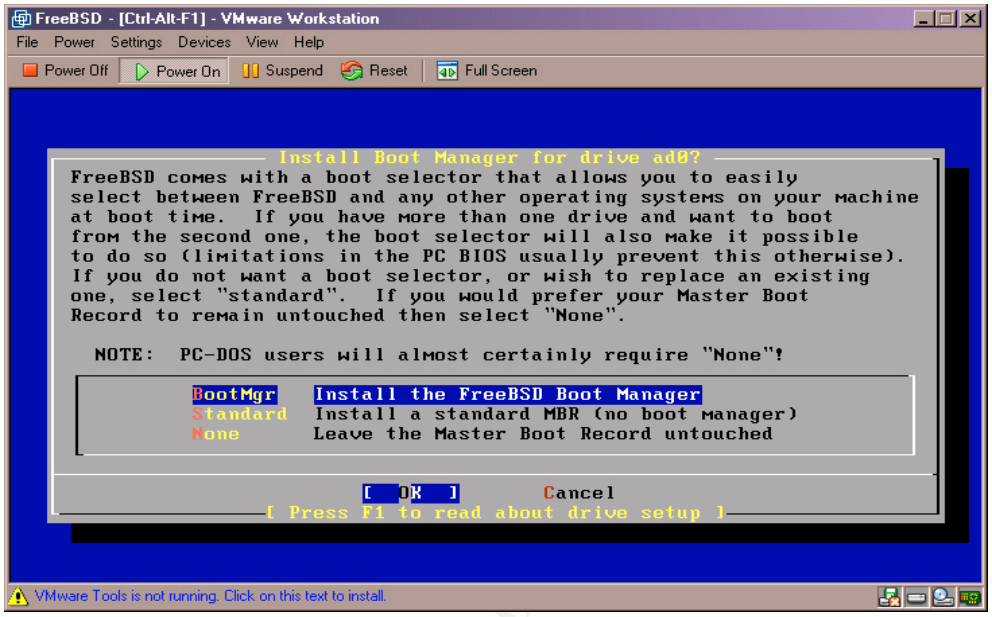
The Custom Installation Options screen should appear listing all the options for a custom install. The first selection, Options, allows the setting of various default installation settings. These defaults are fine for this installation so we will move on to the next option, Partition, where we will allocate disk space for FreeBSD.



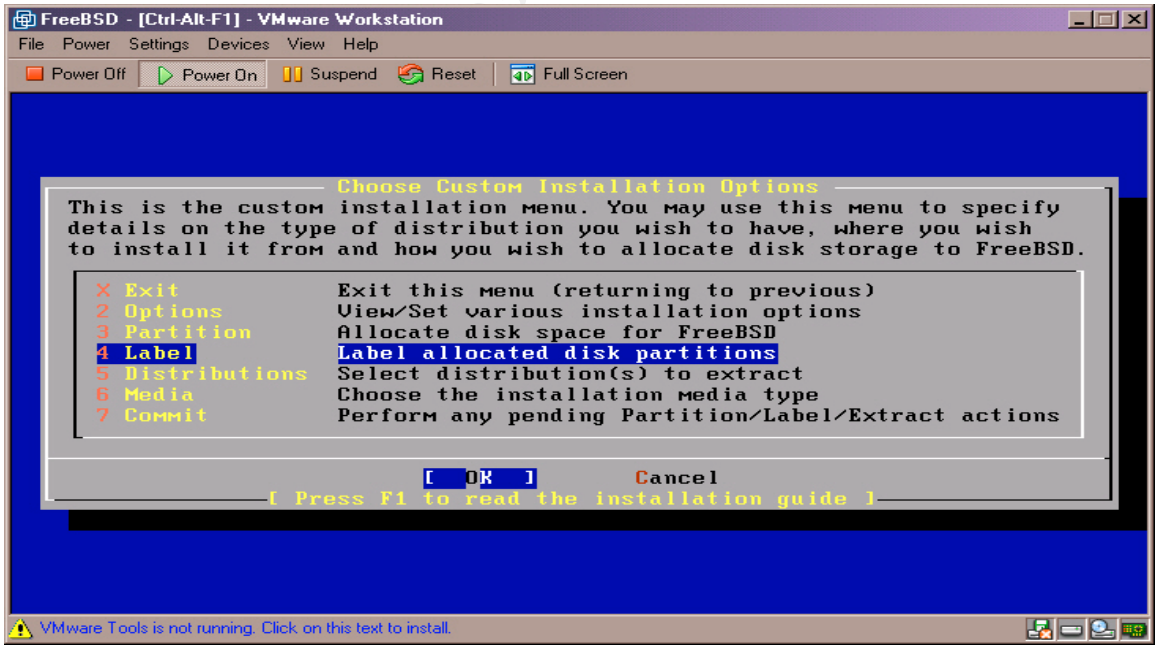
The FDISK Partition Editor screen appears listing the current partitions on your hard disks. At the top of the screen you see a listing of your disks and there attributes. If you have free space on your disks it should show up with a description of unused. Since we intend to use the entire hard disk for FreeBSD, we highlight our unused partition; choose A to use the entire disk and then select S to set the FreeBSD partition as bootable. Select Q to save settings and continue.



Next, we must decide on a boot manager for the drive we just configured. Since this is the only drive in our system, we choose BootMgr to install the FreeBSD boot manager on this disk. The other options or for multiple boot or multiple hard drive situations.



We are returned to the Custom installation Options menu, the next choice is Label. Select label to create file systems within our FreeBSD partition. Select OK to continue.



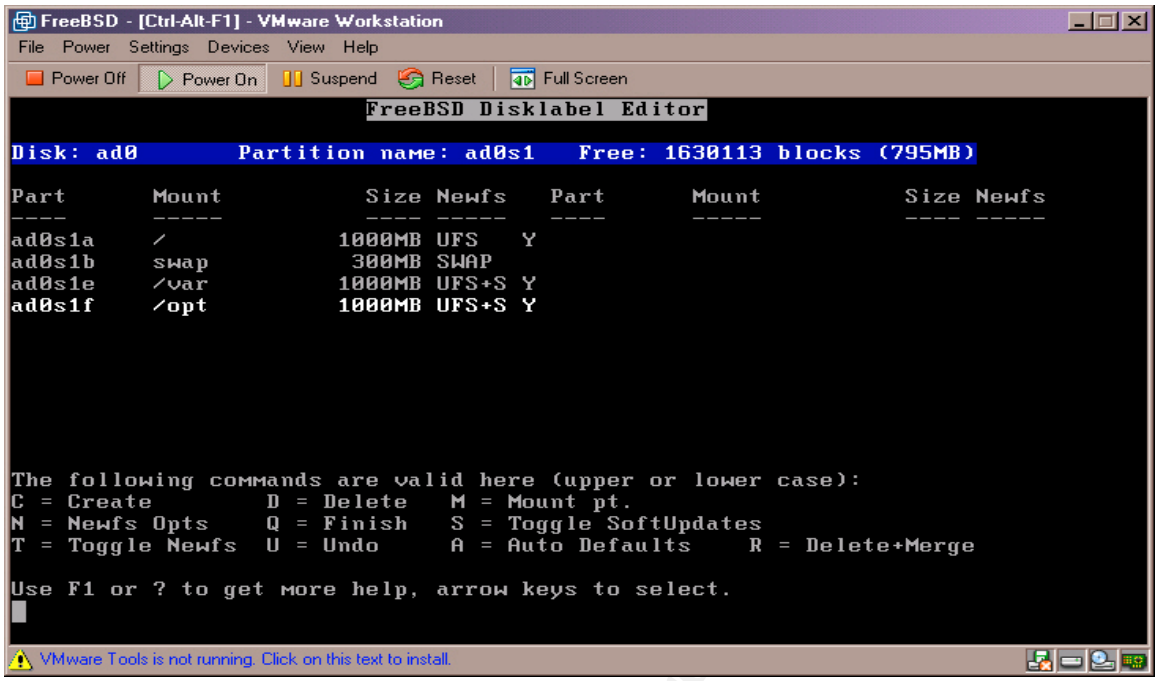
The next screen allows you to select the partition you want to configure and then create file systems within that partition. For our test system we used the following configuration:

File system	Size	Rationale
Root	3000MB	Plenty of space for the future
SWAP	600MB	Should be at least twice the installed memory, plan for adding memory in the future
/var	1000MB	Logging --separate so that it cannot fill our root partition
/opt	5400MB	File transfer – again separate for security and so that it cannot fill our root file system

