MTA based Virus Scanning with Sun One Messaging and Sophos

David A. Evans
GSEC Practical Assignment
Version 1.4 Option B

Abstract

This case study explains how Sun's iPlanet Messaging Server 5.2 (iMS 5.2) and Sophos Anti-Virus Software were configured together in order to stop E-mail viruses from entering a company network. This was achieved by creating four Internet facing MTA gateways that handle all incoming and outgoing mail for the company.

This paper begins with an overview of the messaging environment before the MTA was put in place. Next details on the MTA placement and design are covered. Following this instruction on how to configure iPlanet Messaging and Sophos together are given. Finally I conclude with details on a custom reporting system I developed which provides graphical statistical charts in daily, weekly and monthly views.

Scenario Overview

Here I attempt to paint a picture of the messaging environment before the implementation of the virus scanning solution detailed in this paper. I also explain where it fits into a larger project that had the objective to strengthen messaging security for the entire company.

The global company houses over 300 internal mail servers that serve approximately 60,000 users worldwide. Currently anti-virus (AV) software is deployed at the desktop level and on a few select internal mail servers. The company Internet firewalls allow unmonitored SMTP traffic (port 25) inbound to all 300 internal mail servers and there are no restrictions on outbound SMTP traffic. The company's security team is faced with the daunting task of assuring all 300 internal mail servers, which run a variety of mail server software, on various operating systems are up to date with the latest patches and are configured securely. When the latest fast spreading virus or worm hits the Internet the frantic task of pushing out the latest virus definition files to all 60,000 users begins.

The company security policy states all mail servers should:

- Not accept message attachments that have certain file extensions (such as .exe .vbs etc…)
- Scan E-mail for viruses
- Not relay messages for IP addresses that are not listed as trusted on the company's security website

It is unknown to what degree or how many of the servers are in accordance with this policy.

<u>Review of the before snapshot</u>

The goal of the new message architecture design was to address several security issues outlined above. The external exposure of over three hundred mail servers is a risk. Especially in an environment were there is constant personal change and the servers are geographically distributed all over the world. By making the server not directly accessible from the Internet we are still exposed to the very real internal threat but we reduce our risk from the external threats. In addition it provides four points where all external message traffic flows this gives much greater control to enforce messaging policy.

<u>Product Selection</u>

Several different products were evaluated for the purpose of this project but eventually iPlanet Messaging was selected for its interoperability with LDAP and based on in house experience with the product. Future phases of this project will include configuring the iPlanet MTA servers to operate with the corporate LDAP directory. As the MTA servers receive an incoming message they will do an LDAP query against the corporate directory, which will validate that there is an actual account before accepting the message into the Company network.

When selecting the Anti-virus product it was important to choose a different company other than Mac Fee, which is the software deployed at the desktop level for virus protection. The reasoning behind this was to have two separate companies that provide virus definition files used to detect message viruses and protect our users. This gives us two separate sources to which we rely a pond for virus protection. This follows the defense in depth strategy, which is a core concept in many areas of information security.

MTA Placement and Design

To give explanation where the iPlanet/Sophos anti-virus design was used I begin with the high-level overall architectural design of the project. Four MTAs were built and placed in different geographical regions of the world. Having more than one MTA gateway not only serves better to handle the current and future demands of the companies mail traffic but also provides a level of fault tolerance. For example if one MTA is made inaccessible by perhaps a type of denial of service attack, RFC 2821 states:

```
To provide reliable mail transmission, the SMTP client MUST be able to
try (and retry) each of the relevant addresses in this list in order,
until a delivery attempt succeeds.  However, there MAY also be a
configurable limit on the number of alternate addresses that can be
tried.  In any case, the SMTP client SHOULD try at least two addresses.
```

[1] RFC 2921, p. 60

So if one MTA is not reachable all RFC compliant mail servers should attempt another MX record listed before marking the message undeliverable. In order for this to work all DNS records were changed for all internal domains to list the four MTA servers as domain handling MX records. If the sending server fails to connect to an MTA silenced by the denial of service attack or network outage it will simply attempt another MTA servers. Here is an example of the MX records listed for a sample company domain.

subdomain.company.com	mail is handled (pri=100) by mta01.company.com
subdomain.company.com	mail is handled (pri=100) by mta02.company.com
subdomain.company.com	mail is handled (pri=100) by mta03.company.com
subdomain.company.com	mail is handled (pri=100) by mta04company.com

Figure 1 Example of the MX records for subdomain.company.com after the MTA implementation

Notice the internal mail server that hosts the domain subdomain.company.com is no longer visible from the outside world. All mail will now be routed to one of the four MTA servers and static routes defined on the MTA servers will direct mail to the correct internal mail server. This design allows us to change the firewall rules from allowing port 25 traffic inbound from 300 servers to four and completely close port 25 access outbound with the exception of the four MTA servers. By not allowing anyone external to the company to connect to port 25 across the internal mail severs we greatly reduce the risk of becoming a victim of the latest sendmail or exchange buffer overflow exploit by reducing our exposure to external threats. Before we can limit SMTP traffic outbound to the four MTA servers all internal mail servers must make configuration changes to list the nearest MTA server as a smart host.

