Changes from

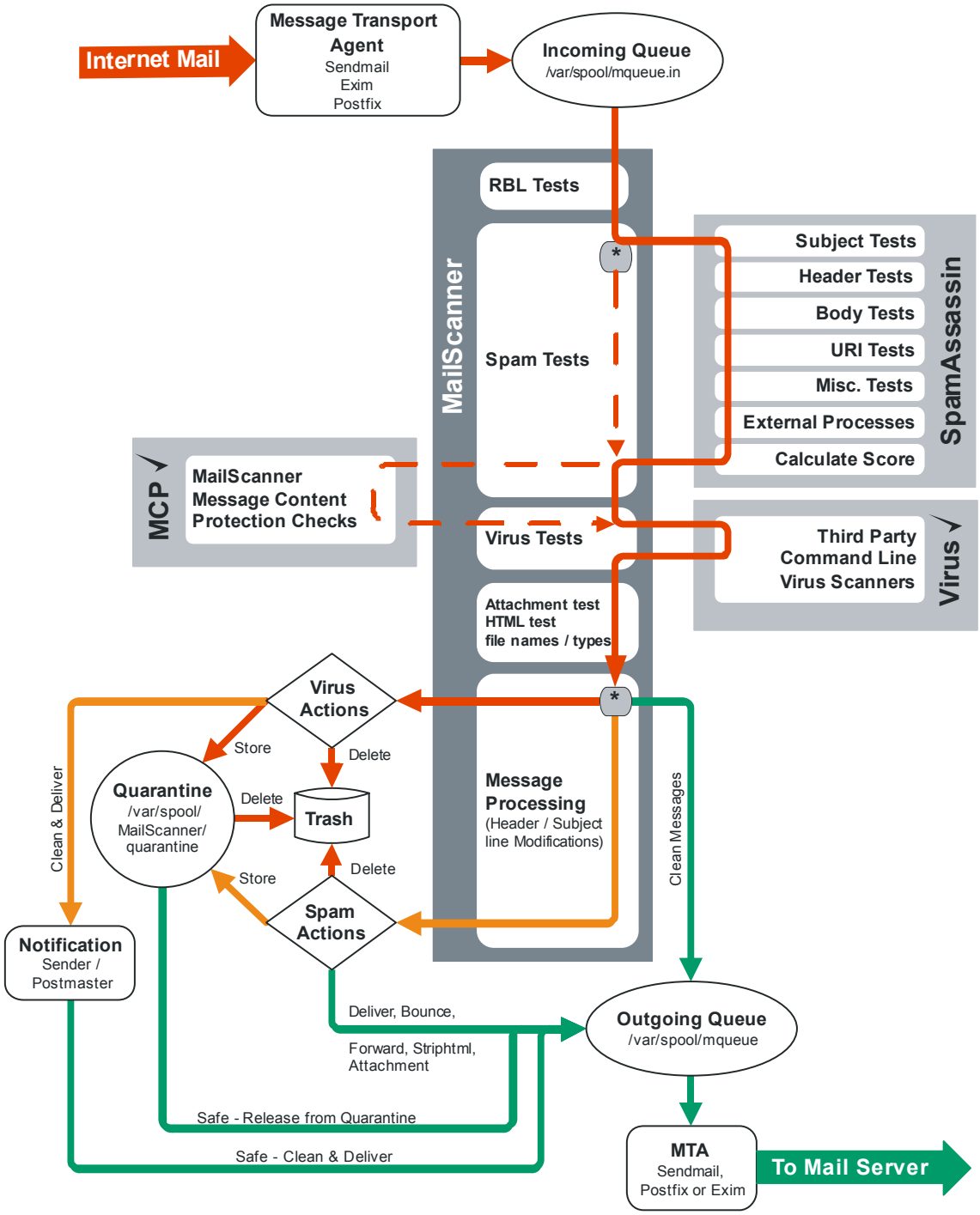
Version 1.1

December 2004

To

Version 1.2

August 2005


Internet Mail
Message Transport Agent
Sendmail
Exim
Postfix
Incoming Queue
/var/spool/mqueue.in
MailScanner
RBL Tests
Spam Tests
SpamAssassin
Subject Tests
Header Tests
Body Tests
URI Tests
Misc. Tests
External Processes
Calculate Score
MCP
MailScanner Message Content Protection Checks
Virus Tests
Virus
Third Party Command Line Virus Scanners
Attachment test
HTML test
file names / types
Virus Actions
Store
Delete
Quarantine
/var/spool/ MailScanner/ quarantine
Delete
Trash
Store
Delete
Spam Actions
Message Processing
(Header / Subject line Modifications)
Clean & Deliver
Clean Messages
Notification
Sender / Postmaster
Deliver, Bounce,
Forward, Striphtml, Attachment
Outgoing Queue
/var/spool/mqueue
Safe - Release from Quarantine
Safe - Clean & Deliver
MTA
Sendmail, Postfix or Exim
To Mail Server

Scan Messages = yes

This setting is used to completely enable or disable all the processing done by MailScanner, except for the "Archive Mail" setting described in the next section. It is intended for use with a ruleset, so that your customers who either do not want their mail processed in any way, or who have not paid you for the service, can have their mail delivered without any processing done to it. No MailScanner headers or modifications of any sort will be applied to the messages, and it is impossible to see from the contents of the message that they have passed through MailScanner. The "Archive Mail" option is still used so that you can take copies of their mail, possibly without their knowledge if your operation or organisation requires it.
This can be the filename of a ruleset, and this is how this setting would normally be used unless it is just set to "yes".

Archive Mail =

Space-separated list of any combination of

1. email addresses to which mail should be forwarded,

2. directory names where you want mail to be stored,

3. file names (they must already exist!) to which mail will be appended in "mbox" format suitable for most Unix mail systems.
   If you give this option a ruleset, you can control exactly whose mail is archived or forwarded. If you do this, beware of the legal implications as this could be deemed to be illegal interception unless the police have asked you to do this.

