Global Information Assurance Certification Paper

SMTP Gateway Virus Filtering with AMaViS and Postfix

Tod A. Sandman
GSEC Practical Assignment, Version 1.4b Option 2
March 27, 2003

Abstract

Most computer viruses these days are spread via email, and relying solely on the PC client along with its various virus detection schemes invariably proves insufficient. Somehow, some way, the viruses manage to slip through, and we are left suffering the consequences. With the threat posed by viruses ever increasing, creating an additional layer of defense by stopping the viruses at the SMTP gateway is essential.

This paper describes the implementation of a freely available SMTP gateway virus filter, AMaViS, in a university environment where Unix mail servers provide mail for an ever growing number of Windows users. It builds upon the work described by Swab with some differences being that I use Postfix instead of Sendmail as the mail transport agent (MTA), and I use a different variant of the AMaViS software. The reader is taken through the steps required to build and configure the AMaViS software and integrate it with Postfix, and he is shown how an extremely effective layer of defense against the virus threat can be established.

1. Background

I work in the IT division of a small research university of approximately 7000 students. One of the responsibilities of my group is to provide "core" support for the various research divisions. The lines of responsibility are often vague, but the idea is for my group to provide support that can be used across research divisions, such as building software packages, maintaining a Solaris jumpstart and Redhat kickstart infrastructure, etc. While my group is very security conscious, the university has been reluctant to sign off on a security policy, and my group has no real authority. Fortunately, the divisions have over the years become more and more willing to work with us toward a more secure environment. For example, clear text passwords are finally a thing of the past as all the divisions have finally accepted OpenSSH and SSL/TSL enabled tools. In any case, though, we can only make security solutions available; we cannot force anyone to make use of them. Any security solutions we provide must be sell-able to academia.

The university has always been very reluctant to spend money on IT. With the economic downturn the situation has gotten worse, and we are now left scrounging around the dumpsters behind Wal-Mart as opposed to the former luxury of shopping inside. The manpower situation is no better. The SANS SE course explains that good security costs money and requires effort; we have very little of either to spare.

2. Initial State

All incoming SMTP traffic is limited at our campus border gateways to about a dozen mail servers. Two are the central campus mail servers supported by a separate group within IT; these are used by all undergraduate students as well as any divisions that opt to use

them. The rest are for the various divisions that, for one reason or another, still prefer to maintain their own mail servers. All the divisional mail servers are well-secured Unix variants running Postfix. The campus mail servers, over which we have less influence, are running Sendmail.

We have a site license for McAfee virus scanning software and make diligent efforts to provide automatic virus scanning and signature updating for the Windows environment. Regardless, new virus outbreaks always send the PC administrators scrambling to fix infected PCs, and even in the absence of a so called outbreak it seems a PC gets infected beyond repair at least once a week. So far the only damage (as far as we know!) has been wasted user down time, wasted system administrator time to re-blast machines, and the resultant frustration from both. But we are all well aware that the damages could be much worse, such as loss and/or theft of intellectual property, grant proposals, or student/staff information. "Viruses cost businesses around the world more than $1.5 trillion in the year 2000, according to a study by PricewaterhouseCoopers" (ActiveState Website). And all the recent "cyber war" stories in the news linking viruses to the new age of electronic warfare are alarming whether they are well founded or not. With the proliferation of Windows in our environment and the lack of authority to completely control them, the threat of viruses is only getting worse.

3. Strengthening Our Defenses with AMaViS

While improving our client anti-virus infrastructure is an ongoing process, a second layer of defense to cut off viruses at the mail gateway is an obvious benefit. AMaViS is an excellent choice. Using one or more commercially or freely available virus scanning utilities, AMaViS scans the email before it is delivered to the PC. If no viruses are found, a mail header is added that indicates the email has been scanned, and the email is simply passed along to its intended destination. Infected mail, on the other hand, can be "quarantined" to a specified directory, mailed to a configurable address, or deleted altogether. Virus notification can be configured for the sender, recipient, and/or a specified address.

Originally developed by Mogens Kjaer, Juergen Quade and Christian Bricart, and then joined by Lars Hecking, Rainer Link and several others, AMaViS has come a long way and is by all means a fully featured and mature product.

> AMaViS decomposes an e-mail into its constituent parts, decoding attachments, and uncompressing/unarchiving any files contained therein. The parts are then scanned by one or more supported virus scanning utilities. ... AMaViS supports e-mail encoded with the following schemes: MIME, uuencode, TNEF and BinHex, along with ordinary ASCII text. The following archival/compression file formats are also supported: .zip, .Z, .gz, .tar, .zoo, .F, .arc, .arj, bz2, .lha, and .rar, along with self-extracting versions of many of the above. Archived/Compressed files can be nested up to a user-definable depth. (Swab)

Somewhat confusing at first may be the fact that AMaViS has branched out into multiple variants, each being actively developed. Four are currently listed on the main AMaViS download website "http://www.amavis.org/download.php3". After reviewing the features of the variants and testing out a few, I chose Mark Martinec's "amavisd-new". Some important features are the following:

* Provides a persistent daemon that handles all mail as opposed to requiring a separate process for each mail message.

* Supports SMTP input from the MTA; most of the others rely on a piping mechanism from the MTA.

* Integrates extremely well as a content filter with Postfix.

* Utilizes a message cache to help avoid duplicate scans of an identical message.

* Includes excellent anti-spam support that can easily be turned on or off. Many users here have long (and loudly) complained about spam, and (of course) they blame us for it.

