Global Information Assurance Certification Paper

Implementing a SPAM Filtering Gateway with Apache James

Kraig P Schario

GIAC Security Essential Certification (GSEC)
Practical Assignment
Version 1.4b
Option 1

January 19, 2004

Abstract

This paper discusses the configuration of a SPAM Filtering Gateway using the Java Apache Mail Enterprise Server, James[1], developed by The Apache Software Foundation. The configuration describes the setup of an Apache James mail server to process incoming mail for spam properties before delivering the mail to an internal server. It focuses on reducing the processing load of the internal mail server by dropping mail relay attempts, flagging messages as SPAM for end-user management, and lowering exposure to common vulnerabilities found in many of the popular mail servers on the market today[2].

Both RedHat Linux 9.0 and Windows 2000/XP installations are discussed, as well as performance and security considerations. This paper assumes basic knowledge of the following: TCP/IP Networking, DNS, SMTP, and firewalls. It is not intended as a HOW-TO on the configuration, installation, or securing of firewalls, RedHat Linux 9, Windows 2000/XP, DNS, or MySQL.

The configuration described herein demonstrates the ability to setup a cost effective SPAM solution using open source software with minimal hardware requirements. Taking advantage of blacklists, whitelists, reverse dns lookups, and a Bayesian filter based upon the research described by Graham in "A Plan for SPAM" and "Better Bayesian Filtering"[3], SPAM is quickly identified for end-user management.

James Overview

The James mail server is a full function mail platform. It supports SMTP, POP3, and NNTP. Written completely in Java, it is an operating system independent mail server. Its powerful Mailet API make it a highly extensible and modular mail platform, offering "new possibilities for what's often been dubbed the Internet's first killer application"[4].

The SpoolManager component is the mail processing engine in James. It is broken into processors that contain matcher and mailet pairs. Matchers determine if the mail message meets the specified condition and mailets process the mail message based on the result of the matcher. Mailets also make it possible to move a message to another processor and prevent further processing of the current processor.

There are two processors required by the SpoolManager: *root* and *error*. All mail is first processed through the *root* processor and if an error occurs, then through the *error* processor. Two additional processors, *transport and spam*, are included to handle mail delivery.

[1] Apache James. The Apache Software Foundation. Web Site.
[2] Cert Coordination Center. CERT/CC Vulnerability Notes Database. Web Site.
[3] Graham, Paul. Web Site.
[4] Duguay, Claude. IBM developerWorks. Web Site.

Listing 1: Sample Matcher/Mailet pair from config.xml
The matcher determines if the message is destined for the domain: *domain-name.com*. If the result is true, the mailet moves the message to the "spamcheck" processor.

```
<processor name="root">
      <mailet match="HostIs=domain-name.com" class="ToProcessor">
            <processor>spamcheck</processor>
      </mailet>
</processor>
```

In addition to its powerful API, James offers a flexible storage system. It supports three types of data storage allowing administrators to optimize James for their needs.

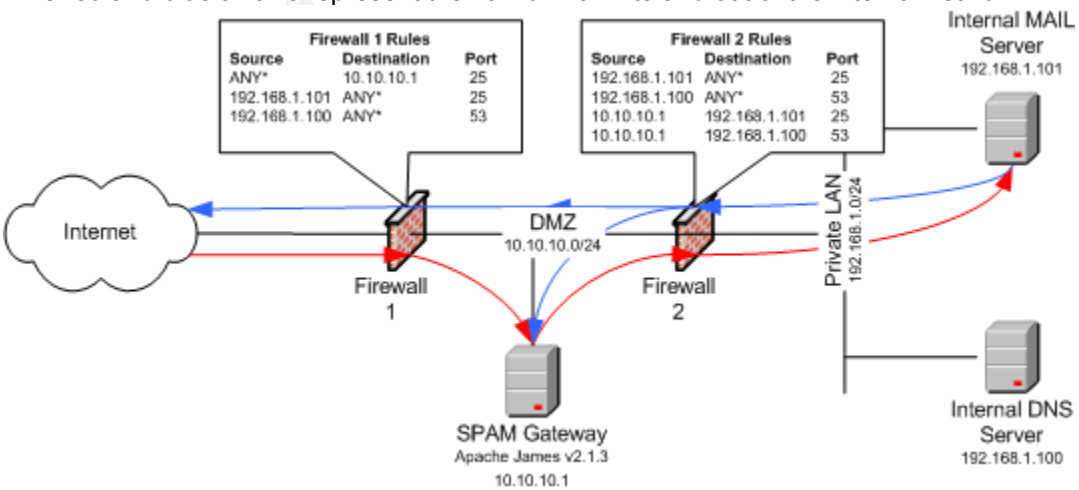
1) File Repositories: Stores data in the file system.
2) Database Repositories: Stores data in a database using a JDBC Driver.
3) DBFile Repositories for mail messages: Stores the message body in the file system and the message headers in a database.

Modification of the default processors' matcher and mailet pairs, the ability to add new processors, multiple mail repositories, and development of custom matchers and mailets creates a highly customizable mail platform. This modularity makes James the ideal platform for a SPAM Gateway.

Gateway Configuration Overview

The gateway configuration described is based on the network diagram shown in Figure 1. The SPAM Gateway is placed in a DMZ on the network. It is possible to place this server on the Private LAN; however, by placing it in the DMZ any compromise of this server should be contained within the DMZ provided the firewall(s) are properly configured.

Figure 1: Gateway Network Diagram
The red and blue arrows represent the flow of mail into and out of the internal network.



A DMZ can consist of one or more firewalls and is both hardware and software dependent. Consult the firewall documentation for specific details on setting up traffic rules, traffic redirection, and DMZ configurations.

The firewalls are configured to allow the following traffic flow:

Firewall 1 Traffic Flow:

1) SMTP Traffic IN from the Internet and redirected to the SPAM Gateway.
2) SMTP Traffic OUT from the Internal MAIL Server.
3) DNS Query Traffic OUT from the Internal DNS Server.

Sample Rules for Firewall 1

Source	Destination	Port
ANY*	10.10.10.1	25
192.168.1.101	ANY*	25
192.168.1.100	ANY*	53

Firewall 2 Traffic Flow:

1) SMTP Traffic IN from the SPAM Gateway
2) SMTP Traffic OUT from the Internal MAIL Server
3) DNS Query Traffic OUT from the Internal DNS Servers

Sample Rules for Firewall 2

Source	Destination	Port
192.168.1.101	ANY*	25
192.168.1.100	ANY*	53
10.10.10.1	192.168.1.101	25
10.10.10.1	192.168.1.100	53

In addition to proper firewall setup, proper configuration of both an external and an internal DNS server are crucial to meeting the following objectives of the Gateway configuration.

Configuration Objectives

1) Flag Mail as SPAM for end user management.
2) Enable address whitelisting on a per user basis.
3) Lower bandwidth requirements:
    a. Only process mail destined for the specified domain.
    b. Lower bandwidth requirements by preventing NDRs (non-deliverable reports) on message relay attempts.
4) Lower exposure to vulnerabilities found in popular mail servers.

Configuring the Gateway

Hardware requirements are dependent upon the size of the network and volume of traffic. The following configuration was successfully tested on a Pentium III 1.0 GHz machine with 512 MB of RAM and a 20 GB hard drive. Both Windows 2000/XP and RedHat Linux 9.0 were used during testing. Any SMTP server software installed on either operating system must be shutdown and disabled for a successful installation, i.e. Sendmail, IIS SMTP Server, etc. A secured installation of either operating system is required.