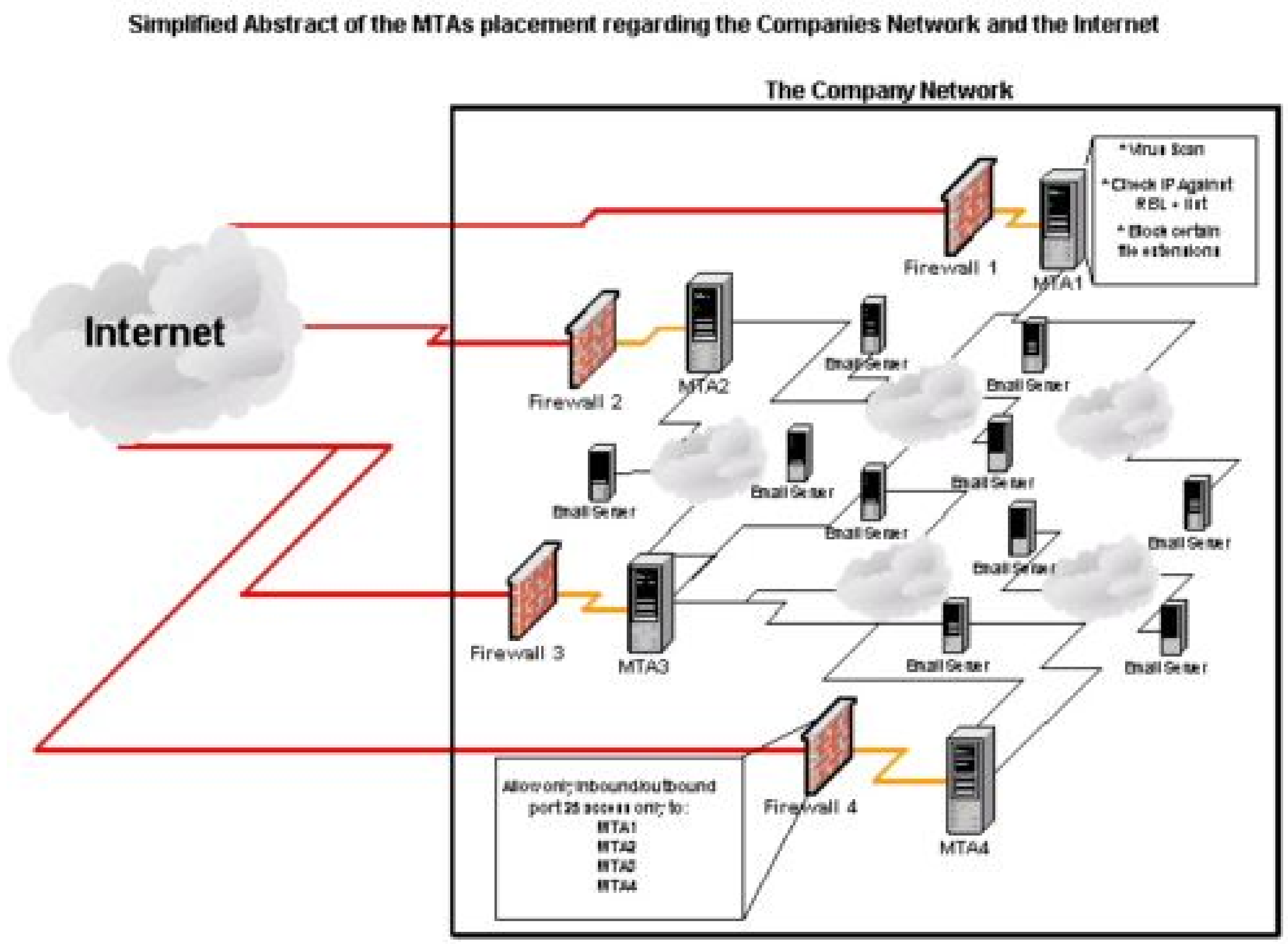

Simplified Abstract of the MTAs placement regarding the Companies Network and the Internet
The Company Network
Internet
*Virus Scan
*Check IP Against RBL + list
*Block certain file extensions
Firewall 1
MTA1
Firewall 2
MTA2
Firewall 3
MTA3
Firewall 4
MTA4
Email Server
Allow only inbound/outbound port 25 access only to:
MTA1
MTA2
MTA3
MTA4

Figure 2 Image showing the placement of the MTA in relation to the company firewalls and the Internet

At this point I would like to bring up the fact that a better design perhaps would be to place the MTA servers off an Internet border firewall interface in a DMZ. With the restrictive firewall rules this would limit the damage that could be done if someone gained unauthorized access to one of the MTA servers. This was not done during this project for internal reasons.

Details of building the Virus-Scanning, Spam-blocking MTA

Each of the MTAs were built on Sun Fire V880s running Solaris 8. After the disk raids were configured the OS hardened was harden. The servers were hardened using a custom company YASSP script; more information on YASSP can be found here http://www.yassp.org/. Next iPlanet Messaging server was installed. The installation is fairly straight forward, the installation guide can be found here:

http://docs.sun.com/source/816-6014-10/index.html

After installing the Messaging software Sophos Anti Virus was installed. Installation instructions for Sophos a can be found here:

http://www.sophos.com/sophos/docs/eng/instguid/unix_ien.pdf

NOTE: During the installation process it is important to check all file and directory permissions and to assure they are owned by the username the mail sever runs as.

Once the Sophos software has been installed the next step is to configure iPlanet to use Sophos to scan its message traffic. Before we do this we must introduce the concept of iPlanet messaging channels. It is important to understand the concept of channels and how the message server uses them. For a good explanation of the term I refer to the iMS 5.2 Administration guide.

> The channel is the fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. The actual hardware connection or software transport or both may vary widely from one channel to the next.
>
> Channels perform the following functions:
>
> - Transmit messages to remote systems, deleting them from their queue after they are sent.
> - Accept messages from remote systems, placing them in the appropriate channel queues.
> - Deliver messages to the local message store.
> - Deliver messages to programs for special processing.
>
> Messages are enqueued by channels on the way into the MTA and dequeued on the way out. Typically, a message enters via one channel and leaves by another. A channel might dequeue a message, process the message, or enqueue the message to another MTA channel

[1] iPlanet Administration Guide, p. 101.
http://docs.sun.com/source/816-6009-10/mtacncpt.htm#22760

There are two default channels that accept all mail received by the MTA and they are:

- tcp_local
- tcp_intranet

All messages that are defined as internal to the MTA (as specified in the mappings file) will first be queued to the tcp_intranet channel and all others --considered external-- will be queued to the tcp_intranet channel. IPlanet determines whether a message is arriving from internal or external by definitions defined in the mappings file. The default location for the mappings file is:

```
server_root/msg-instance/imta/config/mappings
```