From the amavisd-new website:

> amavisd-new grew out of amavisd(-snapshot) (which in turn is a daemonized version of amavis-perl), but over the year of development turned into a separate product, hardly resembling its origin. The code is about three times the size of its predecessor, yet faster in throughput, richer in features, compliant to standards, includes optional support for spam detection, and makes virus scanning optional. Compatibility with helper programs from amavisd(-snapshot) is retained. (Martinec)

4. Building the Software

We have a highly customized software build/package environment. I try to describe the building of the AMaViS components in more general terms to avoid confusing the reader with the specifics of our environment. However, it may be helpful for the reader to be aware of the following:

* My group builds packages for the various divisions, but we do not generally install the packages on the end systems.

* The goal is for each package we release to be a self-contained unit that the divisional system administrators can trivially install and configure on their systems.

* Each package is accessed via /usr/site/NAME-VERSION and possibly /usr/site/NAME; the latter is a symbolic link pointing to the current "default" version of a package.

I assume the user already has Larry Wall's Perl and Wietse Venema's Postfix installed. The home pages for these are "http://www.perl.org/" and "http://www.postfix.org/", respectively, and both are freely available. I assume the reader uses wget to download software, though any method will obviously suffice. Also, I assume the user has gunzip (see Section 4.2.4 if you don't). While I actually built the software on all of our supported platforms (Solaris 8, Aix 4.3.3, Irix 6.5 and Redhat 7.2), the steps shown were done on Solaris 8.

The obvious initial step to building AMaViS is to download and unpack the AMaViS software (note that I have decided to name the package "amavis-20021227p2"):

```
mkdir /var/tmp/amavis; cd /var/tmp/amavis
wget http://www.ijs.si/software/amavisd/amavisd-new-20021227-p2.tar.gz
gunzip -c < amavisd-new-20021227-p2.tar.gz | tar -xf -
mkdir -p /usr/site/amavis-20021227p2/{sbin,etc}
cp -p amavisd-new-20021227/amavisd /usr/site/amavis-20021227p2/sbin
cp -p amavisd-new-20021227/amavisd.conf /usr/site/amavis-20021227p2/etc
```

That's it! amavisd-new consists solely of a daemon (amavisd) and its configuration file (amavisd.conf). The daemon is a perl script, so no compiling is necessary! Actually, amavisd-new requires many external helper programs and Perl modules and can make use of many more, so the software building is just getting started. The best way to figure out exactly what you need is to fire up the daemon with the "debug" option and see what it says - it is quite helpful in saying what it wants. Since you have not yet configured Postfix to use AMaViS, firing up the daemon is harmless. In the following sections I categorize and describe the building of the required components.

4.1. Perl Modules

Of the several required perl modules, only two were missing from our standard perl package. These are Net-Server and Mail-SpamAssassin (required only if you want anti-spam support). We do not alter a package once we release it, so installing the perl modules in the normal place (within the perl package itself) is not an option in my environment. Instead, the modules must be embedded within the amavis-20021227p2 package, and a "use lib" line must be added to amavisd so it can find the modules. You may prefer the more general approach of using the CPAN module to update your perl installation. Either way, the steps are quite trivial, and adding any other modules you may be missing can be done similarly.

4.1.1. Net-Server

```
cd /var/tmp/amavis
wget http://www.cpan.org/authors/id/R/RH/RHANDOM/Net-Server-0.84.tar.gz
gunzip -c < Net-Server-0.84.tar.gz | tar -xf -
cd Net-Server-0.84
perl Makefile.PL PREFIX=/usr/site/amavis-20021227p2 \
  LIB=/usr/site/amavis-20021227p2/lib
make; make test; make install
```

4.1.2. Mail-SpamAssassin

```
cd /var/tmp/amavis
wget http://spamassassin.org/released/Mail-SpamAssassin-2.50.tar.gz
gunzip -c < Mail-SpamAssassin-2.50.tar.gz | tar -xf -
cd Mail-SpamAssassin-2.50
perl Makefile.PL PERL_BIN=/usr/site/perl/bin/perl \
  PREFIX=/usr/site/amavis-20021227p2 \
  LIB=/usr/site/amavis-20021227p2/lib \
  LOCAL_RULES_DIR=/usr/site/amavis-20021227p2/etc \
make; make test; make install
```

4.2. Archive and Compression Tools

AMaViS makes use of several external programs for message decomposition. Some of these were familiar to me and already existed in our environment. The rest were a bit obscure and some hard to find; for these, I copied the binaries into the AMaViS package directory as opposed to creating a unique package for each. Missing some of these tools is not

catastrophic; those message components that cannot be decomposed will simply be passed to the scanner as is. The "enhanced" file command is also required in order to determine which tool to use.

