SANS GSEC Paper:

Spam Filtering: Large Site Principles At A Small Site

Greg Williamson.

October 10, 2003.

Abstract

This paper will describe my experiences in installing spam filtering for my home office. The principles and methods described in this paper can easily be extended to a large site, supporting thousands of users and processing tens or hundreds of thousands of messages per day. The implementation described currently supports three domains, with a small, known set of users. These users have differing needs, differing backgrounds, and accordingly differing tolerances for spam. Although the total number of users behind this protective filter is small, the model described within this paper can easily scale.

The paper covers the various stages of this implementation, from the initial situation, through the steps taken to first start filtering mail. I will then cover the later steps, undertaken to fix some of the problems introduced by the first incarnation of spam filtering. Finally, I will discuss the current state of spam for the internet in general, and also some ideas about what the online community can do to try to minimize the prevalence of spam.

Initial Situation

Spam had become part of every day's email. An average day would bring 10 to 20 unsolicited commercial email messages, but using a text based mail client (elm) meant that they were little more than an inconvenience. No nasty images displayed, no web bugs worked. As the primary user of this mailserver, this was an acceptable situation. Virus scanning was implemented on the desktop, and had been effective in prevention of virus infection.

Although spam was gaining greater awareness in the broader community, there was no perceived benefit in filtering email. Mail had already been accepted, the bandwidth costs incurred, so a bounce would only add to those costs and to the system load. As a home based business, some of the issues that face larger companies were not of any great concern to me. In a larger company, issues such as liability for failing to protect employees from offensive material may require more labour intensive solutions than that I have proposed. Similarly, I have less concern with any privacy issues than a larger company may. It is worth noting that suitable "Acceptable Usage Policies" and user education can usually address any privacy issues.

Like many home-based offices, family members were given email accounts on this server. These users included children, aged 8 and 11. Fortunately either their mother or I was usually around to supervise them reading their mail, which gave us a chance to perform at least some censorship of their email. Then my 2 year old received email – the first in around 18 months. It was a financial scam. The requirement to implement spam filtering increased in importance.

Implementation Details

Hardware and Software

The Internet facing mailserver was an Intel 486 running at 66MHz. It had 32MB RAM, 1GB total disk space, and was running Redhat 7.0.
Obtuse SMTPD was the Mail Transfer Agent (MTA) of choice. SMTPD was chosen because of its simplicity, separation of duties and inbuilt security features. Unfortunately, recent CERT advisories[1] reported buffer overflows in sendmail that utilities such as SMTPD would not prevent, so a sendmail upgrade was required. This was performed using RPM, the RedHat Package Manager. Application of this upgrade stopped SMTPD from functioning properly. Rather than re-install SMTPD, an alternative MTA was desired.
There are a number of non-sendmail MTAs available, with a variety of features and limitations. The most popular include Postfix, Exim and Qmail. I had previously used Postfix, and had enjoyed success with it. It had matured as a product, with a "snapshot", or development release, and a more stable official release. The stable releases were being packaged for a number of operating systems, including RedHat. With all of these factors in mind, Postfix was selected.
I initially installed it using RPM, but had some problems with configuration. As my prior experience had been on a non-Linux system, I had previously only built from source. I uninstalled the RPM, retrieved the source, and followed the instructions to build it.

Anti-Spam Decision Rationale

This mail filtering system was designed with two clear goals:

1. Reduce spam as much as possible.
2. Allow legitimate mail through.

The major constraint was financial. In any small business, financial limitations are often quite sharply defined, and cannot usually be exceeded. This particular constraint encouraged a free solution. Fortunately, spam elimination is an item of concern for a large number of open source developers, and accordingly a wide selection of suitable tools. Even more fortunately, numbered amongst these tools are some of the most effective anti-spam tools.
Research on http://www.google.com/ and the System Administrators Guild of Australia (SAGE-AU) mailing list suggested that SpamAssassin was the industry leader in anti-spam tools. This survey included both commercial and free tools. Its ability to be used as part of a procmail filter, per-user settings and ability to use a number of different methods to evaluate a message helped it receive this recommendation. After the decision was made to implement SpamAssassin, Network Associates International acquired Deersoft[2], the company of the authors of SpamAssassin. They have since commercialised the product and market it as McAfee SpamKiller.

---

[1]CERT Advisories 2003-25, 2003-12, 2003-07.
[2] NAI Press Release.