Required Software

The SPAM Gateway requires the following software packages. The packages are available for each operating system where required. While the software is

available from reliable sources, before installing a program obtained from the Internet, especially those with multiple mirror sites, the source should be verified. The MySQL and Apache James download sites provide MD5 checksums and/or GnuPG signatures of the files to verify their authenticity. Prior to downloading the software, note the checksum and/or signature provided. To verify the checksum or signature of the downloaded file, use the md5sum, rpm, or gpg utilities on RedHat Linux 9.0 and the md5 or gpg utilities on Windows 2000/XP. Compare the resulting checksum/signature to the one provided by the download source. If the package is verified with gpg, be sure to download and import the public key. See the following web sites for more information and specific usage instructions of the commands:

MySQL Manual: Verifying Package Integrity with MD5 Checksum and GnuPG
http://www.mysql.com/doc/en/Verifying_Package_Integrity.html
MD5 Command Line Message Digest Utility for Unix or MS-DOS/Windows
http://www.fourmilab.ch/md5/
GNU Privacy Guard (GnuPG)
http://www.gnupg.org/(en)/download/integrity_check.html

Java Runtime Environment (JRE) version 1.3 or newer

The SUN Java 2 Standard Edition JRE version 1.4.2 was used during testing. The JRE for both RedHat Linux 9 and Windows 2000/XP is available for download at the following URL: http://java.sun.com/j2se/1.4.2/download.html

Open a web browser and navigate to the above URL. Click the appropriate download link for the selected operating system. Accept the SUN License Agreement and proceed to the download page. For a RedHat 9 installation, download the self-extracting RPM package: j2re-1_4_2_03-linux-i586-rpm.bin, and for a Windows 2000/XP installation download the Windows Offline Installation package: j2re-1_4_2_03-windows-i586-p.exe. Save the selected file to one of the following locations (For Linux installations substitute the current user's login name for $USER):

```
RedHat Linux 9:   /home/$USER/downloads
Windows 2000/XP:  C:\Temp
```

MySQL Server and Client versions 3.23 or 4.0

MySQL Server version 4.0.17 was used during testing and is available at the following URL: http://www.mysql.com/downloads/mysql-4.0.html.

MySQL v3.23 may be installed as part of the RedHat Linux 9.0 installation. Only install a new version if required. For RedHat Linux, download the appropriate Server and Client RPM packages: MySQL-server-4.0.17-0.i386.rpm and MySQL-client-2.0.17-0.i386.rpm, and for Windows 2000/XP download the Windows Installer based zip file: mysql-4.0.17-win.zip. Note the MD5 checksum for future reference. Choose the Pick a Mirror link for a list of download sites and download the file from the closest mirror site. Save the selected file to same location as the JRE.

Apache James version 2.1.3

Version 2.1.3 was the latest stable release available at the time of this writing. It is available in binary or source formats. The binary format used by this document is available at the following URL: http://jakarta.apache.org/site/binindex.cgi For RedHat Linux download the james-2.1.3.tar.gz file and for Windows 2000/XP download the james-2.1.3.zip file. Save the selected file to the same location as the JRE. Save the GnuPG signature of the selected file for future reference.

James-Praxis version 1.1.6/16

James-Praxis is a set of open source Matchers and Mailets for James that include the Bayesian SPAM Analysis and Whitelist Management Matchers and Mailets used by the gateway. James-Praxis is available at the following URL: http://portale.praxis.it/pub/james/james-praxis.zip. The same file is used for both the RedHat Linux 9.0 and the Windows 2000/XP operating systems.

Software Installation Steps

Sun Java Runtime Environment (JRE) version 1.4.2

RedHat Linux 9.0

To install the JRE on RedHat Linux, open a terminal window or a telnet session and change directories to /home/$USER/downloads. To install the RPM package, root access is required. Use the su command to obtain root privileges.

```
[$USER@localhost $USER]$ cd downloads
[$USER@localhost downloads]$ su
Password:******
```

When prompted, enter the root password. The prompt should switch from a $ sign to a # sign. At the prompt extract the RPM Package from the binary file. Execute the following command.

```
[root@localhost downloads]# ./j2re-1_4_2_03-linux-i586-rpm.bin
```

The **./** prior to the file name is required to execute the command in the current directory. Accept the license agreement when prompted. The j2re-1_4_2_03-linux-i586.rpm file will be extracted in the current directory. To install the RPM package, execute the following command:

```
[root@localhost downloads]# rpm -v -i j2re-1_4_2_03-linux-i586.rpm
```

The JRE will be installed to the following directory: /usr/java/j2re-1.4.2_03

In order for James to use the SUN JRE, the JAVA_HOME variable must be set in the environment. To set this variable, the /etc/profile file must be modified. With root access, use the vi editor to modify the system profile.

```
[root@localhost downloads]# vi /etc/profile
```

Scroll down to the first blank line after the MAIL variable declaration and press the **I** key to insert new text. Type the following line:

```
JAVA_HOME="/usr/java/j2re-1.4.2_03"
```

Scroll down to the **export** line and move the cursor to the end of the line. The exported environment variables are separated by spaces. At the end of the line, type JAVA_HOME. Press the **ESC** key, then the **<SHIFT> :** key combination. The vi command line will become active at the bottom left of the screen. Type **wq** (write, quit) and press **<ENTER>** to save the file and exit. The JAVA_HOME variable will be available to the James installation on next login or server boot.

Listing 2: JAVA_HOME Variable Declaration in /etc/profile

```
USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"
JAVA_HOME="/usr/java/j2re1.4.2_03"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC JAVA_HOME
```

Windows 2000/XP

To install the JRE on Windows 2000 or XP, login to Windows as a user with administrative rights to the server, and click Start > Run. In the Open field type: c:\temp\j2re-1_4_2_03-windows-i586-p.exe. Click OK to start the installation. Accept the license agreement, and click Next. Select the Typical installation option and click Next. When the installation finishes, click Finish. If prompted to restart the operating system, click Yes.

The JRE will be installed to the following directory: C:\Program Files\java\j2re1.4.2_03

James requires the JAVA_HOME environment variable. To set this variable, open the Control Panel and open the System icon. Click the Advanced tab and then click the Environment Variables button. Click the New... button under the System variables located near the bottom of the dialog window. In the Variable name field type: JAVA_HOME. In the Variable value field type: C:\Program Files\java\j2re1.4.2_03. Click OK three times to set the variable.

Figure 2: System Variable Dialog

MySQL Server version 4.0.17

The following instructions assume that a previous MySQL Server installation does not exist.

RedHat Linux 9.0

To install MySQL on RedHat Linux, open a terminal window or a telnet session and change directories to /home/$USER/downloads. To install the RPM packages, root access is required. Prior to installing the packages, verify their authenticity with the md5sum command. Compare the checksum produced to the checksum provided by the download source before installing the software. Execute the following commands to verify the source file and start the MySQL Server and Client installations.

```
[root@localhost downloads]# md5sum MySQL-server-4.0.17-0.i386.rpm
821f9cf17c1ba0e43828e4f49c93de1b  MySQL-server-4.0.17-0.i386.rpm
[root@localhost downloads]# rpm -v -i MySQL-server-4.0.17-0.i386.rpm
[root@localhost downloads]# md5sum MySQL-client-4.0.17-0.i386.rpm
64ff4c7fb45eb48c307e0f0e80c60d05  MySQL-client-4.0.17-0.i386.rpm
[root@localhost downloads]# rpm -v -i MySQL-client-4.0.17-0.i386.rpm
```

Following the server and client installation, the MySQL server must be started to set the root user's password. Execute the following commands to start the server and set the root password:

```
[root@localhost downloads]# /etc/inti.d/mysqld start
[root@localhost downloads]# /usr/bin/mysqladmin -u root password <new pass.>
```

The RPM installation will configure RedHat to automatically start MySQL on system boot.