4.2.1. arc - extract .arc archive

```
mkdir /usr/site/amavis-20021227p2/libexec
cd /var/tmp/amavis
wget http://pq.ozersk.ru/ftp/unix/freebsd/arc521e.pl8.tar.Z
mkdir arc521e; cd arc521e; zcat ../arc521e.pl8.tar.Z | tar -xf -
perl -i.DIST -p -e 's/DBSD=1/DBSD=1 -DSYSV=1/' Makefile
perl -i -p -e 's:^(.*-o m?arc.*)$:$1 -L/usr/ucblib -lucb:' Makefile
make CC=gcc
strip arc marc; cp -p arc marc /usr/site/amavis-20021227p2/libexec
```

4.2.2. bzip2 - uncompress .bz2 file

```
cd /var/tmp/amavis
wget --passive-ftp ftp://sources.redhat.com/pub/bzip2/v102/bzip2-1.0.2.tar.gz
gunzip -c < bzip2-1.0.2.tar.gz | tar -xf -
cd bzip2-1.0.2
mkdir -p /usr/site/bzip2-1.0.2/{bin,lib,include}
make; make install PREFIX=/usr/site/bzip2-1.0.2
```

4.2.3. file - utility for determining file types

```
cd /var/tmp/amavis
wget --passive-ftp ftp://ftp.astron.com/pub/file/file-3.41.tar.gz
gunzip -c < file-3.41.tar.gz | tar -xf -
cd file-3.41;
mkdir -p /usr/site/file-3.41/{bin,etc}
./configure --prefix=/usr/site/file-3.41 --datadir=/usr/site/file-3.41/etc
make; make install
```

4.2.4. gzip - uncompress .gz files

```
cd /var/tmp/amavis
wget --passive-ftp ftp://ftp.gnu.org/gnu/gzip/gzip-1.2.4a.tar
tar -xf gzip-1.2.4a.tar
cd gzip-1.2.4a
mkdir -p /usr/site/gzip-1.2.4a/{bin,lib,info}
./configure --prefix=/usr/site/gzip-1.2.4a
make; make install
```

4.2.5. lha - extract .lha archive

To download the source code for lha, you must fill out the form:

http://www.infor.kanazawa-it.ac.jp/~ishii/lhaunix/download.html

Alternately, you can download pre-compiled binaries for several Unix variants. Below I grab the pre-compiled Solaris binary.

```
cd /var/opt/amavis
wget http://www.infor.kanazawa-it.ac.jp/~ishii/lhaunix/solaris/lha117.tar.gz
gunzip -c < lha117.tar.gz | tar -xf -
strip lha; cp -p lha /usr/site/amavis-20021227p2/libexec
```

4.2.6. unarj - extract .arj archive

```
cd /var/tmp/amavis
wget --passive-ftp ftp://metalab.unc.edu/pub/Linux/utils/compress/unarj-2.63a.tar.gz
gunzip -c < unarj-2.63a.tar.gz | tar -xf -
cd unarj-2.63a; make; cp -p unarj /usr/site/amavis-20021227p2/libexec
```

4.2.7. unrar - extract .rar archive

```
cd /var/tmp/amavis
wget --passive-ftp ftp://metalab.unc.edu/pub/Linux/utils/compress/unrar-2.71.tar.gz
gunzip -c < unrar-2.71.tar.gz | tar -xf -
cd unrar-2.71; make clean; make; cp -p unarj /usr/site/amavis-20021227p2/libexec
```

4.2.8. zoo - extract .zoo archive

```
cd /var/tmp/amavis
wget http://www.netsw.org/system/tools/fileutils/archive/zoo/zoo-2.10pl1.tar.gz
mkdir zoo; cd zoo; gunzip -c < ../zoo-2.10pl1.tar.gz | tar -xf -
make CC=gcc bsdansi; strip zoo; cp -p zoo /usr/site/amavis-20021227p2/libexec
```

4.3. External Virus Scanners

Last but not least, AMaViS requires one or more virus scanning utilities. amavisd-new supports a huge list of these, and due to its modular design adding new ones should be trivial. At initial start up, amavisd checks for each scanner defined in the amavisd.conf file, generates a subroutine corresponding to each scanner found, and saves the list of subroutines. Thereafter, each email is decomposed into its constituent parts and each subroutine in the list is run on each of those parts. By default, all the definitions of the non-daemonized scanners are not commented out and will thus be checked for at start up. All daemonized scanner definitions are commented out by default since checking for these requires a connection attempt (and time-out if no one is listening) to the daemon. Jumping ahead a bit, here is a log file excerpt to give you an idea of the supported scanners:

```
amavisd[94]: starting.  amavisd at some.host.edu amavis-20021227p2
amavisd[94]: No av scanner: Sophos SAVI
amavisd[94]: Found av scanner Clam Antivirus - clamscan at /usr/site/clamav/bin/clamscan
amavisd[94]: No av scanner: KasperskyLab AntiViral Toolkit Pro (AVP)
amavisd[94]: No av scanner: KasperskyLab AVPDaemonClient
amavisd[94]: No av scanner: H+B EDV AntiVir
amavisd[94]: No av scanner: Command AntiVirus for Linux
amavisd[94]: No av scanner: Symantec CarrierScan via Symantec CommandLineScanner
amavisd[94]: No av scanner: DrWeb Antivirus for Linux/FreeBSD/Solaris
amavisd[94]: No av scanner: FRISK F-Prot Antivirus/Linux
amavisd[94]: No av scanner: F-Secure Antivirus
amavisd[94]: No av scanner: CAI InoculateIT
amavisd[94]: No av scanner: MkS_Vir for Linux (beta)
amavisd[94]: No av scanner: ESET Software NOD32
amavisd[94]: No av scanner: ESET Software NOD32 - Client/Server Version
amavisd[94]: No av scanner: Norman Virus Control v5 / Linux
amavisd[94]: No av scanner: Panda Antivirus for Linux
amavisd[94]: No av scanner: GeCAD RAV AntiVirus 8
amavisd[94]: Found av scanner NAI McAfee AntiVirus (uvscan) at /usr/site/virusscan/bin/uvscan
amavisd[94]: No av scanner: VirusBuster (Client + Daemon)
amavisd[94]: No av scanner: Sophos Anti Virus (sweep)
amavisd[94]: No av scanner: CentralCommand Vexira - engine based on H+BEDV AntiVir/X
amavisd[94]: No av scanner: CyberSoft VFind
amavisd[94]: No av scanner: Trend Micro FileScanner
```

4.3.1. McAfee

Our McAfee site license includes the Unix command line program, so naturally I built a package for it to be used by AMaViS. Creating the package involves downloading a tarball

from a password protected site for each of our supported Unix platforms and fitting the contents into our package mold. The only relevant points here are the location of the program binary and its virus signature files, which are respectively

```
/usr/site/virusscan-4.24.0/bin/uvscan
/usr/site/virusscan-4.24.0/share/dat/*.dat.
```

Keeping the virus signature files and the virus engines themselves up to date is paramount.