Any directory or file name used here may contain the magic string "_DATE". This will be replaced by a number representing the date in the form YYYYMMDD, i.e. 4 digits representing the year, 2 digits representing the number of the month and 2 digits representing the date within that month. This format is used as it will be sorted in date order, should the entries in a list be sorted in alphabetic or numeric order. This can be used to separate the archives by date, so that previous days' archives can be moved or deleted without altering the contents of the present day's archive.

Spam Lists To Be Spam = 1

In order to be treated as spam, the sender of the message must appear in at least this number of "Spam Lists" to be treated as spam. Previously this setting was fixed at 1, so that a message appearing in a single "Spam List" would be treated as spam.

Every word in every processed message is added to the Bayes database, so it can grow quickly. To keep it to a reasonable size, words that have not been seen for some time are expired and deleted from the database. The database is then rebuilt and compacted to optimise future searches. This rebuilding can take a few minutes and is best done with MailScanner's co-operation.

By default, SpamAssassin will automatically rebuild and compact it periodically. If you have a large Bayes database it can take several minutes for this to happen. It can do it at any time when MailScanner asks SpamAssassin to check a message. If the rebuild takes too long, MailScanner will think SpamAssassin has broken (it will trigger a timeout check) and will kill the SpamAssassin check. The result is that a temporary copy of the Bayes database will be left behind in the SpamAssassin working directory. By default this is in a .spamassassin directory under the home directory of the user MailScanner is being run as. If you are using sendmail on a Linux system, this will be in /root/.spamassassin. Every time SpamAssassin attempts to rebuild and compact the database, another temporary copy of it will be added to the .spamassassin directory. This can quickly take up a lot of space.

The best solution is to stop SpamAssassin doing the job itself and tell MailScanner to do it instead. You need to set "bayes_auto_expire 0" in spam.assassin.prefs.conf to stop SpamAssassin doing it. The MailScanner options described below will set up MailScanner to do it. This should stop you getting lots of temporary files left lying around.

It is reasonable to rebuild it every few days on mid-sized sites. During the rebuild you have the choice of whether to suspend all mail processing until it has completed, or to keep processing mail, ignoring the Bayes database until it has completed.

Rebuild Bayes Every = 0

If you are using the Bayesian statistics engine on a busy server, you may well need to force a Bayesian database rebuild and expiry at regular intervals. This is measures in seconds. 1 day = 86400 seconds. To disable this feature set this to 0.

`Wait During Bayes Rebuild = no`

The Bayesian database rebuild and expiry may take a few minutes to complete. During this time you can either wait, or simply disable SpamAssassin checks until it has completed.

`Incoming Work Dir = /var/spool/MailScanner/incoming`

Set where to unpack incoming messages before scanning them This can completely safely use tmpfs or a ramdisk, which will give you a significant performance improvement.

NOTE: The path given here must not include any links at all, but must be the absolute path to the directory. If it is not the absolute path, MailScanner will continue to work correctly, but it will log a warning message about this.

Sometimes multiple Subject: headers are put in the message to confuse email scanning systems so that the Subject: seen by the user does not contain anything indicating that the message might be spam. MailScanner keeps the 1st Subject: header and discards all others.

In order to apply filename and filetype checks on the contents of Zip and Rar archives, they are both unpacked. The Rar archives are unpacked using an external "unrar" program, while Zip archives are handled internally. Any archive found nested deeper than the “Maximum Archive Depth” will make MailScanner reject the message.

In order to stop MailScanner checking the filenames and filetypes of the contents of Zip and Rar archives, the "Maximum Archive Depth" option must be set to 0. Setting this to 0 stops MailScanner unpacking Zip and Rar archives itself. Virus scanners do their own archive unpacking, and so setting this to 0 does **not** stop MailScanner finding viruses in archives.

There are 2 common settings for this option:

- 0 stops MailScanner unpacking Zip and Rar archives. Viruses inside Zip and Rar archives will still be found. No filename checking or filetype checking will be done on the contents of Zip and Rar archives.

- 3 makes MailScanner check the filenames and filetypes of files within archives within archives within the message, which is as deeply nested as any average user will create. Any archive nested deeper than this will cause MailScanner to reject the attachment as a potential denial-of-service attack and it will be replaced by a warning message.

`Unrar Command = /usr/bin/unrar`

If this is set, and the unrar program exists, it will be automatically passed to the ClamAV scanner. No wrapper script adjustments are necessary for unrar to work.

`Unrar Timeout = 50`

The unrar command above will be used when unpacking archives to expand all RAR archives into their original contents. This specifies the maximum amount of time that can be used for the process, after which the unrar command will be terminated.

`Allowed Sophos Error Messages = "corrupt", "format not supported"`

Sophos IDE Dir = /usr/local/Sophos/ide

The directory (or a link to it) containing all the Sophos *.ide files. This is used by the "sophossavi" and "sophos" virus scanners, and is irrelevant for all other scanners.

ClamAVModule Maximum Recursion Level = 5

ClamAVModule only: the maximum depth to which archives will be unpacked. If you set this too high you are inviting denial-of-service attacks, but you should set it deep enough that normal users will not pack archives deeper than this setting.

ClamAVModule Maximum Files = 1000

ClamAVModule only: the max number of files that can be scanned at once.

ClamAVModule Maximum File Size = 10000000

ClamAVModule only: the maximum size of any file that can be scanned.