Windows 2000/XP

Prior to extracting the installation packages, verify their authenticity with the md5 command. Compare the checksum produced to the checksum provided, fa8e81989f7cb62ba3d740b7ced10dbd, by the download source before installing the software. To install MySQL on Windows 2000 or XP, extract the mysql-4.0.17-win.zip file to the c:\temp\mysqld directory using WinZip or other archive utility. Click Start > Run. In the Open field type c:\temp\mysqld\setup.exe and click OK to start the installation. Click Next accepting the defaults. When prompted, select Typical as the Setup Type and click Next to start the installation.

To complete the MySQL installation, the MySQL Service and default user accounts must be setup. Click Start > Run. In the Open field type C:\mysql\bin\winmysqladmin.exe and click OK to start the administration tool. When prompted enter a user name and password. The user name *root* is typically used as the default system administrator name. Remember these values, as they will be used to configure the MySQL database server.

Figure 3: MySQL Quick Setup Dialog

The administration tool will create the c:\%windir%\my.ini configuration file and setup the MySQL service to start automatically when Windows starts.

Listing 3: my.ini created by WinMySQLAdmin tool

```
#This File was made using the WinMySQLAdmin 1.4 Tool
#1/11/2004 5:32:42 PM

#Uncomment or Add only the keys that you know how works.
#Read the MySQL Manual for instructions

[mysqld]
basedir=C:/mysql
#bind-address=192.168.1.101
datadir=C:/mysql/data
#language=C:/mysql/share/your language directory
#slow query log#=
#tmpdir#=
#port=3306
#set-variable=key_buffer=16M
[WinMySQLadmin]
Server=C:/mysql/bin/mysqld-nt.exe
user=root
password=<new password>
```

Apache James version 2.1.3

RedHat Linux 9.0

James will be installed to the to the /usr/local/james-2.1.3 directory. To install James, open a terminal window or a telnet session and change directories to /home/$USER/downloads. Root access is required. Prior to installing the

package, verify its authenticity with the gpg command. Execute the following commands to extract and expand the james-2.1.3.tar.gz file in the /usr/local directory, and create a symbolic link referring to the installation directory.

```
[root@localhost downloads]# cd /usr/local
[root@localhost local]#
            gunzip < /home/$USER/downloads/james-2.1.3.tar.gz | tar xvf -
[root@localhost local]# ln -s james-2.1.3 james
```

To complete the installation, the James mail server must be started to finish extracting the configuration files. Note: James will not start if Sendmail or other SMTP server is running. Execute the following commands to stop the Sendmail daemon and remove the startup script symbolic link disabling automatic start on server boot.

```
[root@localhost local]# /etc/init.d/sendmail stop
[root@localhost local]# cd /etc/rc5.d          (/etc/rc3.d for Non-GUI Boot)
[root@localhost rc5.d]# rm -f S80sendmail
```

The permissions on the startup scripts must modified with the execute permission. Execute the following commands to update the file permissions, start and then stop the James Server.

```
[root@localhost rc5.d]# cd /usr/local/james/bin
[root@localhost bin]# chmod +x run.sh
[root@localhost bin]# chmod +x phoenix.sh
[root@localhost bin]# ./run.sh
```

To stop the server, press the <Control> + C key combination.

Finally, the James startup scripts will be created to start James automatically. The startup scripts listed in Appendixes A and B are directly adapted from the EJB Solutions Out-of-the-Box version 1.0, "James Installation Procedures," documentation.[5] Only slight modifications have been made to the original scripts. A start and stop script, go.sh, will be created in the /usr/local/james/bin directory, a startup script, james, will be created in the /etc/init.d directory, and symbolic links placed in the appropriate run level directories, /etc/rc3.d for run level 3 and /etc/rc5.d for run level 5, to start James on server boot. Execute the following command to create the required scripts.

Copy the script from Appendix A into the vi editor and save the file. Add the execute permission to the file.

```
[root@localhost bin]# vi go.sh
[root@localhost bin]# chmod +x go.sh
```

Create the james startup script in the /etc/init.d directory. Copy the script from Appendix B into the vi editor and save the file. Add the execute permission to the file.

[5] Out-of-the-Box, EJB Solutions. Web Site.

```
[root@localhost bin]# cd /etc/init.d
[root@localhost init.d]# vi james
[root@localhost init.d]# chmod +x james
```

Create the symbolic links in the appropriate run level directories.

```
[root@localhost init.d]# cd /etc/rc5.d    (/etc/rc3.d for Non-GUI Boot)
[root@localhost rc5.d]# ln -s ../init.d/james S99james
```

The james startup script is ready to use. To start or stop the James daemon process, execute one of the following commands.

```
[root@localhost rc5.d]# cd /etc/init.d
[root@localhost init.d]# ./james start
Starting james daemon: [  OK  ]
[root@localhost init.d]# ./james stop
Stopping james daemon: [  OK  ]
```

Windows 2000/XP

James will be installed to the c:\java\james-2.1.3 directory. Alternatively it may be installed to the Program Files directory or other drive. Prior to extracting the james-2.1.3.zip file, verify its authenticity with the gpg command. Create the java directory off the root of the C: drive, and using WinZIP or other archive utility, extract the c:\temp\james-2.1.3.zip file to the c:\java directory.

To complete the installation, the James mail server must be started to finish extracting the configuration files. Click Start > Run. In the Open field type: c:\java\james-2.1.3\bin\run.bat and click on OK. The James server will start up in a command window. Stop the server by pressing the <Control> + C key combination. When prompted to terminate the batch job, press Y + <Enter>. Note: James will not start if the Simple Mail Transport Service is running. This service must be stopped prior to running James and the Startup Type set to Manual or Disabled.

To setup James as a Windows Service, open a command prompt window. Change directories to the c:\java\james-2.1.3\bin directory. Execute the following command to create the 'James Mail Server 2.1' Service.

```
C:\java\james-2.1.3\bin\wrapper.exe -i ..\conf\wrapper.conf
wrapper  | James Mail Server 2.1 installed.
```

The net command or the Services MMC Application may be used to start and stop the James server. By default, the service will automatically start when Windows starts.

```
C:\>net start "James 2.1"
The James Mail Server 2.1 service is starting......
The James Mail Server 2.1 service was started successfully.

C:\>net stop "James 2.1"
The James Mail Server 2.1 service is stopping..
```

```
The James Mail Server 2.1 service was stopped successfully.
```

James-Praxis version 1.1.6

The james-praxis.zip archive contain the source code, readme files, and java classes (it.praxis.james.jar file) that must be integrated into the James SAR (Server Application Resource) file. "A SAR file is a Phoenix concept. James runs in the Phoenix server infrastructure, which part of the Avalon project. A SAR file is JAR file (which in turn is a zip file) with a specified structure."[6] To integrate the jar file, the james-praxis.zip file must be decompressed. On Windows 2000 or XP, extract the james-praxis.zip file to the c:\temp directory using WinZip or other archive utility and on RedHat Linux using the unzip utility. The archive will be extracted into the james-praxis subdirectory.

```
[root@localhost downloads]# unzip james-praxis.zip
```