In the mappings file you can specify which IP address that the server will relay mail for. This IP range is also used to determine what the server considers internal messages. Below is the section of the mappings file that defines this:

```
INTERNAL_IP

 $(192.168.228.0/24)  $Y
 $(192.168.229.0/24)  $Y
 $(192.168.230.0/24)  $Y
 $(192.168.231.0/24)  $Y
 $(192.168.115.0/24)  $Y
 $(192.168.114.0/24)  $Y
 $(192.168.116.0/24)  $Y
 $(192.168.117.0/24)  $Y
 $(192.168.118.0/24)  $Y
 $(192.168.119.0/24)  $Y
  127.0.0.1  $Y
  *  $N
```

A common security mistake made by mail administrators is to configure the server to relay for anyone, hence making the server an open-relay. Spammers commonly exploit this configuration by sending large amounts of unsolicited messages through severs with this configuration in attempt to avoid the repercussions of sending spam. Not only will this activity drain your servers resources but it also may cause your server to end up on a spam prevention black list such as the MAPS RSS list (http://work-rss.mail-abuse.org/rss/). To ensure your server is not configured as an open relay include the following after defining which IP address you will relay for.

```
  *  $N
```

This says if the IP address did not match any of the above cases do not relay messages for it.

This will prevent messages arriving from the Internet from being relayed back out to the Internet. In PMDF terms this prevents messages from taking the channel path tcp_local -> tcp_local. This does not necessarily protect you from spammers relaying messages through your MTA. The default iPlanet configuration allows messages through in the following format somebody@xyz.com@subdomain.company.com. With some mail servers you may find that the server will accept this message and strip the

@subdomain.company.com from the message then send the message to somebody@xyz.com. If you have an internal mail server that exhibits this behavior then there will exist a way to relay messages through your companies mail system. Below is a graph showing such a possible scenario.

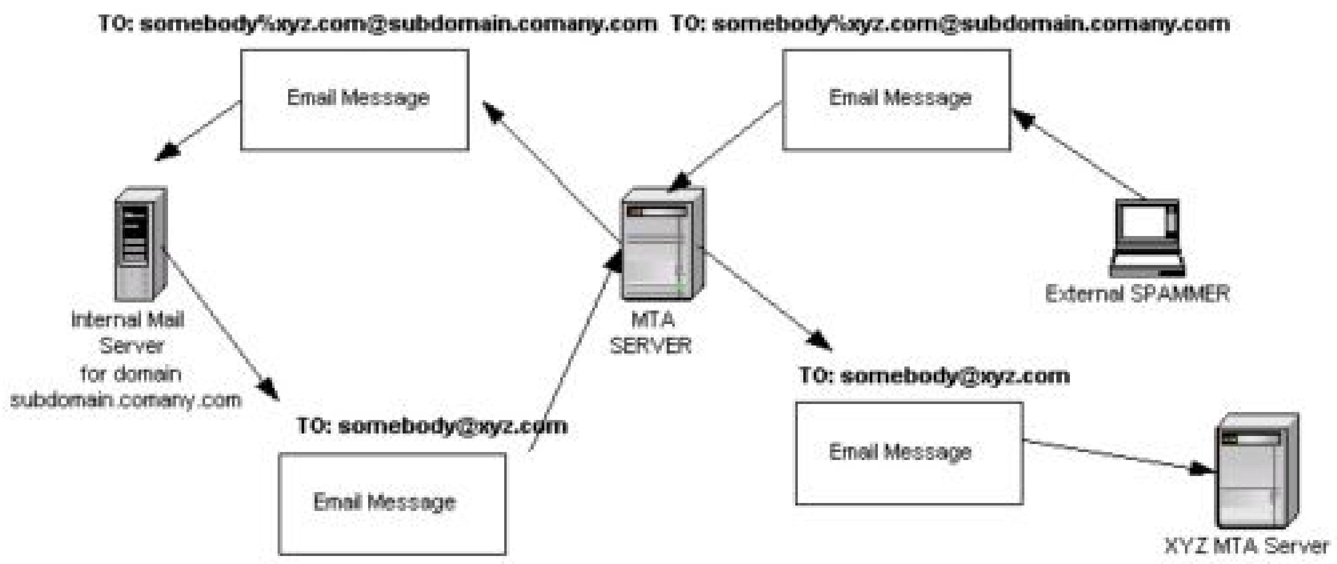

TO: somebody%xyz.com@subdomain.comany.com
TO: somebody%xyz.com@subdomain.comany.com
Email Message
Email Message
Internal Mail Server for domain subdomain.comany.com
MTA SERVER
External SPAMMER
TO: somebody@xyz.com
TO: somebody@xyz.com
Email Message
Email Message
XYZ MTA Server

Figure 3: Example showing the relaying of mail through the MTA gateway

This was discovered during the implementation of the MTA servers and was actually exploited by an external source. To prevent this a few rules need to be added to the iPlanet mappings file. These go in the section below the keyword: ORIG_SEND_ACCESS

```
tcp_local|*|*|*$%*.company.com@*     tcp_local|$0|$1|$2@$4$R
tcp_local|*|*|*$%comany.com@*        tcp_local|$0|$1|$2@$3$R
tcp_local|*|*|*$%*.comany.com$%*@*   tcp_local|$0|$1|$2@$5$R
tcp_local|*|*|*$%comany.com$%*@*     tcp_local|$0|$1|$2@$4$R
tcp_local|*|*|*$%*.*@*               $NRelaying$ not$ permitted
tcp_local|*|*|*.*!*@*                $NRelaying$ not$ permitted
tcp_local|*|*|"*@*"@comany.com       $NRelaying$ not$ permitted
tcp_local|*|*|"*@*"@*.comany.com     $NRelaying$ not$ permitted
tcp_local|*|*|@*:"*@*"@comany.com    $NRelaying$ not$ permitted
tcp_local|*|*|@*:"*@*"@*.comany.com  $NRelaying$ not$ permitted
tcp_local|*|*|@*:*@*.comany.com      $Y
tcp_local|*|*|@*:*@comany.com        $Y
tcp_local|*|*|@*:*@*                 $NRelaying$ not$ permitted
```

Reference: http://www.innosoft.com/app-notes/relay.html

Notice in the rule set above we are blocking messages formatted with the % @ email address format as well as others.