First Steps

First was an upgraded CPAN, which in turn updated Perl.
The Redhat Package Manager, RPM, was not used for this upgrade work, primarily for the freedom to mix and match versions. This also allowed me to bypass the preferences and peculiarities of the various package maintainers. This non-RPM approach has the advantage that it is portable to other versions of UNIX, or even other distributions of Linux. OpenPkg has some potential in the portability area, but only provides a part solution. The decision not to use RPM creates some issues in other areas, where libraries for particular utilities are now no longer available. This led to maintenance issues in the longer term.
Once CPAN is installed, almost any perl module can be installed, using a friendly, automated interface. This includes the ability to process and optionally automatically resolve dependencies. This feature is lacking in RPM, and is one of the main reasons I chose to use CPAN instead.

The Postfix documentation reports a factor of 4 slowdown in mail processing for a simple content filter. This increases by another factor for each temporary file that is created by the filter program[3]. For a small site, a simple filter such as the SpamAssassin spamc/spamd program suite is generally acceptable, but at a larger site a large backlog can easily occur. Several programs exist to improve this performance, but AMaViSd-new was chosen after research into the postfix users mailing list and newsgroup. According to Mark Martinec, "amavisd-new is a high-performance and reliable interface between mailer (MTA) and one or more content checkers: virus scanners, and/or Mail::SpamAssasin Perl module."[4]

Many of the pre-requisite packages required are available via CPAN, and the dependency tracking that CPAN provides means that this is the easiest way to install them. Not all of the optional packages are available via CPAN, especially Razor2, which is only available from Sourceforge. It does come with a full set of pre-requisite packages as a separate bundle, thus simplifying the installation process. The pre-requisite packages for AMaViSd-new are a little more complicated to manage, but were achieved using CPAN dependencies. All are specified in the INSTALL file that comes with the source.
AMaViSd-new requires a number of changes to the Postfix configuration, and some modifications to its own defaults. This involves configuring a second Postfix SMTP listener on a different port, and adding a content filter to the usual Postfix parameters. This is covered in README.postfix, in the source distribution for AMaViSd-new . For those who wish to use a different MTA, several README files are available, covering a number of different mailers. Some generic files are also provided, allowing easy integration with a less well known MTA.

[3] Venema. FILTER_README
[4] Martinec. "amavisd-new – Introduction".

Components Used

RedHat 7.0 and 9 – RedHat has been in use on this gateway for many years. Version 7.0 is now out of support, and an upgrade to 9 on new hardware was begun. RedHat 9 won't support some hardware in the old server.
Virus Scanner – This is a command line scanner only, which introduces some latency into mail processing, with a new process spawned for every email with attachments. It is one of the many virus scanners that AMaViSd-new supports by default. A daemonised scanner would provide a performance increase, but at a financial cost. This is not an issue with the current load on this server, and there are a number of larger sites that only use a command line scanner.
SpamAssassin – SpamAssassin includes a number of different tests, including local, lexical tests and remote, collaborative checks such as Realtime Blackhole Lists, Razor2 and DCC. If remote tests are enabled they are initiated first, then lexical analysis of the email occurs. Once the results of the remote checks are evaluated, the spam rating is calculated and the fate of the message determined. Remote tests fall into two broad categories – free and commercial. Only support for the free services is currently configured.
AMaViSd-new – This daemonised Perl program comes with SpamAssassin hooks built in. It breaks the mail down to its component parts into a temporary directory. Then, the directory is virus scanned. Support for a large number of anti-virus products is included in the distributed source, including both daemonised and command line scanners. Others can be easily incorporated, with options to deploy multiple virus scanners simultaneously, or in sets of primary and secondary. The secondary scanners are only invoked if the primaries are unable to decipher an attachment, meaning a fast, less functional virus scanner can be used for the bulk of scanning, but a slower, more functional scanner used for the hard cases. It is important to note that the secondary will not be called if the primary reports a clean file, only a failure to decide. It is similarly important to note that all primaries are called, so using a tool like OpenAV (fast but not as current with virus pattern files) as a primary may buy a performance gain at the cost of allowing viruses through the gateway. Assuming no virus is found, SpamAssassin is invoked. As both are written in Perl, this is a simple subroutine call, and no extra overhead is invoked, thus improving overall scanning performance.
Postfix – Also known as the IBM Secure Mailer. A very popular mail package.

According to Wietse Venema, "Postfix attempts to be fast, easy to administer, and secure"[5]. The author distributes it without open relaying configured, as do most package maintainers. The postfix mailing lists contain a number of caveats around the default RPMs, and according to Chuck Moon, author of the official RedHat Postfix FAQ, RedHat dropped support for Postfix at one stage[6].

