SpamAssassin

SpamAssassin is an extensible email filter which is used to identify spam.

Yung-Zen Lai (yzlai@tp.edu.tw)

2004/10

Agenda

- Introducing SpamAssassin
  - How It Works
  - Mailer and SpamAssassin
- SpamAssassin Basics
  - Prerequisites
  - Building SpamAssassin
  - Invoking SpamAssassin
- SpamAssassin Rules
- SpamAssassin as a Learning System
  - AutoWhiteListing
  - Bayesian Filtering

Agenda (cont.)

- Integrating SpamAssassin with Postfix
  - Postfix Architecture
  - Spam-Checking During Local Delivery
  - Spam-Checking All Incoming Mail
    - AMaViSd.conf
- Using SpamAssassin as a Proxy
  - Using Pop3proxy
  - Using SAproxy Pro

Introducing SpamAssassin

- Software for analyzing email messages, determining how likely they are to be spam, and reporting its conclusions.
  - Uses a large number of different kinds of rules and weights them.
  - Easy to tune the scores associated with each rule or to add new rules.
  - Can adapt to each system's email environment, learning to recognize which senders are to be trusted and to identify new kinds of spam.

From: "xbwhc" <acfxi@ms73.hinet.net>

Reply-To: "xbwhc" <acfxi@ms73.hinet.net>

To: netadm@tp.edu.tw

X-Mailer: punk digestible egregious

Subject: =?ISO-8859-1?Q? *SPAM-Mail*
XYZ=B8=EA=B0T=A4u=A7{=A7=F3=B7s=A7=D6=B3=F8chosen?=

Date: Mon, 18 Oct 2004 22:02:19 +0400

MIME-Version: 1.0

Organization: punk digestible egregious

Content-Type: multipart/alternative; boundary="=====8516744314768 3=_“

X-Amavis-Alert: BAD HEADER Non-encoded 8-bit data (char B8 hex) in message header 'Subject' Subject: XYZ\270\352\260T\244u\247{\247\363\267s\247\326\263\370... ^

X-Spam-Status: Yes, hits=15.9 tagged_above=5.0 required=8.0 tests=BAYES_99, HTML_FONTCOLOR_RED, HTML_MESSAGE, RAZOR2_CF_RANGE_11_50, RAZOR2_CHECK, RCVD_IN_DSBL, RCVD_IN_NJABL, RCVD_IN_NJABL_DIALUP, RCVD_IN_RFCI, SUBJ_ILLEGAL_CHARS

X-Spam-Level: **********

X-Spam-Flag: YES

How SpamAssassin Works

- Check consistency and adherence to Internet standards. (RFC 2821, 2822, …)
- Check headers and body for phrases or messages elements commonly found in spam.
- looked up headers and body in several online databases that track message checksums of verified spam messages.
- Lookup up the sending system’s IP address in several online lists of sites that have been used by spammers or are otherwise suspicious. (RBL)