ClamAVModule Maximum Compression Ratio = 250

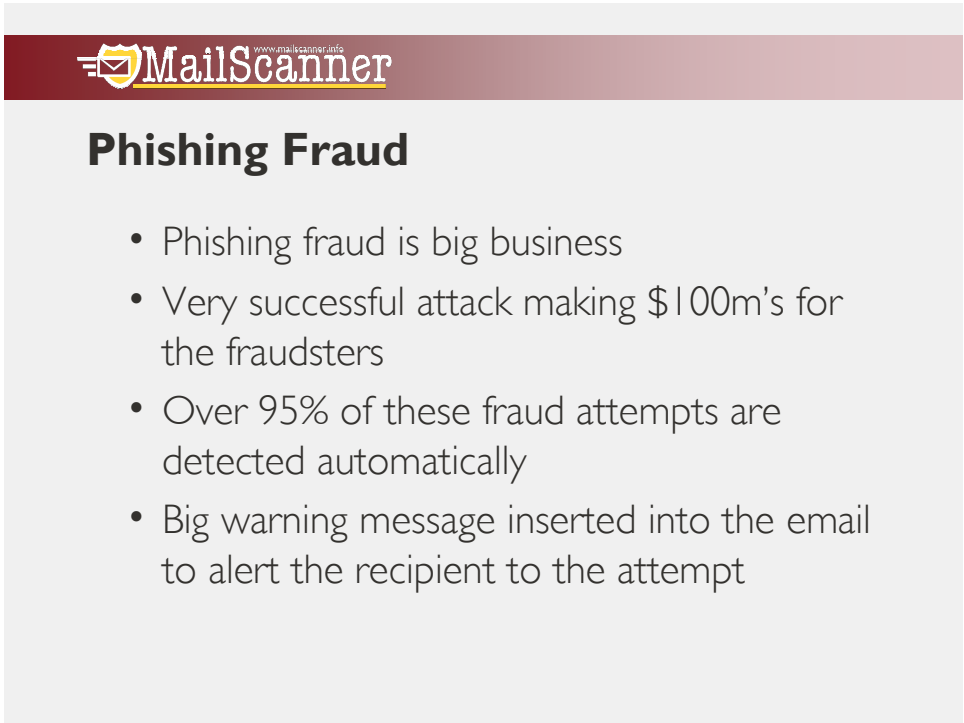
ClamAVModule only: the maximum compression ratio of any archive scanned. Any archives which expand more than this amount will be flagged as errors and will be quarantined.

Unrar Command = /usr/bin/unrar

If this is set, and the unrar program exists, it will be automatically passed to the ClamAV scanner. No wrapper script adjustments are necessary for unrar to work.

Unrar Timeout = 50

The unrar command above will be used when unpacking archives to expand all RAR archives into their original contents. This specifies the maximum amount of time that can be used for the process, after which the unrar command will be terminated.



"Phishing" is a term coined by hackers. It is the use of email messages in which there is a link you are encouraged to click which appears to link to your bank or credit card company's web site; in reality it takes you to a fraudulent realistic copy of the site. The aim is to lure you into entering confidential information, including items such as

- Bank account numbers
- Internet banking passwords
- PIN numbers
- Account details
- Your mother's maiden name

This information is then used by the fraudsters to empty your bank account, duplicate your credit card or acquire large loans in your name.

It is estimated to have costed US banks and credit card companies well over $100m in 2004. By using viruses to infect PC's, the fraudsters then gain control over a large number of PC's and use these "zombie" machines to send out huge numbers of the phishing emails, making them impossible to trace. Fortunately the global internet community, in particular the banks, have so far been quite efficient at tracing the fake copies of their websites and quickly having them shut down.

Find Phishing Fraud = yes

MailScanner can detect most of the phishing fraud attempts, and will place a large warning in the email message, right next to the potentially fake link.

The text added into the message is set in 2 settings in the languages.conf file, one placed before the real address of the link, and the other placed before the displayed address of the link:

```
PossibleFraudStart = <font color="red"><b>MailScanner has
detected a possible fraud attempt from
PossibleFraudEnd = claiming to be</b></font>
```

The only action taken is to add the warning, as some companies persist in issuing email messages to their customers containing links that look like phishing fraud attacks. As a result this trap does occasionally trigger with a false alarm, but that cannot be totally avoided. Steps have been taken to adapt to all the cases of false alarms that have been reported, and the heuristic rules used to detect these are now very complex and reliable.

I am not going to attempt to describe here exactly how the heuristics work, they are far too complex to easily explain. But if you see some messages which are mistakenly triggering the "phishing net" then please send them to support@mailscanner.info for analysis.

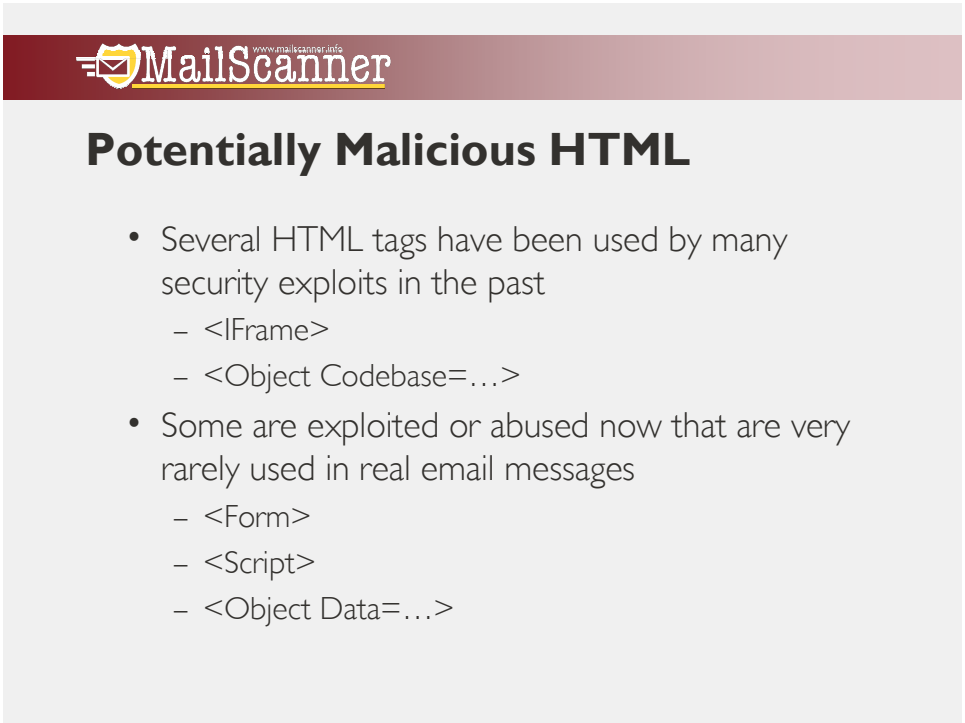
The heuristics used by the phishing net are given online at http://www.phishingnet.info.