[5] Venema. "The Postfix Home Page"
[6] Moon. Archived mailing list post "Re: Postfix for the first time..."

Interim Post Implementation Situation

Mail is accepted by Postfix. This mail is then checked against a small set of known rejection conditions, such as banned MIME types, blacklisted senders and content strings. If one or more of these tests results in a message rejection, the SMTP session is terminated and the mail is not accepted. This rejection at session time removes the need to notify senders that their mail has been rejected, in accordance with RFC 2821.[7] It also saves resources on the server. If the message passes these initial tests, the message is stored on non-volatile storage and the SMTP session with the remote site is terminated.

Next, an SMTP connection is made via the loopback interface, to port 10024. This is where AMaViSd-new has been configured to listen for mail to scan. AMaViSd-new unpacks the message into a temporary directory, mounted on swap. AMaViSd-new does not confirm successful receipt of the message to the originating Postfix instance at this point in time. The message is checked for banned file types, and then the temporary directory is virus scanned. If a virus is detected, processing is stopped, and the message placed into quarantine. Depending on the virus type, a warning message can be created and sent to the sender. This is not done for viruses that are known to forge the sender, such as Swen or Sobig.

If no virus is found, SpamAssassin is invoked to evaluate the message. This involves header analysis, comparisons to known spam and lexical analysis. Building on the ideas postulated by Paul Graham in "A Plan for Spam[8]," a Bayes database is also used. If the message is determined to be spam, it is quarantined and a notification email sent to the spam administrator. If not, new headers are added and the message is forwarded to port 10025 on the loopback interface, where Postfix is listening.

Once this second Postfix instance confirms successful receipt of the message, AMaViSd-new reports success to the originating Postfix instance. The delay in notification means that configuring AMaViSd-new to use swap to unpack the message is safe. Mail will not be lost by AMaViSd-new. There is a possibility that a message may be sent twice, but this is considered preferable, and the performance gains are important on a busy server.

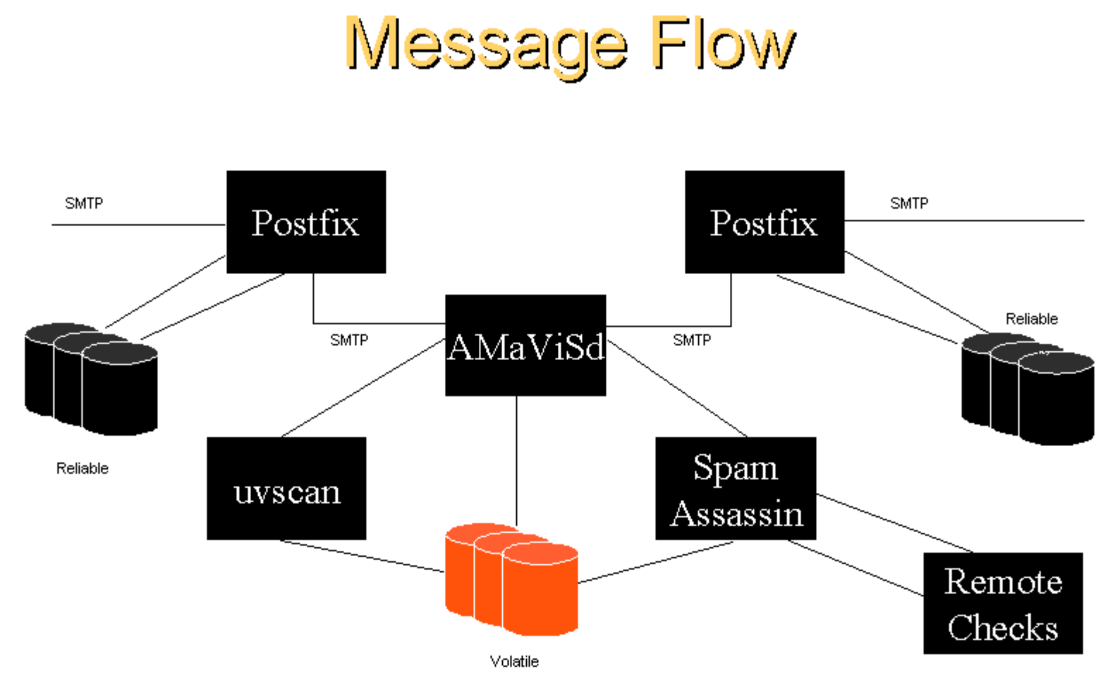
Having Postfix as the first relay is important, as AMaViSd-new has not been designed as a reliable MTA. It provides no anti-relay protection, except for allowing only certain hosts to connect via a network ACL. This obviously is not suitable for general Internet access. Having the second Postfix instance is important to allow low-cost retries. Although AMaViSd-new could be configured to send directly to the remote SMTP server, any interruption to this service would result in email being scanned for viruses and spam rating on multiple occasions.

