Global Information Assurance Certification Paper

How to build a secure mail server using QMail and FreeBSD

Name: Donald Nathaniel Holloway, III

Assignment Version:1.4b

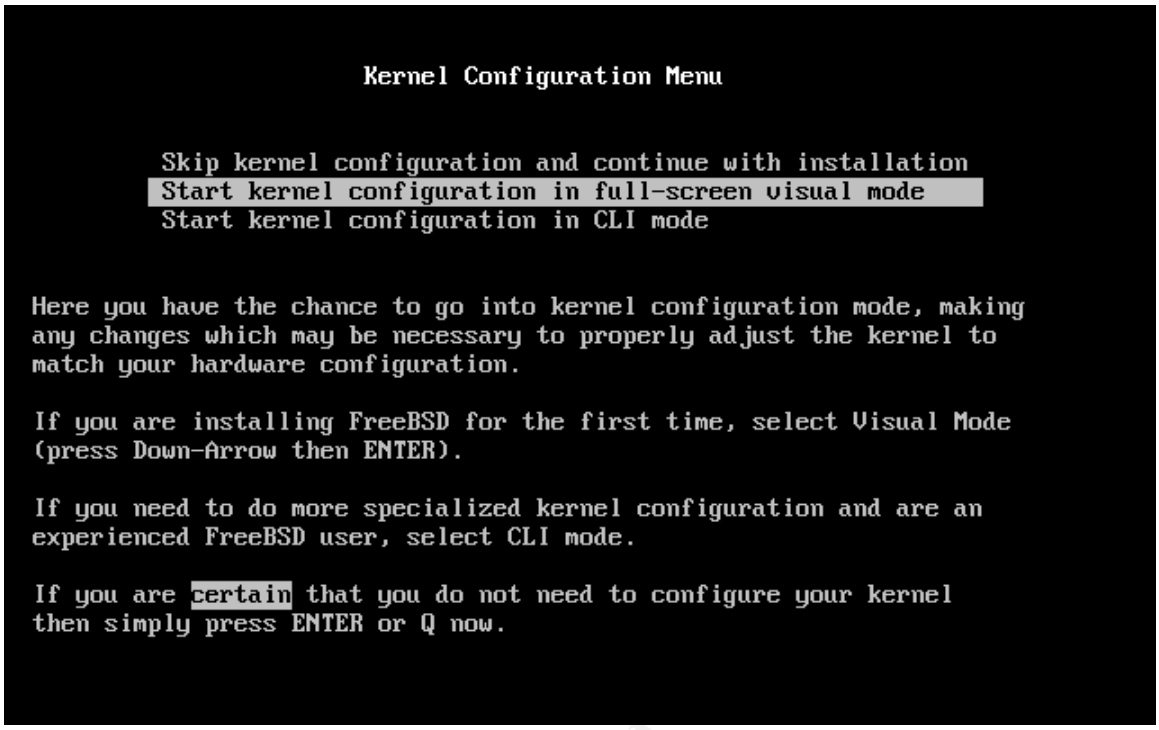
Introduction

In this tutorial we will be using FreeBSD 4.10 (current release as of this writing) and QMail to produce a simple and secure mail server for small or home offices (SOHO). FreeBSD is easy to maintain and keep patched and QMail is designed with security in mind. We'll begin by getting FreeBSD installed and patched to the latest patch level. That way we'll have a good base to begin with our QMail installation. We'll also cover Sqwebmail, QMailadmin, and dovecot which make QMail easier to manage and more user-friendly without the cost of lessened security. We'll also introduce the concept of a jailed environment, since we will be running QMail and its supporting processes in a jailed environment.

The goal of this how-to is to have a stable, secure, and easy to manage mail system that is suitable for a small or home office. This guide will walk you through the entire process of inserting the installation media to sending and receiving mail through a web interface. After going through this guide you should become familiar with the FreeBSD operating system and its supporting package management system. You should also learn some easy techniques to securing your application environment that you can use to support other services. The only prerequisites to this installation how-to is that you have Internet connectivity, an i386-based PC (processor speeds of 100Mhz or higher), and name services set up with the mail servers records already present.

FreeBSD installation

To begin the install simply download an iso image from the FreeBSD website ftp.freebsd.org. Once you have a working CD boot up your system and begin the install process. You will not need to reconfigure the kernel so press enter when this screen pops up.

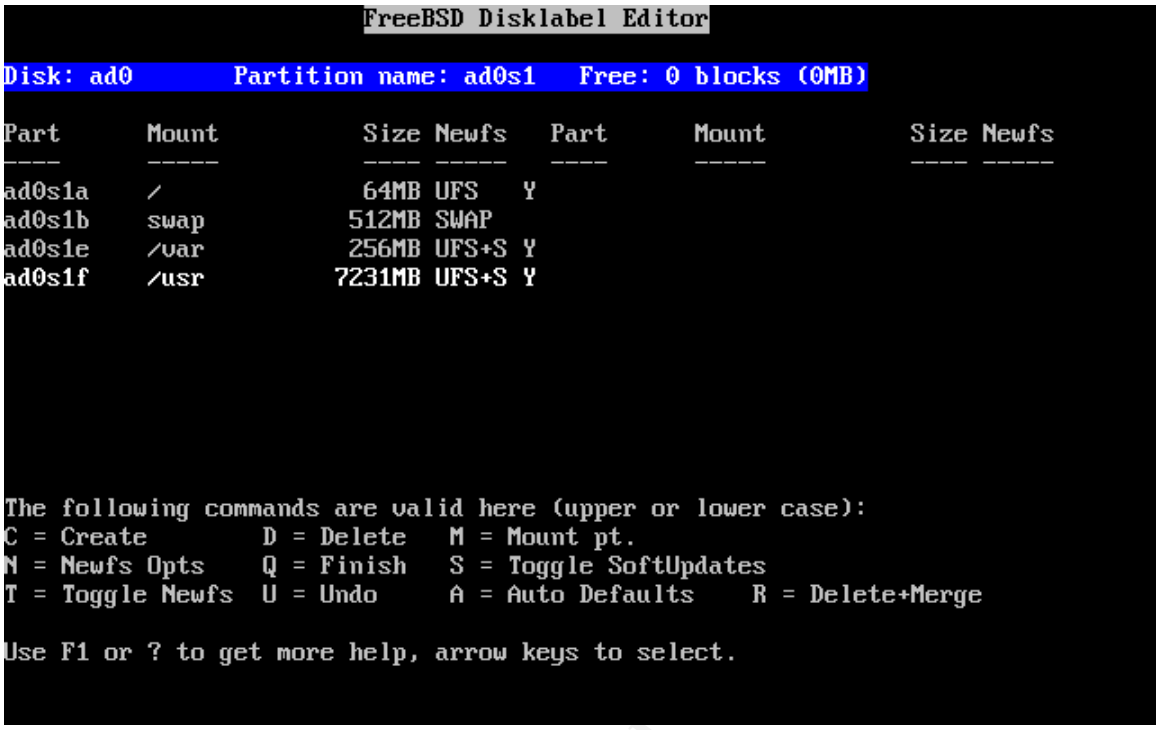


[1]

Select a “Standard” Install so we can select user accounts and set the time zone. After you select “Standard Install” it will bring you to the disk partitioning utility. We will assume that you are only using this disk for FreeBSD and will allocate the entire disk for FreeBSD. Enter “A” for add and select the default for each entry. Once completed select “Q” to quit the utility and then select “Standard” and this will install a standard MBR since you will only be using this computer with FreeBSD.

