Filtering Mail with Milter

David F. Skoll
Roaring Penguin Software Inc.

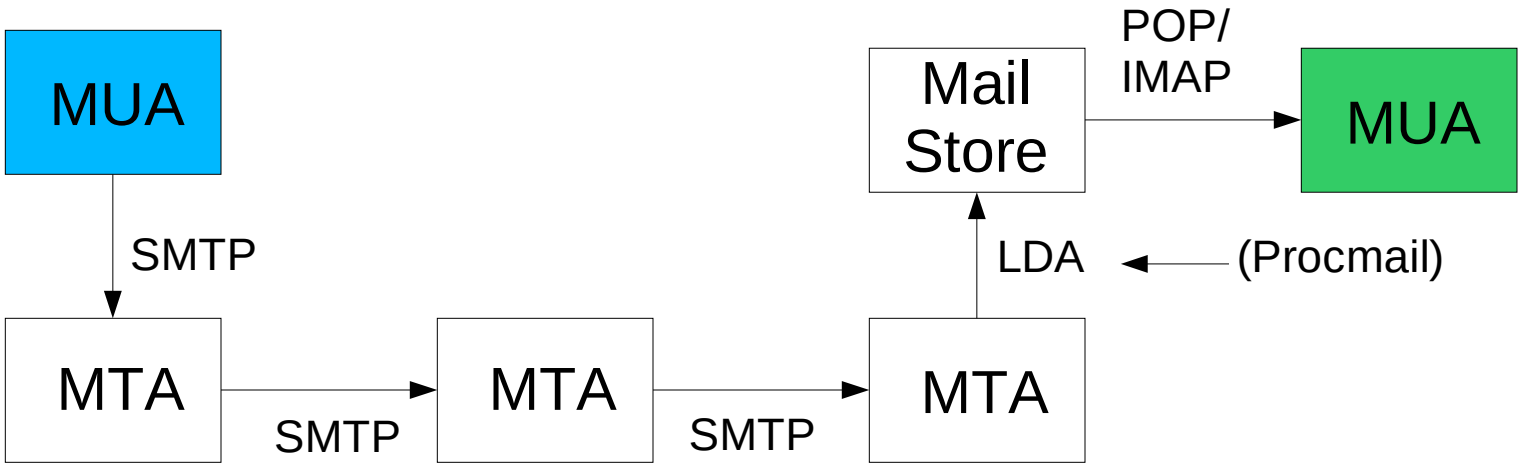
Overview

- Why filter mail?
- Different filtering approaches
  - Delivery agent (e.g. Procmail)
  - Central filtering (Milter)
- Milter Architecture
- Milter API
- Pros and cons of Milter API
- MIMEDefang Architecture
- Conclusions

Why Filter Mail?

- The old reason: to stop viruses.
- The new reason: to stop spam and inappropriate content.
- Blocking viruses is easy. Block .exe and friends, and test against signature databases.
- Blocking spam is hard, but becoming increasingly important. Organizations can even face lawsuits over inappropriate content.

One-slide Overview of Mail Delivery



MUA = Mail User Agent. What you use to send and read mail. Eg: Mozilla, Evolution
MTA = Mail Transfer Agent. Eg: Sendmail, Postfix, Exim, qmail
Mail Store = Where mail is stored until it is read. Usually in /var/spool/mail
LDA = Local Delivery Agent. Program which performs final delivery into the mail store
POP = Post Office Protocol. A method for retrieving mail from a mail store
IMAP = Internet Message Access Protocol. A better protocol than POP/POP3.
SMTP = Simple Mail Transfer Protocol. The language used by MTA's to talk to each other.

Filtering with Procmail

- The local delivery agent does the filtering.
- PRO: Easy to customize filter rules per-recipient. Procmail well-known and tested.
- CON: Inefficient. Messages sent to multiple users are processed multiple times.
- CON: Local users only. Difficult to use procmail on a store-and-forward relay.
- CON: SMTP transaction is complete by the time Procmail is invoked.