Lastly, this configuration minimises interruption to service in the event of a new virus being reported but prior to the availability of update anti-virus definitions. AMaViSd-new can be stopped, and the mail will spool on the local machine, subject only to disk capacity. Once the new definitions are loaded,

[7] Klensin. RFC2821

[8] Graham. "A Plan for Spam".

mail will be processed relatively quickly. If appropriate, the queue can be flushed to speed this process further.



This was implemented as the first anti-spam configuration. The administrator can control almost all items mentioned above. These include the use of quarantine, per-user settings, the levels at which spam is discarded, whether to forward virus infected email to certain users. The configuration is contained in a Perl file, with sensible defaults distributed. The configuration is also heavily commented, with many examples of alternate uses supplied. Apart from changing some obvious defaults, such as local domain names and spam administrator, the software works almost "out of the box."

Modifications to Standard Packages

The most important change to the standard AMaViSd-new was multiple disposition options for a message based on "spamminess": drop the most obvious spam, quarantine intermediate, tag and forward indeterminate mail via subject rewrite, leave legitimate mail alone. This was partially implemented by the author of AMaViSd-new, Mark Martinec, and a two level solution is now available. The changes made to allow this are easily identifiable and thus extending the code to allow more possible results is a relatively simple undertaking.

Interim Results

The implementation was successful at this point in time, but the administration and maintenance of the system had become more difficult, to the point of being almost unmanageable. Some system utilities were no longer functional, and the upgrade to the new hardware was deferred.
Initial testing showed that a "tag and forward" level of 4.0 and a "kill" level of 7.0 was effective in blocking a large percentage of spam, with a very low

percentage of false positives. The false positives that were detected scored much higher than these levels, and consisted almost entirely of newsletters. The senders of these newsletters were added to the "whitelist" of known good senders, and any future mail from these senders was allowed through. No bounce messages were generated. The requirements of RFC 2821 section 6.1 were met by notifying an administrator that the mail had been sent. Any blocked messages were stored in a quarantine mailbox, which was checked for correctness. Any false positives were removed, and the remaining mail fed to the Bayesian database. Similarly, any spam that was missed by the system was re-learned as spam. These steps improved the accuracy of this database, with very little mail in the intermediate areas – instead, the Bayes database reported most messages with a >70% or <20% confidence rating.

Later Steps

The increased administrative overheads introduced by a move away from RPM led to a Linux distribution change on the spam-filtering server. Debian "Woody" was chosen as the best distribution to use for this purpose. This was based on both the long-term stability of each major release and on the package methodology. This includes the requirement to package everything, the ability to import packages from other distributions, the requirement to preserve local configuration changes during upgrades and on the ability to mix packages and package versions between multiple distribution streams, including stable, testing, and unstable. This allows core stability in areas that require it, whilst allowing bleeding edge code to be used. This is particularly useful in anti-spam software, where software more than a few months old loses its effectiveness. One piece of spam I recently received was scored at 9.9 on SpamAssassin 2.55, but as 13.9 on SpamAssassin 2.60. These are sequential releases of SpamAssassin.