Disk Partitioning

Now you will create your partitions. You will want to at least create separate partitions for “ / “, “ /usr “, swap, and “ /var “. It’s important that you do this since most logs will sit in /var and if something goes haywire on the system and starts filling up the logs it won’t completely fill the disk disabling you to log on to the system and remedy the situation. For an 8 gigabyte disk I would recommend allocating 3 gigabytes for /var, 1 gigabyte for /, 256 megabytes for swap, 2.5 gigabytes for /jail and the rest for /usr. You can modify this to your liking and depending on what hardware you are using. Use “C” to create and “Q” to quit. You can allocate 2.5 gigabytes as 2500M, 1 gigabytes as 1G, 256 megabytes as 256M, and accept the defaults for /usr. Once you are complete it should look something similar to this:



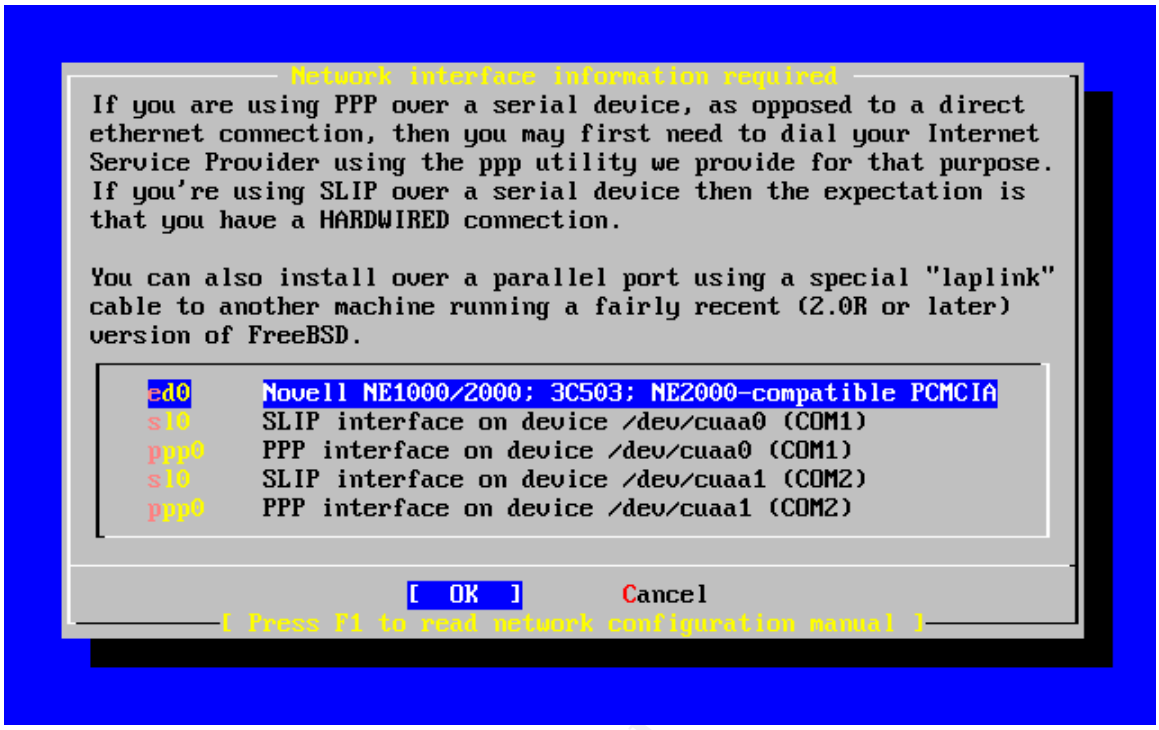
[1]

Your partitions may look different if you have a larger disk than 8 gigabytes. Select “Q” to quit and it will bring you to the installation type you want to pick. Pick “User” for simplicity.

Exit from that menu and exit from the previous menu. It will ask you where you want to install the operating system (heretofore referred to as “OS”) from and select CD-ROM. It will then ask you if you are sure if you want to wipe out the disk and install the OS. Select “Yes” here. It will then begin to partition and format the disks and then install the OS.

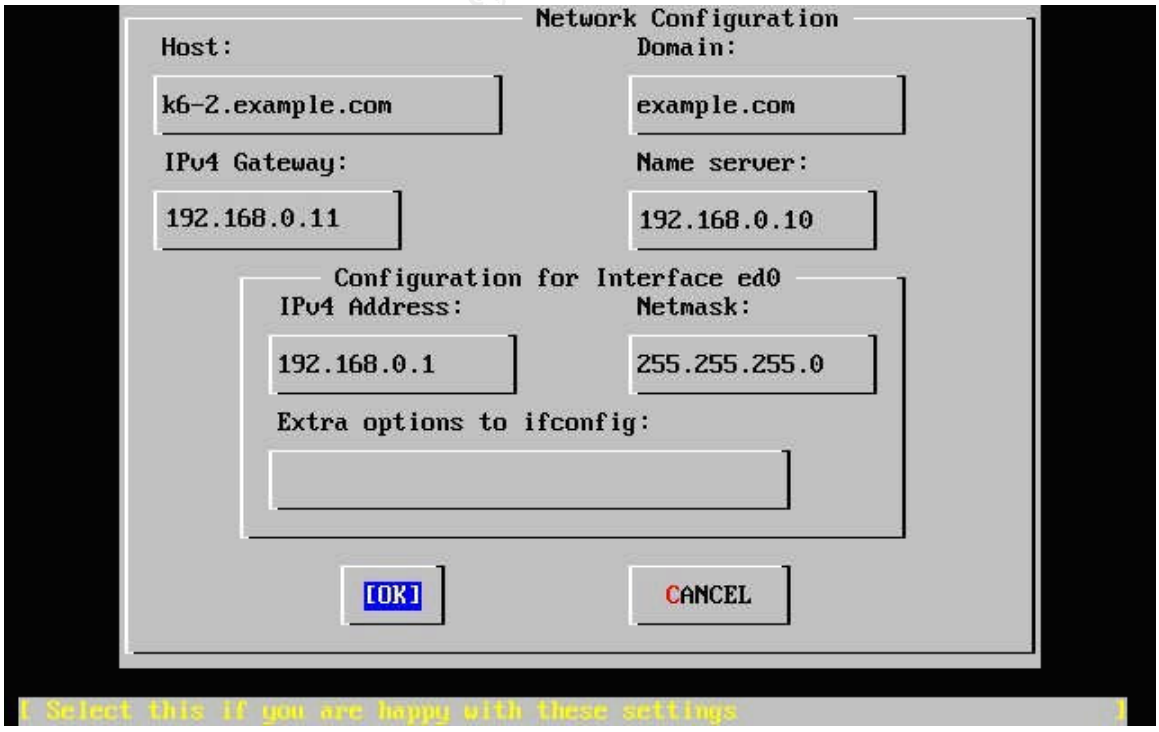
Network Interface Configuration

Once it’s complete it will give you a congratulations screen and then will ask if you want to configure any network devices. Select “Yes” here. The device you will want to configure will almost always be the first device listed. I am using a “lnc” adapter, so I will select this one. The configuration screen is similar to the screen below:



[1]

It will then ask you if you want to use IP v6 and then if you want to use DHCP. Since this is going to be a mail server it should have a static IP address. Input the values in the menu like so:



[1]

Your selections will, of course, be different. Once you are happy with the selection select OK.