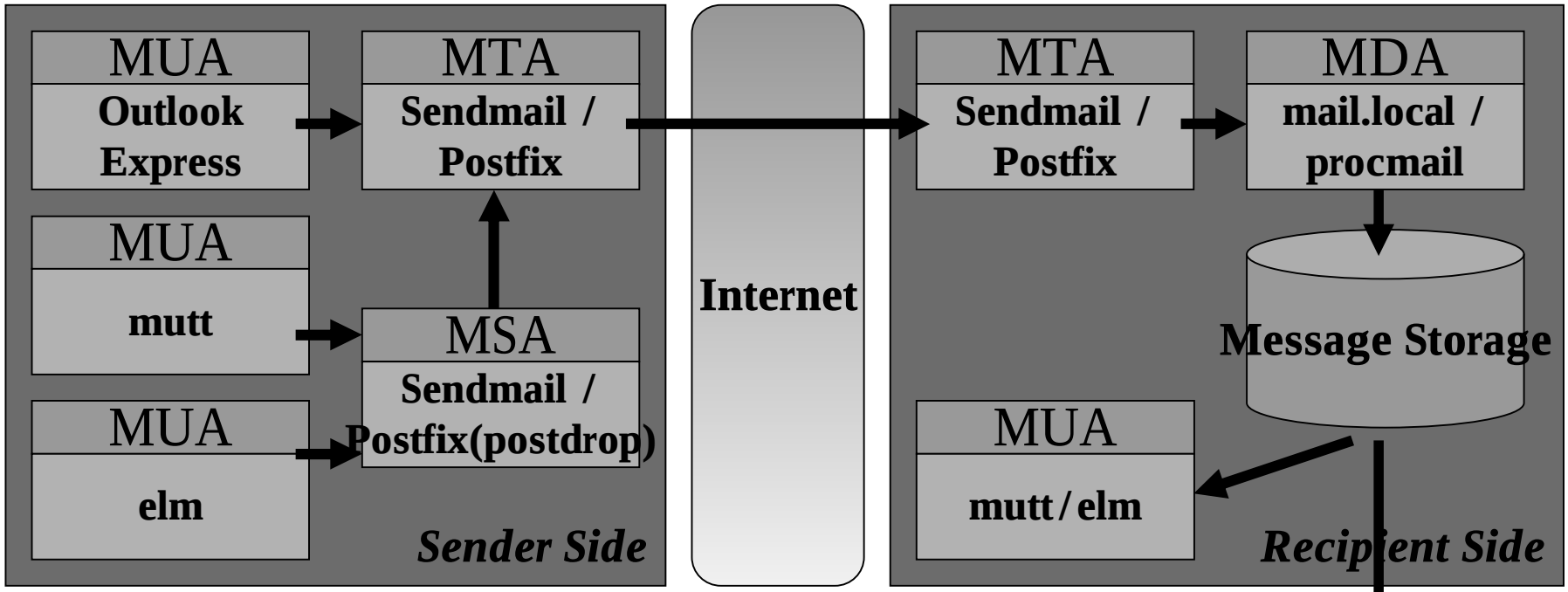
How SpamAssassin Works (cont.)

- Specific addresses, hosts, or domains can be blacklisted or whitelisted.
- Can be trained to recognize the types of spam that you received.
- The sending system's IP address can be compared to the sender's domain name using the Sender Policy Framework (SPF) protocol. (version 3.0)
- Can privilege senders who are willing to expend some extra computational power in the form of Hashcash. (version 3.0)

Mailers and SpamAssassin

- Typical mail transmission
- Scanning at the MTA
- Scanning at the MDA
- Scanning with a POP Proxy

Message flow



UA: User Agent
TA: Transfer/Transport Agent
DA: Deliver Agent
SA: Submission Agent
AA: Access Agent

SpamAssassin Basics -- Prerequistes

- Written in Unix or Unix-Like environment that includes Perl 5, preferably 5.6.1 or later.
- Up-to-Date versions of the following Perl modules
  - ExtUtils::MakeMaker
  - File::Spec
  - Pod::Usage
  - HTML::Parser
  - Sys::Syslog
  - DB_File
  - Digest::SHA1
  - Net::DNS

Building SpamAssassin

- perl -MCPAN -e shell
  - o conf prerequisites_policy ask
  - install Mail::SpamAssassin
- Build from source tarball
  - http://spamassassin.apache.org
  - perl Makefile.PL
- Linux: Download the source RPM
  - http://spamassassin.apache.org
- FreeBSD: Build from the ports collection
  - /usr/ports/mail/p5-Mail-SpamAssassin-3.xx

Basic Configuration & Test

- [/usr/local] /etc/mail/spamassassin/local.cf
  - required_hits/required_score
  - report_safe
  - rewrite_subject (2.x only)
  - rewrite_header (Subject, From, or To headers 3.0 only)
  - skip_rbl_checks
- Test
  - spamassassin --test-mode < sample-spam.txt
  - spamassassin --test-mode < sample-nonspam.txt
- Untagging
  - spamassassin --remove-markup < marked-message > unmarked-message

Invoking SpamAssassin

- With procmail
  - Scanning at the MDA (/etc/procmailrc)

```
DROPPRIVS=yes
PATH=/bin:/usr/bin:/usr/local/bin
SHELL=/bin/sh


# Spamassassin
:0fw
< 300000
|/usr/bin/spamassassin
```

Invoking SpamAssassin (cont.)