Also Find Numeric Phishing = yes

Many phishing attacks rely on the use of numeric IP addresses, as the servers used to host the fake sites are often run on systems that have been automatically hacked into and had the fake web site copied onto them. In order not to give away the name, the fraudsters just use the numeric IP address of the server. It is rare for a real website to use a numeric IP address in a valid link, and so all uses of numeric IP addresses in links are flagged as possible phishing attacks.

Phishing Safe Sites File = %etc-dir%/phishing.safe.sites.conf

This file contains the list of web sites which are known to be safe and whose owners send out newsletters and other email messages containing links that look like phishing attacks. The hosts listed in this file are the real hostnames of the sites mentioned in the email message. A leading "*" wildcard character may be used in the entry to represent any hostname in the given domain. For example, "*.example.com" would match both "www.example.com" and "mail.example.com". The entries must each be put on a separate line. Comments start with a "#" character and continue to the end of the line. Blank lines are ignored.



A few HTML tags are recognised as they have been used in so many attacks over the years. They are for features that would never normally be used in email as they are more suited to complex web pages.

About the only one that is legitimately used is <IFrame> as the Dilbert daily cartoon uses it. As a result, the sources of <IFrame> tags can be logged so that you can construct a whitelist for the tags so you allow them from a few addresses but ban them from all others.

`Log Dangerous HTML Tags = no`

Banning some HTML tags completely is likely to break some common HTML mailing lists, such as Dilbert and other important things like that. So before you implement any restrictions on them, you can log the sender of any message containing a dangerous HTML tag, so that you can set the “dangerous HTML tags” options to be rulesets allowing some of the tags from named "From" addresses and banning all others. This can also be the filename of a ruleset.

Allow IFrame Tags = disarm

Do you want to allow <IFrame> tags in email messages? This is not a good idea as it allows various Microsoft Outlook security vulnerabilities to remain unprotected, but if you have a load of mailing lists sending them, then you will want to allow them to keep your users happy.

yes Allow these tags to be in the message

no Ban messages containing these tags

disarm Allow these tags, but stop these tags from working

This can also be the filename of a ruleset, so you can allow them from known mailing lists but ban them from everywhere else.

Allow Form Tags = disarm

Do you want to allow <Form> tags in email messages? This is a bad idea as these are used as scams to pursuade people to part with credit card information and other personal data.

yes Allow these tags to be in the message

no Ban messages containing these tags

disarm Allow these tags, but stop these tags from working

Note: Disarming can be defeated, it is not 100% safe! This can also be the filename of a ruleset.

Allow Script Tags = disarm

Do you want to allow <Script> tags in email messages? This is a bad idea as these are used to exploit vulnerabilities in email applications and web browsers.

yes Allow these tags to be in the message

no Ban messages containing these tags

disarm Allow these tags, but stop these tags from working

Note: Disarming can be defeated, it is not 100% safe! This can also be the filename of a ruleset.

Allow Object Codebase Tags = disarm

Do you want to allow <Object Codebase=...> or <Object Data=...> tags in email messages? This is a bad idea as it leaves you unprotected against various Microsoft-specific security vulnerabilities. But if your users demand it, you can do it.

Yes	Allow these tags to be in the message
No	Ban messages containing these tags
disarm	Allow these tags, but stop these tags from working

This can also be the filename of a ruleset, so you can allow them just for specific users or domains.

Disarmed Modify Subject = yes

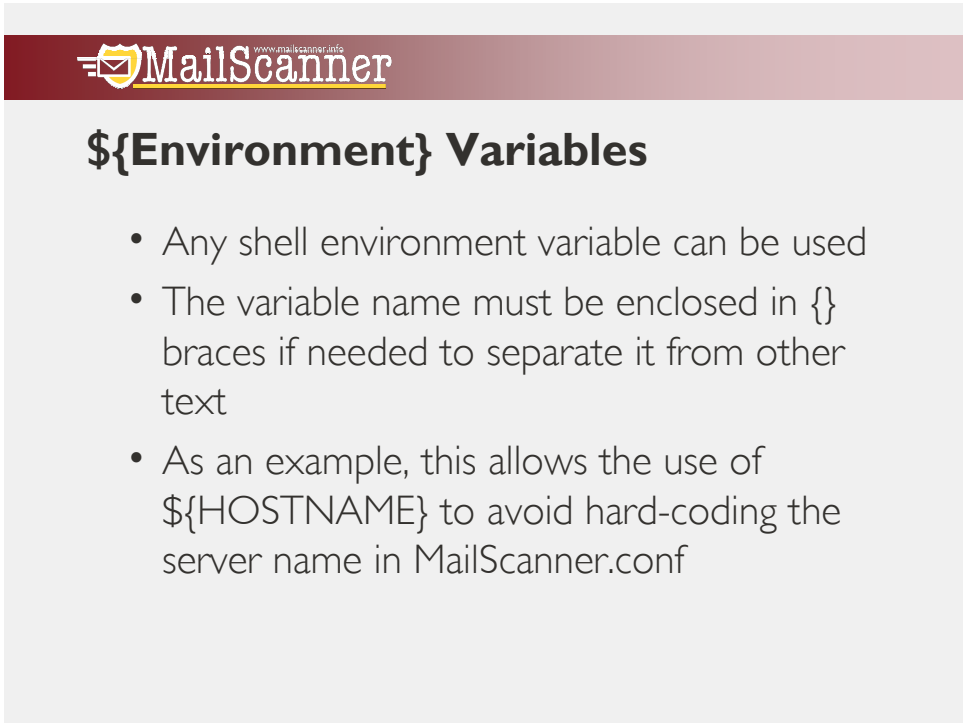
If the message has had any dangerous HTML tags disarmed, do you want to modify the subject line? Users find this very helpful to avoid them wasting time discovering that a form in their email message has been disarmed and hence will not work. Note that this will not be triggered by a phishing attack on its own, but only if there are disarmed dangerous HTML tags present. This can also be the filename of a ruleset.

Disarmed Modify Text = {Disarmed}

This is the text to add to the start of the subject if the "Disarmed Modify Subject" option is set. This can also be the filename of a ruleset.

To be able to personalise the signature for different users, or to be able to place a personalised link in the inline signature, any of the strings “$from”, “$to” and “$subject” may be used in the inline signature files. These will be replaced with the message sender’s address, the message recipients’ addresses and the message subject line respectively.