System Services

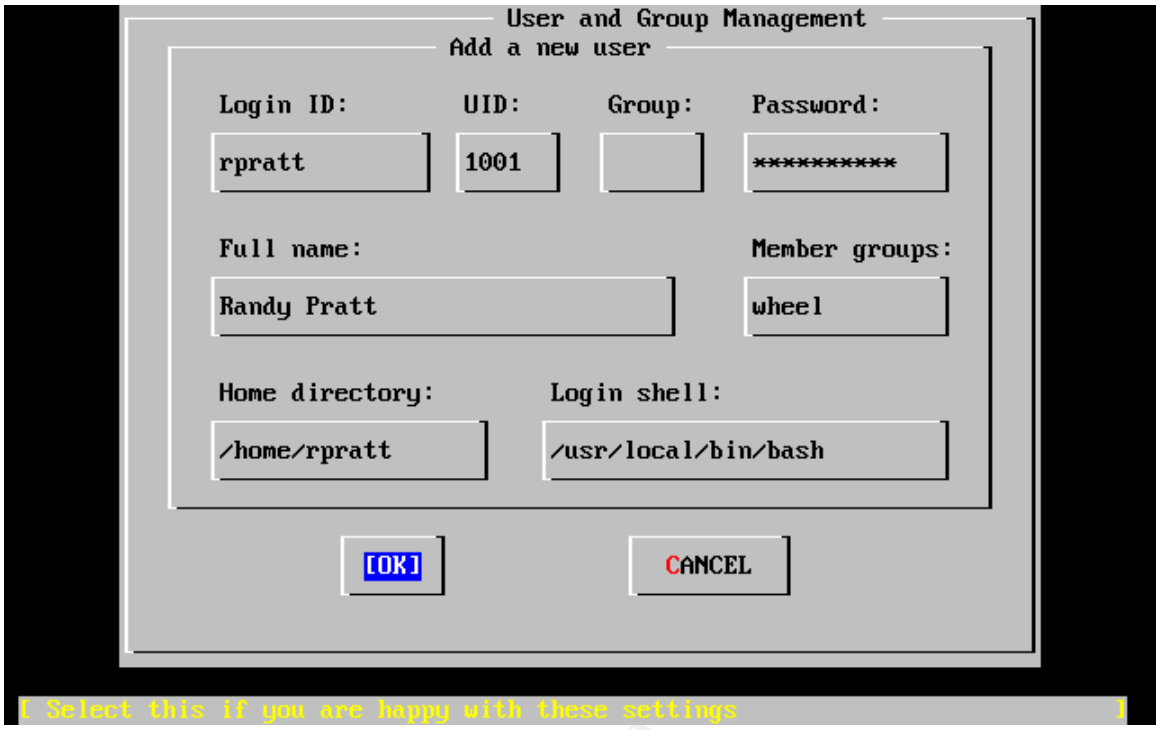
It will then ask you if you want the server to function as a gateway (firewall or router), to have inetd run services, if you want anonymous ftp access, or have the device serve as an NFS server or client. Select “No” for all, which is the default. The reason for this is that you only want to have this device to be a host system running nothing but ssh for remote access. Keep the configuration as simple as possible and that will help keep it more secure and less troublesome should something go wrong. It’s also a good policy to keep a single service on a single machine if you can.

It will then ask you if you want to select a different kernel security level than “moderate”. Select “No” here. Having a higher kernel security level will impede us from updating our source code later. It will then ask you if you want to change some of the system settings such as font or screensaver. You can select “No” if you want, or make changes to suit your liking. After that you can select the time zone. Select the time zone that is appropriate for your location. It will then verify the time zone you have selected and then ask you if you want to install Linux binary compatibility. This is not needed, so select “No” here. All the applications we will install have been ported or are native to FreeBSD. It will ask you if you want to install any applications, select No here as well. Many of the versions of the applications are out of date with this release and we will add the up-to-date programs at a later time.

User Accounts

You will then be asked if you want to create any users or groups. You will need to create another user that you can use to “su –“ to root. This user needs to be in the “wheel” group. This is important because you will not be able to ssh for remote access to this device since FreeBSD disables root ssh login by default. It’s also a good habit to get into using an administrative account to administer the box and use root as sparingly as possible. The root user has the keys to the kingdom and if you log into the console as root and forget to log out anyone can come right behind you and do whatever they please with the box. Possibly creating a security nightmare, since there’s no accountability and the logs can be wiped away concealing the user’s actions. [2]

The user admin tool looks similar to this:



[1]

You will then be asked to enter the root password. Input this and then it will ask you if you want to go to the main menu for any other changes. Select "No" here and then select exit installation. It will then ask you if you've removed the CD-ROM. Select "Yes" here and then it will reboot. Make sure to take the CD-ROM out before the system starts to boot from it again.

System Update – recompiling the OS from source with cvsup

At this point we will update the system and get it up to the latest patch levels. You'll need a working DNS server that already has your mail server's records. To update the system we'll need CVSup and we can get it from the ftp.freebsd.org site. The current version as of this writing is located here ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/4.10-RELEASE/packages/net/cvsup-without-gui-16.1h.tgz. Ftp the file down and then add the package as "pkg_add cvsup-without-gui-16.1h.tgz". Next we will need to copy the config file for cvsup to /etc and then modify it. [3] Copy /usr/share/examples/cvsup/stable-supfile to /etc and edit the file to look similar to this:

```
*default host=cvsup6.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_4
*default delete use-rel-suffix
#*default compress
src-all
```

`ports-all tag=.` [3]

Once you have this in place and edited like the above you can start updating from source. Cd into /usr/src and run these commands:

```
cvsup –g –L 2 /etc/stable-supfile && make world
```

This will take a while depending on the processor and the Internet connection speed. Once it has completed you then recompile the kernel. Cd into /usr/src/sys/i386/conf. Copy GENERIC to MAIL and then modify at your leisure. For this example we will take out the following lines:

```
options     FFS_ROOT
options     MD_ROOT
options     NFS_ROOT
options     CD9660_ROOT
```
[4]

The reason we want to remove these lines is that you don't want any application or user to gain root level permissions of any device or file system. It is not needed for the system to function properly.