The /james-praxis/jars/SAR-INF/lib/it.praxis.james.jar file must be added to the james.sar file located in the /usr/local/james-2.1.3/apps directory on RedHat Linux and in the c:\java\james-2.1.3\apps directory on Windows 2000/XP. Presently the only way to add custom Java classes, in JAR form, is to recreate the james.sar file.[6] On Windows 2000 or XP, create a subdirectory called james-sar under the c:\temp directory. Extract the james.sar file to the c:\temp\james-sar directory using WinZip or other archive utility. On RedHat Linux, in the /home/$USER/downloads directory create a james-sar directory. Execute the following commands to extract the james.sar file.

```
[root@localhost downloads]# mkdir james-sar
[root@localhost downloads]# cd /usr/local/james/apps
[root@localhost apps]# unzip james.sar -d /home/$USER/downloads/james-sar
```

Copy the /james-praxis/jars/SAR-INF/lib/it.praxis.james.jar to the /james-sar/SAR-INF/lib directory. On Windows 2000 or XP use WinZIP or other archive utility to recreate the james.sar file. Using WinZIP, in the c:\temp\james-sar directory, highlight the conf, META-INF, and SAR-INF directories. Right click and select Add to Zip file from the drop-down menu. In the Add to Archive field type: c:\temp\james-sar\james.sar. Make sure the Save Full Path Info option is not selected and click Add. Copy the new c:\temp\james-sar\james.sar file to the c:\java\james-2.1.3\apps directory. On RedHat Linux using the zip utility, recreate the james.sar archive and copy it to the /usr/local/james/apps directory by executing the following commands.

```
[root@localhost apps]# cd /home/$USER/downloads/james-sar
[root@localhost james-sar]# zip -r james.sar *
[root@localhost james-sar]# cp james.sar /usr/local/james/apps/james.sar
```

To complete the James-Praxis installation, the MySQL database that will hold the database tables for the Whitelist Manager and the Bayesian Filters must be configured. The MySQL commands are the same for both RedHat Linux and Windows 2000/XP and are executed from the MySQL Monitor. A

[6] Duguay, Claude. IBM developerWorks. Web Site.

database named **Spam** and a user named **spamd** will be created. Prior to starting the monitor, the whitelist.sql file for MySQL must be created in the src/it/praxis/james/tools subdirectory of the james-praxis directory. See Appendix C for the SQL commands required to create the whitelist.sql file.

Open the MySQL Monitor. On RedHat Linux execute the following command to open the monitor: `[root@localhost root]# /usr/bin/mysql -u root -p`. On Windows 2000/XP click Start > Run. In the Open field type: c:\mysql\bin\mysql -u root -p. Enter the root user's password when prompted. Enter the following commands to configure the database. A semicolon must be at the end of each command to signal the Monitor to execute it.

Create the database:

```
mysql> CREATE DATABASE Spam;
```

Create the spamd user and set permissions and password:

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER
    -> ON Spam.* TO spamd@localhost
    -> IDENTIFIED BY 'password';
```

Create the Bayesian Filter and Whitelist tables. Place the correct path before each file name, or copy the files to the current directory: The files are located in the /james-praxis/src/it/praxis/james/tools directory.

```
mysql> USE Spam;
mysql> source <path to file>SpamDatabase.sql
mysql> source <path to file>whitelist.sql
```

Configuration

To complete the gateway configuration, the DNS namespaces must be added and the James configuration file modified.

DNS Server Namespaces

Add the *domain-name.com* and sub-domain spam.*domain-name.com* DNS namespaces to the Internal DNS Server. Substitute the appropriate domain name for *domain-name.com*. The Internal MAIL server must use the Internal DNS server as its Primary DNS Server to ensure its ability to deliver mail to the SPAM Gateway. It is important that the Internal DNS server is configured to process DNS queries for domains not hosted by the server. Based on the diagram in Figure 1, the following DNS entries would be added.

```
@      IN    A     192.168.1.101
mail   IN    A     192.168.1.101
@      IN    MX    10    mail.domain-name.com.
spam   IN    A     10.10.10.1
spam   IN    MX    10    spam.domain-name.com.
```

James Configuration File

James uses XML for its configuration files. The main configuration file, config.xml is located in the following directory on Windows: C:\java\james-2.1.3\apps\james\SAR-INF and /usr/local/james/apps/james/SAR-INF on RedHat Linux. The file is very intuitive and well documented. The XML blocks will be modified in the following order: <James>, <fetchpop>, <spoolmanager>, <dnsserver>, <remotemanager>, <pop3server>, <smtpserver>, <nntpserver>, and <data-sources>. Open the config.xml file with a standard text editor, vi on RedHat Linux or WordPad on Windows 2000/XP and make the following changes.

<James> XML Block

Modify the postmaster's address. This is the address where administrative messages will be sent.

```
<postmaster>postmaster@domain-name.com</postmaster>
```

Identify the DNS namespaces serviced by the James server and change the autodetect settings to *false,* explicitly defining the server names and IP addresses to be used.

```
<servernames autodetect="false" autdetectIP="false">
   <servername>domain-name.com</servername>
   <servername>spam.domain-name.com</servername>
</servernames>
```

<fetchpop> XML Block

The fetchpop service is not required by the gateway and will be disabled.

```
<fetchpop enabled="false">
```

<spoolmanager> XML Block

The SpoolManager controls the flow of mail through the James server. This XML Block contains the processors that control the SPAM Gateway, and custom matcher and mailet declarations. The following setup of the SpoolManager was adapted from the James-Praxis documentation, "Matchers and Mailets for James."[7] To enable the James-Praxis Matchers and Mailets, add the following Java package declarations to the <mailetpackages> and <matcherpackages> XML Blocks.

```
<mailetpackages>
   <mailetpackage>org.apache.james.transport.mailets</mailetpackage>
   <mailetpackage>it.praxis.james.mailets</mailetpackage>
</mailetpackages>
<matcherpackages>
   <matcherpackage>org.apache.james.transport.matchers</matcherpackage>
   <matcherpackage>it.praxis.james.matchers</matcherpackage>
</matcherpackages>
```

[7] Gianferrari Pini, Vincenzo. Praxis Calcolo S.p.A. Web Site.

Seven processors will be defined to aid in the identification of SPAM. Within the <spoolmanager> XML Block, configure the following processors.

1) The *root* processor is the first processor executed by the SpoolManager. It is configured to prevent message relaying, and redirect mail to other processors to maintain user's whitelists, update the Bayesian databases, and process mail for spam properties. The *root* processor is configured as follows:

```
<processor name="root">
   <!-- Important check to avoid looping -->
   <mailet match="RelayLimit=30" class="Null"/>

   <!-- Never block messages to the postmaster -->
   <mailet match="RecipientIs=postmaster@domain-name.com"
      class="ToProcessor">
      <processor> transport </processor>
   </mailet>

   <!-- Bayesian Feeder: Update the HAM and SPAM Bayesian DB -->
   <!-- Tables with mail forwarded from end users.           -->
   <mailet match="RecipientIs=ham@spam.domain-name.com
      spam@spam.domain-name.com" class="ToProcessor">
      <processor>bayesianfeeder</processor>
   </mailet>

   <!-- Whitelist management -->
   <!-- Updates the Whitelist DB Tables. End users can insert or -->
   <!-- remove addresses, and display their current whitelist    -->
   <mailet match="RecipientIs=whitelist@spam.domain-name.com"
      class="ToProcessor">
      <processor>whitelistmgr</processor>
   </mailet>