Virus Scanning

With iMS 5.2 the virus scanning is invoked from the conversion channel. Here are two sentences taken from the iPlanet Administration 5.2 guide that give some insight to what the conversion channel is and does.

"The conversion channel allows you to perform arbitrary body part-by-body part processing on specified messages flowing through the MTA."

"This is a default channel that is created in the MTA configuration file and can be used as is with no modification."
[1] iPlanet Administration Guide, p. 278.
http://docs.sun.com/source/816-6009-10/channel2.htm#42159

Below is a block diagram that shows the path an external message takes as it passes through the MTA.

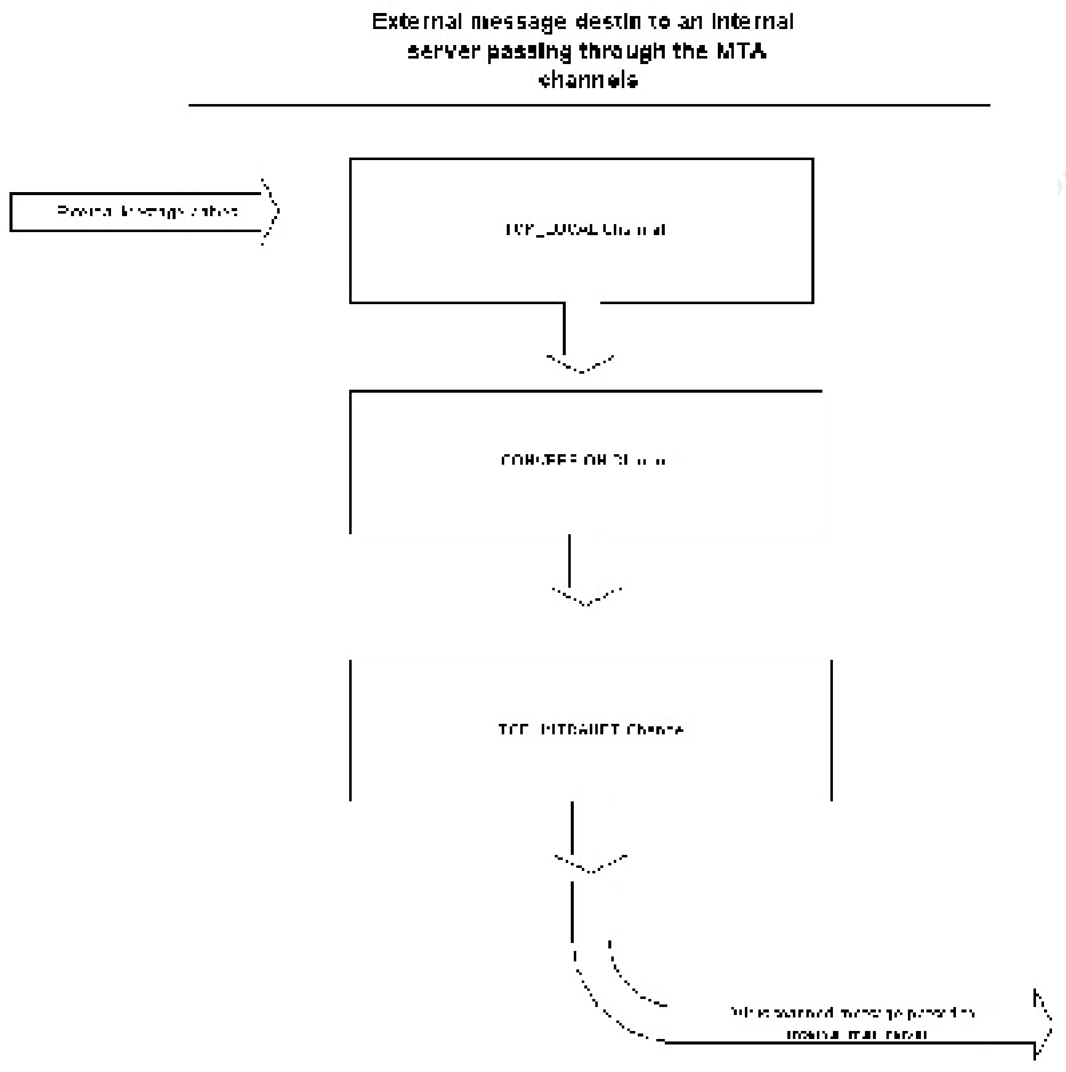

External message destin to an internal server passing through the MTA channels
TCP_LOCAL Channel
CONVERSION Channel
TCP_INTRANET Channel

Figure 4: Image showing an example path an external message would take as it passes through the MTAs channels

We can specify which attachments are to be scanned with another configuration file, which is simply called *conversions.* The default location of this file is *server_root*/msg-*instance*/imta/conversions. In this file we specify the criteria for the types of message attachments we want to scan. Here is a simple example of the entry you would include if you wanted to scan all message attachments of MIME type application.

```
!
! Message parts which have a MIME-type  of "APPLICATION/octet-stream",
! should be processed by the "multiscan.sh" command.
!
in-channel=*; in-type=application; in-subtype=*;
  parameter-symbol-0=NAME; parameter-copy-0=*;
  dparameter-symbol-0=FILENAME; dparameter-copy-0=*;
  message-header-file=2; original-header-file=1;
  override-header-file=1; override-option-file=1;
  command="/server_root/msg-instance/imta/bin/multiscan.sh"
```

There are two things to mention here. One is comment lines are specified by beginning with the "!" character. Second is the last line.

```
command="/server_root/msg-instance/imta/bin/multiscan.sh"
```

With this line we are stating all messages that match the criteria specified by the first line should be processed by the script multiscan.sh. This leads us to the next section with explains what this script is and how it is used to marry iPlanet and Sophos.