Add these lines to the bottom:

```
options     TCP_DROP_SYNFIN
options     RANDOM_IP_ID
```
[4]

We want to add those lines to help protect the system against TCP SYN+FIN attacks. It also makes the system more difficult to enumerate with scanning tools like nmap. The RANDOM_IP_ID helps prevent observers to figure out the rate of packet generation. [4]

Once you have completed this type:

```
config MAIL && cd ../../compile/MAIL && make depend && make && make
install
```
[4]

This will compile your new kernel and install it in place. Once it has successfully completed reboot the computer by typing "`shutdown –r now`".

When the system comes back online you will have to run "`mergemaster`". [4] This will update all of your configuration files to the latest versions. If there is a discrepancy between the new file and the old file it will pull up the difference (heretofore known as "diff") between the two files. Since this is the first time you've run this type "q" to quit the diff and "i" to install the new file. You will not want to install the "hosts" file, the "rc.conf" file, "passwd" file, or the "master.passwd" file. These are specific to your machine and mergemaster will overwrite them if you let it. Use "d" to delete the mergemaster files when it asks

to update. Once this is completed you can type “uname –a” and it will show you you are at 4.10-Stable. This is the latest production release as of this writing.

Creating the Jailed environment

Now we’ll create our jail to house our mail environment. A jailed environment is basically a virtual FreeBSD system within the core operating system. It has no access to any processes running on the core system, nor does it have access to any disk area other than the folder you have installed the jail in. Another benefit is that if the processes in the jail are ever compromised the intruder will not have access to the host system or the host disk. Therefore, you can minimize the damage the intruder will cause and you can also log the intrusion. [5]

To create your jail we’ll need to create a small script to make the installation easier. Since /var has the most space you can use /var/jail as a jail base directory. This script will create the mail jail for you:

```
#!/bin/sh

D=/jail/mail
cd /usr/src
mkdir -p $D
make world DESTDIR=$D
cd etc
make distribution DESTDIR=$D -DNO_MAKEDEV_RUN
cd $D/dev
sh MAKEDEV jail
cd $D
ln -sf dev/null kernel

exit 0
```
[5]

This will basically put an entire FreeBSD system in the /jail/mail directory. Once this is complete you will have to copy sysinstall (install and configuration utility) into the jail. Run “mkdir /jail/mail/stand” and copy /stand/sysinstall to that new directory. To start the jail and run some basic configuration options type the following:

```
jail /jail/mail mail 192.168.75.250 /bin/tcsh
```
[5]

Now you will configure the jail. Run `/stand/sysinstall` and set the time zone. Go to Configure > Time Zone and configure the time zone. The jail will get its time from the host system but you still need to set the correct time zone. Once completed you need to set the root password for the jail, which should be different than the host system. [5] After that edit the /etc/crontab file and comment out this line:

```
1,31    0-5     *       *       *       root    adjkerntz –a
```

The jailed system won't be able to set the system time since it gets that from the host system. If you don't comment out the above line your syslog will be filled with error messages about the above process since the jailed system will keep trying to match the kernel's time with the CMOS settings of the device.

Also, edit (create) the /etc/hosts file to read something similar to this:

```
::1                     localhost.example.com localhost
127.0.0.1               localhost.example.com localhost
192.168.75.250           mail.example.com mail
192.168.75.250           mail.example.com.
```

Edit the /etc/rc.conf to look similar to this:

```
hostname="mail.example.com"
network_interfaces=""
sendmail_enable="NONE"
portmap_enable="NO"
tcp_keepalive="NO"
```

Also create an empty /etc/fstab to quell start up error messages [5]:

```
touch /etc/fstab
```

Create the /etc/resolv.conf to input the nameservers you will be using. These nameservers must be able to resolve the name and IP address of the mail server that you will be using in this jail. After you have completed this exit out of the jail by hitting CTRL + D or exit.

You then need to create a start-up script so the jail will start each time the host system boots. Here is a sample script that will boot the mail jail:

```
#!/bin/sh

ifconfig lnc0 inet alias 192.168.75.250/32
mount -t procfs proc /jail/mail/proc
jail /jail/mail mail 192.168.75.250 \
/bin/sh /etc/rc

exit 0
```

This file needs to be executable and in the /usr/local/etc/rc.d/ directory. [5]

One other thing that will need to be done on the host system is to modify the /etc/rc.conf to have the following:

```
sendmail_enable="NONE"
icmp_drop_redirect="YES"
tcp_drop_synfin="YES"
icmp_log_redirect="YES"
portmap_enable="NO"
```

```
fsck_y_enable="YES"
inetd_flags="-wW -a 192.168.75.134"
```

This will keep the system from starting sendmail and not allowing the jail environment to bind to port 25. The others help keep the system from being scanned or a victim of an ICMP or TCP based Denial of Service attack.

Once this is completed you can start the jail by running "`sh /usr/local/etc/rc.d/mail.sh`" (or whatever you named the start up script). This will start up the jail and alias the IP address to the interface you specified in the script. You'll see interface properties as such:

```
> ifconfig -a
lnc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.75.134 netmask 0xffffff00 broadcast 192.168.75.255
        inet6 fe80::20c:29ff:fea5:4e7c%lnc0 prefixlen 64 scopeid 0x1
        inet 192.168.75.250 netmask 0xffffffff broadcast 192.168.75.250
        ether 00:0c:29:a5:4e:7c
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
        inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

To get the applications you need you will need to copy the /usr/ports directory to the /var/jail/mail/usr directory. Once that is complete you can enter the jail and begin the application installation.

To get into the jail run the jail command you ran earlier:

```
jail /jail/mail mail 192.168.75.250 /bin/tcsh
```

Installing services via the Ports system

You will need to get the ports system on the jail and the easiest way to do this is again with CVSup. You will have to add the CVSup package to the jail system and the easiest way to do this is to copy the cvsup-without-gui-16.1h.tgz file over to /jail/mail/tmp. When you start the jail with the above command you can cd into /tmp and you will see the file there. Run the pkg_add command and type "rehash" and you will be able use CVSup. Your stable-supfile will be a little different since you won't want to install the OS source, but just install the ports system. Your /etc/stable-supfile should look like this:

```
*default host=cvsup6.freebsd.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_4
*default delete use-rel-suffix
ports-all tag=.
```

Once you have completed running CVSup you will have the ports system installed in /usr/ports. The beauty of the ports system is that when you install an application it will go out and gather the dependencies if they are not already installed. One nice feature is that if there is a serious security flaw in the ported application it will refuse to install. It also integrates with FreeBSD's package management system well. If you keep your ports regularly updated by running the CVSup update utility you can check to see if new version are available by running pkg_version –c. This will print out scripts to update the packages. You will have to exercise some judgment when updating ported applications since some applications depend on other older versions of the applications you have installed.. For example, most library packages shouldn't be upgraded. Use your best judgment for this. [6]

Now let's begin to install the applications.

Installing QMail

cd into /usr/ports/mail/qmail

Type "make install clean" and this will install QMail and clean up the source files after installation.

Installing Apache

Next, cd into /usr/ports/www/apache2 and type "make install clean".

Installing QMailAdmin

Go into /usr/ports/mail/qmailadmin and "make WEBDATADIR=/usr/local/www/data-dist CGIBINDIR=/usr/local/www/cgi-bin-dist install clean".

Installing SqwebMail

After that go into /usr/ports/mail/sqwebmail and type "make WITH_HTTPS=yes WITH_VCHKPW=yes WITH_ISPELL=yes install clean".

Installing Dovecot

After that go into /usr/ports/mail/dovecot and "make install clean".
It will pull up a menu and select "VPOPMAIL" and select "OK".

Installing Daemontools

Cd into /usr/ports/sysutils/daemontools and "make install clean".

Configuring QMail and its supporting cast

Cd into /var/qmail/configure and type "./config".

Cd into /usr/local/vpopmail/bin and run `./vadddomain example.com`. It will then ask you for a password for the postmaster account. Give a password and make sure it's not a password you have used before.