   <!-- Process all mail destined for our domain for SPAM Properties -->
   <mailet match="HostIs=domain-name.com" class="ToProcessor">
      <processor>spamcheck</processor>
   </mailet>

   <!-- Drop All other Mail. Message relay prevented by only   -->
   <!-- processing mail destined for the domain-name.com above -->
   <mailet match="All" class="Null"/>
</processor>
```

2) The *bayesianfeeder* processor updates the Bayesian database tables used by the *spamcheck* processor when statistically evaluating messages for SPAM properties. Mail forwarded to the spam@spam.*domain-name.com* address updates the message tokens indicating SPAM, and mail forwarded to the ham@spam.*domain-name.com* address updates the message tokens indicating Non-SPAM. Refer to "A Plan for SPAM" by Graham[3] and "A Statistical Approach to the Spam Problem" by Robinson[8] for a detailed

[3] Graham, Paul. Web Site.
[8] Robinson, Gary. Linux Journal. Web Site.

explanation on the use of statistical analysis to identify SPAM. The *bayesianfeeder* processor is configured as follows.

```
<processor name="bayesianfeeder">
    <!-- "not spam" bayesian analysis feeder. -->
    <mailet match="RecipientIs=ham@spam.domain-name.com
       class="JDBCBayesianAnalysisFeeder">
       <repositoryPath> db://spamdb </repositoryPath>
       <feedType>ham</feedType>
       <hamTable>bayesiananalysis_ham</hamTable>
       <messageCountTable>bayesiananalysis_messagecounts</messageCountTable>
    </mailet>

    <!-- "spam" bayesian analysis feeder. -->
    <mailet match="RecipientIs=spam@spam.domain-name.com"
       class="JDBCBayesianAnalysisFeeder">
       <repositoryPath> db://spamdb </repositoryPath>
       <feedType>spam</feedType>
       <spamTable>bayesiananalysis_spam</spamTable>
       <messageCountTable>bayesiananalysis_messagecounts</messageCountTable>
    </mailet>

    <!-- Sent here by mistake -->
    <mailet match="All" class="ToProcessor">
       <processor>error</processor>
          <notice>Entered Feeder by Mistake</notice>
    </mailet>
</processor>
```

3) The *whitelistmgr* processor enables end users to manage their personal whitelist. Users can *insert*, *remove*, and *display* their whitelist by sending a ***text based*** email message to the Whitelist Manager's address: whitelist@spam.*domain-name.com*. The subject line contains the command executed by the manager, and the body of the message contains the list of addresses to process. One address per line. The *insert* command will add addresses to a user's whitelist. The *remove* command will delete addresses from a user's whitelist. And the *display* command will send a message back to the user with a list of addresses in their whitelist.

   To enable whitelisting for end users, an email account must be created on the SPAM Gateway. Using James' Remote Manager, add user accounts with the *adduser <username> <password>* command. Set the user name equal to the userid of their email address, and the password can be anything as the account will only be used for whitelisting. The *whitelistmgr* processor is configured as follows:

```
<processor name="whitelistmgr">
    <!-- Process Whitelist -->
    <mailet match="RecipientIs=whitelist@spam.domain-name.com"
       class="WhiteListManager">
       <table>db://spamdb/whitelist</table>
       <whitelistManagerAddress>whitelist@spam.domain-name.com
                                                  </whitelistManagerAddress>
          <displayFlag>display</displayFlag>
```

```
            <insertFlag>insert</insertFlag>
            <removeFlag>remove</removeFlag>
    </mailet>

    <!-- Sent here by mistake -->
    <mailet match="All" class="ToProcessor">
       <processor>error</processor>
           <notice>White List Manager</notice>
    </mailet>
</processor>
```

4) The *spamcheck* processor analyzes messages for SPAM properties. Messages are checked against the recipients' whitelist before any SPAM Analysis is performed. Messages that pass the analysis are moved to the *transport* processor for remote delivery to the Internal Mail Server. Messages that fail the analysis are moved to the *FlaggedAsSpam* processor. The following analysis is performed to determine if a message is likely to be SPAM: 1) Bayesian Statistical Analysis 2) Reverse DNS lookup (SenderInFakeDomain) 3) Check message against Black Holes, Open Relays, and known SPAM servers 4) Check sender against Blacklist. Use the XML <!-- ... --> comment block to disable any unwanted analysis. As end users forward Non-SPAM and SPAM messages to the Bayesian Feeder, the active SPAM corpus will need to be updated and reloaded. To update the corpus, send a message to the manager@spam.*domain-name.com* address with the subject line: **rebuild spam corpus**. The *spamcheck* processor is configured as follows:

```
<processor name="spamcheck">
    <!-- Check the whitelist -->
    <mailet match="IsInWhiteList=db://spamdb/whitelist" class="ToProcessor">
       <processor> transport </processor>
    </mailet>
    <!-- Anti spam bayesian analysis -->
    <mailet match="All" class="JDBCBayesianAnalysis">
       <repositoryPath>db://spamdb</repositoryPath>
       <hamTable>bayesiananalysis_ham</hamTable>
       <spamTable>bayesiananalysis_spam</spamTable>
       <messageCountsTable>bayesiananalysis_messagecounts</messageCountsTable>
       <spamManagerAddress>manager@spam.domain-name.com</spamManagerAddress>
       <rebuildSubjectFlag>rebuild spam corpus</rebuildSubjectFlag>
       <headerName>X-MessageIsSpamProbability</headerName>
       <ignoreLocalSender>true</ignoreLocalSender>
    </mailet>
    <!-- Check Spam Header. In conjunction with the Bayesian -->
    <!-- Analysis the SPAM Header is checked. Message with a -->
    <!-- probability of 95% or greater are moved spam processor -->
    <mailet match="HeaderIsGreaterThan=X-MessageIsSpamProbability 0.95"
       class="ToProcessor">
       <processor> FlaggedAsSpam </processor>
       <notice>Spam not accepted</notice>
    </mailet>
    <!-- Reverse DNS Lookup. Checks that the email Sender is -->
    <!-- associated with a valid domain. -->
    <!-- Useful for detecting and eliminating spam. -->
    <mailet match="SenderInFakeDomain" class="ToProcessor">
```

```
      <processor> FlaggedAsSpam </processor>
   </mailet>
   <!-- Check for delivery from a known spam server -->
   <!-- This set of matchers/mailets redirect all emails from known -->
   <!-- black holes, open relays, and spam servers to the spam processor -->
   <mailet match="InSpammerBlacklist=blackholes.mail-abuse.org"
      class="ToProcessor">
      <processor> FlaggedAsSpam </processor>
      <notice>Rejected - see  http://www.mail-abuse.org/rbl/</notice>
   </mailet>
   <mailet match="InSpammerBlacklist=dialups.mail-abuse.org"
      class="ToProcessor">
      <processor> FlaggedAsSpam </processor>
      <notice>Dialup - see http://www.mail-abuse.org/dul/</notice>
   </mailet>
   <mailet match="InSpammerBlacklist=relays.mail-abuse.org"
      class="ToProcessor">
      <processor> FlaggedAsSpam </processor>
      <notice>Open spam relay -
              see http://www.mail-abuse.org/rss/</notice>
   </mailet>
   <!-- Sample matching to kill a message (Blacklist) -->
   <mailet match="RecipientIs=badboy@badhost badboy2@badhost"
      class="Null"/>

   <!-- Message passed SPAM Analysis - Deliver to Recipients -->
   <mailet match="All" class="ToProcessor">
      <processor> transport </processor>
   </mailet>
</processor>
```

5) The *FlaggedAsSpam* processor attaches the original message to a new message from the postmaster. The subject line is prepended with [SPAM] and the message is sent on the original recipients. End users are able to configure rules in their mail client to move or delete messages based on the modified subject line. As testing is done and the accuracy of the SPAM Analysis is determined, messages may be discarded at the SPAM Gateway to reduce the processing load and storage requirements of the Internal Mail Server. The *FlaggedAsSpam* processor is configured as follows:

The Redirect Matcher and Mailet pair configuration used by the *FlaggedAsSpam* processor was adapted from the "RE: subject prefix mailet or unaltered recipients in Redirect"[9] mailing list article.