> A virus-scanning engine that is out of date is almost as bad as not having virus scanning at all. It should be possible to configure the SMTP gateway to automatically download and install updated virus definitions on a predetermined schedule. (Steen)

McAfee provides anonymous FTP access to their virus signature files, which are updated at least weekly. I created a separate package, vscan_update-0.0, to make this as simple and foolproof as possible for the divisional system administrators. Two scripts are provided: "update_dat" updates the virusscan dat files via FTP, and "update_dat_nfs" does so via NFS. With the following crontab entry, updates are checked for hourly:

```
## Update virusscan dat files.
42 * * * *  /usr/site/vscan_update/bin/update_dat -qm virusalert
```

With the -q option, notification is sent to virusalert only if an update (or an error) occurs:

```
To: virusalert@somewhere.edu
Subject: update_dat on host.somewhere.edu
Date: Wed, 26 Mar 2003 12:41:39 -0600 (CST)

Dat file update from version 4253 to 4254 successful!
```

The McAfee virus engine itself (the program) is updated much less often. Unfortunately, we currently have no procedure in place for obtaining the new binaries in a timely manner. Once we obtain the binaries, creating a new "virusscan-xx" package is trivial, but we currently rely on the vendor to notify us when a new version is released. Resolving this issue is a work in progress.

4.3.2. Clam AntiVirus

Adding virus scanning at the gateway as a layer on top of the virus scanning at the PC client is an excellent step toward defense in depth. Utilizing multiple virus scanners decreases our vulnerability even more. As explained by Evans,

> When selecting the Anti-virus product it was important to choose a different company other than McAfee, 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 upon for virus protection. This follows the defense in depth strategy, which is a core concept in many areas of information security. (Evans)

An excellent virus scanning tool, "Clam AntiVirus", is freely available and easily builds

on each of our supported Unix platforms. From the Clam AntiVirus website:

> Clam AntiVirus is an anti-virus toolkit written from scratch. It is licensed under GNU GPL2 and uses the virus database from OpenAntiVirus, which is an another free anti-virus project. In contrast to OpenAntiVirus (which is written in Java), Clam AntiVirus is written entirely in C and its database is KEPT UP TO DATE. It also detects polymorphic viruses as well. (Kojm)

Clam AntiVirus requires bzip2 and zlib, and you may need to use CFLAGS and LDFLAGS to specify your bzip2 and/or zlib locations.

```
CFLAGS="-O2 -I/usr/site/bzip2/include -I/usr/site/zlib/include"
LDFLAGS="-L/usr/site/bzip2/lib -L/usr/site/zlib/lib"
CC=gcc; export CC CFLAGS LDFLAGS
./configure --prefix=/usr/site/clamav-20030123 \
    --sysconfdir=/usr/site/clamav-20030123/etc \
    --with-dbdir=/usr/site/clamav-20030123/share/dat \
    --disable-clamuko --disable-clamav --with-user=vscan \
    --with-group=vscan --with-db1=viruses.db --with-db2=viruses.db2
make; make install
```

Clam AntiVirus includes a fully featured signature updating command, "freshclam", that can be run via cron or as a daemon to keep the data files up to date:

```
/usr/site/clamav/bin/freshclam -v --stdout
Checking for a new database - started at Thu Mar 27 11:11:30 2003
Current working dir is /usr/site/clamav-20030123/share/dat
Connected to clamav.elektrapro.com.
Reading md5 sum (viruses.md5): OK
Reading md5 sum (viruses2.md5): OK
viruses.db2 is up to date.
Downloading viruses.db ..........done
Database updated (containing in total 7764 signatures).
```

5. AMaViS Configuration

AMaViS supports a vast array of configuration options. The amavisd.conf file (which is actually perl code sourced by amavisd) that comes with the package describes each option, and even though it is well organized it can be quite daunting at first glance. As commented at the top of the file, amavisd.conf is organized into several sections:

```
#Sections:
# Section I   - Essential daemon and MTA settings
# Section II  - MTA specific
# Section III - Logging
# Section IV  - Notifications, quarantine
# Section V   - Per-recipient and per-sender handling, white-listing, etc.
# Section VI  - Resource limitations
# Section VII - External programs, virus scanners, SpamAssassin
```

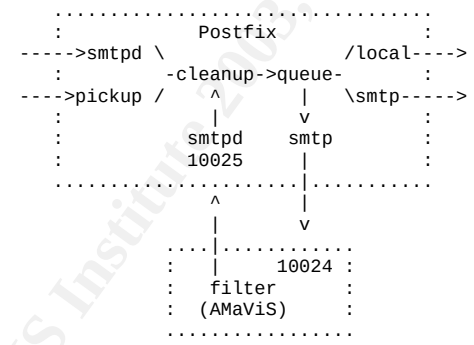
In the following sections I show a few of the available options. While I have basically cut and pasted from the amavisd.conf file, note that I have left out many, many options in between. Hopefully, the reader will get an idea of how feature rich and configurable AMaViS is without getting lost in the details.