```
mkdir /service
```

This is a great control file written by Dave Sill. I've modified it to remove support for POP3 since we will not be using it. You can cut and paste this script into the /var/qmail/bin directory. Make it executable (chmod 755 /var/qmail/bin/qmailctl) and then link it to /usr/local/sbin. [7] Using this system will keep QMail up and running at all times. Svscan will scan the /service directory and restart any daemons that have died. This keeps your system available at all times.

Here is the modified file:

```
#!/bin/sh

PATH=/var/qmail/bin:/bin:/usr/bin:/usr/local/bin:/usr/local/sbin
export PATH
LOG=/var/log/qmailctl

echo `date` `tty` $* >>$LOG

case "$1" in
  start)
    echo "Starting qmail"
    if svok /service/qmail-send ; then
      svc -u /service/qmail-send 2>&1 | tee -a $LOG
    else
      echo qmail-send service not running
    fi
    if svok /service/qmail-smtpd ; then
      svc -u /service/qmail-smtpd 2>&1 | tee -a $LOG
    else
      echo qmail-smtpd service not running
    fi
    if [ -d /var/lock/subsys ]; then
      touch /var/lock/subsys/qmail
    fi
    ;;
  stop)
    echo "Stopping qmail..."
    echo "  qmail-smtpd"
    svc -d /service/qmail-smtpd 2>&1 | tee -a $LOG
    echo "  qmail-send"
    svc -d /service/qmail-send 2>&1 | tee -a $LOG
    if [ -f /var/lock/subsys/qmail ]; then
      rm /var/lock/subsys/qmail
```

```
    fi
    ;;
  stat)
    svstat /service/qmail-send
    svstat /service/qmail-send/log
    svstat /service/qmail-smtpd
    svstat /service/qmail-smtpd/log
    qmail-qstat
    ;;
  doqueue|alrm|flush)
    echo "Sending ALRM signal to qmail-send."
    svc -a /service/qmail-send 2>&1 | tee -a $LOG
    ;;
  queue)
    qmail-qstat
    qmail-qread
    ;;
  reload|hup)
    echo "Sending HUP signal to qmail-send."
    svc -h /service/qmail-send 2>&1 | tee -a $LOG
    ;;
  pause)
    echo "Pausing qmail-send"
    svc -p /service/qmail-send 2>&1 | tee -a $LOG
    echo "Pausing qmail-smtpd"
    svc -p /service/qmail-smtpd 2>&1 | tee -a $LOG
    ;;
  cont)
    echo "Continuing qmail-send"
    svc -c /service/qmail-send 2>&1 | tee -a $LOG
    echo "Continuing qmail-smtpd"
    svc -c /service/qmail-smtpd 2>&1 | tee -a $LOG
    ;;
  restart)
    echo "Restarting qmail:"
    echo "* Stopping qmail-smtpd."
    svc -d /service/qmail-smtpd 2>&1 | tee -a $LOG
    echo "* Sending qmail-send SIGTERM and restarting."
    svc -t /service/qmail-send 2>&1 | tee -a $LOG
    echo "* Restarting qmail-smtpd."
    svc -u /service/qmail-smtpd 2>&1 | tee -a $LOG
    ;;
  cdb)
    tcprules /etc/tcp.smtp.cdb /etc/tcp.smtp.tmp < /etc/tcp.smtp 2>&1 |
tee -a $LOG
    chmod 644 /etc/tcp.smtp.cdb
    echo "Reloaded /etc/tcp.smtp."
    ;;
  help)
    cat <<HELP
   stop -- stops mail service (smtp connections refused, nothing goes
out)
  start -- starts mail service (smtp connection accepted, mail can go
out)
  pause -- temporarily stops mail service (connections accepted,
nothing leaves)
   cont -- continues paused mail service
```

```
   stat -- displays status of mail service
    cdb -- rebuild the tcpserver cdb file for smtp
restart -- stops and restarts smtp, sends qmail-send a TERM & restarts
it
doqueue -- sends qmail-send ALRM, scheduling queued messages for
delivery
 reload -- sends qmail-send HUP, rereading locals and virtualdomains
  queue -- shows status of queue
   alrm -- same as doqueue
  flush -- same as doqueue
    hup -- same as reload
HELP
    ;;
  *)
    echo "Usage: $0
{start|stop|restart|doqueue|flush|reload|stat|pause|cont|cdb|queue|help
}"
    exit 1
    ;;
esac

exit 0
```
[7]

Remove the qmail.sh start up link in /usr/local/etc/rc.d directory. [7] We will be starting QMail with the svscan application. [8]

Enter the following commands to create the file structure to support the qmailctl script:

```
echo ./Maildir > /var/qmail/control/defaultdelivery

mkdir –p /var/qmail/supervise/qmail-send/log
mkdir –p /var/qmail/supervise/qmail-smtp/log
chmod +t /var/qmail/supervise/qmail-send
chmod +t /var/qmail/supervise/qmail-smtpd
ln –s /var/qmail/supervise/* /service
```

These scripts are also written by Dave Sill and integrate with the qmailctl script. You need to install them in their respective locations listed below. [7]

This is the /var/qmail/supervise/qmail-send/run script:

```
#!/bin/sh
exec /var/qmail/rc
```

This is the /var/qmail/supervise/qmail-smtpd/run script:

```
#!/bin/sh

QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`
MAXSMTPD=`head -1 /var/qmail/control/concurrencyincoming`
if [ -z "$QMAILDUID" -o -z "$NOFILESGID" -o -z "$MAXSMTPD" ]; then
```

```
    echo QMAILDUID, NOFILESGID, or MAXSMTPD is unset in
    echo $0
    exit 1
fi
exec /usr/local/bin/softlimit -m 2000000 \
    /usr/local/bin/tcpserver -v -R -H -l 0 -x /etc/tcp.smtp.cdb -c
"$MAXSMTPD" \
        -u "$QMAILDUID" -g "$NOFILESGID" 0 smtp /var/qmail/bin/qmail-
smtpd 2>&1 [7]
```

This is the /var/qmail/supervise/qmail-send/log/run script:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog t
/var/log/qmail [7]
```

This is the /var/qmail/supervise/qmail-smtpd/log/run script:

```
#!/bin/sh
exec /usr/local/bin/setuidgid qmaill /usr/local/bin/multilog \
  t /var/log/qmail/smtpd [7]
```