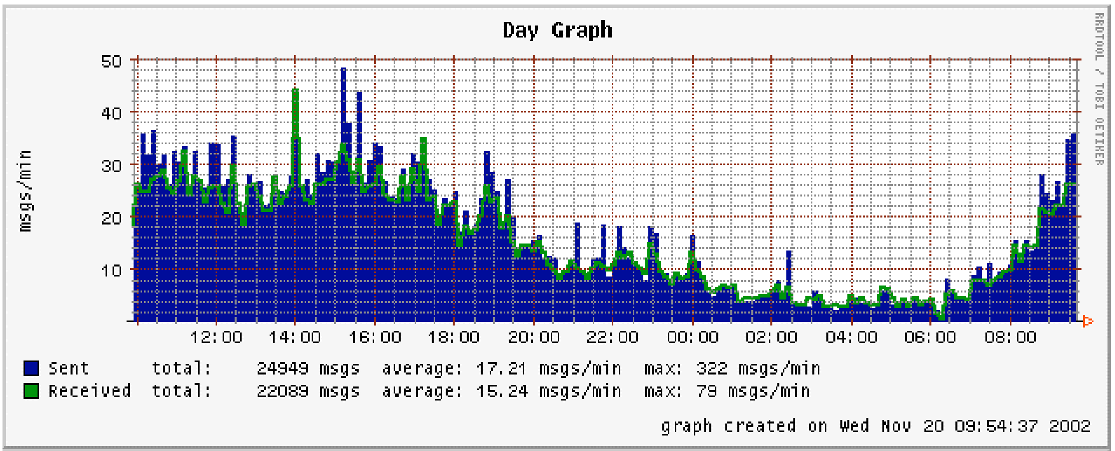
Mailgraph has been implemented to allow some realtime monitoring of the system in question. This uses syslog over the network to consolidate the log entries onto the internal host. Prior to this, the graph was relatively flat, with only rejections and spam reported.

In order to increase the accuracy of spam detection, support for Pyzor was added. Pyzor is a python re-implementation of Razor, but uses different databases. The server source code is available, allowing a totally local engine to be used. Alternately, UDP port 24441 is used to communicate with a remote, collaborative server. At a small site, use of the remote server gives a much better result than a local only cache. At a larger site, a local server may be able to provide an acceptable level of protection, although this would depend on the userbase.

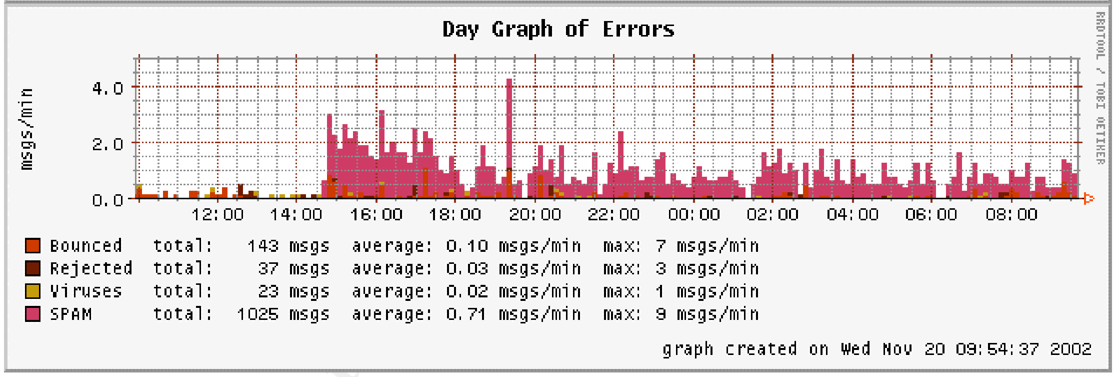
DCC, the Distributed Checksum Clearinghouse, is another collaborative spam service. This has not yet been configured, but will be in the near term. The delay is primarily due to difficulty locating a Debian supplied package. Experience in mixing both packages and non-packaged software with tight interdependencies has encouraged caution in this area.

Example Large Site

This configuration has been implemented for Charite.de, a large German hospital. Their email administrator, Ralf Hildebrandt, has implemented a configuration similar to that I describe above. The following text is copied from his Postfix website[9]. The graphs are produced with mailgraph and RRDtool.



```
🟦 Sent      total:    24949 msgs  average: 17.21 msgs/min   max: 322 msgs/min
🟩 Received  total:    22089 msgs  average: 15.24 msgs/min   max: 79 msgs/min
                                        graph created on Wed Nov 20 09:54:37 2002
```



```
🟧 Bounced   total:   143 msgs  average: 0.10 msgs/min  max: 7 msgs/min
🟫 Rejected  total:    37 msgs  average: 0.03 msgs/min  max: 3 msgs/min
🟨 Viruses   total:    23 msgs  average: 0.02 msgs/min  max: 1 msgs/min
🟥 SPAM      total:  1025 msgs  average: 0.71 msgs/min  max: 9 msgs/min
                                        graph created on Wed Nov 20 09:54:37 2002
```

amavisd-new is configured to use McAfee's uvscan, a non-daemonized scanner and SpamAssassin 2.50
We merely "tag" the spam instead of rejecting or quarantining it. This way each user can filter his/her mail individually.
Architecture

Linux, 2.4.20-ac1
AMD Athlon 1.2GHz
1 GB RAM
Spool, logs, virusscanning and system distributed over multiple disks for lower latencies
Postfix as MTA
amavisd-new
McAfee uvscan
avcheck with kavdaemon

Performance
On a normal day, this machine easily handles 25.000 Mails with a load of 0.80.