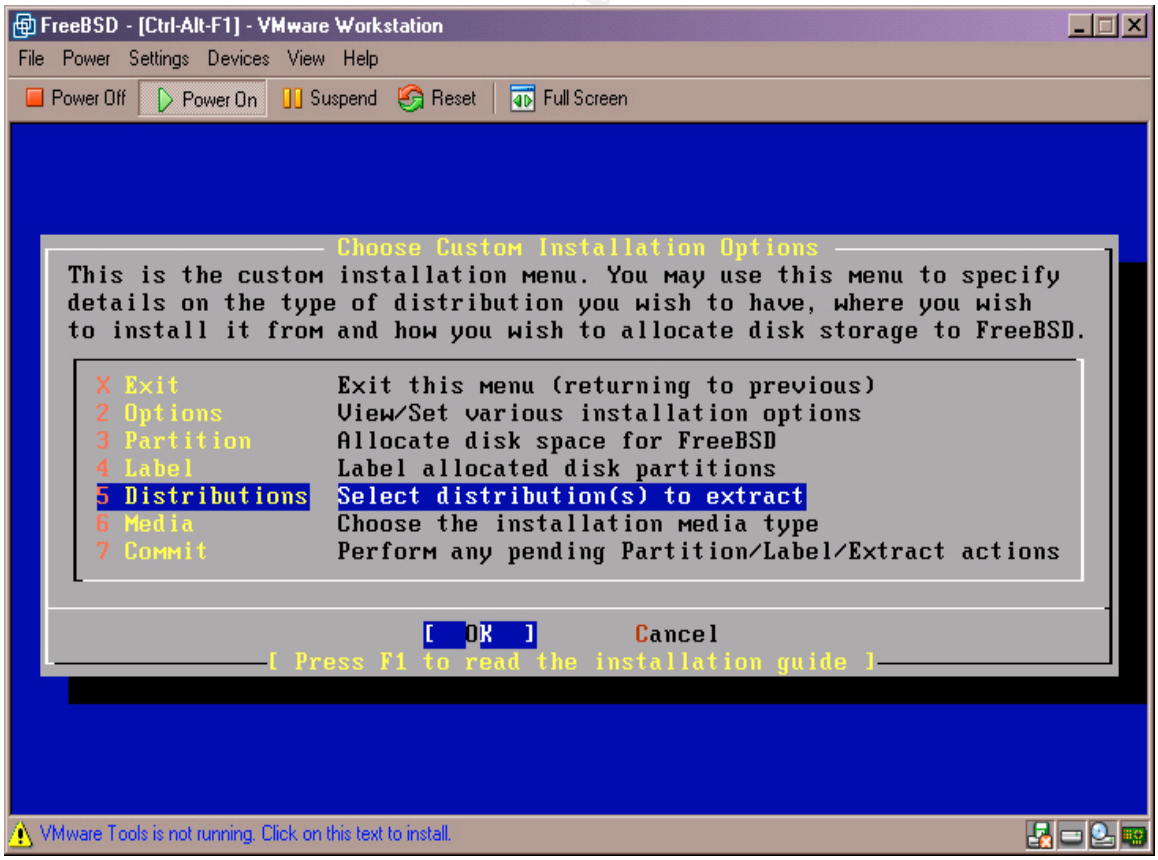
For the Production system we will use something closer to the following:

File system	Size	Rationale
Root	3000MB (MIRRORED)	Plenty of space for the future
SWAP	3000MB (MIRRORED)	Should be at least twice the installed memory, plan for adding memory in the future
/var	5000MB (MIRRORED)	Logging --separate so that it cannot fill our root partition
/opt	25000MB (MIRRORED)	File Exchange data repository--separate for security and so that it cannot fill our root file system

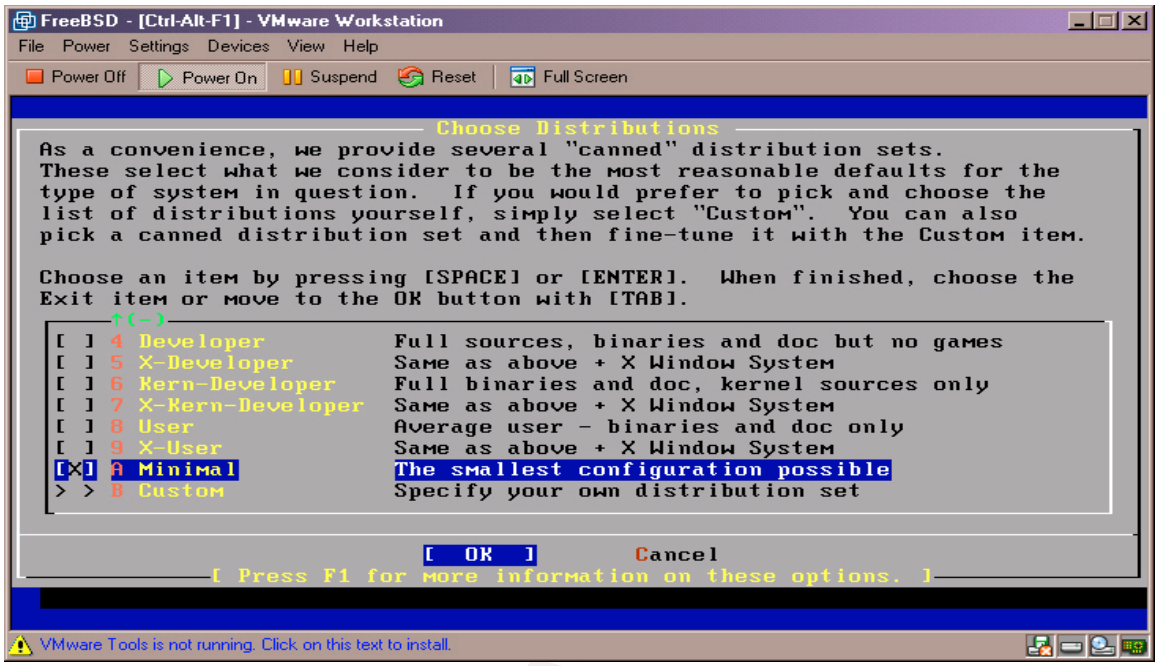
Once you have made your selections, choose Q to save your changes and continue.



Once again you are returned to the Custom Installation Options menu, the next option in the list is Distributions. Select this and then select OK to continue.



There are many different pre-packaged distributions (related packages of software) meant to support certain types of predefined system functions. Since we only want to load the very minimum software required (we don't need anything extra to provide vulnerabilities for our system) select minimal and then OK, to continue.

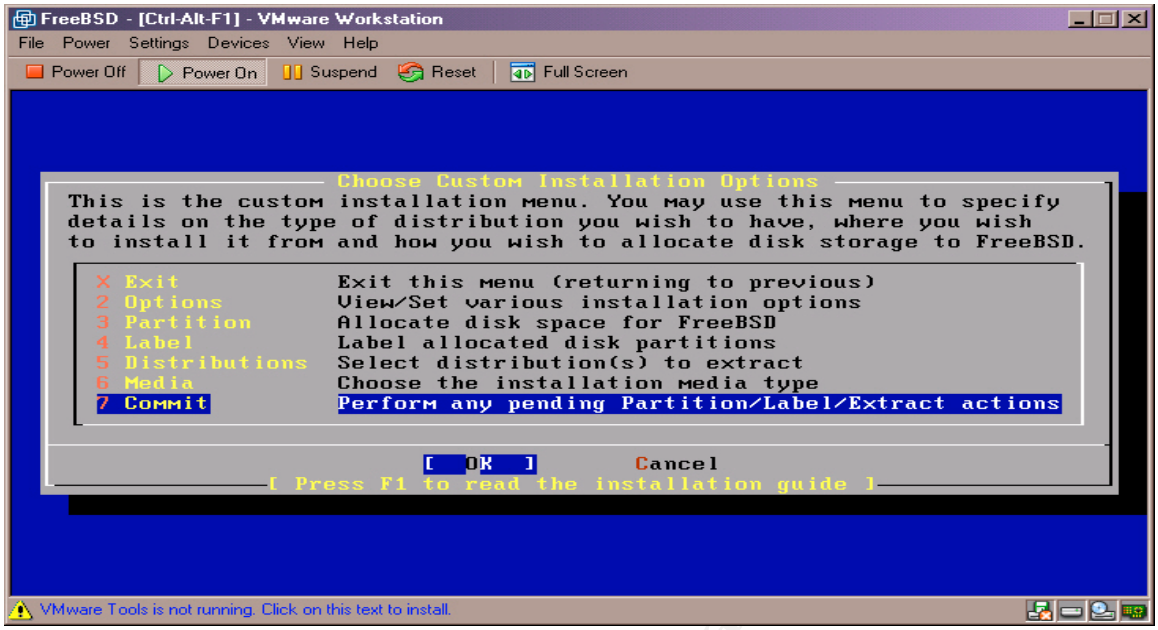


Once again we are returned to the Custom installation Options menu. The next selection is Media. This is used to choose the installation media type, select this option and then choose CDROM.
Next, we need to choose the last option in the list commit, to install the minimal software packages for FreeBSD.