This is the /var/qmail/rc script:

```
#!/bin/sh

DELIVERY=`cat /var/qmail/control/defaultdelivery`
if [ -z "$DELIVERY" ]; then
        echo "/var/qmail/control/defauldelivery is empty or does not
exist" 1>&2 exit 1
fi
exec env - PATH="/var/qmail/bin:$PATH" qmail-start "$DELIVERY" [7]
```

To prevent your mail server from turning into an open relay you'll use tcpserver to limit the hosts that can relay mail on the system. [9] You'll create a file called /etc/tcp.smtp and it should have the localhost and your internal network or clients listed. Here is an example:

```
127.:allow,RELAYCLIENT=""
192.168.75.:allow,RELAYCLIENT=""
```

Once you have added this file run the following command to load this into the tcpserver database:

```
qmailctl cdb [7]
```

Configuring Dovecot

Edit the /etc/rc.conf file inserting the following:

```
dovecot_enable="YES"
```

Copy /usr/local/etc/dovecot-example.conf to /usr/local/etc/dovecot.conf

Edit dovecot.conf to enable POP3S and IMAPS and specify the ports they listen on:

```
protocols = imaps pop3s

imaps_listen = 192.168.75.250:2810
pop3s_listen = 192.168.75.250:2811

ssl_disable = no
```
[10]

Also edit the directory so dovecot will know where to serve mailboxes. [10] Since we are using vpopmail to serve mailboxes the location will be /usr/local/vpopmail/domains/example.com/testuser/Maildir. Also, when we authenticate we will need to specify the username and the domain as so testuser@example.com since vpopmail serves virtual domains and can serve as many different domains as you wish.

```
default_mail_env = maildir:/usr/local/vpopmail/domain/%d/%u/Maildir
```
[10]

We also need to tell it what method we'll be using to authenticate users:

```
auth_userdb = vpopmail
auth_passdb = vpopmail
login_user = dovecot-auth
```
[10]

Next we need to create some certificates to use with Dovecot since we'll be using SSL enabled POP3 and IMAP. Cd into /usr/local/share/doc/dovecot/ and edit the dovecot-openssl.conf with your information. [11] Here is an example:

```
[ req ]
default_bits = 1024
encrypt_key = yes
distinguished_name = req_dn
x509_extensions = cert_type
prompt = no

[ req_dn ]
# country (2 letter code)
C=US

# State or Province Name (full name)
ST=Georgia

# Locality Name (eg. city)
L=Atlanta

# Organization (eg. company)
```

```
O=Example Corp.

# Organizational Unit Name (eg. section)
OU=IMAP server

# Common Name (*.example.com is also possible)
CN=mail.example.com

# E-mail contact
emailAddress=postmaster@example.com

[ cert_type ]
nsCertType = server
```

Once you have this to your liking it's time to create your cert. Simply run ./mkcert.sh to create your certificates. [11] Once it's completed you'll just have to rename the certificates to match what is in your dovecot.conf directory. Copy the imap.pem files in /var/dovecot/ssl/certs and the /var/dovecot/ssl/private to dovecot.pem. [11]

Once that is set dovecot is fully configured.

Sqwebmail

You need to copy over a few files to get sqwebmail working correctly. Cd into /usr/local/share/sqwebmail and copy authdaemonrc.dist to authdaemonrc. Copy sqwebmaild.dist to sqwebmaild.

Qmailadmin

Qmailadmin is already set up. Just make sure that you see qmailmailadmin in /usr/local/www/data and in /usr/local/www/cgi-bin.

Apache

There are a few things to configure in the httpd.conf file to make apache work in our environment. Cd into /usr/local/etc/apache2 and edit the httpd.conf file to change these lines:

```
ServerAdmin www@example.com

ServerName mail.example.com:443
```

Comment out the `Listen 80` [12]

We'll now need to edit the ssl.conf file in the apache2 directory.

Change these lines to fit your environment [12]:

```
ServerName www.example.com:443
ServerAdmin www@example.com

SSLCertificateFile /usr/local/etc/certs/server.cert
SSLCertificateKeyFile /usr/local/etc/certs/server.key

mkdir /usr/local/etc/certs
```

Now, we'll generate our own certificate for this server to use [13]:

```
openssl req -new > server.csr

openssl rsa -in privkey.pem -out server.key

openssl x509 -in server.csr -out server.cert -req -signkey server.key -
days 999
```

Once completed, you'll have your own signed certificate to use with Apache. By the book you should just add this line to /etc/rc.conf and apache configuration would be complete:

```
apache2ssl_enable="YES"
```

However, if you do have problems getting apache to start like I did, then this start up script will start it every time:

```
#!/bin/sh
PREFIX=/usr/local

case "$1" in
start)
        [ "ssl" = "ssl" -a -f "$PREFIX/certs/server.crt" ] && SSL=ssl
        [ -x ${PREFIX}/sbin/apachectl ] && ${PREFIX}/sbin/apachectl
start${SSL}
 > /dev/null && echo -n ' apache2'
        ;;
stop)
        [ -r /var/run/httpd.pid ] && ${PREFIX}/sbin/apachectl stop >
/dev/null &
& echo -n ' apache2'
        ;;
*)
        echo "Usage: `basename $0` {start|stop}" >&2
        ;;
esac

exit 0
```

Move the existing apache2.sh out of /usr/local/etc/rc.d and put the above script in its place. Remember to "chmod 655" the script to make it executable.

You will want to modify the /usr/local/www/data and cgi-bin directories to add your front pages and remove unnecessary files that you do not need. Make

sure to keep the Sqwebmail and QMailadmin directories. You may also want to limit access to the QMailadmin application by allowing access only by username/password. This can be accomplished by the following:

```
cd /usr/local/etc/apache2

htpasswd –c admins QmailAdmin

cd /usr/local/www/data/qmailadmin
```

Create a file called .htaccess with the following [14], [15]:

```
AuthName "Restricted Area"
AuthType Basic
AuthUserFile /usr/local/etc/apache2/admins

require valid-user
```

Create the same file in /usr/local/www/cgi-bin/qmailadmin. Edit the /usr/local/etc/apache2/httpd.conf file as follows [14], [15]:

```
<Directory "/usr/local/www/cgi-bin">
    AllowOverride AuthConfig
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

This will force apache to look for .htaccess files in this directory and force authentication when it finds this file.

You should now be all set with your installation and configuration. Exit out of the mail jail and go into the host environment. Reboot the device to make sure everything will come up after a reboot by typing “shutdown –r now”.

Testing

Processes, Users, and Ports

You should have processes that are similar to the following depending on which ports you have chosen for your applications:

```
Proto Recv-Q Send-Q  Local Address          Foreign Address
(state)
tcp4       0      0  192.168.75.250.443     *.*
LISTEN
tcp4       0      0  192.168.75.134.22      192.168.75.1.1418
ESTABLISHED
tcp4       0      0  192.168.75.250.25      *.*
LISTEN
```