Multiscan.sh

The multiscan.sh is a shell script that gives instructions on what tests are to be preformed against the selected message attachment. The multiscan.sh script contains several function calls; the actual functions are defined in the script ScriptFunctions. Anthony Waldron wrote these scripts while he worked at Innosoft International. Innosoft was acquired by Sun but before the acquisition Innosoft sold their PDMF product to Process software, which is where you can find the two scripts.

http://www.process.com/techsupport/pmdf/conversion/library/multiscan.sh

http://www.process.com/techsupport/pmdf/conversion/library/scriptfunctions

To configure the script to run the Sophos sweep command against the selected message attachments simply uncomment the line that contains sophos_sweep. The virus scan directory (VSCANDIR) should be set to a directory in which you want to keep the virus scan reports. Each time a message is found to be virus infected Sophos outputs a report. Shown below is the section of the multiscan.sh script where you define the shell variable VSCANDIR.

```
# Prep for a sophos sweep.

VSCANDIR="/pmdf/log/VIRUS"
RECORD_FILE="${VSCANDIR}/VirusScan-${REFERENCE_ID}"
DIAGNOSTIC_MSG="Possible Virus Detected"
ERROR_POINTER_URL="http://www.duhmayne.eju/email/viruses.html"

 #===>>>># sophos_sweep
#^^^^^^^^^ Uncomment if you wish to launch sophos_sweep
```

After every sub routine call in multiscan.sh there is an IF statement that tests to see if the variable RED_FLAG was set in the function call. This variable is set to 1 if the sweep command finds the attachment positive for a virus infection. After the function sophos_sweep has been called the IF statement checks to see if the message contained a virus. If it is true a number of function calls may be called. The default provides functions perform the following:

- Create a virus report of the infected message and place this in the directory defined by the variable VSCANDIR
- Bounce the entire message
- Replace the infected attachment with a substitute note
- Mark the message as held
- Silently delete the infected attachment

Here is an example where we have configured the script to create a virus report and then substitute the virus-infected attachment with a subsitue.txt file. The text message included in the substitute.txt message is defined in the scriptfunctions file.

```
if [ $RED_FLAG -eq 1 ]
then
    write_record       # Log the MIME information of this message part
    substitute_part    # Replace affected part with a substitute note.
    #force_delete      # Delete the current message part.
    #force_bounce      # Bounce the entire message.
    #force_hold        # Mark the message as .HELD
    exit
fi
```

Although these are the only functions offered by default it is fairly easy to customize the script to include your own subroutine that performs an action on a message that is not included here.

If you have installed sweep in the default location of /usr/local/ and the ide files in the directory /usr/local/ide/ then no modifications are needed to scriptfunctions. Else modify the lines LD_LIBRARY_PATH and SAV_IDE to match your installation.

```
sophos_sweep () {
      LD_LIBRARY_PATH=/usr/local/sweep/lib
      SAV_IDE=/usr/local/sweep/ide
      export LD_LIBRARY_PATH SAV_IDE
      CONVERTER_COMMAND="/usr/local/sweep/bin/sweep -sc -ss"
      ${CONVERTER_COMMAND} $INPUT_FILE >${RECORD_FILE}.scan 2>&1
      RETURN_STATUS=$?
      if [ $RETURN_STATUS -eq 3 ]
      then
         RED_FLAG=1
      else
         rm -f ${RECORD_FILE}.scan
      fi
}
```

In the multiscan.sh script you can also configure it to perform other tests against the message. Here we could have another virus scanner product also scan the message. Arguments for this were made in a paper posted in the Sans reading room titled "One Virus Engine Is Not Enough: The Case for Maximizing Network Protection with Multiple Anti-virus Scanners".

A lot of companies security policies today state that their servers should reject messages that contain attachments ending in certain extensions such as .vbs or .exe. This is often in attempt to protect their users from malware. To do this with multiscan locate the sub routine call suffix_scan in the multiscan.sh script and configure it as the example below.

```
# Prep for a suffix scan

RECORD_DIR="${SERVER_ROOT}/log/scan"
RECORD_FILE="${RECORD_DIR}/MyScan-${REFERENCE_ID}"
DIAGNOSTIC_MSG="A potentially executable attachment was found in this
email."
ERROR_POINTER_URL="http://security.com/

# if you wish to launch this type of scan
   suffix_scan .exe .src .vbs

# See if the FLAG was changed after the converter was 'launch'ed

if [ $RED_FLAG -eq 1 ]
then
    write_record       # Log the MIME information of this message part
#    substitute_part   # Replace affected part with a substitute note.
#    force_delete      # Delete the current message part.
#    send_email
     force_bounce      # Bounce the entire message
#    force_hold        # Mark the message as .HELD
    exit
fi
```

In the above example we cause the MTA to bounce all message back to the recipient if they contain an attachment that contains the extension .exe, .src or .vbs.

We have now gone over how iPlanet/Sophos were configured together to scan viruses and reject messages based on extensions. After the server has been built and Sophos configured and put in production we may find all of our work in vain if the Antivirus definition files are not regularly updated. In the next section we go over how to configure the server to automatically update these virus definitions.

Procedures for automating the updating of Sophos software

Sophos anti-virus consists of a command line utility called *sweep*. As with most virus scanning software there are two main parts to sweep, the virus-scanning engine and the virus definition files. As it is crucial to stay up to date with the latest version of each I will explain one solution to automate this process.

Sophos does two things that make this possible. One they make their newly published versions of the virus engine and definition files available from their web site. Second they provide a subscription-based mailing list, which sends an email indicating when the new versions of their software are available. A few software packages are required to be installed on the system for this to work.