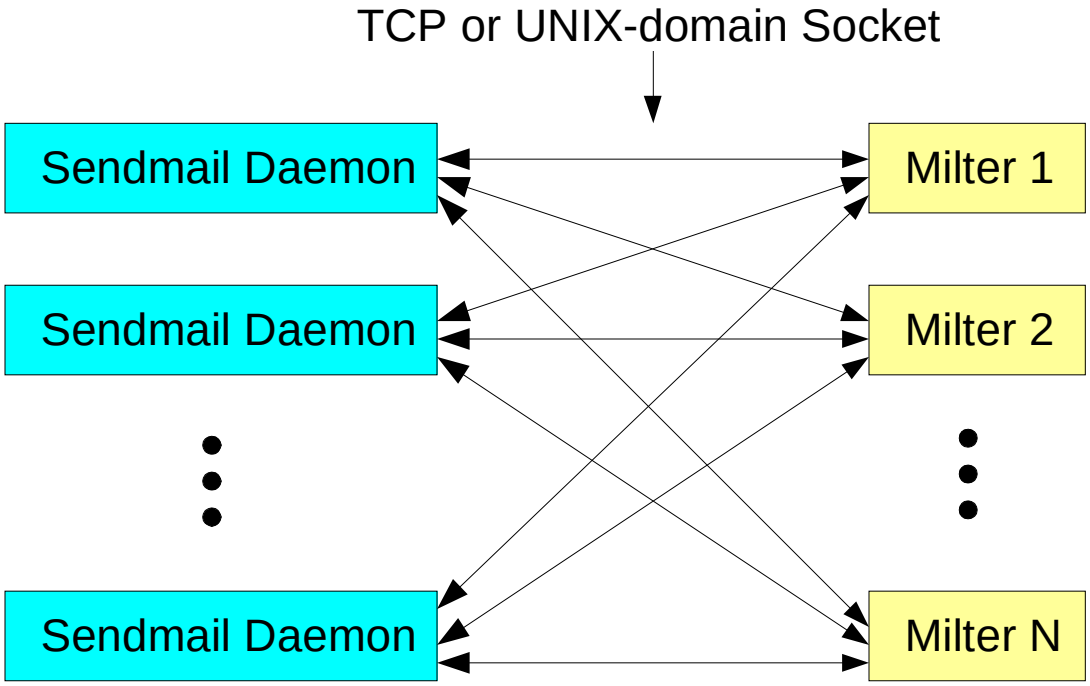
Central Filtering (Topic of This Talk)

- The mail transfer agent does the filtering.
- PRO: Efficient. Each message filtered once, regardless of number of recipients.
- PRO: Can modify the SMTP dialog. Amazingly useful, as you'll see...
- PRO: Can filter relayed mail, not just local.
- CON: Harder (but not impossible) to implement per-recipient filter rules.

Milter

- Milter (Mail fILTER) is both a *protocol* and a *library*.
- Milter was designed by the Sendmail development team, and has been part of Sendmail since 8.10. Really matured in 8.12.
- Milter lets filters “listen in” to the SMTP conversation and modify SMTP responses.
- Milter is extremely powerful and can let you do amazing things with e-mail.

Milter Architecture



Sendmail is single-threaded, but forks into multiple processes. The milter library is multi-threaded, and a given Sendmail installation can have multiple mail filters (filters 1 through N above.)

Milter API Operation

- At various stages of the SMTP conversation, Sendmail sends a message over the socket to the milter.
- The milter library invokes a *callback* into your code. It then sends a reply message back to Sendmail containing the return value from your callback.
- In addition, you can call milter library functions that send special messages to Sendmail to modify aspects of the message.

Typical SMTP Conversation

```
C: Connect to server
S: 220 server_hostname ESMTP Sendmail 8.12.7/8.12.7...
C: HELO client_hostname
S: 250 server_hostname Hello client_hostname, pleased...
C: MAIL FROM:<dfs@roaringpenguin.com>
S: 250 2.1.0 <dfs@roaringpenguin.com>... Sender ok
C: RCPT TO:<foo@roaringpenguin.com>
S: 250 2.1.5 <foo@roaringpenguin.com>... Recipient ok
C: RCPT TO:<bar@roaringpenguin.com>
S: 250 2.1.5 <bar@roaringpenguin.com>... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: (transmits message followed by dot)
S: 250 2.0.0 h0AJVcGM007686 Message accepted for delivery
C: QUIT
S: 221 2.0.0 server_hostname closing connection
```

Typical SMTP Conversation with Milter

```
C: Connect to server                                                    *
S: 220 server_hostname ESMTP Sendmail 8.12.7/8.12.7...
C: HELO client_hostname                                                 *
S: 250 server_hostname Hello client_hostname, pleased...
C: MAIL FROM:<dfs@roaringpenguin.com>                                   *
S: 250 2.1.0 <dfs@roaringpenguin.com>... Sender ok
C: RCPT TO:<foo@roaringpenguin.com>                                     *
S: 250 2.1.5 <foo@roaringpenguin.com>... Recipient ok
C: RCPT TO:<bar@roaringpenguin.com>                                     *
S: 250 2.1.5 <bar@roaringpenguin.com>... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: (transmits message followed by dot)                                  *
S: 250 2.0.0 h0AJVcGM007686 Message accepted for delivery
C: QUIT
S: 221 2.0.0 server_hostname closing connection
```

* = response-modification opportunity * = filtering opportunity

Milter API Functions

- Initialization:
  - **`smfi_register`** Register a filter
  - **`smfi_setconn`** Specify socket
  - **`smfi_settimeout`** Set timeout
  - **`smfi_main`** Enter main milter loop
- You “register” a filter and tell what kinds of callbacks you're interested in, and whether or not you might modify the message headers and body.