- Using spamc/spamd
  - Client/Server Architecture
  - spamc is a lightweight client, written in C

```
$ /usr/bin/spamd –daemonize –pidfile /var/run/spamd.pid


$ spamc –c < sample-nonspam.txt
0.0/5.0
$ spamc –c < sample-spam.txt
1000.0/5.0
```

Invoking SpamAssassin (cont.)

- Invoking spamc with procmail (procmailrc)
  - Just as spamc is run in place of the spamassassin

```
DROPPRIVS=yes
PATH=/bin:/usr/bin:/usr/local/bin
SHELL=/bin/sh


# Spamassassin
:0fw
< 300000
|/usr/bin/spamc
```

Invoking SpamAssassin (cont.)

- spamd caches the spam-checking rules in memory when it starts up.
- The daemon must be signaled whenever changes the SpamAssassin rulesets or systemwide configuration file.
- spamd reloads configuration files when it receives a HUP signal.

```
$ kill –HUP `cat /var/run/spamd.pid`
```

Invoking SpamAssassin (cont.)

- In a Perl Script

```
#!/usr/bin/perl


use Mail::SpamAssassin;


my @lines = <STDIN>;
my $mail = Mail:SpamAssassin::NoMailAudit->new(data => \@lines);
my $spamtest = Mail::SpamAssassin->new();
my $status = $spamtest->check($mail);
$status->rewrite_mail() if ( $status->is_spam() );
print $status->get_full_message_as_text();
```

SpamAssassin and the End User

- True Nagatives(ham)
  - Both you and SpamAssassin agree are non-spam
- True Positives(spam)
  - Both you and SpamAssassin agree are spam.
- False Positives
  - SpamAssassin marks a message as spam that you actually wanted to receive.
- False Negatives
  - A missed spam.

SpamAssassin Rules

- All *static* tests
  - Don’t change over time
- Pattern-based & Network-based tests
- How they are written and scored
- How to modify the score of a build-in test
- How to write our own custom tests
- Whitelist and Blacklist rules

A test definition and score

```
header    FROM_STARTS_WITH_NUMS       From =~ /^\d\d/
describe FROM_STARTS_WITH_NUMS       From: starts with nums
score     FROM_STARTS_WITH_NUMS       0.390 1.574 1.044 0.579
```

- The header directive defines it as a test that will be applied to the message headers and give the test name (FROM_STARTS_WITH_NUMS)
- The describe directive provides a human-readable description
- SpamAssassin will add 0.390, 1.574, 1.044, or 0.579 to the spam score with network and Bayesian tests are in use or not in use

Modifying the Score of a test

- Adjustment scores in systemwide configuration file
  - /etc/mail/spamassassin/local.cf
  - score HTML_WIN_OPEN 2
- Write our own systemwide tests
  - Regular expression matching
  - Eval test defined in Mail::SpamAssassin::Evaltests

Header Tests

- Existence of a header
  - header TESTNAME exists:headername
- Test single header
  - header TESTNAME headername =~ /regexp/modifiers
- Test single header not match
  - header TESTNAME headername !~ /regexp/modifiers
- Test multiple header
  - header TESTNAME =~ /^(?:headername1|headername2|…):.*/modifiers
- All capital letter
  - header SUBJ_ALL_CAPS eval:subject_is_all_caps()
- RBL
  - header TESTNAME =~ eval:check_rbl('ZoneID', 'Server')
  - check_rbl_txt()
  - check_rbl_sub()

Body Tests

- The body, rawbody, full directives
  - Can be tested against a regular expression
  - Can be submitted to an eval test
- The body directive appears to a person reading the message in an text-based MUA
- The rawbody directive appears to a person reading the message in an HTML-based MUA
- The full directive defines a test to be applied to the full text of a message