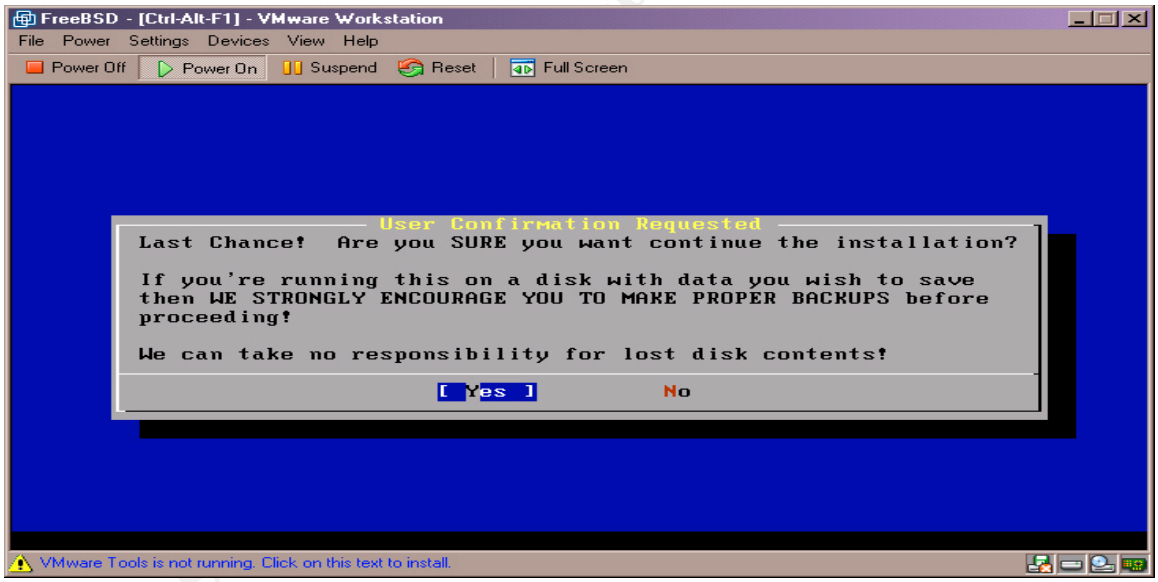
The FreeBSD Minimal Install includes the following software critical to our installation:

Name	*Version*	*Notes*
Openssh	2.9	Though required for initial connectivity, the version included is rarely the current version of the application. Since this software is very critical to the security of our system, we will eventually replace this software with the most current version during hardening
TcpWrappers	7.6	This is current

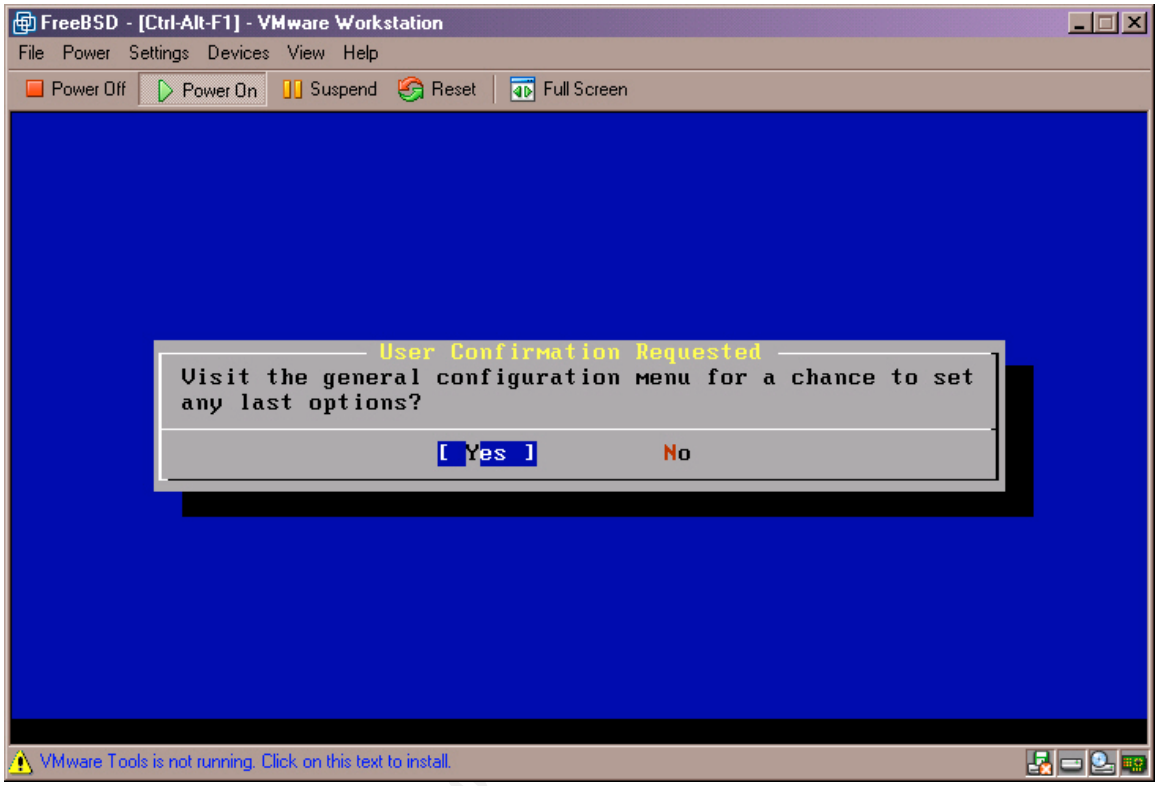
Select OK to continue.



The installation warns you that you are about to commit changes (write) to your hard disk. Any previous data will be overwritten. Choose Yes to continue.



After the installation finishes we have completed the install of the minimal FreeBSD system. The installation program presents you with the option to return to the Main Configuration menu for a chance to set any last options. There is quite a bit more customizing we need to do so select Yes, to return.

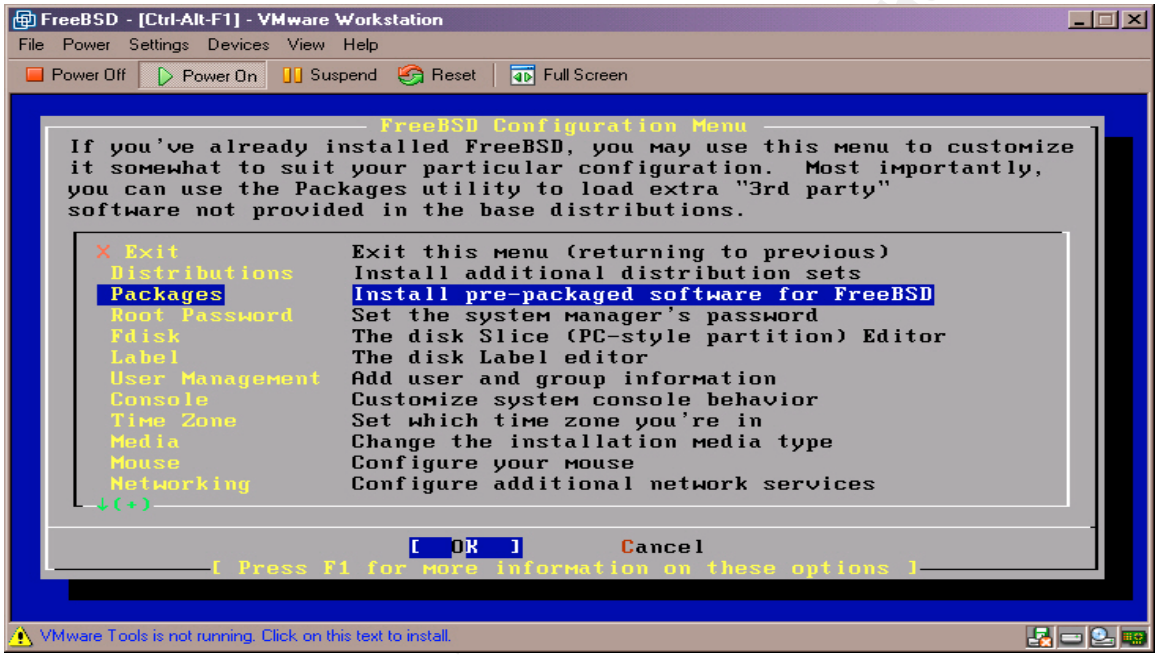


The Full Options for the FreeBSD 4.5 Configuration menu include the following:

FreeBSD Configuration Menu:	
X Exit	
Distributions	Install additional distribution sets
Packages	Install pre-packaged software for FreeBSD
Root Password	Set the system manager's password
Fdisk	The disk slice (PC style Partition) editor
Label	The disk label editor
User Management	Add user and group information
Consol	Customize the system console behavior
TimeZone	Set the system time zone
Media	Change the installation media
Mouse	Configure a mouse
Networking	Configure additional network services
Security	Select default system security profile
Startup	Configure system startup options
TTYs	Configure system ttys

Options XFree86 Desktop HTML Docs.	View / set various installation options Configure the XFree86 server Configure the XFree86 desktop Go to the HTML documentation menu (POST INSTALL)
Load KLD	Load KLD from floppy

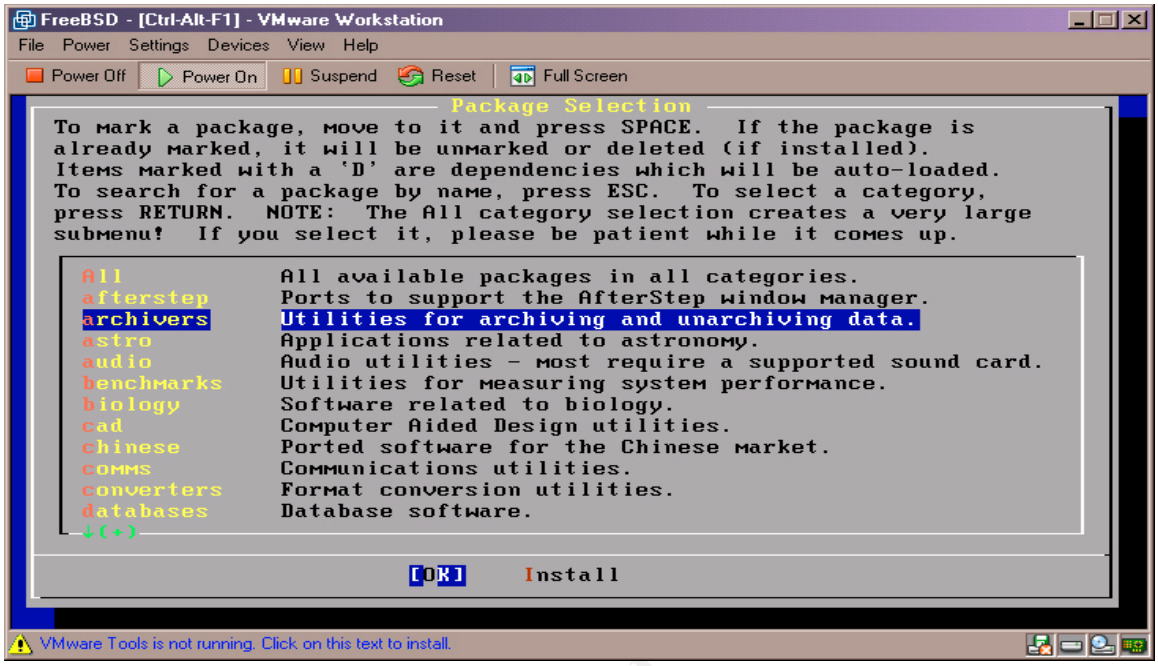
Select Packages from the menu and then choose OK to install additional software on the system.



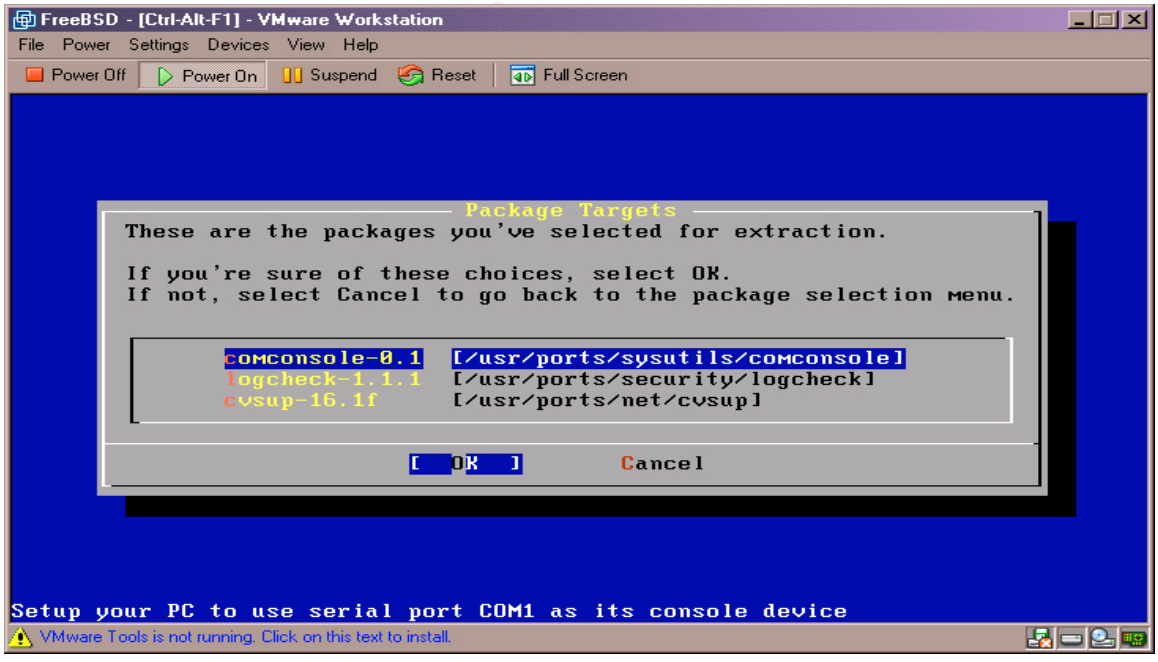
The FreeBSD Package categories are displayed, each containing a set of individual packages which can be installed. For this installation, we have added the following packages:

Category	Package	Purpose
Networking	Cvsup-16.1f	**Maintaining FreeBSD Source Code and Updates- This will not be covered in detail as we are running the latest stable code, for more information, see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html**
Security	Logcheck-1.1.1	**Log file auditing and alerting**
Sysutils	**ComConsole-0.1**	**For security reasons, this system will be installed "headless" (without a monitor or keyboard). This software will allow the use of a serial console as needed**

Once you have made your selections, choose Install to have them added to the system.



The Installation program presents the full listing of software selected for install. Confirm by selecting OK to continue.



After our package are loaded, the installation returns to the Configuration Options Menu again. At this point we need to add a single account to the system to be used for systems administration. Using the root account creates an audit problem as we have no way to determine who did what. By using a sysadmin account and then having individual

administrators su to root when needed, the system will keep an audit log of this activity in /var/log/messages.

Select User management and add the following:

Group:

Name	GID (Group ID #)
Sysadmin	2000

User:

Name	UID (User ID #)	Group	Password	Full Name	Home Dir.
Sysadmin	2001	2000 wheel (allows su access)	XXXXX	System Administrator	/opt/home/sysadmin (though this will need to be created manually)

Once you have finished you will be returned to the Configurations Options menu.

Next, we need to configure the Startup Services for our system. From the Configuration Options menu, select the Networking and Startup options individually and configure them as follows:

Services

Of all the services listed, only the following should be left enabled:

Startup Option	Notes
Pccard mem	Actually disabled if no PCMCIA cards exist in the system
Pccard ifconfig	See above
Startup dirs	Should contain: usr/local/etc/rc.d ONLY
NIS domainname	Actually disabled since NIS wasn't installed
Quotas	Will give us the ability to limit the disk space used by each of the end user's of the system. This will not be covered in detail, for more information, please see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/quotas.html

Networking

Of all the networking options listed only the items listed in the table below should be left enabled: Keep in mind that our goal is to only provide the minimum required to securely support our server.