5.1 Daemon and MTA Settings

```
## Specify a path string, which is a colon-separated string of directories
## (no trailing slashes!) to be assigned to the environment variable PATH
## and to serve for locating external programs below.
```

```
$path = '/usr/site/amavis-20021227p2/libexec:/usr/site/bin:' .
  '/usr/site/virusscan/bin:/usr/site/clamav/bin:/usr/sbin:/sbin:/usr/bin:/bin';

## Set the user and group to which the daemon will change when started as root:
$daemon_user  = 'vscan';        # (no default;  customary: vscan or amavis)
$daemon_group = 'vscan';        # (no default)

## POSTFIX or EXIM V4 or dual MTA setup (set host IP and port num as required)
$forward_method = 'smtp:127.0.0.1:10025';  # where to forward checked mail
$notify_method  = 'smtp:127.0.0.1:10025';  # where to submit notifications

## Net::Server pre-forking settings
## You may want $max_servers to match the width of your MTA pipe
## feeding amavisd, e.g. with Postfix the 'Max procs' field in the
## master.cf file, like the '2' in the:  smtp-amavis unix - - n - 2 smtp
$max_servers  =  2;   # number of pre-forked children          (default 2)

## Here is a QUICK WAY to completely DISABLE some sections of code
## that WE DO NOT WANT (it won't even be compiled-in).
## For more refined controls leave the following two lines commented out,
## and see further down what these two lookup list really mean.
#@bypass_virus_checks_acl = qw( . );  # uncomment to DISABLE anti-virus code
@bypass_spam_checks_acl  = qw( . );  # uncommend to DISABLE anti-spam code

## SMTP-INPUT PROTOCOL SETTINGS (e.g. with Postfix, Exim v4, ...)
## (used when MTA is configured to pass mail to amavisd directly via SMTP)
$inet_socket_port = 10024;        # accept SMTP on this local TCP port
```

The $inet_socket_port setting specifies the local TCP port on which amavisd will listen for incoming mail from the MTA. Using its content filtering mechanism, we will configure Postfix to pass all mail to this port (except, of course, for mail coming from amavisd). The $forward_method specifies where to send mail that has been proccessed and has received a passing grade. We will pass it back to the MTA via a Postfix SMTP daemon listening on 127.0.0.1:10025. The following sketch from the Postfix "FILTER_README" file (Venema) illustrates our setup:



5.2 Notifications, Quarantine, Etc.

```
## Notify virus sender/recipient?
## (for admin notifications see $virus_admin and $spam_admin)
$warnvirussender = 0;   # (defaults to false (undef))
$warnvirusrecip  = 1;   # (defaults to false (undef))

## Notify spam sender?
$warnspamsender = 0;    # (defaults to false (undef))

## - the administrator address may be a simple fixed e-mail address (a scalar),
##   or may depend on the SENDER address (e.g. its domain), in which case
##   a ref to a hash table can be specified (specify lower-cased keys,
##   dot is a catchall, see README.lookups).
##   Empty or undef lookup disables virus admin notifications.
# $virus_admin = undef;   # do not send virus admin notifications (default)
# $virus_admin = {'not.example.com' => '', '.' => 'virusalert@example.com'};
$virus_admin = 'virusalert';

## equivalent to $virus_admin, but for spam admin notifications:
```

```
$spam_admin = 'spamalert';

## Location to put infected mail into:
## - empty for not quarantining, may be a file (mailbox),
##   or a directory (no trailing slash)
## (the default value is undef, meaning no quarantine)
$QUARANTINEDIR = '/var/opt/amavis/virusmails';

## Add X-Virus-Scanned line to mail?     # (default: undef)
$X_HEADER_TAG = 'X-Virus-Scanned';
## Leave empty to add no header          # (default: undef)
$X_HEADER_LINE = 'by amavis-20021227p2';

$remove_existing_x_scanned_headers = 0; # leave existing X-Virus-Scanned alone
#$remove_existing_x_scanned_headers= 1; # remove existing headers
                                       # (defaults to false)

## set $bypass_decode_parts to true if you have a good virus scanner
## that can deal with compression and recursively unpacking archives by itself,
## and save amavisd the trouble. It is a double-edged sword, make sure you know
## what you are doing!
$bypass_decode_parts = 0;              # (defaults to false)

## Specify what to do with infected mail and spam
$final_virus_destiny  =  0;  # -1=reject, 0=discard, 1=pass;  (defaults to  0)
$final_spam_destiny   =  1;  # -1=reject, 0=discard, 1=pass;  (defaults to -1)
```

Clearly, AMaViS is extremely flexible in how it handles infected mail and/or spam. Precisely what gets sent where can be controlled with extremely fine granularity. Although not shown above, flexible lookup tables can also be used if you want certain users to receive their infected mail and/or spam. Likewise, white-listing and black-listing of specific senders is also fully supported.