```
<processor name="FlaggedAsSpam">
   <!-- Send Original Message to Recipients as an attachment.  -->
   <!-- Modify Subject to flag as SPAM -->
   <!-- Include message from postmaster -->
   <mailet match="All" class="Redirect">
      <static>true</static>
      <sender>postmaster</sender>
      <replyto>postmaster</replyto>
      <returnPath>null</returnPath>
      <inline>none</inline>
```

[9] Gianferrari Pini, Vincenzo. The Mail Archive. Web Site.

```
        <attachment>message</attachment>
        <prefix>[SPAM]</prefix>
        <passThrough>false</passThrough>
        <message>Based on our analysis,  the attached message has been
                 marked as SPAM.  From quick review of the subject
                 line one should discern whether it is a legitimate
                 message, and if true delete it without opening it.

                 Postmaster doimain-name.com</message>
    </mailet>
</processor>
```

6) The *transport* processor delivers mail to the Internal Mail Server. This processor is included as part of the default config.xml file. It has been modified to perform remote deliver only. Message relay is prevented by only processing messages for the selected domain: *domain-name.com*. The *transport* processor is configured as follows:

```
<processor name="transport">
<!-- Attempt remote delivery using the specified repository for -->
<!-- the spool, using delay time to retry delivery and the      -->
<!-- maximum number of retries -->
    <mailet match="All" class="RemoteDelivery">
       <outgoing> file://var/mail/outgoing/ </outgoing>

       <!-- Number of milliseconds between delivery attempts -->
       <delayTime> 21600000 </delayTime>

       <!-- Number of failed attempts before returning to the sender -->
       <maxRetries> 5 </maxRetries>

       <!-- The number of threads that should be trying to deliver -->
       <!-- outgoing messages -->
       <deliveryThreads> 1 </deliveryThreads>

       <!-- A single mail server to deliver all outgoing messages. -->
       <!-- This is useful if this server is a backup or failover  -->
       <!-- machine, or if you want all messages to be routed      -->
       <!-- through a particular mail server, regardless of the    -->
       <!-- email addresses specified in the message -->
       <!-- The gateway element specifies the gateway SMTP server  -->
       <!-- name. If your gateway mail server is listening on a    -->
       <!-- port other than 25, you can set James to connect to it -->
       <!-- on that port using the gatewayPort  element. -->
       <!--
       <gateway> otherserver.domain-name.com </gateway>
       <gatewayPort>25</gatewayPort>
       -->
    </mailet>
</processor>
```

7) The *error* processor handles a message when errors occur, or when a message is routed to it by another processor. The processor is configured to store the message using the *error* File Repository and notify the Postmaster. The *error* processor is configured as follows:

```
<!-- The error processor is required.  James may internally set   -->
```

```
   <!-- emails to the error state.  The error processor is generally -->
   <!-- invoked when there is an unexpected error either in the      -->
   <!-- mailet chain or internal to James. -->
   <processor name="error">
       <!-- Logs any messages to the repository specified -->
       <mailet match="All" class="ToRepository">
          <repositoryPath> file://var/mail/error/</repositoryPath>
          <passThrough> true </passThrough>
       </mailet>

       <!-- If you want to notify the postmaster that a message -->
       <!-- generated an error, uncomment this  -->
       <mailet match="All" class="NotifyPostmaster"/>
   </processor>
```

<dnsserver> XML Block

By default James will use the localhost as its DNS server. Configure James to use the Internal DNS Server, 192.168.1.100.

```
<dnsserver>
    <servers>
    <!-- Enter ip address of your DNS server, one IP address per  -->
    <!-- server element.  The default configuration assumes a DNS -->
    <!-- server on the localhost. -->
       <server>192.168.1.100</server>
    </servers>
    <authoritative>false</authoritative>
</dnsserver>
```

<remotemanager> XML Block

The Remote Manager service is used to manage user accounts. The Whitelist Manager requires that a user account is defined for each user. The Remote Manager is accessed via a telnet session to port 4555: *telnet 10.10.10.1 4555*. It possible to access the manager via a SSL connection. Uncomment the <useTLS> XML block to enable this functionality. Based on Figure 1, modify the default configuration to bind the manager to IP Address 10.10.10.1, to use the *domain-name.com* as its Hello Name, and configure the administrator account and password.

```
<remotemanager>
    <port>4555</port>
    <bind>10.10.10.1</bind>
    <!--  Uncomment this if you want to use TLS (SSL) on this port -->
    <!--
    <useTLS>true</useTLS>
    -->
    <handler>
    <!-- This is the name used by the server to identify itself in -->
    <!-- the RemoteManager protocol.  If autodetect is TRUE, the   -->
    <!-- server will discover its own host name and use that in    -->
    <!-- the protocol.  If autodetect is FALSE, James will use the -->
    <!-- specified value. -->
       <helloName autodetect="false">domain-name.com</helloName>
       <administrator_accounts>
          <!-- Change the default login/password. -->
          <account login="root" password="<new password>"/>
       </administrator_accounts>
```

```
        <connectiontimeout> 60000 </connectiontimeout>
    </handler>
</remotemanager>
```

<pop3server> XML Block

The pop3server service is not required by the gateway and will be disabled.

```
<pop3server enabled="false">
```

<smtpserver> XML Block

The SMTP Server supports SSL and SMTP Authentication. Based Figure 1, modify the default settings to bind the SMTP Server to IP Address 10.10.10.1 and set the Hello Name used to identify the server.

```
<smtpserver enabled="true">
    <!-- port 25 is the well-known/IANA registered port for SMTP -->
    <port>25</port>
    <bind>10.10.10.1</bind>
    <!-- Uncomment this if you want to use TLS (SSL) on this port -->
    <!--
    <useTLS>true</useTLS>
    -->
    <handler>
    <!-- This is the name used by the server to identify itself in   -->
    <!-- the SMTP protocol.  If autodetect is TRUE, the server will  -->
    <!-- discover its own host name and use that in the protocol. If -->
    <!-- autodetect is FALSE, James will use the specified value. -->
       <helloName autodetect="false">domain-name.com</helloName>
       <connectiontimeout>360000</connectiontimeout>
       <!-- Uncomment this if you want to require SMTP authentication. -->
       <!-- <authRequired>true</authRequired> -->
       <!-- Uncomment this if you want to verify sender addresses,    -->
       <!-- ensuring that the sender address matches the user who has -->
       <!-- authenticated. This prevents a user of your mail server   -->
       <!-- from acting as someone else -->
       <!-- <verifyIdentity>true</verifyIdentity> -->
       <!-- This sets the maximum allowed message size (in kilobytes) -->
       <!-- for this SMTP service. If unspecified, the value defaults -->
       <!-- to 0, which means no limit. -->
       <maxmessagesize>0</maxmessagesize>
    </handler>
</smtpserver>
```

<nntpserver> XML Block

The nntpserver service is not required by the gateway and will be disabled.