[9] Hildebrandt. "Enterprise-Wide Spam and Virus Protection".

If we were to drop avcheck in favor of an integration of kavdaemon into amavisd-new, we could increase performance even more.

Conclusion

This implementation of content filtering has been successful. The solution detailed above is able to handle the email load it receives, tuned to meet my requirements, and delivers excellent value for money. It also has the advantage that it can be deployed at many sites using a similar model. Most of the components can be replaced with another similar product, allowing sufficient flexibility to meet contractual or philosophical requirements. The configuration of both SpamAssassin and AMaViSd-new is easily modified, and can be tuned to meet the requirements of most sites.

From both a corporate and parental perspective, the biggest item lacking is the ability to filter attachments that are pornographic. There is little (if anything) in the Open Source area that provides this functionality, although this may change in the future. Commercial software provides around 80% detection, with an acceptable false positive rate. The only solution applicable for the system described within this paper is to mark all image attachment types as banned, and then to quarantine them. The end user can be notified that a banned file has been quarantined, and can then ask for it to be released. This only works at very small sites, such as a home network. For a larger user base, some kind of quarantine management software is required. Another alternative is to be more selective and careful with mail clients. A client such as SquirrelMail will not show images by default, providing some protection. Similar options are available in other mail clients.

Things never stand still. Most anti-spam practitioners use terms like “arms race” and “war on spam” to describe their activities. Unfortunately this is true. The major disadvantage of using SpamAssassin is that it is popular. This makes bypassing it one of the prime aims of spammers. Bayesian databases, collaborative databases, and elimination and reporting of open relays are the tools that we can all use to fight back. The more people contributing quality information to these services, the better it becomes for all of us.

References

Computer Emergency Response Team, Various CERT Advisories
September 29, 2003 http://www.cert.org/advisories/CA-2003-25.html
May 29, 2003 http://www.cert.org/advisories/CA-2003-12.html
June 9, 2003 http://www.cert.org/advisories/CA-2003-07.html

Network Associates International, January 9, 2003
http://www.networkassociates.com/us/about/press/corporate/2003/20030106.htm

Venema, Wietse. FILTER_README,
http://www.advosys.ca/papers/FILTER_README

Martinec, Mark.October 7, 2003, “amavisd-new – Introduction”.
http://www.ijs.si/software/amavisd/

Venema, Wietse. “The Postfix Home Page”, http://www.postfix.org/

Moon, Chuck. September 5, 2001. Archived mailing list post “Re: Postfix for the first time...”
http://archives.neohapsis.com/archives/postfix/2001-09/0067.html

Klensin, J. April 2001. “Simple Mail Transfer Protocol”
http://www.ietf.org/rfc/rfc2821.txt
section 6.1, “Reliable Delivery and Replies by Email”

Graham, Paul. August 2002, “A Plan for Spam”
http://www.paulgraham.com/spam.html

Hildebrandt, Ralf. March 17, 2003. “Enterprise-Wide Spam and Virus Protection”.
http://www.stahl.bau.tu-bs.de/~hildeb/postfix/enterprise_spam_tagging.shtml

Sources of Further Information

Product Name	**URL for more information**
Postfix	http://www.postfix.org/
AMaViSd-new	http://www.ijs.si/software/amavisd/
SpamAssassin	http://www.spamassassin.org/
Perl	http://www.perl.org/
CPAN	http://cpan.perl.org/
SPAM	http://www.spam.com/ci/ci_in.html
OpenAV	http://www.openantivirus.org/
Obtuse SMTPD	http://www.obtuse.com/smtpd.html
Razor2	http://razor.sourceforge.net/
Pyzor	http://pyzor.sourceforge.net/
DCC	http://www.rhyolite.com/anti-spam/dcc/
Bayesian Filtering	http://www.bayesian.org/ http://www.paulgraham.com/spam.html
RRDTool	http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/
MailGraph	http://people.ee.ethz.ch/~dws/software/mailgraph/
Larger Site (Charite.de)	http://www.stahl.bau.tu-bs.de/~hildeb/postfix/enterprise_spam_tagging.shtml
Debian GNU/Linux	http://www.debian.org/