.



Normally, shell environment variable names are all upper-case. They can be of the form $HOSTNAME or ${HOSTNAME}. The latter format is used when it is not obvious how to separate the name of the environment variable from any text following it. It is usually safer to use the ${} form, and cannot do any harm.

Any configuration option can also contain the string "\n" which will be replaced by a newline character in the result.

The most common use for this is in a setting such as

`Hostname = The %org-long-name% (${HOSTNAME}) MailScanner`

This nicely demonstrates the use of both %variables% and shell environment ${variables} in order to produce a neat but informative server name in MailScanner reports.

1) 152.78
   Translated into 152.78.0.0-152.78.255.255 and matches email which was delivered to MailScanner from a matching IP address

2) 152.78.
   As above

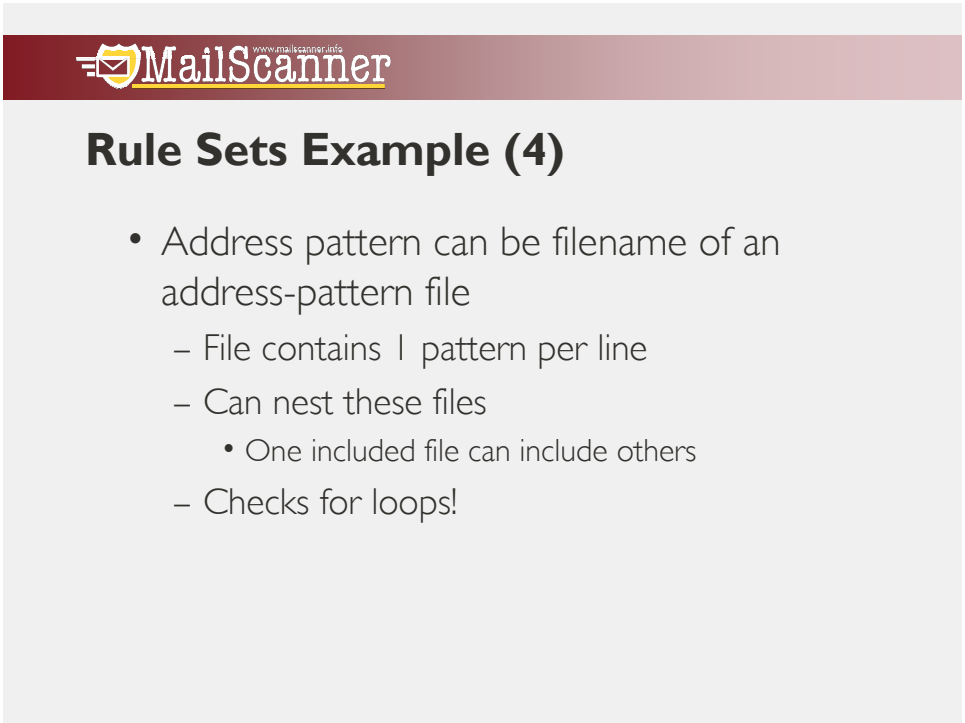
3) 152.78.0.0/16
   As above

4) 152.78.0.0/255.255.0.0
   As above

5) 152.78.0.0-152.78.255.255
   As above

6) 192.168.2.1
   The exact IP address given, not just any IP address starting with the given number. It will not match 192.168.21.11 or 192.168.21.123, for example

7) /numerical-regular-expression/
   Similar to the regular expression example earlier, except this is matched agains the originating IP address



If you have a large number of similar rules with the same right-hand sides, ie the same result, you can list all the address patterns in another file and refer to it by putting the filename in instead of the address pattern itself. So for the "Spam Checks" option, if you wanted to have a file which listed all the domains for which you provide a spam checking service, you could write a ruleset like this:

To:	/etc/MailScanner/spam.check.domains	yes
From Or To:	default	no

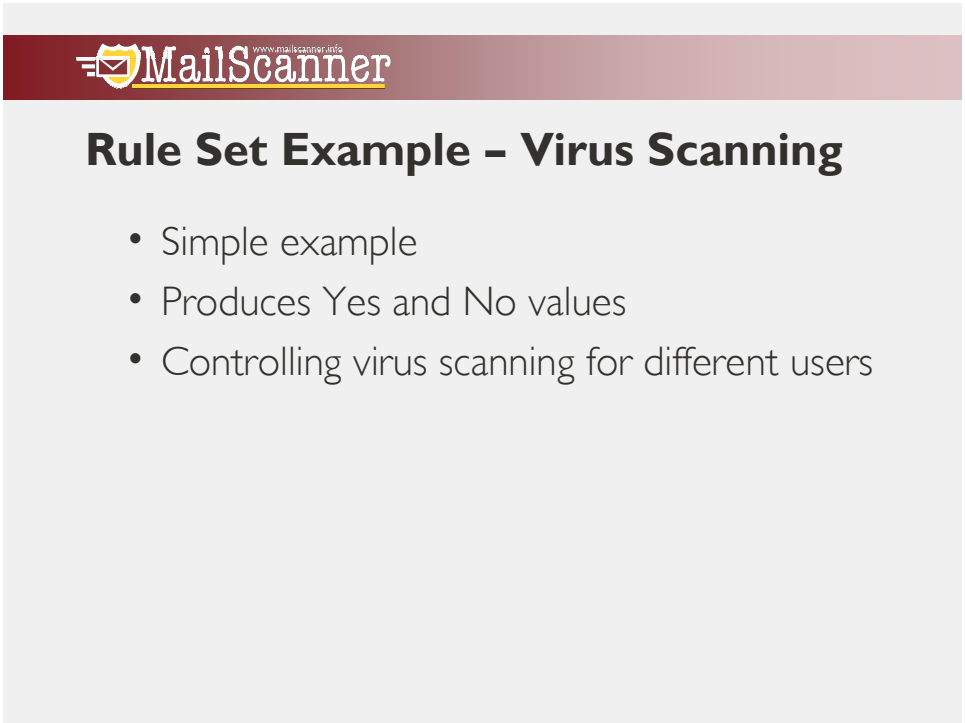
The /etc/MailScanner/spam.check.domains could just contain a list of all the domain names for which you want the "yes" result, one domain per line. This is just a short-cut for having to manually write a rule for each domain who pays you for spam checking. This short-cut may help quite a lot if you automatically generate the list of domain names.