```
<nntpserver enabled="false">
```

<data-sources> XML Block

The Bayesian filters and Whitelist Manager require access to the MySQL database: Spam. Within the <data-sources> XML block, configure the following <data-source>.

```
<!-- SPAM Database-->
<data-source name="spamdb"
             class="org.apache.james.util.mordred.JdbcDataSource">
```

```
        <driver>org.gjt.mm.mysql.Driver</driver>
        <dburl>jdbc:mysql://10.10.10.1/Spam</dburl>
        <user>spamd</user>
        <password>password</password>
        <max>20</max>
    </data-source>
```

Gateway Configuration Considerations

If the gateway will process large messages, Java Memory Exceptions may occur. They are particularly noticeable in the JDBCBayesianAnalysis Mailet. The exception occurs when the message is copied from one Java class to another. To resolve this issue the Java Heap Size must be defined in the startup files.[10] On Windows 2000/XP modify the c:\java\james-2.1.3\conf\wrapper.conf file and make the following changes.

```
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=128

# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=256
```

On RedHat Linux, modify the -Xms128m, initial heap size, and -Xmx256m, maximum heap size, settings in the /usr/local/james/bin/go.sh startup script.

It is important to review the log files to look for errors, because as Graham bluntly puts it, "For most users, missing legitimate email is an order of magnitude worse than receiving spam."[3] The log files are located in the apps/james/logs directory of the James installation folder. To modify the amount of detail logged, update the log level settings in the apps/james/SAR-IN/environment.xml file.

"Apache James has a careful, security-oriented, full multi-threaded design, to allow performance, scalability and mission-critical use."[11] However, never assume anything is completely secure. Even Java has had recent security flaws exposed.[12] While these did not affect James directly, it does reinforce that good security and patch management practices must be in place to mitigate the risks. To improve James' security, the ability to run James as a non-root user on RedHat Linux and as an account other than System on Windows 2000/XP should be explored.

Conclusion

Spam will continue to be a problem in future. Recent lawsuits will deter some, but international borders make enforcement difficult. As more and more people connect to the Internet via high speed connections, securing those systems becomes a larger concern. Recent viruses have been used to take over home

[10] Johnson, Corey. The Mail Archive. Web Site.
[3] Graham, Paul. Web Site.
[11] Apache James. The Apache Software Foundation. Web Site.
[12] SunSolve. Sun Alert Notification. Web Site.

PCs and spew out SPAM from unwilling participants.[13] There is hope however. New ideas to improve Bayesian filtering techniques that account for low frequency tokens or tokens that haven't appeared before,[8] and Greylisting, the idea of sending a temporary smtp failure code to trick automated spamming software,[14] can reduce the volume of SPAM reaching inboxes.

The gateway described delivers an affective SPAM solution on a company wide basis. James' Mailet API and ability to manipulate mail processing in multiple ways make it the ideal platform for hosting a SPAM Gateway. Its use of open source software levels the playing field for small business, putting an affordable SPAM solution in reach.

[13] Reuters. Wired News. Web Site.
[8] Robinson, Gary. Linux Journal. Web Site.
[14] Harris, Evan. Web Site.

Appendix A
/usr/local/james/bin/go.sh James Start and Stop Script [5]

```
#!/bin/sh
#
# go.sh
# Shell script to start and stop James

# OS specific support.  $var _must_ be set to either true or false.
cygwin=false
case "`uname`" in
CYGWIN*) cygwin=true;;
esac

# Checking for JAVA_HOME is required on *nix
if [ "$JAVA_HOME" = "" ] ; then
  export JAVA_HOME=/usr/java
  echo "ERROR: JAVA_HOME not found in your environment."
  echo
  echo "Please, set the JAVA_HOME variable in your environment to match the"
  echo "location of the Java Virtual Machine you want to use."
  echo "Trying to use /usr/java as default"
  exit 1
fi
JAVACMD=$JAVA_HOME/bin/java

# resolve links - $0 may be a softlink
THIS_PROG="$0"

while [ -h "$THIS_PROG" ]; do
  ls=`ls -ld "$THIS_PROG"`
  link=`expr "$ls" : '.*-> \(.*\)$'`
  if expr "$link" : '.*/.*' > /dev/null; then
    THIS_PROG="$link"
  else
    THIS_PROG=`dirname "$THIS_PROG"`/"$link"
  fi
done

# Get standard environment variables
PRGDIR=`dirname "$THIS_PROG"`
PHOENIX_HOME=`cd "$PRGDIR/.." ; pwd`

unset THIS_PROG

# For Cygwin, ensure paths are in UNIX format before anything is touched
if $cygwin; then
  [ -n "$PHOENIX_HOME" ] && PHOENIX_HOME=`cygpath --unix "$PHOENIX_HOME"`
fi

if [ -z "$PHOENIX_TMPDIR" ] ; then
  # Define the java.io.tmpdir to use for Phoenix
  PHOENIX_TMPDIR="$PHOENIX_HOME"/temp
  mkdir -p "$PHOENIX_TMPDIR"
fi

# For Cygwin, switch paths to Windows format before running java
if $cygwin; then
  PHOENIX_HOME=`cygpath --path --windows "$PHOENIX_HOME"`
fi

# ----- Execute The Requested Command -----------------------------------------

echo "Using PHOENIX_HOME:   $PHOENIX_HOME"
```

[5] Out-of-the-Box, EJB Solutions. Web Site.

```
  echo "Using PHOENIX_TMPDIR: $PHOENIX_TMPDIR"
  echo "Using JAVA_HOME:      $JAVA_HOME"

  #
  # Command to override JVM ext dir
  #
  # This is needed as some JVM vendors
  # like placing jaxp/jaas/xml-parser jars in ext dir
  # thus breaking Phoenix
  #
  JVM_OPTS="-Djava.ext.dirs=$PHOENIX_HOME/lib"

  if [ "$PHOENIX_SECURE" != "false" ] ; then
    # Make phoenix run with security manager enabled
    JVM_OPTS="$JVM_OPTS -Djava.security.manager"
  fi

  if [ "$1" = "start" ] ; then
    shift
    $JAVACMD $JVM_OPTS \
      $JVM_OPTS \
      -Djava.security.policy=jar:file:$PHOENIX_HOME/bin/phoenix-loader.jar!/META-
INF/java.policy \
      $PHOENIX_JVM_OPTS \
      -Xms128m \
      -Xmx256m \
      -Dphoenix.home="$PHOENIX_HOME" \
      -Djava.io.tmpdir="$PHOENIX_TMPDIR" \
      -jar "$PHOENIX_HOME/bin/phoenix-loader.jar" $* > /dev/null 2>&1 &
    echo $! > /var/run/james.pid

  elif [ "$1" = "stop" ] ; then
    shift
    kill -15 `cat /var/run/james.pid`
    rm -rf /var/run/james.pid

  elif [ "$1" = "run" ] ; then
    shift
    $JAVACMD $JVM_OPTS \
      $JVM_OPTS \
      -Djava.security.policy=jar:file:$PHOENIX_HOME/bin/phoenix-loader.jar!/META-
INF/java.policy \
      $PHOENIX_JVM_OPTS \
      -Xms128m \
      -Xmx256m \
      -Dphoenix.home="$PHOENIX_HOME" \
      -Djava.io.tmpdir="$PHOENIX_TMPDIR" \
      -jar "$PHOENIX_HOME/bin/phoenix-loader.jar" $* "$@"

  else
    echo "Usage:"
    echo "james (start|run|stop)"
    echo " start - start james in the background"
    echo " run - start james in the foreground"
    echo " stop - stop james"
    exit 0
  fi
```

Appendix B
/etc/init.d/james Startup Script [5]