The GNU Wget package, which can be found here:
http://wget.sunsite.dk/

and the GNU grep which can be found here:
http://www.gnu.org/software/grep/grep.html

With the availability of wget and grep we are able to create shell scripts that will download the latest software updates from the Sophos web site and proceed with the installation when invoked. Included below is the source to a script that will update the virus engine when called. Both of the scripts below were originally obtained from a Sophos Sales engineer some small modifications were made. Sophos redesigned their web site and the location of where their software updates were kept so the scripts were adjusted.

```
#!/bin/bash

file2get="solaris.sparc.tar.Z"
#file2get="linux.intel.libc6.tar.Z"
#uzipdfile="linux.intel.libc6.tar"
uzipdfile="solaris.sparc.tar"
```

```
sav_install_dir="/usr/local/sweep/sav"
ide_install_dir="/usr/local/sweep/ide"
sav_directory="sav-install"
PATH=/usr/local/bin:/bin:/sbin:/usr/local/sweep; export PATH
LD_LIBRARY_PATH=/usr/local/sweep/lib; export LD_LIBRARY_PATH
tmpdir="/tmp/sav"
WGET='/usr/local/wget/bin/wget -C off'
if [ ! -d $tmpdir ]; then
   echo "No temp directory"
   echo "Creating temp dir"
   mkdir $tmpdir
   echo "Created temp dir"
fi
user="Username given by Sophos"
passwd="Password given by Sophos"

echo "Checking installed version of sweep..."
get_cur_ver=`sweep -v | sed -e '/[Pp]roduct [Vv]ersion/!d'`
cur_ver=`echo $get_cur_ver | sed -e 's/[^0-9]//g'`
echo "Current version of SAV running is $cur_ver"

cd $tmpdir

$WGET --http-user=$user --http-passwd=$passwd
http://downloads.sophos.com/sophos/products/full/$file2get
uncompress $file2get
tar -xvf $uzipdfile

if [ -d $tmpdir/$sav_directory ]; then
   web_ver=`ls $tmpdir/$sav_directory/*dat | sed -e 's/[^0-9]//g'`
   echo $web_ver
else
   echo "Could not download/untar the current web version SAV"
fi

if [ "$web_ver" -gt "$cur_ver" ]; then
   echo "SAV running on your system is outdated"
    mv $ide_install_dir/*.ide $ide_install_dir/archive/
      sh $tmpdir/$sav_directory/install.sh -v -ni -d /usr/local/sweep
      sleep 5
      ksh /usr/local/sweep/bin/getide
else
   echo "The current version of SAV installed on your system is up to
date"
fi
cd /tmp
if [ -d $tmpdir ]; then
   rm -rf $tmpdir
fi
```

Notice if the script finds that it has downloaded a newer virus engine than the one installed it performs the installation and then calls a script called getide. This script is defined below and when called downloads and installs the latest IDE files released on the Sophos web site. Below is the source to this script.

```
#!/bin/ksh
#
# Sample shell script to download and update IDE files
# Relies on environment variable SAV_IDE being set
# This should usually point to /usr/sav or /usr/local/sav
# depending on how you have installed sav on your system
#

SAV_IDE=/usr/local/sweep/ide

# Put your username and password for the Sophos site here

sav_user="Username given by Sophos"
sav_pw="Password given by Sophos"

# Change the following line to point to local copy of wget, or change
all
# three if you want to use something other that wget that can get a
file
# from a web site and dump it to a file
WGET='/usr/local/wget/bin/wget -C off'

# Change this line to show where SAV is installed

#sav_install_root=/usr/local
sav_install_root=/usr/local/sweep

# This script doesn't know where sweep is
SWEEP="$sav_install_root/bin/sweep"
LD_LIBRARY_PATH="$sav_install_root/lib";export LD_LIBRARY_PATH


tmpdir='/tmp/ides'
gotfile="$tmpdir/ides.zip"
#geturl='http://www.sophos.com.au/downloads/ide'
geturl='http://www.sophos.com/downloads/ide'

if [ $SAV_IDE ] ; then
    idedir=$SAV_IDE
else
    echo "SAV_IDE not defined - don't know where to put ides"
    exit
fi

echo $idedir
if [ ! -d "$idedir" ] ; then
    echo "Target IDE dir $idedir doesn't exist - creating it"
    mkdir $idedir
fi


if [ -e "$tmpdir" ] ; then
   rm -Rf "$tmpdir"
fi

mkdir "$tmpdir"
```

```
echo "sweep is `which $SWEEP`"

if [ "$1" ] ; then
   filename="$geturl/$1_ides.zip"
else
   filename="$geturl/`$SWEEP -v | \
     sed -e '/[Pp]roduct [Vv]ersion/!d' -e 's/[^0-9]//g'`_ides.zip"
fi

echo "Getting $filename into $gotfile"
$WGET -O "$gotfile" "$filename"

if [ -f "$gotfile" ] ; then
    echo Extracting IDEs
    cd "$tmpdir"
    unzip "$gotfile"
    if [ $? -eq 0 ] ; then
        echo Extraction successful
        echo $idedir
        echo `ls $idedir/*.ide`
        echo These files will be deleted
        rm -f $idedir/*.ide
        mv *.ide $idedir/
    else
        echo File not really a zip
#        cat < "$gotfile"
    fi
else
    echo No file downloaded
fi

cd
rm -Rf "$tmpdir"
```

Note that when updating Sophos identity files if new ones are available we must remove all of the old IDE files and install a complete new set and not just add the newly created IDE files that appear on the web site. This is important, as often the old IDE files have been updated.

Now that we have a working script that will pull down the latest Sophos Anit-Virus software and proceed with the installation of this software when run, we need to customize a solution that will execute this script when ever an update or new virus identity files are released.

One way of keeping up to date is to simply create a cron job that runs the script every four hours. This gives us a four-hour window where the MTA may not be up to date with the latest virus identity files. This may or may not be an acceptable risk for your organization.

In a discussion with another iPlanet administrator I learned of another solution, which reduces this risk by updating to the latest IDE files as soon as Sophos announce their availability by email. This is where the Sophos product mailing list comes into play. Here are the steps to make it work

- Subscribe to Sophos mailing list, ie. subscribed-user@pseudo.domain.

- In your dns server: setup an MX record for pseudo.domain.com to point to your mail server

- In imta.cnf make the following definition:
  ```
  pseudo.domain $U%pseudo.domain@pipe-daemon
  ```

- In pipe_option:
  ```
  subscribed-user@pseudo.domain=/your/script/here/xxx %s
  ```

You can read more about the pipe channel in the iPlanet Administration guide.

Reporting

Overview

The purpose of the MTA virus reporting system is to provide a web accessible report on the current MTA operations. The report presents a series of graphs and tables based on the virus detect activities of the MTA servers. All of the source code used to create the graphs is included.