URI Tests

- Test on all URIs that appear in an email message
  - uri MAILTO_TO_REMOVE /^mailto:.*?remove/is
- SpamAssassin 3.0 includes a plug-in to test the uridnsbl directive

```
uridnsbl    URIBL_SBLXBL        sbl-xbl.spamhaus.org.   TXT
header      URIBL_SBLXBL        eval:check_uridnsbl('URIBL_SBLXBL')
describe    URIBL_SBLXBL        Contains a URL listed in the SBL/XBL
score       URIBL_SBLXBL        10
```

Test Examples

- For a To, From, or Cc header
  - header XXX ALL =~ /^(?:to|cc|from):.*abc@def/im
- For the existence of the X-PMFLAGS header
  - header XXX exists:X-PMFLAGS
- For long lines of hexadecimal code in the message body
  - header XXX /[0-9a-fA-F]{70,}/
- For Subject header in all capital letters
  - header XXX eval:subject_is_all_caps()
- For a message that include HTML to open a new window
  - body XXX eval:html_test('window_open')
- For an HTTP URI in the message
  - uri XXX /^http:ooxxabcd/is

The Built-in Tests

- Over 700 test rules
  - /usr/share/spamassassin
  - http://spamassassin.apache.org/tests.html
- 10_misc.cf
  - Defines special rules that are not spam tests
  - Include templates for the spam report
- 20_fake_helo_tests.cf
  - Defines a set of rules that use the eval test check_for_rdns_helo_mismatch()

Whitelists and Blacklists

- Whitelisting senders
  - whitelist_from yzlai@hotmail.com
  - whitelist_from *@tp.edu.tw
- Whitelisting senders by relay
  - whitelist_from_rcvd *@tp.edu.tw smtp.tp.edu.tw
- Whitelisting recipients
  - whitelist_to netadm@tp.edu.tw
- Blacklist
  - blacklist_from netadm@tp.edu.tw
  - blacklist_to spamtrap@tp.edu.tw

SpamAssassin as a Learning System

- Learns each sender's history of sending spam or non-spam messages
- Modifies the spam score of their subsequent mailings on the basis of this history
- Trying to reduce false positives
- Can also reduce false negatives, but this happens infrequently enough.

Autowhitelisting

- SpamAssassin maintains a database keyed on message senders' email address and the IP addresses of their nearest untrusted relay.
- The average sender score – the total score divided by the number of messages received – is used to modify the spam score of new messages from that sender
- Use *auto_whitelist_factor* directive to set the multiplier between a message's spam score and the sender's history average score.

Autowhitelisting (cont.)

Message number	Message score (before)	Sender average score	Score after autowhitelist with give AWF			
			0	.5	.75	1
1	2	(none)	2	2	2	2
2	1	2	1	1.5	1.75	2
3	1	1.5	1	1.25	1.375	1.5
4	0	1.33	0	0.67	1.00	1.33
5	2	1.0	2	1.5	1.25	1.0
6	6	1.2	6	3.6	2.4	1.2

Using Autowhitelists in Perl

```
#!/usr/bin/perl


use Mail::Spamassassin;


my @lines = <STDIN>;
my $mail = Mail:SpamAssassin::NoMailAudit->new(data => \@lines);
my $spamtest = Mail::SpamAssassin->new();
my $awl = Mail::SpamAssassin::DBBasedAddrList->new();
$spamtest->set_persistent_address_list_factory($awf);
my $status = $spamtest->check($mail);
$status->rewrite_mail() if ( $status->is_spam() );
print $status->get_full_message_as_text();
```

Bayesian Filtering

- Bayesian filtering is based on Bayes' Theorem, a statement of probability theory
- Can reduce both false positives and false negatives
- The probability of some event given a test result depends on the baseline probability of the event before the test result is known and on the discriminating power of the test
- The more the test result can increase (or decrease) the probability from baseline, the stronger the test.
- Some are patterns that only occur in spam (and never in non-spam) …

Bayesian Filtering (cont.)