For example, if using a ruleset for the "Spam Actions" configuration option, a ruleset might containg "store forward user@domain.com" on the right-hand side of a rule. If you were

using a ruleset for the “SpamAssassin Required Score” option, then the number “8” would be suitable for the right-hand side of the rule. If the ruleset was for the “Spam Checks” option, then the only valid right-hand sides would be “yes” or “no”.

The contents of the right-hand side is simply the setting that you want to apply to the configuration option, when the condition in the rule matches the message.

Rulesets cannot be nested. The only nesting allowed is when using files full of address patterns, as described a few slides ago.



In this example, the aim is to do virus scanning for all the domains of your customers, except for a few who have specifically requested that they do not want it for mail going to or coming from them. Also, you do not want to do virus scanning for mail sent to your support staff. You want to scan all other mail for viruses.

The domains who do not want it are nocheck1.com and nocheck2.com. The addresses of all the support staff at your customers' sites are "support" at each site.

In MailScanner.conf, set this:

```
Virus Scanning = %rules-dir%/virus.scanning.rules
```

It is worth mentioning a couple of points about this line. Firstly it uses the %rules-dir% variable which is set at the top of MailScanner.conf, so that if you move it all then changing all the directory names is a lot easier. Secondly, it is good practice to make the name of

ruleset files end in ".rules". This is usually not absolutely necessary, but will avoid a few problems in specific situations.

Assuming %rules-dir% is set to /etc/MailScanner/rules, then the following ruleset will be stored in /etc/MailScanner/rules/virus.scanning.rules. In it, put these lines:

```
FromOrTo:       nocheck1.com        no
FromOrTo:       nocheck2.com        no
To:             support@*           no
FromOrTo:       default             yes
```

The last line sets the default value used for all mail that does not have any of the addresses given in any other rule. Note that the default rule does not have to be at the end of the ruleset file, but it is often more clear if you do.

After changing this configuration, you will need to restart or reload MailScanner. On RedHat systems, this can be done most easily with the "service" command:

```
service MailScanner reload
```

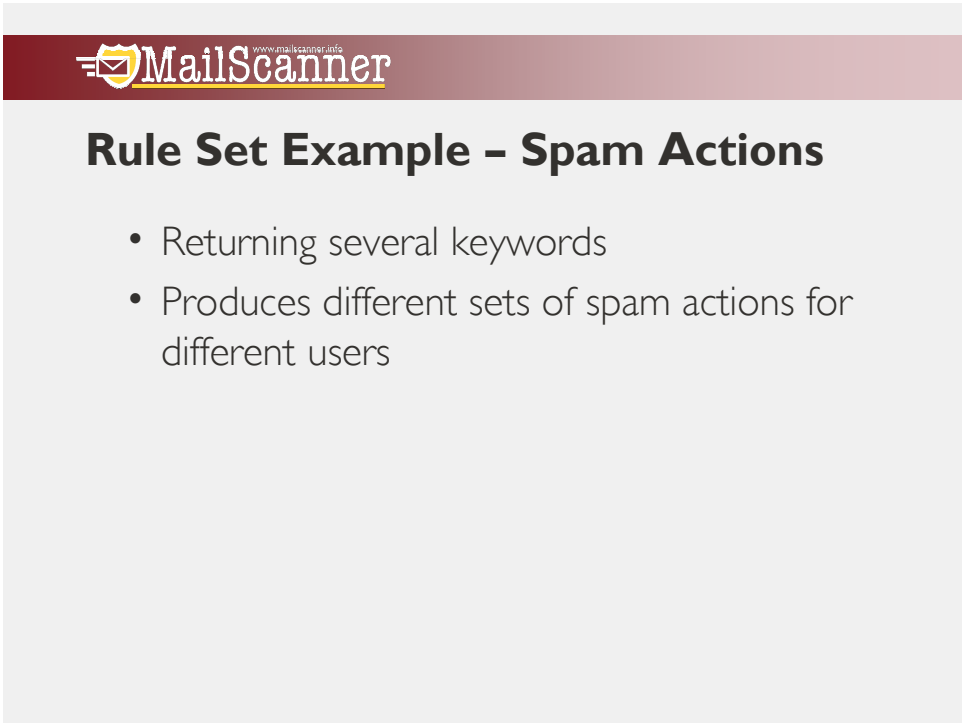
On other Linux systems you can do it like this:

```
Kill –HUP `cat /var/run/MailScanner.pid`
```

On non-Linux systems where MailScanner is installed under /opt you can do it like this:

```
Kill –HUP `cat /opt/MailScanner/var/MailScanner.pid`
```

If you do not do this, the configuration will automatically updated at the next regular MailScanner restart (governed by the "Restart Every" configuration setting).



This example is a bit more complex than the previous one, in that it returns a string of Spam Actions instead of a simple yes or no value. This example uses the "Spam Actions" setting. This does not affect the actions required for high-scoring spam at all, that is outside the scope of this example.

- Some of your customers want all their spam forwarded to a single account so the support staff there can review it and deliver any messages that were incorrectly tagged as spam. The customers wanting this are forwarding1.com and forwarding2.com.

- One of your customers want all their spam to be delivered but also stored in an archive for later review. The customer wanting this is archive3.com.

- Another customer wants to turn each spam message into an attachment and deliver the resulting message, so that they do not have to look at the real spam message, but still receive it all as they are new users of your MailScanner service and are not yet

convinced by the accuracy of MailScanner's spam detection. The customer wanting this is attach4.com.