```
tcp4       0      0  192.168.75.250.2811    *.*
LISTEN
tcp4       0      0  192.168.75.250.2810    *.*
LISTEN
```

Each port that we have enabled on this machine for external access is encrypted in some form or fashion with the exception of smtp (TCP port 25). Since we need to talk with other mail servers we cannot have encrypted communication due to the limitations in the smtp protocol.

You'll also notice that a lot of the processes running are running under a non-privileged account. Therefore, if any of these processes is compromised it will be difficult for the intruder to do much damage. One odd point is that when you pull up a list of processes on the host system it will order them by User ID. So, you may see the accounts being run by another user than what the jailed system is running. If there is no user ID match on the host system as there is on the jailed system it will simply show a User ID number. For example, on the host system the user "testuser" has a UID of "1001" and on the jailed system the user "dovecot" has a UID of "1001". The dovecot daemon is running under the dovecot User ID, but will show up on the host system as "testuser".

One benefit with running these processes in the jail is that once the processes have started the jailed environment has no control over stopping or killing them. This can only be done on the host system. In fact, if you try to list processes in the jail you won't get much. Another nice benefit is that since the jail has no access to the physical box it has no access to interfaces. Therefore, your intruder won't be able to sniff traffic if the jailed system ever becomes compromised.

QMail

To test if QMail is working correctly you can run the following test. Telnet to the IP address of the mail server on port 25:

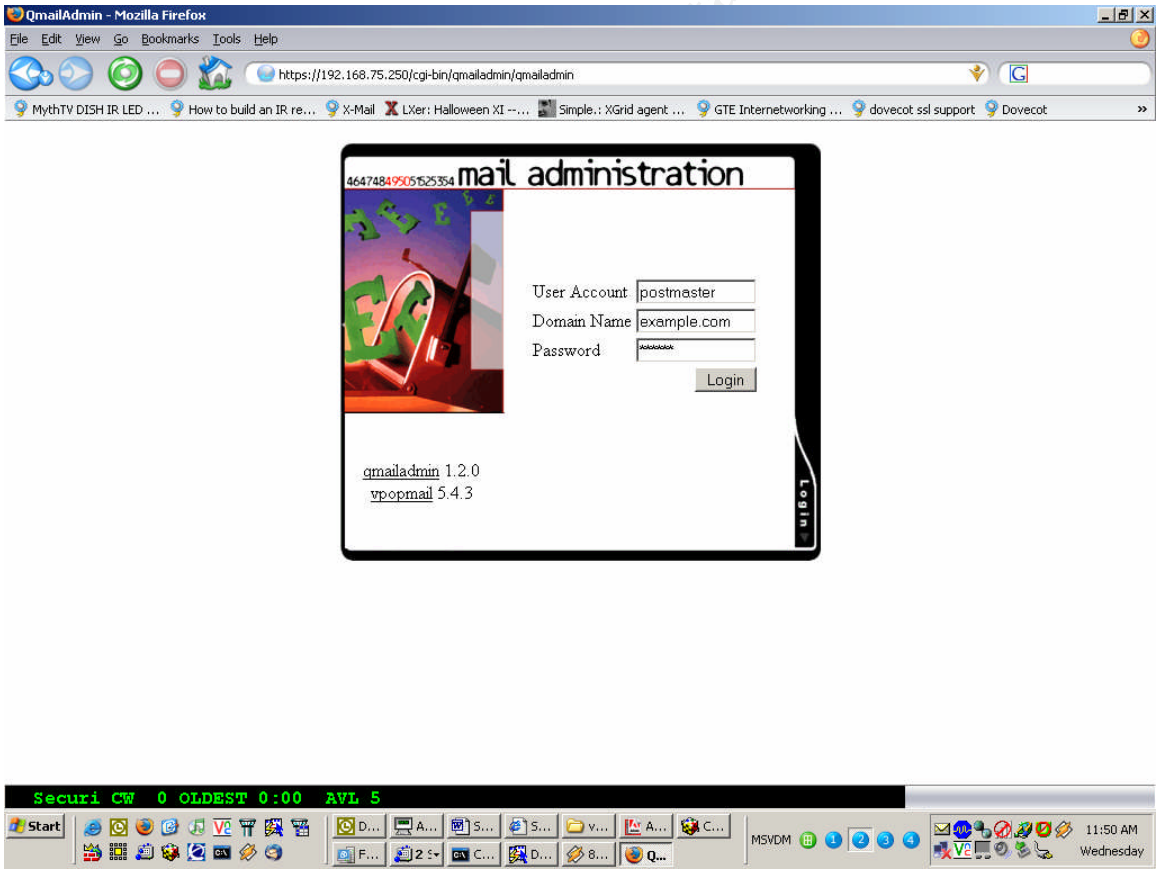
```
> telnet 192.168.75.250 25
Trying 192.168.75.250...
Connected to mail.example.com.
Escape character is '^]'.
220 mail.example.com ESMTP
helo
250 mail.example.com
mail from:<test@somewhere.com>
250 ok
rcpt to:<testuser@example.com>
250 ok
data
354 go ahead
Subject: test