Networking Option	Notes
AMD Flags if enabled	Disabled in services above
Sshd	This will be the main method of connecting to this server and transferring files (using the GUI sftp client). Though the version included with FreeBSD is not the latest, we will update it when we harden the system

In addition to the enabled network services, we need to configure our network interface. From the Networking menu, select Interfaces and then choose the interface you would like to configure from the list (by default, the first interface is xl0).

The following information will need to be supplied:

Parameter	Configured on our mock system
Hostname	SafeXFer
Domain	mtnmorts.org
IPv4 Gateway	10.10.1.1
Name Server	206.196.128.1 204.147.80.5
IPv4 interface address	10.10.1.2
Netmask	255.255.255.0

NOTE: These represent the setting for the build environment. I have made every effort to create a build environment which is as similar to the production environment as possible, including firewall protection. A pre-hardened system should never be connected to an unprotected network. Once the edits have been made, select OK to save the changes and return to the Configuration Options menu. The final configurations required are to set the time zone for the server and then the root password

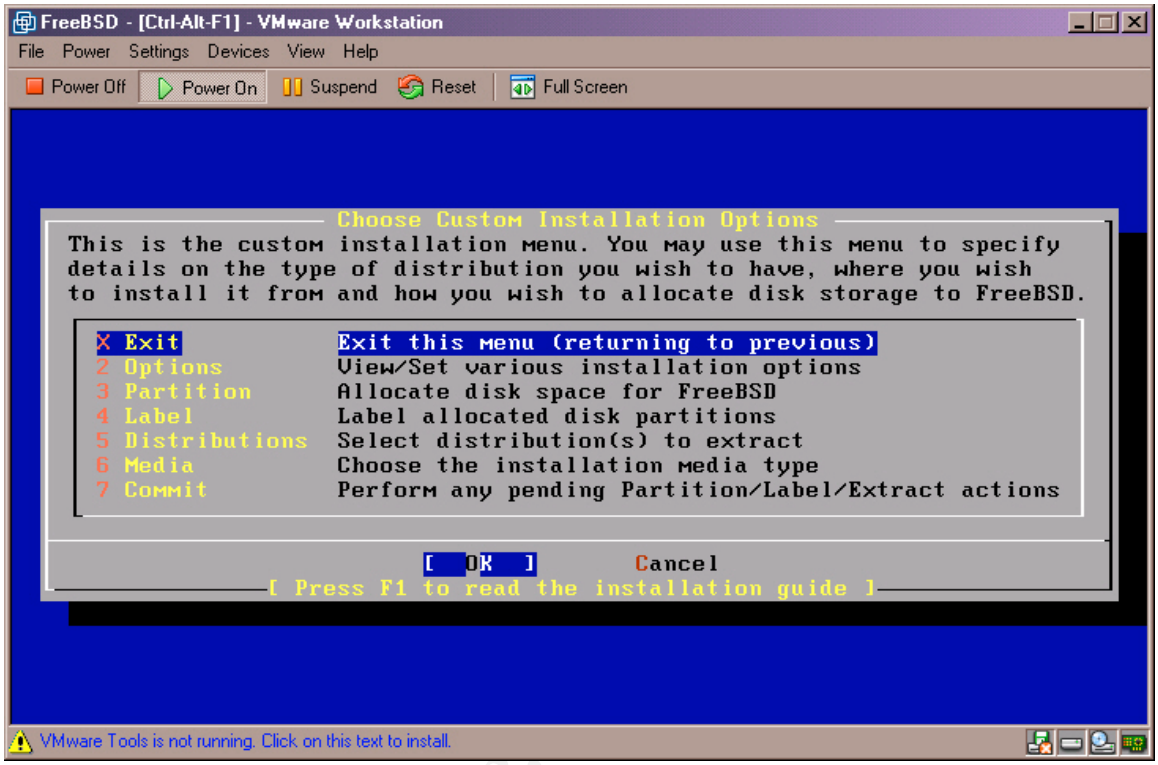
Timezone

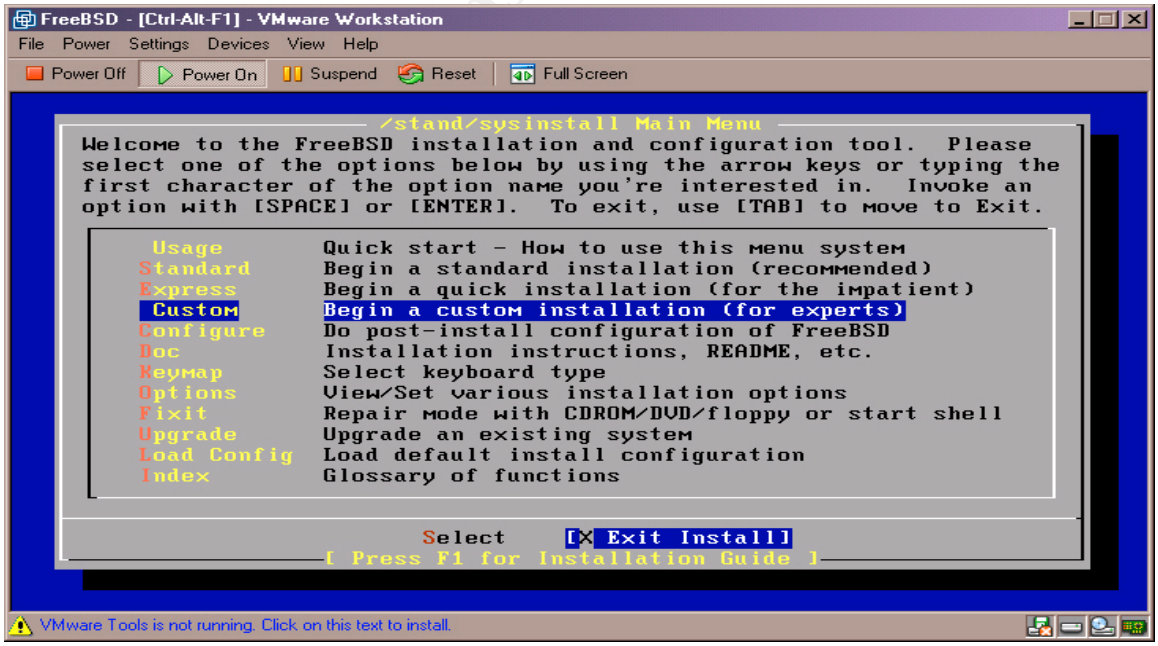
Select timezone from the Configuration Options menu and select the number which represents your geographic location

Root Password

Choose Root Password from the Configuration Options menu and then enter the password and press the return key. You will be prompted to verify the password by entering it a second time. NOTE: It is critical that you use strong passwords for this system as they are a very important part of our layered security approach. Choose passwords that are not commonly found in a dictionary and ensure that they contain a combination of letters, numbers and special characters. Passwords must be a minimum of 6 characters long.

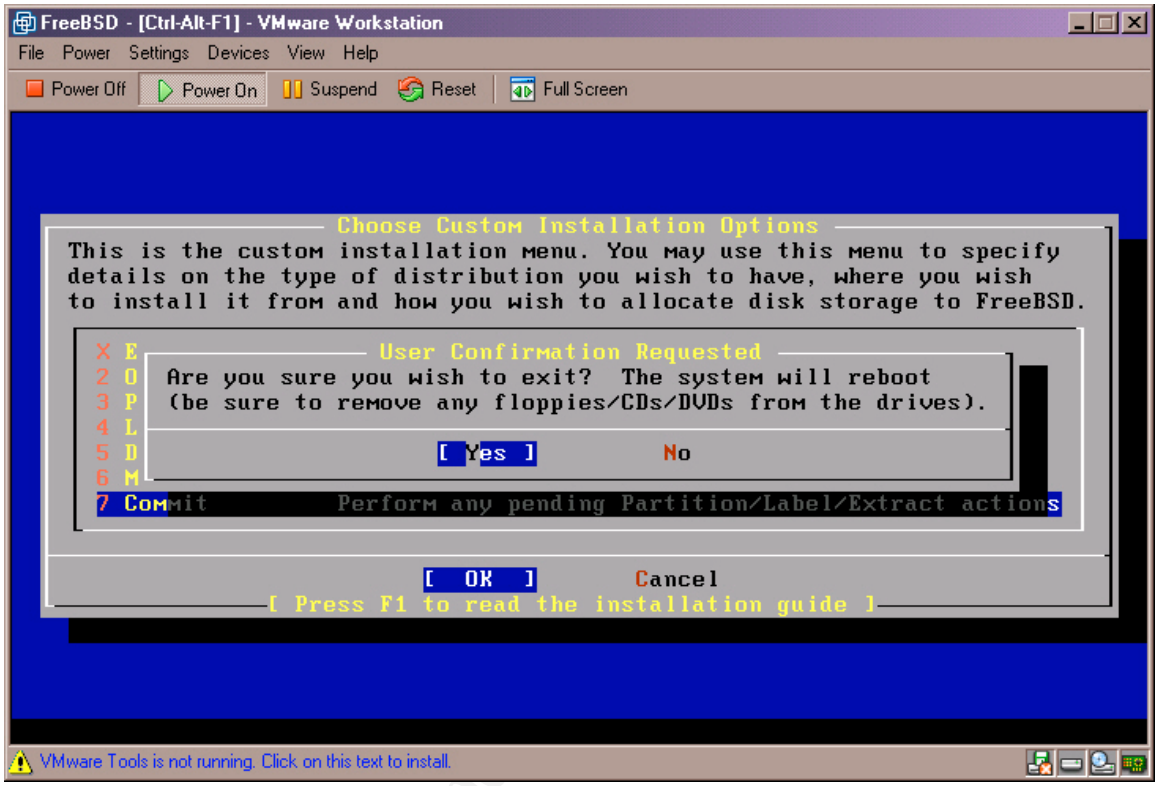
At this point, our installation is complete, select Exit from the Configuration Options menu to end the installation. You have to select this option from the following screens as well.