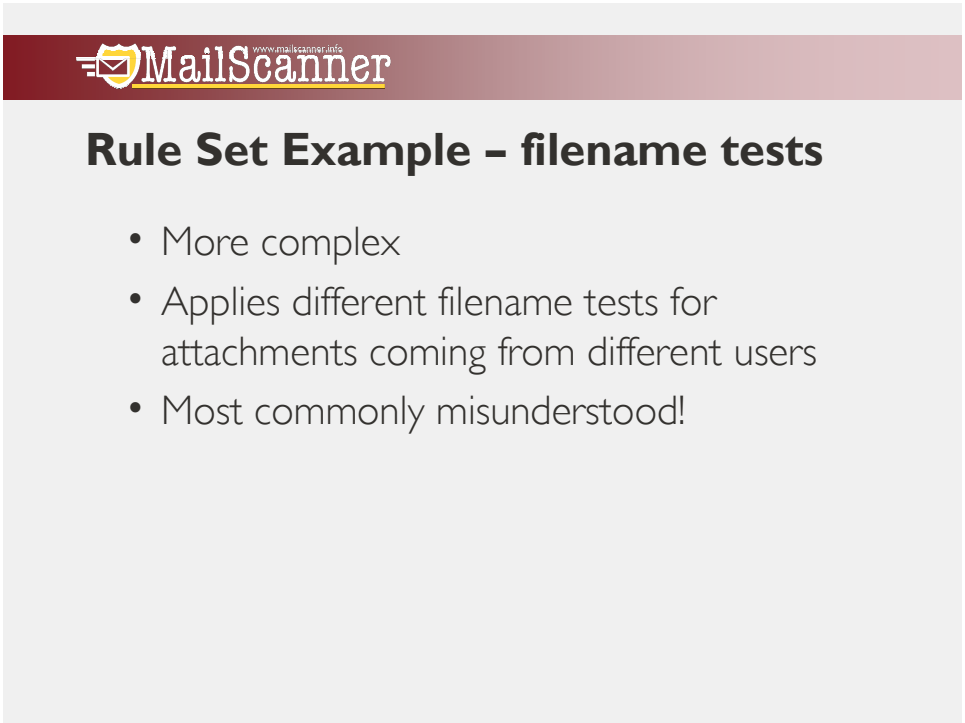
- You want to simply delete all other spam for other customers.

In MailScanner.conf you need to set

Spam Actions = %rules-dir%/spamactions.rules

In /etc/MailScanner/rules/spamactions.rules put these lines:

```
To:        forwarding1.com     forward reviewme@forwarding1.com
To:        forwarding2.com     forward reviewme@forwarding1.com
To:        archive3.com        store deliver
To:        attach4.com         deliver attachment
FromOrTo: default              delete
```



This example is rather more complex, but is one of the most common requests from the MailScanner user community.

Most of your customers are happy with the filename checking provided by the default allow/deny tests supplied with MailScanner.

However, 2 of your customers have slightly different requirements.

- One of them wants to ban its staff from sending out Zip archives ending in ".zip", but allow all Microsoft Word ".doc" documents, regardless of the rest of the filename. Their domain name is domain1.com.

- Another customer wants to allow its support staff to send out ".ini" files, which are used by the software they sell and they need to be able to send these files to their own clients to assist them. Their domain name is domain2.com.

What you need to do first is to use a ruleset which points the "Filename Rules" setting to different combinations of "filename.rules.conf" files. Point the setting at your ".rules" ruleset that will determine which files to use. Put this in MailScanner.conf:

```
Filename Rules = %rules-dir%/filenames.rules
```

In addition to the original filename.rules.conf file supplied with MailScanner, you need to create 2 new ".rules.conf" files containing the differences required by your customers. These new ".rules.conf" files **must** use **tab** characters to separate the 4 parts of each line, and not just spaces.

Create the 2 new ".rules.conf" files. Create a "filename.zip.doc.rules.conf" file containing this:

```
deny      \.zip$    No zip files!  Zip files are blocked
allow     \.doc$    -              -
```

And create a "filename.ini.rules.conf" file containing this:

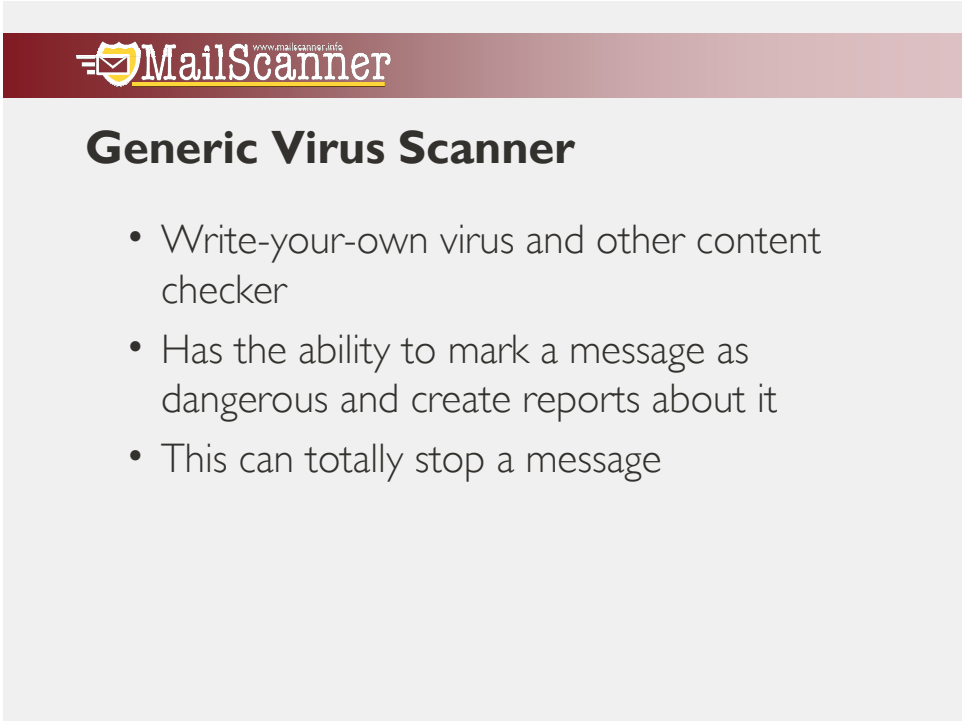
```
allow     -    \.ini$    -    -
```

The last part is to tell MailScanner which combination of ".rules.conf" files are needed for the different email addresses. Create /etc/MailScanner/rules/filenames.rules containing this:

```
From:     domain1.com
     /etc/MailScanner/filename.zip.doc.rules.conf
     /etc/MailScanner/filename.rules.conf
From:     domain2.com
     /etc/MailScanner/filename.ini.rules.conf
     /etc/MailScanner/filename.rules.conf
FromOrTo: default   /etc/MailScanner/filename.rules.conf
```

Unfortunately, the 3 lines above are too long to easily fit on one line of this book. The first and second lines start with "From:", and the third line starts with "FromOrTo:".