```
#!/bin/sh
#
# Startup script for James, the Jakarta Mail Server
#
# chkconfig: 2345 95 15
# description: James is a Mail Server
# processname: james
# pidfile: /var/run/james.pid
# config: /usr/local/james/apps/james/SAR-INF/config.xml
# logfiles: /usr/local/james/apps/james/logs
#
# version 1.0 -
#
# Source function library.
. /etc/rc.d/init.d/functions

#SET THE FOLLOWING LINE TO YOUR JAVA_HOME (This was done in /etc/profile)
#export JAVA_HOME=/usr/java/j2re1.4.2_03

#SET THE FOLLOWING LINE TO YOUR CORRECT JBOSS_HOME
export JAMES_HOME=/usr/local/james
export PATH=$PATH:$JAMES_HOME/bin:$JAVA_HOME/bin

#IF YOU NEED SPECIAL CLASSES IN YOUR CLASSPATH
#AT STARTUP, ADD THEM TO YOUR CLASSPATH HERE
#export CLASSPATH=

RETVAL=0

# See how we were called.
case "$1" in
  start)
      cd $JAMES_HOME/bin
      echo -n "Starting james daemon: "
      daemon $JAMES_HOME/bin/go.sh start
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && touch /var/lock/subsys/james
      ;;
  stop)
      echo -n "Stopping james daemon: "
      killproc james
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/james
      ;;
  restart)
      echo -n "Restarting james daemon: "
      $0 stop
      sleep 2
      $0 start
      ;;
  esac
```

[5] Out-of-the-Box, EJB Solutions. Web Site.

Appendix C
MySQL whitelist.sql Script

```
create table whitelist
(
      localUser varchar (80) not null,
      localHost varchar (80) null,
      remoteUser varchar (80) not null,
      remoteHost varchar (80) null
) ;

ALTER TABLE whitelist ADD INDEX ix_whitelist (remoteUser, remoteHost);
ALTER TABLE whitelist ADD INDEX ix_whitelist_1 (localHost);
ALTER TABLE whitelist ADD INDEX ix_whitelist_2 (remoteHost);
```

References

[1] Apache James. The Apache Software Foundation.
URL: http://james.apache.org

[2] Cert Coordination Center. CERT/CC Vulnerability Notes Database.
URL: http://www.kb.cert.org/vuls
Sendmail Vulnerability Search
URL: http://www.kb.cert.org/vuls/byid?searchview&query=Sendmail&searchorder=3
Microsoft Exchange Vulnerability Search
URL: http://www.kb.cert.org/vuls/byid?searchview&query=Microsoft%20Exchange&searchorder=3

[3] Graham, Paul. "A Plan for SPAM." August, 2002.
URL: http://www.paulgraham.com/spam.html

Graham, Paul. "Better Bayesian Filtering." January, 2003.
URL: http://www.paulgraham.com/better.html

[4] Duguay, Claude. "Working with James, Part 1: An introduction to Apache's James enterprise e-mail server." IBM developerWorks. June 10, 2003.
URL: http://www-106.ibm.com/developerworks/java/library/j-james1.html

[5] EJB Solutions. "Chapter 20. James Installation Procedures." Out-of-the-Box verison 1.0 Installation and Configuration. February 2003.
URL: http://www-106.ibm.com/developerworks/java/library/j-james2.html

[6] Duguay, Claude. "Working with James, Part 2: Build e-mail based applications with matchers and mailets." IBM developerWorks. June 10, 2003.
URL: http://www-106.ibm.com/developerworks/java/library/j-james2.html

[7] Gianferrari Pini, Vincenzo. "Matchers and Mailets for James." Praxis Calcolo S.p.A. April 29, 2003. URL: http://portale.praxis.it/pub/james/readme.htm

[8] Robinson, Gary. "A Statistical Approach to the Spam Problem." Linux Journal. March 1, 2003. URL: http://www.linuxjournal.com/print.php?sid=6467

[9] Gianferrari Pini, Vincenzo. "RE: subject prefix mailet or unaltered recipients in Redirect." The Mail Archive. June 23, 2003.
URL: http://www.mail-archive.com/james-dev@jakarta.apache.org/msg07648.html

[10] Johnson, Corey. "RE: Determine Message Size in Mailet." The Mail Archive. December 29, 2003.
URL:http://www.mail-archive.com/server-user%40james.apache.org/msg01845.html

[11] Apache James. "Design Objectives." The Apache Software Foundation.
URL: http://james.apache.org/design_objectives.html

[12] SunSolve. "A Vulnerability in JRE May Allow an Untrusted Applet to Escalate Privileges." Sun Alert Notification. December 9, 2003.

URL: http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert%2F57221

[13] Reuters. "Spammers Tap Unwitting Users' PCs." Wired News. December 3, 2003. URL: http://www.wired.com/news/technology/0,1282,61457,00.html

[14] Harris, Evan. "The Next Step in the SPAM Control War: Greylisting." August 21, 2003. URL: http://projects.puremagic.com/greylisting/

Additional Resources:
108th Congress, United States of America. "CAN-SPAM Act of 2003". January 1, 2004. URL: http://www.spamlaws.com/federal/108s877enrolled.pdf

Computer Associates. "Win32.Sobig.F" Virus Information Center. August 19, 2003. URL: http://www3.ca.com/virusinfo/virus.aspx?ID=36376

Computer Associates. "Win32.Mimail.P" Virus Information Center. January 7, 2004. URL: http://www3.ca.com/virusinfo/virus.aspx?ID=37946

Delio, Michelle. "Random Acts of Spamness." Wired News. January 13, 2004. URL: http://www.wired.com/news/infostructure/0,1377,61886,00.html

Glasner, Joanna. "Open Up a Can of SPAM." Wired News. January 16, 2004. URL: http://www.wired.com/news/politics/0,1283,61928,00.html

MySQL AB. "2.1.4 Verifying Package Integrity Using MD5 Checksums or GnuPG." MySQL Manual. URL: http://www.mysql.com/doc/en/Verifying_Package_Integrity.html

Reuters. "Lawsuits Target Alleged Spammers." Wired News. December 18, 2003. URL: http://www.wired.com/news/business/0,1367,61660,00.html

Security Focus. "Sun Java Virtual Machine Slash Path Security Model Circumvention Vulnerability." October 22, 2003.
URL: http://www.securityfocus.com/bid/8879
Security Focus, Vulnerabilities Archive
URL: http://www.securityfocus.com/bid/vendor/

The Sans Institute. "(1) MODERATE: Sun Java Virtual Machine Security Bypass." SANS Critical Vulnerability Analysis. October 29, 2003 Vol. 2. No. 42.
URL: http://www.sans.org/newsletters/cva/vol2_42.php

The Last Stage of Delirium Research Group, "Java and Java Virtual Machine Vulnerabilities and their Exploitation Techniques."
URL: http://lsd-pl.net/java_security.html

Ulbrich, Chris. "Can SPAM? Or New Can of Worms?" Wired News. December 22, 2003. URL: http://www.wired.com/news/politics/0,1283,61679,00.html