5.3 Resource Limitations

```
## Maximum recursion level for extraction/decoding (0 or undef disables limit)
$MAXLEVELS = 14;               # (default is undef, no limit)

## Maximum number of extracted files (0 or undef disables the limit)
$MAXFILES = 1500;              # (default is undef, no limit)

## For the cumulative total of all decoded mail parts we set max storage size
## to defend against mail bombs. Even though parts may be deleted (replaced
## by decoded text) during decoding, the size they occupied is _not_ returned
## to the quota pool.
##
## Parameters to storage quota formula for unpacking/decoding/decompressing
##   Formula:
##     quota = max($MIN_EXPANSION_QUOTA,
##                 $mail_size*$MIN_EXPANSION_FACTOR,
##                 min($MAX_EXPANSION_QUOTA, $mail_size*$MAX_EXPANSION_FACTOR))
##   In plain words (later condition overrules previous ones):
##     allow MAX_EXPANSION_FACTOR times initial mail size,
##     but not more than MAX_EXPANSION_QUOTA,
##     but not less than MIN_EXPANSION_FACTOR times initial mail size,
##     but never less than MIN_EXPANSION_QUOTA
##
$MIN_EXPANSION_QUOTA =      100*1024;  # bytes  (default undef, not enforced)
$MAX_EXPANSION_QUOTA = 300*1024*1024;  # bytes  (default undef, not enforced)
$MIN_EXPANSION_FACTOR =   5;  # times original mail size  (must be specified)
$MAX_EXPANSION_FACTOR = 500;  # times original mail size  (must be specified)
```

Since AMaViS decomposes each message, there is always the danger of DOS attacks. AMaViS allows you to choose the trade-offs. If a limit is hit, amavisd will stop decomposing the message any further and pass the whole message (or partially decomposed components from before the limit was hit) on to the virus scanners.

6. AMaViS Installation

Finally it is time to fire up AMaViS! Here is the basic recipe (see the "README.postfix" file in the amavisd-new distribution for more details):

```
* Create the "vscan" user and group.  Sample entries are, resp.,
    vscan:x:80:80:AMaViS User:/NO_SUCH_HOME_DIR:/bin/false
    vscan:*:80:

* Create a "virusalert" mail alias.

* Append the following lines to your Postfix main.cf file:
    ## Content filtering with amavis:
    content_filter = amavis:[127.0.0.1]:10024

* Append the following lines to your postfix master.cf file:
    ## Content filtering with amavis, MTA -> AMaViS:
    amavis unix - - n - 2 smtp
    ## Content filtering with amavis, AMaViS -> MTA:
    127.0.0.1:10025 inet n - n - - smtpd -o content_filter=

* Run the following:
    mkdir -p /var/opt/amavis/virusmails
    chown -R vscan:vscan /var/opt/amavis;
    chmod -R 700 /var/opt/amavis
    /usr/site/amavis-20021227p2/sbin/amavisd -c /usr/site/amavis/etc/amavisd.conf
    /etc/init.d/postfix reload
```

Note that testing and debugging your AMaViS configuration without interfering with your mail system's normal operation is simple and straightforward. For instance, if you follow all the steps above except the main.cf update step (the third), you can send test messages directly to amavisd (port 10024) and watch amavisd process the message (pass it to Postfix, quarantine it, or whatever). Furthermore, adding the word "debug" to the amavisd start up line causes the daemon to stay in the foreground and print out a wealth of information about what it is doing.

I first installed AMaViS on the IT division's mail server, a Sun E250 (2x300MHz, 1G RAM) that handles about 14,000 messages per day. CPU load and disk I/O increased a bit as I expected, but the server easily handled it. The first week AMaViS blocked 97 infected emails.

The following week I officially released amavis-20021227p2 and assisted the divisional system administrators in updating their systems. Very confident with the AMaViS software (and Postfix, of course - hats off to Wietse Venema), our main concern was with overloading some of the less powerful machines. The most heavily loaded server (for its size) is a Sun Blade 100 (1G Ram) that handles about 18,000 messages per week day, and it has had no problems whatsoever.

Within 2 months, spurred on by the results my group had with AMaViS and Postfix on the divisional servers, the campus administrators installed Postfix and amavis-20021227p2 on the campus servers. These handle over 65% of the university's mail, over 3 million messages weekly, so the system administrators were more concerned than my group over load issues. The servers, actually a system composed of 2 Sun Fire V480s, 2 Sun Fire 280Rs, a dozen back-end Sun Fire V120s, and 4 Sun T3 disk arrays, have purred along flawlessly.

7. Issues

The only issues we ran into were per division policy issues dealing with the handling of

infected mail, and these were easily resolved with simple changes in the configuration file. Actually, these might not be considered issues at all, but rather changes to my initial default configuration. These issues would certainly be handled differently in the commercial world.

7.1 Recipient Notification

I initially decided to turn on "recipient notification"; if a message got blocked at the mail gateway, the intended recipient would receive a notification like this:

```
From: amavis@some.host.edu
Subject: VIRUS (W32/Nimda.eml)  IN MAIL TO YOU (from (someone))
To: undisclosed-recipients: ;

VIRUS ALERT

Our content checker found
    W32/Nimda.eml
virus in an email to you from someone sender.

The message has been quarantined as:
   /var/opt/amavis/virusmails/virus-20030228-143633-20898-01

Please contact your system administrator for details.
```

I did not want infected mail to just disappear with no one knowing, portraying a faulty mail system. I did not want to activate sender notification due to issues with spammers, faked sender addresses, etc., and I didn't want the system administrators to be responsible for tracking down a user or two for every infected mail message. Our users tended to prefer the black hole approach over being notified, so in most of the divisions recipient notification has been turned off.