Finally, you are prompted to confirm that you want to completely exit the FreeBSD installation. Remove any disks or CDs from your system and select Yes to restart.

NOTE: You can return to the installation program at any time in the future by executing the following command:
/stand/sysinstall



Hardening the FreeBSD Installation:

Follow the steps below to harden our FreeBSD installation. For each step, one of the following priority levels is listed:

Priority	Meaning	Required for this Server?
Optional	This may worsen security and must be considered carefully	NO- not recommended
Moderate	These could have a significant effect on the security of the system	YES
Critical	These are directly related to the security of the system	YES
Paranoid	These have a limited impact on this system's security and /or may be excessive for this installation	With Caution ** Note these should be done cautiously as they could adversely impact the system, i.e. make it harder to upgrade, etc.

Step 1:

Recommendation:	**Double Check that you are running the latest Stable release of FreeBSD**
Vulnerability **Reasoning**	New vulnerabilities are discovered all the time Newer releases of the Operating System incorporate security and performances fixes
Tasks Required	• Download the latest Stable Version from http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mirrors.html • Install according to installation instructions presented in this document
Additional Resources / Information:	The following sites lists resources to help you keep up on FreeBSD releases: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html
PRIORITY:	**CRITICAL**

Step 2:

Recommendation:	**Install Security and other patches regularly**
Vulnerability **Reasoning**	New vulnerabilities are discovered all the time, this is your best chance at staying one step ahead of the blackhats Only the bugs which are publicly known are fixed, it only stands to

Tasks Required	reason that many vulnerabilities exist which have not been publicized. You need to limit your chances for compromise as much as possible. As with burglary, the easier the job, the better the chances it will happen! • Check the FreeBSD Errata (vulnerability listings) frequently: http://www.freebsd.org/releases/4.5R/errata.html • Current 4.5 Errata include (neither of these affect our version of FreeBSD 4.5-STABLE or OpenSSH version 3.1p1: A race condition existed whereby a file could be removed between a fstatfs(2) call and the point where the file is accessed, causing a kernel panic. Only the procfs(5) filesystem was known to be vulnerable to this attack. This bug was fixed in FreeBSD 4.5-RELEASE, but the security advisory describing the bug was issued after the release. For more information, including a workaround and bug fix, see security advisory FreeBSD-SA-02:09. An ``off-by-one'' bug has been fixed in **OpenSSH**'s multiplexing code. This bug could have allowed a an authenticated remote user to cause sshd(8) to execute arbitrary code with superuser privileges, or allowed a connecting SSH client to execute arbitrary code with the privileges of the client user. Various workarounds and bugfixes, for versions of **OpenSSH** in both the base system and Ports Collection, can be found in security advisory FreeBSD-SA-02:13. Version 3.1 is not vulnerable • Update system and software as required
Additional Resources / Information:	For more information on staying current with FreeBSD, see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/synching.html
PRIORITY:	**CRITICAL**

Step 3:

Recommendation :	**Install the latest version of OpenSSH and configure for SSH v2 ONLY**
Vulnerability Reasoning	Bugs in the code and exposed vulnerabilities This is our only internet exposed daemon service, as such we need to be sure we are running the latest bugfixed code at all times. It is also a good idea to build this software from source code so that we know it hasn't been tampered with.
Tasks Required	• Download the latest stable source code from www.openssh.org • Unpack the archive and follow the INSTALL document Typical steps to build from source:

	cd /unpacked directory ./configure –with-tcp-wrappers (--with-tcp-wrappers enables support for hosts.allow and hosts.deny authentication) – TCPWrappers must be installed. make make install • Note: The current release (3.1.p1 installs into /usr/local/ instead of /usr so the /etc/rc.conf file must be updated to reflect this change • Make sure the following lines exist, as follows in /etc/rc.conf sshd_enable=”YES” sshd_program=”/usr/local/sbin/sshd” • Edit /usr/local/etc/sshd_config and change the line Protocol 2,1 to Protocol 2 and save the file • Also, change the following to deny ssh as root # PermitRootLogin yes (default) to PermitRootLogin no
Additional Resources / Information:	If you are new to building software from source code, please see the following primer: http://www.icewalk.com/doclib/howtos/Software-Building-HOWTO.html (Linux reference but still applicable)
PRIORITY:	**CRITICAL**

Step 4:

Recommendation:	**Shut down un-needed applications**
Vulnerability Reasoning	May provide unanticipated access for intruders Anything running on the system could provide a useful avenue of attack. The less running, the less you have to keep patched and updated
Tasks Required	• Inetd was disabled during install, double check that the following line is in /etc/rc.conf inetd_enable="NO" This disables, telnet, ftp and many other services we don’t need • Check, the FreeBSD master configuration file /etc/defaults/rc.conf and note the items you need to disable. This files should not be edited directly • Disable anything which isn’t needed, i.e. Sendmail, USB, etc., by adding a the lines noted above to the /etc/rc.conf file and changing the text after the = from “YES” to “NO” • After reboot, check what services are running by using the command: netstat –na |grep LISTEN NOTHING BUT SSH (port 22) should show up, because

Additional Resources / Information:	everything else should be disabled.
PRIORITY:	**CRITICAL**

Step 5:

Recommendation:	**Configure TCPWrappers to control authentication to SSH via IP addresses**
Vulnerability **Reasoning** **Tasks Required** **Additional Resources / Information:**	Provides an additional layer of machine authentication, though it can be spoofed Security in layers • Edit /etc/host.deny and add the following line **ALL : ALL :** • Edit /etc/hosts.allow and add the following line for each host that will access this server **ssh : [host ip address] : allow** **i.e. ssh : 10.10.1.1 : allow** For more information on configuring TCPWrappers, please see: http://www.defcon1.org/html/Networking_Articles/apsfilter/Config-TCPWrappers/config-tcpwrappers.html
PRIORITY:	**MODERATE, can also be controlled by the Firewall**

Step 6:

Recommendation:	**Secure the Root (superuser) account –Disable network root logins**
Vulnerability **Reasoning** **Tasks Required** **Additional Resources / Information:**	Many protocols pass passwords in the clear over remote network connections, also remote root logins do not leave an audit trail Sysadmins should use the sysadmin account to login to the system and then su to root when needed. Passwords should never be allowed to pass over the network in clear text. Someone with access on the network could potential sniff this traffic off the network and obtain the root password • Edit /etc/ttys and change all tty* entries from secure to insecure
PRIORITY:	**MODERATE**

Step 7:

Recommendation:	**Secure the Console**
Vulnerability Reasoning	Unless specified no password is required for the console connection If physical security is breached, this allows free access to the system

Tasks Required	• edit /etc/ttys and change the line below from secure to insecure # If console is marked "insecure", then init will ask for the root password when going to single-user mode. console "/usr/libexec/getty std.9600" vt100 on secure should be console "/usr/libexec/getty std 9600" vt100 on insecure
Additional Resources / Information:	
PRIORITY:	**MODERATE**

Step 8:

Recommendation:	**Remove or Lock un-necessary system accounts**
Vulnerability **Reasoning** **Tasks Required**	Un-need accounts offer extra doors for an attacker to break into Limit that which needs to be monitored and maintained • use the vipw command to edit the password file • place an * in the location of the password field on each account which can be disabled • All account but sysadmin, root and legitimate users should be locked • Alternatively, delete the unused accounts all together
Additional Resources / Information:	For more information on using vipw, see: http://www.freebsd.org/cgi/man.cgi?query=vipw&sektion=8
PRIORITY:	**MODERATE**

Step 9:

Recommendation:	**Secure Cron and at services**
Vulnerability **Reasoning** **Tasks Required**	Non root users could use Cron or at to launch a set-UID exploited program of their choice and gain root access through a buffer overflow or other exploit. Cron and at run as root, and should be limited to root use only • Edit /var/cron/allow and add only the entry: root • Edit /var/at/allow and add only the entry: Root • Only run allow Cron to call scripts created and owned by root
Additional Resources / Information:	
PRIORITY:	**MODERATE**

Step 10:

Recommendation	**Set the Kernel Security level (unset by default)**

:	
Vulnerability **Reasoning**	Root can do anything Limits root from the following being able to perform the following: • unset certain file flags, such as `schg` (the system immutable flag), • write to kernel memory via `/dev/mem` and `/dev/kmem`, • load kernel modules, and • alter ipfirewall(4) rules. Could help to limit the damage if the system is compromised, could help to signal an alarm that the system has been compromised.
Tasks Required	• edit /etc/rc.conf and add the following lines kern_securelevel_enable="YES" kern_securelevel="1"
Additional Resources / Information:	For more information on Kernel security levels, please see: http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/admin.html#SECURELEVEL
PRIORITY:	**PARANOID**

Step 11:

Recommendation :	**Create a system Banner**
Vulnerability **Reasoning**	Assumed public availability If it isn't stated in writing it may be hard to prove in court
Tasks Required	• edit /etc/motd and add a system warning message
Additional Resources / Information:	
PRIORITY:	**MODERATE**

Step 12:

Recommendation :	**Find and Evaluate Set-UID / Set-GID root Programs**
Vulnerability	Could be abused by anyone with access to the system to obtain root access
Reasoning	Program errors, buffer overflows, etc could allow a user to break out of one of these programs as it is running as root, or modify the system during root access
Tasks Required	• Using the following command, locate all the set-UID root programs on the system and disable as many as possible: Find / -perm -4000 –user root -print

The following files were found to be set-UID root on the system (the ones in bold should have the set-UID permission removed for this system)
Use the command: chmod u-s [filename]
to remove the Set-UID bit

/bin/rcp
/**sbin/ping**
/sbin/ping6
/sbin/route
/sbin/shutdown
/usr/bin/at
/usr/bin/atq
/usr/bin/atrm
/usr/bin/batch
/usr/bin/chpass
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/ypchpass
/usr/bin/ypchfn
/usr/bin/ypchsh
/usr/bin/keyinfo
/usr/bin/keyinit
/usr/bin/lock
/usr/bin/login
/usr/bin/passwd
/usr/bin/yppasswd
/usr/bin/quota
/usr/bin/rlogin
/usr/bin/rsh
/usr/bin/su
/usr/bin/crontab
/usr/bin/lpq
/usr/bin/lpr
/usr/bin/lprm
/usr/libexec/sendmail/sendmail
/usr/local/bin/ssh
/usr/sbin/mrinfo
/usr/sbin/mtrace
/usr/sbin/sliplogin
/usr/sbin/timedc
/usr/sbin/traceroute
/usr/sbin/traceroute6
/usr/sbin/ppp
/usr/sbin/pppd

Additional Resources / Information:	• Using the following command, locate all the set-GID root programs on the system and disable as many as possible: Find / -perm -2000 –user root –print The following files were found to be set-GID root on the system (the ones in bold should have the set-GID permission removed for this system) Use the command: chmod g-s [filename] to remove the Set-GID bit **/sbin/ccdconfig** **/usr/bin/fstat** **/usr/bin/ipcs** **/usr/bin/netstat** **/usr/bin/systat** **/usr/bin/top** **/usr/bin/vmstat** **/usr/bin/wall** **/usr/bin/write** **/usr/bin/lpq** **/usr/bin/lpr** **/usr/bin/lprm** **/usr/sbin/iostat** **/usr/sbin/pstat** **/usr/sbin/swapinfo** **/usr/sbin/trpt** **/usr/sbin/lpc** • These disabled Set-UID and Set-GID root files will now require the sysadmin user to su to root before executing • Since we don't expect general users to do anything but transfer files to our server, they should not be affected by these changes
PRIORITY:	**MODERATE**

Step 13:

Recommendation:	**Configure logcheck to watch the system logs**
Vulnerability Reasoning **Tasks Required**	Critical system events could go un-noticed Human error, lack of time, etc. • Use the following information to configure logcheck to

	audit the system log files http://www.bsdtoday.com/2000/August/Features247.html • Here is an example should be done: cd /usr/local/etc/ cp logcheck.ignore.sample logcheck.ignore cp logcheck.violations.ignore.sample logcheck.violations.ignore cp logcheck.violations.sample logcheck.violations cp logcheck.hacking.sample logcheck.hacking Note that logcheck is started with /usr/local/etc/logcheck.sh
Additional Resources / Information:	For more information on Logcheck—now called logsentry, please see: **http://www.psionic.com/products/logsentry.html**
PRIORITY:	**MODERATE**

Step 14:

Recommendation:	**Consider configuring CVsup for automatic updating of the system**
Vulnerability Reasoning	Provides automatic updates to the system source May help improve the security of the system
Tasks Required	• Use the following information to evaluate using CVsup for system updates http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html
Additional Resources / Information:	
PRIORITY:	**OPTIONAL**

Ongoing Maintenance:

We now have a FreeBSD secure file exchange server which can be used to allow project members to share data all over the world. In order to keep this system secure and running as efficiently as possible, it is critical that the following ongoing maintenance take place at regularly scheduled intervals:

Maintenance Task	Information—HOWTO
Stay current on vulnerabilities which affect this system	The following mailing lists will help: FreeBSD related mailing lists: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html CERT Advisories: http://www.cert.org/contact_cert/certmaillist.html OpenSSH mailing lists: http://www.openssh.org/list.html

Review system logs and logcheck reports	Log files are stored in /var/log Review logcheck output daily to get comfortable with what is "normal" operation for this server and also to pick up on irregularities as quickly as possible
Check for updates to key system software, OS, SSH, etc	See mailing lists above
Backups	Though we intend to mirror the disks on the production system, it is important to perform and test regular backups of the system using the local tape drive. Some of the tapes should be stored offsite in case of disaster
Consider future need for more advanced tools	Consider the necessity of tripwire (though this would require a re-build to insure the system is pure at install), local virus scanning, and other applications as needed to combat issues which may arise after production release of this system
Documentation	This is critical—JUST DO IT!

Checking the Configuration:

To check the security of the system we will first check it using one of the most popular scanning tools available, NMAP and the best freeware vulnerability scanner, NESSUS. Then we will run five individual tests to zeroing in to test some of the specific tasks we took during the hardening of the system. Regular scanning and testing should be something which continues after the system is put into production. This is very useful in capturing mis-configurations and / or changes to the system.

NMAP

NMAP is probably the most popular port scanner available today www.nmap.org . It can be used (with lots of different options which enable it to work in almost any situation) to map which ports are listening on a remote system (which daemons are available). In addition, it can do a fairly decent job of determining the underlying operating system of a host. NMAP gives you a good idea of how the outside world views (will attack) your host. The results of the NMAP scan are displayed below. We scanned our host without firewall protection so that we could get a clear idea of what is left available. From the internet, the only service available is SSH, tcp port 22 (as we would expect). All other services have been successfully turned off. In addition to the protection from our production firewall, there is very limited available access to our system.