These graphs are recreated on a daily bases allowing management to view recent virus trends in the company mail. In order to create these graphs software package called Gdchart 0.94b was used. This package provides a rich C library that allows us to create graphs based on MTA virus detect data. The output of these graphs is in the standard GIF format which allowing anyone with a web browser to view them.

Web page for gdchart : http://www.fred.net/brv/chart/
Download area for the software: http://www.fred.net/brv/chart/gdchart0.94b.tar.gz

Data Collection

In order to get a one-line record of each virus infected message detected by the MTA the logger command is used in the ScriptFunctions.sh script mention previously. This script contains the functions that are called by the multiscan.sh script. We need to create a particular function that is only called when a virus infected messages is found, this is where will place our one-liner logger command. The multiscan.sh script contains an IF statement just below the function call to run the Sophos sweep command against the respective attachment. If this test results in a positive virus detect the sub routines in the IF statement are performed. In order to have a unique subroutine that is only used with the virus scan case and not say for other possible test such as the checking of the file extension we create a new subroutine.

First make a copy of the subroutine code for write_record within the ScriptFunctions.sh script and give it a unique name, such as write_record_vir_detect. Then add the custom logger command, which will push a log entry to syslog whenever a virus is detected.

Logger command:

```
/usr/bin/logger -p mail.alert -t iMS $VIRUS_NAME virus found and dealt
```

The $VIRUS_NAME variable contains the virus name that was detected in the attachment. You can modify your syslog.conf file to direct these logs into a separate file if you wish or just grep them out from the system syslog.

The log entries will appear as follows in the log file:

```
Sep  8 16:45:44 mta.company.com iMS: [ID 702911 mail.alert] W32/Klez-H virus
found and dealt
```

Now we have a date, time and virus name for the logged virus detect. A simple way to get a tally of the number of virus detects for each day is do use the following combination of commands.

```
#> grep “virus found” /var/adm/syslog | wc -l
```

Or if you have configured these message to go to a separate file such as /var/adm/virus_detects you can use:

```
#> wc –l /var/adm/virus_detects
```

This assumes you have configured your syslog.conf to direct these messages to the file /var/adm/virus_detects and this log is rolled over each day. The below script takes this number and writes it along with the date to a log file. This log file is written with the most recent date listed at the top.

```
#!/usr/bin/perl –w
# Written by David Evans

$date=` date +%Y%m%d`;
$output_file=”/usr/local/data_files/vir_data.dat”;
$mta_dir=”/var/adm/”;
$virfile=”virus_detects”;

$VIR_HITS = `wc -l $mta_dir$virfile`;
#  We want to separate and grab only the number
```

```
@virsplit=split(/\s+/, $VIR_HITS);
 $VIR = $virsplit[1];


#
# Write data to output_file in comma seperated format.  If the output file exist it is overwritten.
#

open READIN, "$output_file" or die "Can't open $output_file: $!\n";
while (<READIN>) {
    push(@lines, $_);
}

$new_line = "$date,$VIR \n";
close READIN or die "Can't close FILE: $!\n";

my @lines = reverse @lines;
push(@lines,$new_line);
@lines = reverse @lines;

# Here we make a copy of the old file before writing to it
copy("/usr/local/scripts/vir_rbl_file.dat","/usr/local/scripts/vir_rbl_file.dat.old");

open OUTPUT, ">$output_file" or die "Can't open $output_file: $!\n";
foreach(@lines){
   print OUTPUT $_ ;
}
close OUTPUT or die "Can't close FILE: $!\n";
```

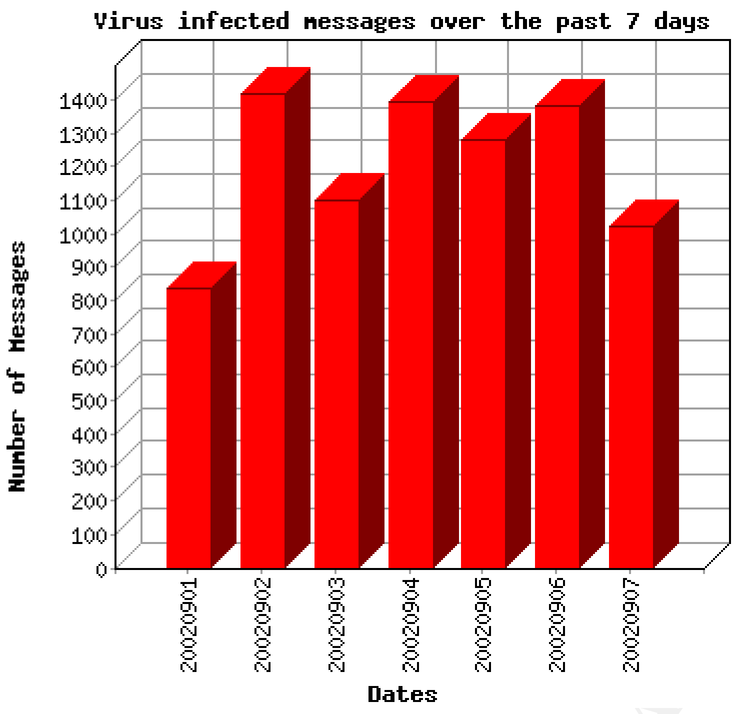
The log file looks like this:

```
20020908, 1248
20020907, 1321
20020906, 1157
…
```

Finally this script is run from a cron job nightly.