7.2. System Administrator Notification

A similar situation arose with "admin notification". For every infected email blocked at the gateway, the "virus admin" would receive a notification like this:

```
From: amavis@some.host.edu
Subject: VIRUS (W32/Nimda.eml) FROM LOCAL (somewhere)
To: <virusalert@some.host.edu>

A virus (W32/Nimda.eml) was found.

Scanner detecting a virus: NAI McAfee AntiVirus (uvscan)

The mail originated from: <somewhere>

According to the 'Received:' trace, the message originated at:
   by some.host.edu (Postfix, from userid 2796) \
   id 2028C8BCA; Fri, 28 Feb 2003 14:36:27 -0600 (CST)

Notification to sender will not be mailed.

The message WAS NOT delivered to:
<someone@some.host.edu>:
   250 2.7.0 Ok, discarded, id=20898-01 - VIRUS: W32/Nimda.eml

Virus scanner output:
   Scanning /var/opt/amavis/amavis-20030228T143631-20898/parts/*
   Scanning file /var/opt/amavis/amavis-20030228T143631-20898/parts/part-00001
   /var/opt/amavis/amavis-20030228T143631-20898/parts/part-00001
           Found the W32/Nimda.eml virus !!!

   Summary report on /var/opt/amavis/amavis-20030228T143631-20898/parts/*
   File(s)
           Total files: ...........       1
```

```
          Clean: .................       0
          Possibly Infected: .....       1

The message has been quarantined as:
   /var/opt/amavis/virusmails/virus-20030228-143633-20898-01

------------------------ BEGIN HEADERS -----------------------------
Received: by some.host.edu (Postfix, from userid 2796)
        id 2028C8BCA; Fri, 28 Feb 2003 14:36:27 -0600 (CST)
To: someone@some.host.edu
Subject: test
Message-Id: <20030228203627.2028C8BCA@some.host.edu>
Date: Fri, 28 Feb 2003 14:36:27 -0600 (CST)
From: someone@some.host.edu (Some One)
------------------------- END HEADERS -----------------------------
```

This is excellent information. It was kind of fun at first but got old fast. We simply don't have the time to attend to all the alerts, and most of us now filter them away or have deactivated admin notification altogether.

7.3 To Save or Not to Save

The final issue we encountered regarded dealing with all the infected mail. Always paranoid about loosing a single message, I had initially set up quarantining of infected messages into a local directory. With disks filling up and no time to pay attention to those infected messages anyway, most of us have simply turned off quarantining.

8. Conclusion

The number of infected messages blocked at the gateway has been staggering, and the occurrence of viruses on the PCs has practically vanished. The divisional servers average about 100 blocked messages each per week. Below is a typical weekly mail check of one of the divisional servers using a simple script derived from David Schweikert's Mailgraph program (Schweikert):

```
Mail stats from Sat Mar  1 04:05:04 2003 to Sat Mar  8 04:05:03 2003 (7 days; 06:23:59:59).

      sent   received    bounced   rejected      virus       spam
     69257      44945       1677       4567        132          0

Exploit.IFrame                          2
Exploit.IFrame, W32/Brid.Worm           1
Exploit.IFrame, W32/Yaha.g.dam         38
Exploit.IFrame.HTML, Worm/Klez.H       39
Sircam                                  3
W32/Gibe.b@MM                           1
W32/Lirva.dam                           2
W32/Yaha.g.dam                         27
Worm.Sobig.A                            6
Worm/Klez.H                            12
Yaha.K                                  1
```

The campus servers collectively block about 20,000 viruses weekly. Here are their combined statistics for last week, cut and pasted from their website running the Mailgraph program:

```
Sent      1623821;  average: 167.71 msgs/min;   max: 3408 msgs/min
Received  1914120;  average: 197.62 msgs/min;   max: 1905 msgs/min
Viruses     20985;  average:   2.17 msgs/min;   max:   36 msgs/min
```

The value of an SMTP gateway virus filter in my type of environment is apparent. Besides improving our security posture by strengthening our resilience to infections, wasted time

spent tending to infected machines is a thing of the past. The extra horse power required at the gateway is well worth the cost, and in most cases it is already there.

AMaViS is a very well designed gateway filter. It takes full advantage of the power of perl and many time tested perl modules as much as possible, allowing it to concentrate on its main objective - handling infected mail and spam. And Postfix, with its robust and modular design and simple configuration, is a godsend.

References

ActiveState Website.
URL: http://www.activestate.com/Products/PureMessage/viruspack.plex (March 27, 2003)

Evans, David A. "MTA based Virus Scanning with Sun One Messaging and Sophos".
Global Information Assurance Certification.
URL: http://www.giac.org/practical/David_Evans_GSEC.doc (March 27, 2003)

Heinrichs, Thomas A. "Stopping Viruses at a Unix Mail Gateway".
SANS Institute Information Security Reading Room. August 20, 2001.
URL: http://www.sans.org/rr/email/stop_virus.php (March 27, 2003)

Kojm, Tomasz. Clam AntiVirus Homepage.
URL: http://clamav.elektrapro.com/ (March 27, 2003)

Martinec, Mark. amavisd-new. March 24, 2003
URL: http://www.ijs.si/software/amavisd/ (March 27, 2003)

Schweikert, David. "Mailgraph - a RRDtool frontend for Postfix statistics".
URL: http://people.ee.ethz.ch/~dws/software/mailgraph/ (March 27, 2003)

Steen, Eric. "The Case for an SMTP Gateway Anti-Virus System".
SANS Information Security Reading Room. March 9, 2001.
URL: http://www.sans.org/rr/email/SMTP.php (March 27, 2003)

Swab, Kevin. "SMTP Gateway Virus Filtering with Sendmail and AMaViS"
SANS Information Security Reading Room. August 8, 2001
URL: http://www.sans.org/rr/toppapers/amavis.php (March 27, 2003)

Venema, Wietse. "The Postfix Home Page".
URL: http://www.postfix.org/ (March 27, 2003)

Appendix

A. Spam Support

As mentioned earlier, amavis-20021227p2 has built-in SpamAssassin support. I disabled it by default in order to concentrate on the virus issue. Recently I enabled it on the IT divisional server, and the results are very promising. From the SpamAssassin man page:

> SpamAssassin is a module to identify spam using text analysis and several internet-based realtime blacklists. Using its rule base, it uses a wide range of heuristic tests on mail headers and body text to identify spam.