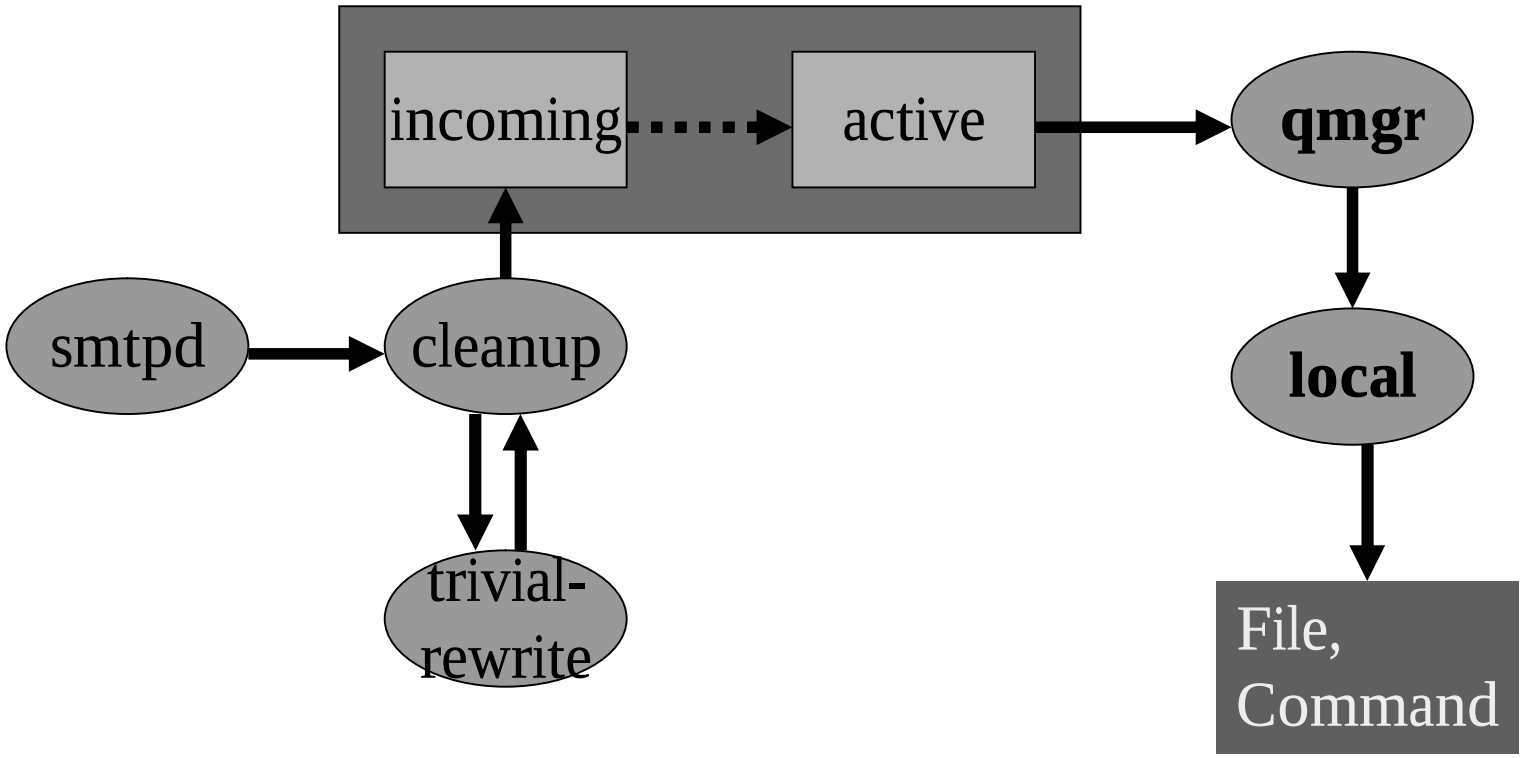
- Configuration
  - use_bayes
  - bayes_auto_learn
  - bayes_auto_learn_threshold_nonspam
  - bayes_auto_learn_threshold_spam
  - bayes_ignore_from address
  - bayes_ignore_to address
  - bayes_auto_expire
  - bayes_expiry_max_db_size

Bayesian Filtering (cont.)