Graph Creation

With the data file in comma-separated format we can easily read this data from a small C program, which will use the data to generate the graphs. The gdchart package includes several example programs and the all the options are explained on the web site http://www.fred.net/brv/chart/. Here is an example program that was used to generate the following graph.



```
* ----------------------------------------------------
MTA project
Written by David Evans March 25, 2002

This program creates a bar chart of the number of
Virus infected messages handled by the mta servers for
The past seven days
----------------------------------------------------- */
#include <stdio.h>
#include <values.h>
#include <stdlib.h>
#include "gdc.h"
#include "gdchart.h"

main()
{
    int           num_points = 7;
    int i;
    float data[20];
    int num[29];
    char lbls[10][10];
    FILE          *fl = fopen( "/usr/local/data_files/vir_data.dat"; ", "r" );

    /* ----- set some data ----- */
    float   a[7]  ; //= {6368,5332,5195,5269};
    /* ----- X labels ----- */
    char    *t[7]; //= { "NAMMTA01", "EURMTA01", "EURMTA02",
"FEAMTA01","cgad","dfad","dfads"};
    /* ----- data set colors (RGB) ----- */
    unsigned long   sc[7]    = { 0xFF0000, 0xFF0000, 0xFF0000, 0xFF0000, 0xFF0000, 0xFF0000,
0xFF0000 };
 FILE           *fp = fopen( "/var/www/graphs/past7_vir.gif", "wb" );
```

```
   GDC_BGColor   = 0xFFFFFFL;                  /* backgound color (white) */
   GDC_LineColor = 0x000000L;                  /* line color      (black) */
   GDC_ExtColor  = sc; //&(sc[0]);                   /* assign set colors */
   GDC_bar_width = 60;                   /* (%)              */
   GDC_title = "Virus infected messages over the past 7 days";
   GDC_ytitle = "Number of Messages ";
   GDC_xtitle = "Dates";
   GDC_stack_type = GDC_STACK_BESIDE;
/*
     num[1] = Total Messages recived by all four MTA Servers
     num[2] = Total Messages Reject by RBL on all four MTA Servers
*/

 for(i=num_points; i >= 0;i--) {
   fscanf(fl, "%[^,],%f\n",lbls[i],&data[I]);
}

fclose( fl );
t[0] = lbls[1];
t[1] = lbls[2];
t[2] = lbls[3];
t[3] = lbls[4];
t[4] = lbls[5];
t[5] = lbls[6];
t[6] = lbls[7];

a[0] = data[1];
a[1] = data[2];
a[2] = data[3];
a[3] = data[4];
a[4] = data[5];
a[5] = data[6];
a[6] = data[7];
  /* ----- call to the library ----- */
 out_graph( 400, 350,      /* short       width, height */
              fp,        /* FILE*       open FILE pointer */
              GDC_3DBAR,     /* GDC_CHART_T chart type */
              7,             /* int        number of points per data set */
              t,             /* char*[]     array of X labels */
              1,             /* int        number of data sets */
              a );            /* float[]     data set 1 */
fclose( fp );
   exit(0);
}
```

Compiling Notes: Here are the commands to compile the above program using gcc. This assumes the program source is in the directory gdchart0.94b/ was installed. The default location is /usr/local/gdchart0.94b/

```
gcc -Igd1.3 -c past7_vir.c
gcc -o past7_vir gdc.o gdchart.o price_conv.o past7_vir.o -Lgd1.3 -lgd -lm
```

With the gif image written to /var/www/graphs/past7_vir.gif and this program ran from cron job nightly you can display this graph from a static HTML page. Each time this program is ran it will over write the existing graph and be refreshed with the new data.

One final note in order to process the virus detect data file and create a top ten list of virus detects the following can be used:

```
grep "virus found" /usr/local/data_files/vir_data.dat |awk -F] '{print $2}' |awk '{print $1}'|sort -n
|uniq -c |sort -nr |head -10 > /usr/local/data_files/virus_top10.dat
```

This creates the file /usr/local/data_files/virus_top10.dat which looks like this:

```
    834 W32/Klez-H
    115 W32/Sircam-A
     38 W32/Magistr-B
     21 W32/Yaha-E
      4 W32/Magistr-A
      2 XM97/Laroux-NW
      2 W95/CIH-10xx
      2 W32/Klez-E
      2 W32/Fix2001
      1 WM97/Thus-T
```

With PHP server side scripting you can dynamically create a table and display this in the reporting page. Here is an example of such a PHP script.

```
<?php

// Declare the log file
$input_file = file('/usr/local/data_files/virus_top10.dat');

// Go through each line of the log file
for ($k=0; $k<=count($input_file)-1; $k++) {
    $fields = split("\t",$input_file[$k]);
    print "<tr>";
    print"<td bgcolor=\"#00FFFF\">$fields[1]</td>";
    print"<td bgcolor=\"#00FFFF\">Virus found </td>";
    print"<td bgcolor=\"#00FFFF\"><strong>$fields[0]</strong></td>";
    print"</tr>";
} // ends for loop

?>
```

The table appears in the next section.

Conclusion

After implementing this solution on any given day we are seeing over 1000 virus detection per day and at the height of the Klez outbreak in 2002 over 2000 messages were detected in one day. For example here were the top 10 number of virus detects for August 14, 2002.

Virus Name	DESCRIPTION	Number Of Detects
W32/Klez-H	Virus found	**1262**
W32/Sircam-A	Virus found	**128**
W32/Flcss	Virus found	**121**
W32/Magistr-B	Virus found	**40**
W32/Klez-E	Virus found	**19**
W32/Yaha-E	Virus found	**8**
W95/CIH-10xx	Virus found	**6**
W32/Yaha-D	Virus found	**5**
W95/Spaces	Virus found	**3**
W32/Yaha-A	Virus found	**3**

Our external message server exposure was reduced from hundreds of servers to four and our organization now has a much better handle on its messaging environment.

References

Sun Docs. "iPlanet Messaging Administration Guide" http://docs.sun.com/source/816-6009-10/contents.htm(Aug. 14, 2002).

Sun Docs. "iPlanet Messaging Reference Manual" http://docs.sun.com/source/816-6020-10/contents.htm(Aug. 14, 2002).

Sophos Unix Installation Guide, http://www.sophos.com/sophos/docs/eng/instguid/unix_ien.pdf (Aug. 12, 2002).

RFC 2821, http://www.ietf.org/rfc/rfc2821.txt (Aug. 12, 2002).

Location of wget software, http://wget.sunsite.dk/ (Aug. 12, 2002).

Location of gdchart software, http://www.fred.net/brv/chart/ (Sep. 8, 2002).

Location of YASSP, http://www.yassp.org/ (Sep 2, 2002).

Location of GNU grep, http://www.gnu.org/software/grep/grep.html (Aug. 11, 2002)

MAPS RSS Website, http://work-rss.mail-abuse.org/rss/ (Aug. 12, 2002).