As in the case of infected mail, how spam is handled is completely configurable. After decomposing a message and checking it for viruses, the components of the message (including the headers) are analyzed with a variety of tests, the results of which are added to provide a fuzzy spam rating for the message. Based on configurable thresholds, a header can be added that shows the overall spam rating of the message and/or the individual tests that contributed to the spam rating. Here is an example of the headers added for a message that passed the "notify" threshold but not the "this is spam" threshold:

```
X-Virus-Scanned: by amavis-20021227p2
X-Spam-Status: No, hits=4.0 tagged_above=4.0 required=6.9 tests=HTML_40_50,
 HTML_FONT_COLOR_BLUE, HTML_FONT_COLOR_UNSAFE, HTML_MESSAGE,
 MAILTO_TO_SPAM_ADDR, MORTGAGE_BEST
X-Spam-Level: ****
```

Here are 2 examples of more blatant spam:

```
X-Virus-Scanned: by amavis-20021227p2
X-Spam-Status: Yes, hits=10.3 tagged_above=4.0 required=6.9 tests=CLICK_BELOW,
 EXCUSE_14, HAIR_LOSS, HGH, HTML_40_50, HTML_FONT_BIG_B, HTML_LINK_CLICK_HERE,
 HTML_MESSAGE, HTML_TAG_EXISTS_TBODY, INCREASE_SEX, MAILTO_TO_SPAM_ADDR,
 SOME_BREAKTHROUGH
X-Spam-Level: **********
X-Spam-Flag: YES

X-Virus-Scanned: by amavis-20021227p2
X-Spam-Status: Yes, hits=25.0 tagged_above=4.0 required=6.9
 tests=BANG_EXERCISE, BAYES_90, CLICK_BELOW, DATE_IN_PAST_12_24, DCC_CHECK,
 FORGED_MUA_OIMO, FROM_ENDS_IN_NUMS, HTML_20_30, HTML_LINK_CLICK_HERE,
 IMPOTENCE, INVALID_DATE, MIME_HTML_ONLY, MISSING_MIMEOLE, MONEY_BACK,
 MSG_ID_ADDED_BY_MTA_3, PENIS_ENLARGE, PENIS_ENLARGE2, SOME_BREAKTHROUGH
X-Spam-Level: *************************
X-Spam-Flag: YES
```

I left out the message bodies for obvious reasons, but I think the point is clear.

We have many users who have been practically demanding anti-spam support. They certainly seem to care more about this than virus protection, and we will begin using it in the divisions within the month.

As with the virus detection, one concern is the additional load required to analyze each message for spam. A convenient option allows you to set a threshold above which spam analysis will be skipped. The idea is to not waste time on huge messages since these are more difficult to analyze and make up only a tiny percentage of spam. This parameter is specified in amavisd.conf:

```
$sa_mail_body_size_limit = 64*1024;  # don't waste time on SA if mail is larger
                          # (less than 1% of spam is > 64k)
                          # default: undef, no limitations
```

Though it certainly shouldn't be, this anti-spam support may be what you need to sell AMaViS to your users (or boss).

B. AMaViS Startup Script

```
#!/bin/sh
################################################################################
# AMaViS init script.
#
# description: AMaViS Mail Virus Scanner
#
```

```
#############################################################################

PATH=/usr/sbin:/bin:/usr/bin:/usr/ucb
program=‘basename $0‘

service="AMaViS"
daemon="/usr/site/amavis/sbin/amavisd"
config_file="/usr/site/amavis/etc/amavisd.conf"
pid_file="/var/opt/amavis/amavisd.pid"
request=$1

## If your OS does not allow PERL to change the real uid of the process
## (variable $<), you will have to su by uncommenting out "app_user=vscan".
#app_user=vscan
su_opts=

Error () { echo $@ 1>&2; exit 1; }

case $request in

  ’start’)
    message="Starting $service"
    ## Clean up a bit, but make sure amavis is not running. See the FAQ for details.
    if [ ! -f /var/opt/amavis/amavisd.pid ]; then
      find /var/opt/amavis -type d -name ’amavis-20??????T*’ -prune -exec rm -rf {} \;
    fi
    command="$daemon -c $config_file"
    ;;

  ’stop’)
    message="Shutting down $service"
    command="$daemon -c $config_file stop"
    ;;

  ’reload’)
    message="Reloading $service"
    command="$daemon -c $config_file reload"
    ;;

  *)
    Error "${program}: usage: $program {start|stop|reload}"
    ;;

esac

echo $message

if [ -n "$app_user" -a x‘whoami‘ != x$app_user ]; then
  su $su_opts $app_user -c "$command"; status=$?
else
  $command; status=$?
fi

[ $status -eq 0 ] || echo "${program}: FAILED to $request $service"
exit $status
```