- Training
  - sa-learn --mbox --spam Mail/spam
  - sa-learn --mbox --ham mail/nonspam
  - sa-learn --mbox --spam /var/spool/mail/spamtrap
  - sa-learn --dir --spam …
  - sa-learn --no-rebuild --spam ...
  - sa-learn --rebuild

Integrating with Postfix

- Postfix architecture



Integrating with Postfix (cont.)

- Checking During Local Delivery
  - Easy to set up
  - Run *spamd*, and the *procmail* recipe can use *spamc* for faster spam-checking
  - User preference files, autowhitelisting, and Bayesian databases can be used
- Postfix's mail.cf
  - mailbox_command = procmail

Integrating with Postfix (cont.)

- Checking All Incoming Mail
  - Postfix's *content_filter* parament
  - To a Program
  - To a Daemon
    - More effective
    - Building a Spam-Checking Gateway

Using a Program as a Content Filter

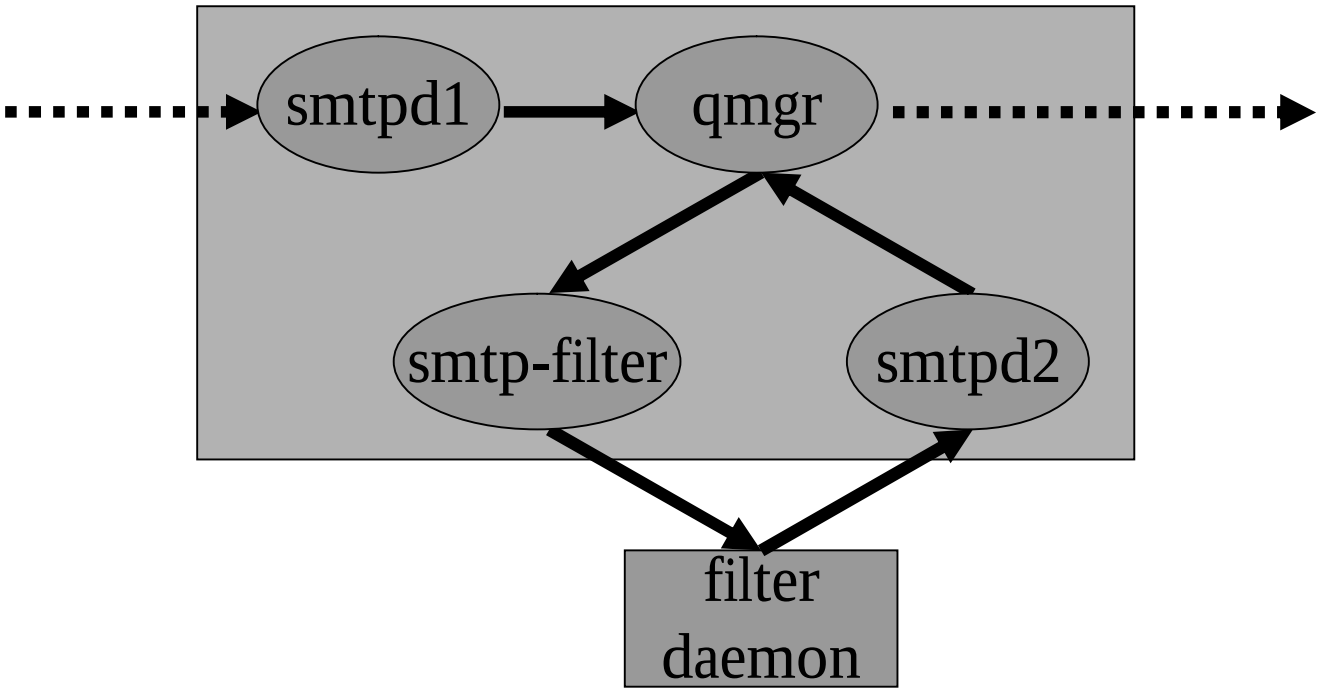
```
#!/bin/sh

sender=$1
shift
recip=“$@”
If [ “$#” - eq 1 ]; then
  /usr/bin/spamc –u $recip
else
  /usr/bin/spamc
fi | /usr/sbin/sendmail -i –f $sender -- $recip
exit $?
```