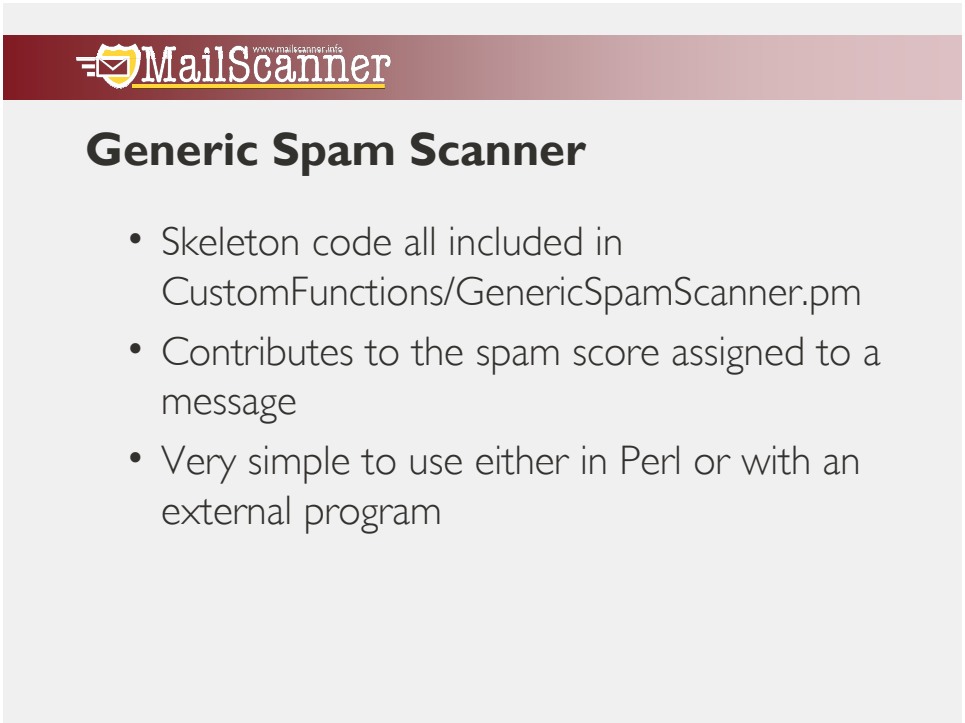
The result of this ruleset is that mail from domain1.com will have its attachments checked according to the allow/deny tests in filename.zip.doc.rules.conf followed by the allow/deny tests in filename.rules.conf. Mail from domain2.com will have its attachments checked according to the allow/deny tests in filename.ini.rules.conf followed by the allow/deny tests in filename.rules.conf. All other mail is just checked against filename.rules.conf on its own.



There are situations in which you will want to user a content checker you have written yourself, or you want to use some other content checker that is not directly supported by MailScanner.

The “Generic Virus Scanner” provides a means whereby you can do your own content checks. There is a “generic-wrapper” script provided for you to start and run your content checker. You will have to write your own parser, which should go into MailScanner/SweepViruses.pm, to process all the output from your checker.

If your checker needs any form of regular updates, such as new virus signatures, then you should customise the “generic-autoupdate” script as needed.



The "Generic Spam Scanner" gives you the ability to easily implement support for unsupported spam scanners such as CRM114 and dspam. All the code you need to get started is in MailScanner/CustomFunctions/GenericSpamScanner.pm.

You can either implement it using a Perl function, or as an external program which is run for each message. There is skeleton code in the file already where you can simply slot in your Perl function, and skeleton code for running an external program.

You do not need to check that your function or program does not take too long or otherwise fails. Full timeout support is already provided by MailScanner itself.

If you choose to do it as an external program, the suggested specification of your program is that it will accept a few lines of envelope information followed by the message. Once it has been processed, it should print the message's score and a 1-line report which will be included in the MailScanner headers:

Input:

1. SMTP connection IP address
2. Envelope sender
3. List of envelope recipients, one per line
4. 1 blank line
5. The whole message, including all the headers and the body in rfc822 format

Output:

1. Floating point or integer giving the spam score of the message
2. 1-line report giving a brief summary of the results of the program

`Use Custom Spam Scanner = no`

If this is set to yes, then the custom (or "generic") spam scanner will be applied to the message. Base your code on the GenericSpamScanner.pm file in the CustomFunctions directory. A timeout wrapper is already included, so you do not need to worry about your program or function not terminating properly.
This can also be the filename of a ruleset.

`Max Custom Spam Scanner Size = 20000`

This is the maximum amount of the message that is passed to the custom spam scanner. The message passed to the scanner will be truncated at this point. This can greatly speed up the scanning of a message, and most spam detection techniques do not require the whole body of a very long message.
This can also be the filename of a ruleset.

`Custom Spam Scanner Timeout = 20`

This is the maximum time in seconds allowed for your spam scanner to run. If it does not complete within this time, it will be automatically killed and will effectively generate a score of 0.
This cannot be the filename of a ruleset.

Max Custom Spam Scanner Timeouts = 10

If your spam scanner fails to complete more than this number of times in any sequence of messages, it is considered to have failed permanently. It will not be tried again until the next periodic restart of MailScanner. See the “Restart Every” setting for more details about this.

This cannot be the filename of a ruleset.

Custom Spam Scanner Timeout History = 20

This is the length of the history of timeout occurrences. More than “Max Custom Spam Scanner Timeouts” in any sequence of messages of this length will cause the spam scanner to be considered to have failed permanently. It will not be tried again until the next periodic restart of MailScanner.

A good example is provided by the default values. 10 failures in any sequence of 20 messages will cause it to be considered dead. So over a sample of 20 messages, a 50% failure rate will stop it being used.

This cannot be the filename of a ruleset.