Milter API Functions: Data Access

- **`smfi_getsymval`** Get value of a Sendmail macro
- **`smfi_getpriv`** Get arbitrary private data
- **`smfi_setpriv`** Set arbitrary private data
- **`smfi_setreply`** Set SMTP reply code
- Accessing Sendmail macros lets you access a lot of useful info.
- Private data useful for storing thread-specific data.
- Can set SMTP reply to anything you like

Milter API: Message Modification

- **`smfi_addheader`** Add a header
- **`smfi_chgheader`** Change a header
- **`smfi_addrcpt`** Add a recipient
- **`smfi_delrcpt`** Delete a recipient
- **`smfi_replacebody`** Replace message body
- Recipient functions affect *envelope recipients*, not headers.

Milter API: Callbacks

- **`xxfi_connect`** Called when connection made
- **`xxfi_helo`** Called after HELO
- **`xxfi_envfrom`** Called after MAIL FROM:
- **`xxfi_envrcpt`** Called after RCPT TO:
- **`xxfi_header`** Called for each header
- **`xxfi_eoh`** Called at end of all headers
- **`xxfi_body`** Called for each “body block”
- **`xxfi_eom`** Called at end of message
- **`xxfi_abort`** Called if message aborted
- **`xxfi_close`** Called when connection closed

Callback Types

- Connection oriented: Apply to connection as a whole. **`xxfi_connect`**, **`xxfi_helo`**, **`xxfi_close`**.
- Recipient oriented: Apply to a single recipient only. **`xxfi_envrcpt`**
- Message-oriented: Apply to a message. All other callbacks are message-oriented.

Callback Return Values

- **`SMFIS_CONTINUE`**: Continue processing
- **`SMFIS_REJECT`**: For connection-oriented callback, close the whole connection. For message-oriented, reject the message. For recipient-oriented, reject only this recipient.
- **`SMFIS_DISCARD`**: For message- or recipient-oriented routine, silently discard message.
- **`SMFIS_ACCEPT`**: Accept without further filtering.
- **`SMFIS_TEMPFAIL`**: Return a temporary-failure code for recipient, message or connection.