- main.cf
  - content_filter = spamcheck:
- master.cf
  - smtpcheck unix - n n - - pipe
    flags=Rq user=spamfilt argv=/usr/local/bin/pf-spamfilt ${sender} ${recipient}

Using a Daemon as a Content Filter

- Daemon-based filtering
  - more efficient



Using a Daemon as a Content Filter (cont.)

- Install a daemon that performs content-filtering
  - amavisd-new
- main.cf

  content-filter = smtp-filter:[127.0.0.1]:10024
- master.cf

  smtp-filter unix - - n - 10 smtp

  -o myhostname=localhost

  localhost:10025 inet n - n - 10 smtpd

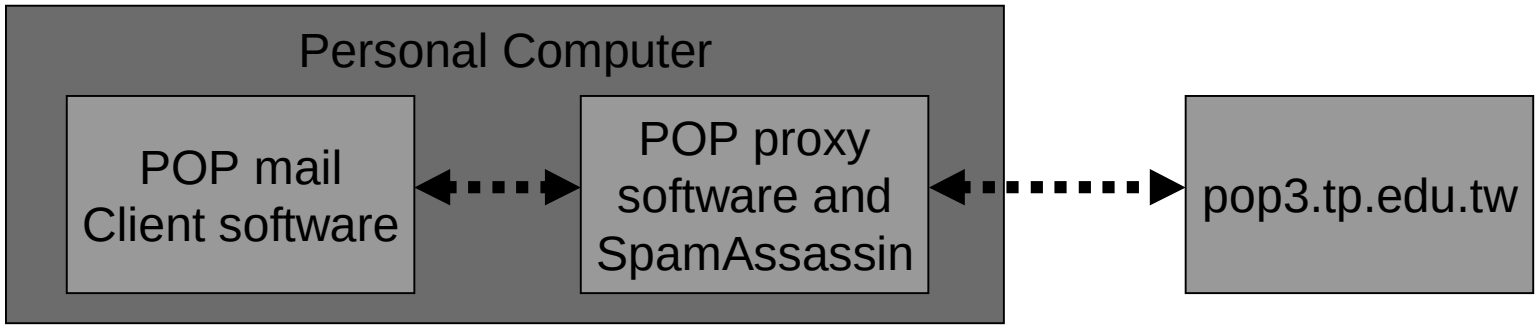
  -o content_filter=

AMaViSd.conf

- $mydomain = ‘tp.edu.tw’;
- $forward_method = 'smtp:127.0.0.1:10025‘;
- $max_servers = 15;
- $final_spam_destiny = D_DISCARD;
- $sa_tag_level_deflt = 5.0;
- $sa_tag2_level_deflt = 8.0;
- $sa_kill_level_deflt = 20.0;
- $sa_spam_subject_tag = '*SPAM-Mail* ‘;
- $sa_... = …;

Using as a Proxy

- Message flow



Using as a Proxy (cont.)

- Using Pop3proxy
  - Download and Extract Pop3proxy from http://mcd.perlmonk.org/pop3proxy/
  - Install Perl for Windows which includes Time::HiRes module
  - Download and Extract SpamAssassin from http://spamassassin.apache.org/
  - `$ perl pop3proxy.pl --host pop3.tp.edu.tw`
  - Reconfigure the POP Client
    - Connect to *localhost* instead of the usual POP server

Using as a Proxy

- Using SAproxy Pro
  - A commercialized version of Pop3proxy
  - http://www.statalabs.com/
  - Reconfigure the POP Client
    - Connect to *localhost* instead of the usual POP server
    - <account>:<pop server> in Account Name

The End

- Reference
  - *SpamAssassin – The Open Source Solution to SPAM*, Alan Schwartz, O'Reilly, July 2004
- Thank you!