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on  (207.225.105.194):
(The 1546 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
```

No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/cgi-bin/nmap-submit.cgi).
TCP/IP fingerprint:
SInfo(V=2.54BETA30%P=i686-pc-linux-gnu%D=3/11%Time=3C8D3C33%O=21%C=1)
TSeq(Class=RI%gcd=1%SI=5BAA8%IPID=I%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=94BB5%IPID=I%TS=100HZ)
TSeq(Class=RI%gcd=1%SI=75354%IPID=I%TS=100HZ)
T1(Resp=Y%DF=Y%W=2297%ACK=S++%Flags=AS%Ops=NNTNWME)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=N)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=Y%DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(Resp=N)
PU(Resp=Y%DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=0%ULEN=134%DAT=E)

Uptime 17.883 days (since Thu Feb 21 19:11:05 2002)

Nmap run completed -- 1 IP address (1 host up) scanned in 38 seconds

Nessus

Nessus is one of the most robust vulnerability scanners on the market today, and it also happens to be free. Nessus is a client server application available from http://www.nessus.org. The server runs on most flavors of Unix and the client runs on many different Operating systems. You do not need to load any software on the host you want to scan. Nessus can discern rich detail about a system, including the underlying operating system, versions of software available and susceptibility to vulnerabilities. It can even offer suggestion for how to fix any problems it encounters. Nesses's checks are based on signature files which it tries to match against the system being checked. When finished, a report will be presented with the vulnerabilities and solutions identified. The results of our Nessus scan are displayed below. Again, there is nothing surprising or alarming in the Nessus report. Again, this offers verification that our system is ready for production.

Nessus Scan Report

Number of hosts which were alive during the test : 1
Number of security holes found : 0
Number of security warnings found : 0
Number of security notes found : 2

List of the tested hosts :

- 207.225.105.194

207.225.105.194 :

List of open ports :

- *ssh (22/tcp) (Security notes found)*
- *general/udp (Security notes found)*

Information found on port ssh (22/tcp)

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

. 1.99
. 2.0

Information found on port ssh (22/tcp)

Remote SSH version : SSH-2.0-OpenSSH_3.1p1

Information found on port general/udp

For your information, here is the traceroute to 207.225.105.194 :
162.18.91.1
?
162.18.67.131
160.81.57.113
144.232.12.165
208.172.129.201
208.172.162.61
208.172.162.13
208.172.173.90
205.171.16.13
205.171.8.86
206.196.128.248
207.225.101.20
207.225.105.198
207.225.105.193
207.225.105.194

This file was generated by Nessus,

Since things are looking good after scanning with NMAP and Nessus we will move on to test some the specific security steps which we applied when we hardened the system.

Local Test 1: running services

From a system on the internet (which is allowed to pass all traffic through our firewall) I will attempt to connect to our server via telnet and ftp to prove that these are not available.

From the outside machine, attempt to connect to the system via telnet:

```
[root@tornado root]#
[root@tornado root]# telnet 207.225.105.194
Trying 207.225.105.194...
telnet: connect to address 207.225.105.194: Connection refused
```

From the outside machine, attempt to connect to the system via ftp

```
[root@tornado root]#
[root@tornado root]# ftp 207.225.105.194
421 Service not available, remote server has closed connection
ftp> quit
```

From the outside machine, attempt to connect to the system via rlogin

```
[root@tornado root]# rlogin 207.225.105.194
207.225.105.194: Connection refused
[root@tornado root]#
```

From the outside machine, attempt to connect to the system via ssh version 2

```
[root@tornado root]# ssh2 207.225.105.194
Host key not found from database.
Key fingerprint:
xorok-lufin-kikuv-fukyc-sobit-menop-fepas-bulav-bimyh-mypyf-boxex
You can get a public key's fingerprint by running
% ssh-keygen -F publickey.pub
on the keyfile.
Are you sure you want to continue connecting (yes/no)? yes
Host key saved to /root/.ssh2/hostkeys/key_22_207.225.105.194.pub
host key for 207.225.105.194, accepted by root Mon Mar 11 2002 20:55:17 -0700
root's password:
```

So clearly, of all the methods attempted only SSHv2 is available to the outside world.

Local Test 2: remote SSH logins as root are disabled

From the outside machine, attempt to connect to the system using SSHv2 as root:

```
[root@tornado root]# ssh2 207.225.105.194
root's password:
```

root's password:
root's password:
warning: Authentication failed.
Disconnected; no more authentication methods available (No further authentication methods available.).

Now try to connect as sysadmin (which has been configured temporarily with the same password as root):

[root@tornado root]# ssh2 -l sysadmin 207.225.105.194
sysadmin's password:
Authentication successful.
Last login: Mon Mar 11 13:26:40 2002 from 10.10.1.4
FreeBSD 4.5-RELEASE (GENERIC) #0: Mon Jan 28 14:31:56 GMT 2002

****
WARNING-
THIS SYSTEM CONTAINS PRIVATE INFORMATION FOR THE SOLE USE
OF ITS OWNERS. ALL ACCESS IS EXPLICITLY LOGGED AND
MONITORED. VIOLATORS WILL BE PROSECUTED!
****
$

Local Test 3: local root logins are disabled

From the local system attempt to login locally as root:

login: root
password: xxxxxxxx
Mar 11 21:31:52 SafeXFer login: LOGIN root REFUSED (NOROOT) on TTY ttyv0

However, it is possible to login as sysadmin and su to root.

Local Test 4: SSH v1 is not supported

From another machine, using Putty a freeware SSH client (set to default to using SSH v1, if possible) attempt to connect to the system:

2002-03-11 21:34:07 Looking up host "207.225.105.194"
2002-03-11 21:34:07 Connecting to 207.225.105.194 port 22
2002-03-11 21:34:07 Server version: SSH-2.0-OpenSSH_3.1p1
2002-03-11 21:34:07 We claim version: SSH-2.0-PuTTY-Release-0.52
2002-03-11 21:34:07 Using SSH protocol version 2

2002-03-11 21:34:07 Doing Diffie-Hellman group exchange
2002-03-11 21:34:07 Doing Diffie-Hellman key exchange
2002-03-11 21:34:12 Host key fingerprint is:
2002-03-11 21:34:12 ssh-rsa 1024 7b:1e:25:57:2f:3c:9d:b0:ed:93:b1:a2:33:71:1f:2b
2002-03-11 21:34:12 Initialised AES-256 client->server encryption
2002-03-11 21:34:12 Initialised AES-256 server->client encryption
2002-03-11 21:34:17 Keyboard-interactive authentication refused
2002-03-11 21:34:19 Sent password
2002-03-11 21:34:19 Access granted
2002-03-11 21:34:19 Opened channel for session
2002-03-11 21:34:19 Allocated pty
2002-03-11 21:34:19 Started a shell/command

NOTE: Even though the client is configured to use SSHv1, the negotiated version is 2, SSHv1 is not available.

Local Test 5: Cron is limited to root

From the local system, connected as sysadmin, try and create and implement a new Cron file:

$ whoami
sysadmin
$
$ pwd
/opt/home/sysadmin
$
$ cat testcron
This is a test!
$
$ crontab testcron
crontab: you (sysadmin) are not allowed to use this program
$
$

Conclusion:

The FreeBSD 4.5-STABLE system has been successfully configured and secured to provide secure file exchange services. While our tests have proved that things went as expected, this is no time to get complacent. New vulnerabilities appear daily and this system will need to be maintained consistently in order to keep it secure. While the risk of compromise, even after all we have done, is still very real (especially, given that we have only really attempted to protected against known issue and attack methods) the risks of providing this service should be far outweighed by the benefits of providing this much needed new service. I believe this system fills a very critical need which has not been addressed in our current environment. This system will provide a secure way to meet the growing need for dynamic, secure file exchange.

References:

Lehey, Greg. The Complete FreeBSD, 3rd edition. 1999. Walnut Creek

The FreeBSD Documentation Project. The FreeBSD Handbook. 2002. **http://www.freebsd.org/doc/en_US.ISO88591/books/handbook/index.html**

The FreeBSD Documentation Project. Frequently Asked Questions for FreeBSD 2.x, 3,x and 4,x. 2002. **http://www.freebsd.org/doc/en_US.ISO88591/books/handbook/index.html**

Raborn, Timothy. "Installing & Securing Solaris 8" 2001. http://www.giac.org/GCUX.php

Sery, Paul. "RedHat Linux 7.1 Installation Hardening Checklist " 2001. http:// www.giac.org/GCUX.php

Thompson, Michael. "Building a Secure RedHat Web & FTP Server" 2001. http:// www.giac.org/GCUX.php

Zeltser, Lenny. "Consultant's Report From Auditing Unix" 2001. http:// www.giac.org/GCUX.php

Frederick, Karen. "OpenBSD 2.8 Security Checklist" 2001. http:// www.giac.org/GCUX.php

Koconis, David. "Step by Step Guide to Configuring an SSL Enabled Web Server that Accesses a Backend Database using RedHat 7.0" 2001. www.giac.org/GCUX.php