this is a test.
```

```
.
250 ok 1088560033 qp 324
quit
221 mail.example.com
Connection closed by foreign host.
```

QMailadmin

In order to begin using QMailadmin you will already had to have created the domain with the postmaster password using the vpopmail vadddomain program. If you need to add a domain simply run `/usr/local/vpopmail/bin/vadddomain postmaster_password`. To use the application point your browser to https://mail.example.com/qmailadmin/qmailadmin. You'll have to use the username and password combination that you specified earlier in the htpasswd application if you are restricting access to this part of the web server. If successful you will see a screen similar to this one:



Log in using the username "postmaster", domain "example.com", and the postmaster password you supplied earlier. From here you can administer users. The brilliant part about creating users on this system is that users you create will not have accounts on the jailed system. If someone happens to acquire a mail user's username and password combination they will not have any other access

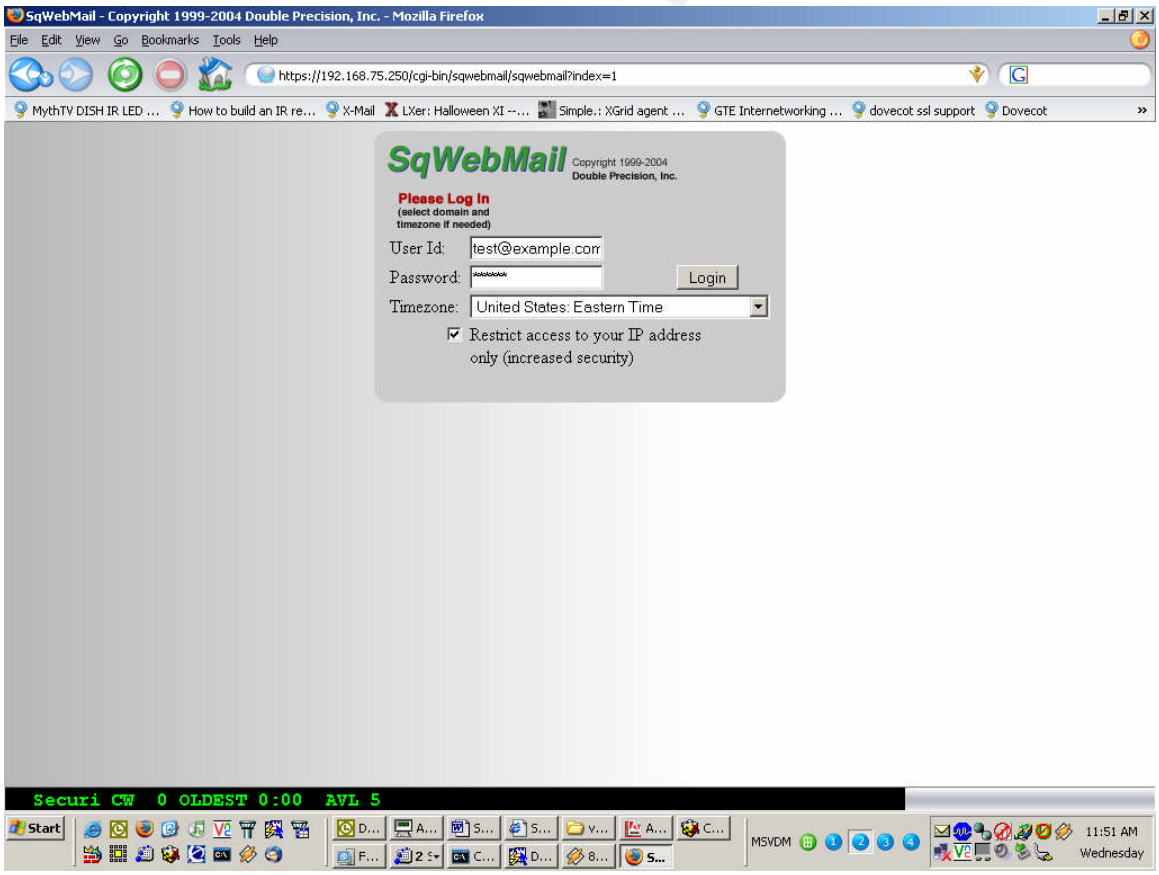
than reading and sending mail with that user's account. Therefore, it's important not to duplicate system accounts and e-mail accounts solely for that exact purpose.

Dovecot

Configure a mail client to connect on the ports you specified for POP3S or IMAPS. You will need to authenticate to the mail server as testuser@example.com. It will first ask you if you want to approve the certificate from the mail server and once you accept it with then authenticate you with a username and password. Now, all the mail that you send to and from the mail server from your client will be encrypted, thus ensuring confidentiality.

Sqwebmail

To test Sqwebmail simple fire up a browser and point it to the following url: https://mail.example.com/cgi-bin/sqwebmail/sqwebmail. If successful you'll see a screen like so:



You will then see the log in screen for Sqwebmail. Once you log in using the entire e-mail address as your login, i.e. testuser@example.com you will see a nice web interface in which you can send and receive messages from anywhere.

The nice part of this is that this setup uses an SSL tunnel. So, your entire session is encrypted in that tunnel ensuring session confidentiality.

File and application locations

In case something goes wrong or a situation calls for troubleshooting it's good to know where logs and files are kept. User's mailboxes will be in /usr/local/vpopmail/domains/domain_name/user_name/Maildir. To check the stats on QMail you can run the "qmailctl stat" command. If for some reason you need to restart QMail you can do so by typing "qmailctl restart".

Apache will log to a few files in /var/log http-access.log, http-error.log, and http-ssl_request.log. Dovecot will log to syslog or /var/log/messages. QMail will log in its own directory /var/log/qmail. You should be able to tackle most any situation with a peek through these logs.

Maintenance and Upgrading

Keeping your system and applications up to date is simple using the ports system and CVSup. A must is to subscribe to the FreeBSD security mailing list at http://lists.freebsd.org/mailman/listinfo/freebsd-security-notifications. [16] FreeBSD is very good at supplying patches to vulnerabilities as soon as they come out. It's a must to patch your system and applications when new security advisories come out. They don't come out often, but when they do you need to be on top of it.

To update your system you can wait until new milestones are announce by keeping current with the FreeBSD announcement list. You can sign up for that here: http://lists.freebsd.org/mailman/listinfo/freebsd-announce. [16] You could also set up a cron job to run "`cvsup -g -L 2 /etc/stable-supfile && make world`" a certain day of the week at late hours.

The least that you want to do is update your ports system daily or weekly. This can be accomplished by running the cvsup command string in the jailed environment. Be conscious that you may want to back up your configuration files first before upgrading just to be sure nothing gets overwritten.

Conclusion

Now you should have a secure mail server environment that is easy to manage and maintain. With this type of installation you have a tiered security structure that makes no one process a dam breaker to compromising the system. Of course, no system architecture is perfect, but with the separation of non-privileged processes running in a jailed environment the possibility of a remote root compromise of the host system is fairly diminished. Also, with this system you maintain confidentiality and privacy issues that arise from clients sending and receiving mail. No mail or authentication from a client is ever transferred in clear text. Simply put, a SOHO can easily implement a secure and cheap mail solution that easy to manage and maintain. Using the jail system and FreeBSD

the SOHO can also migrate other services to this model, providing fairly secure solutions with a very small investment.

References

[1] FreeBSD Project, The. “Starting the Installation”, June 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-start.html

[2] Sterns, William. “The Top Ten Reasons Not to Run as Root v.8”, October 2000. http://www.ists.dartmouth.edu/IRIA/knowledge_base/linuxinfo/topten-noroot.v8.htm

[3] FreeBSD Project, The. “Using CVSup”, June 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html

[4] FreeBSD Project, The. “Configuring the FreeBSD Kernel”, June 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-config.html

[5] FreeBSD Project, The. “FreeBSD Hypertext Man Pages: jail(8)”, December 2001. http://www.freebsd.org/cgi/man.cgi?query=jail&apropos=0&sektion=0&manpath=FreeBSD+4.10-RELEASE&format=html

[6] FreeBSD Project, The. “Installing Applications: Packages and Ports”, June 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports-using.html

[7] Sill, Dave. “Life with QMail”, March 2004. http://www.lifewithqmail.org/lwq.html#configuration

[8] Bernstein, D.J. “daemontools”, June 2004. http://cr.yp.to/daemontools.html

[9] Bernstein, D. J. “ucspi-tcp”, June 2004. http://cr.yp.to/ucspi-tcp.html

[10] Dovecot Project, The. “Secure IMAP server”, June 2004. http://www.dovecot.org/doc/configuration.txt

[11] Op de Beeck, Steven. “Dovecot with SSL support”, March 2004. http://annika.studentenweb.org/documents/dovecot/dovecot_ssl_support.html

[12] Apache Software Foundation, The. “Apache HTTP Server Version 2.0”, June 2004. http://httpd.apache.org/docs-2.0/configuring.html

[13] de Louw, Luc. “Apache Compile HOWTO (Linux edition)”, June 2002. http://www.learninglinux.com/HOWTO/Apache-Compile-HOWTO/apache.html

[14] Apache Week. "Using User Authentication", October 1996. http://www.apacheweek.com/features/userauth

[15] Apache Software Foundation, The. "Apache HTTP Server Version 2.0", June 2004. http://httpd.apache.org/docs-2.0/howto/htaccess.html

[16] FreeBSD Project, The. "List Summary", June 2004. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/eresources.html#ERESOURCES-SUMMARY