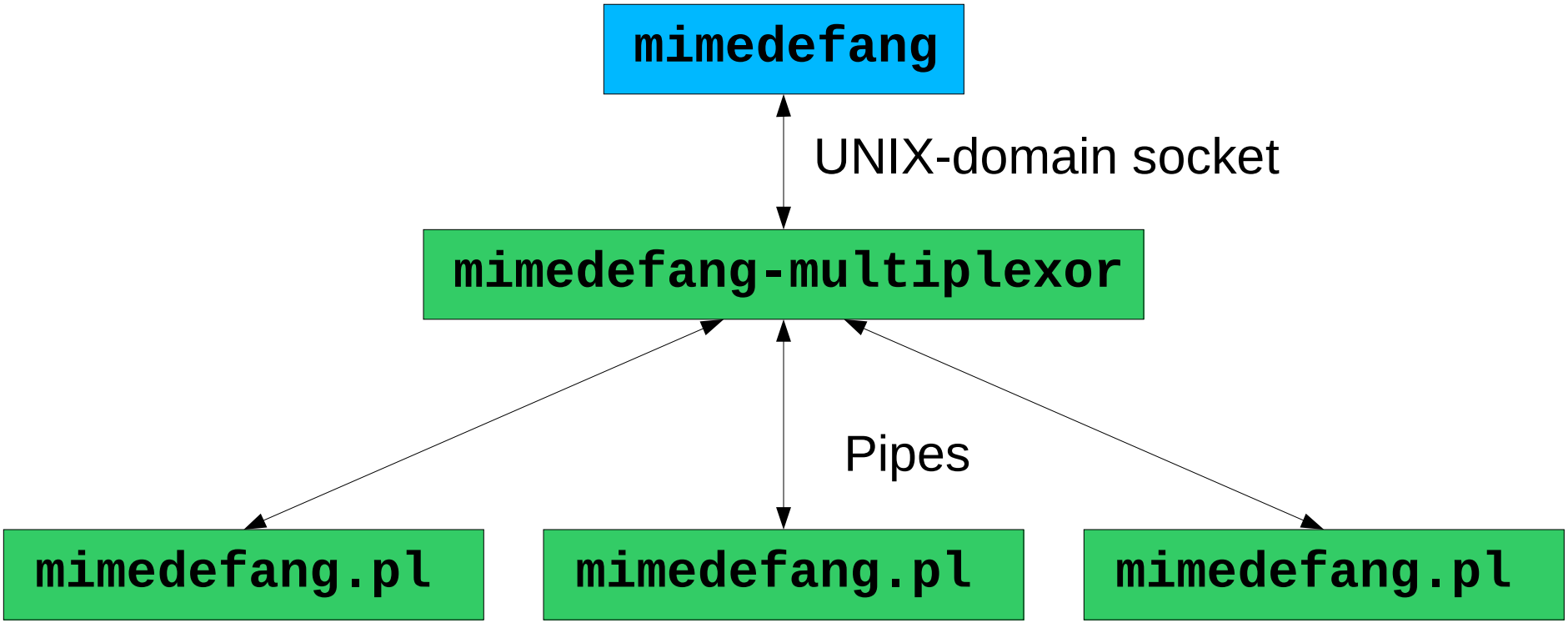
Pros and Cons of Milter API

- Pro: Sendmail's implementation is not too buggy.
- Pro: Quite efficient. Filters do what they need and no more.
- Pro: Can modify SMTP return codes.
- Con: Written in C. C is not the best language for parsing e-mail messages. Hard for end-users to write filters.
- Con: Multi-threaded. Easy to make dumb programming errors; some pthreads implementations leave a lot to be desired.

MIMEDefang

- MIMEDefang is a GPL'd, Perl-based mail filter. It uses Milter to interface with Sendmail.
- Currently used in thousands of different sites.
- Larger MIMEDefang installations process >1 million e-mail messages per day.
- MIMEDefang runs on Linux, FreeBSD, Solaris, Tru64 UNIX, HP-UX, AIX, ...
- http://www.roaringpenguin.com/mimedefang/

MIMEDefang Architecture



- Multithreaded mimedefang threads talk to *single-threaded* mimedefang-multiplexor
- Multiplexor manages a pool of single-threaded Perl slaves that do the actual filtering.

MIMEDefang Advantages

- Robust: Perl slaves are killed after they process a certain number of messages. Perl memory leaks are therefore not a problem.
- Efficient: Idle Perl processes are fed work. No per-message Perl startup overhead. Similar to Apache's pre-forked MPM.
- Scalable: Just add processors and RAM...
- Easy to use: Single-threaded Perl filters much easier to write than multi-threaded C. Can take advantage of CPAN goodies.

MIMEDefang Advantages – 2

- Lots of built-in functionality:
  - Hooks to anti-virus software
  - Hooks to SpamAssassin
  - Easy to drop attachments or reject mail based on filenames (*.exe, for example)
  - Can add/delete recipients, add/delete/change headers.
- Almost all Milter functionality is exposed to MIMEDefang.

MIMEDefang in the Real World

- Large US financial institution filters >1 million messages/day on a couple of Sun E-450's.
- Large US PC manufacturer filters > 1.5 million messages/day on 10 Linux PC's.
- One crazy MIMEDefang list member reported filtering almost 2 million messages/day on a single specially-tuned Intel/Linux box. (He modified the Perl filter quite extensively.)

Example MIMEDefang Filter

- Block Korean mail (filter fragment):

```
$head = $entity->head;
$charset = $head->mime_attr("content-type.charset");
if (defined($charset)) {
    $charset =~ tr/A-Z/a-z/;
    if ($charset eq "ks_c_5601-1987" or
        $charset eq "euc-kr") {
        return action_bounce(
          "Spam from Korea not accepted.");
    }
}
```

Example MIMEDefang Filter 2

- Tempfail open relays and disbelieve spamware that uses our domain in HELO

```
sub filter_relay ($$$) {
 my ($ip, $name, $helo) = @_;
 if ($helo =~ /roaringpenguin\.com$/i) {
   return('REJECT', 'Liar... not roaringpenguin');
 }
 if (relay_is_blacklisted($ip, 'ordb.org')) {
   return('TEMPFAIL',
   'Try again when you are no longer an open relay');
 }
 return('CONTINUE', 'OK');
}
```

More on MIMEDefang

- Perl callbacks are called after MAIL FROM:, RCPT TO: and end of DATA phase.
- All Sendmail macro values are accessible.
- Can do things like:
  - Replace large attachments with URLs
  - Omit spam-scanning for users authenticated with SMTP AUTH
  - Quarantine viruses
  - Make copies of incoming/outgoing mail

CanIt

- CanIt and CanIt-PRO are commercial (I.E., not open-source) solutions built around MIMEDefang.
- Features include:
  - Sophisticated SQL-based spam trap.
  - Lots of anti-spam tricks.
  - Simple Web interface for managing spam.
  - Per-recipient traps (CanIt-PRO) with individual rules, thresholds, etc.

CanIt

- CanIt is not open-source because I have to eat. (You get the source, but can't redistribute it. You can modify it for internal use.)
- However, CanIt revenues support continuing development of MIMEDefang, and some CanIt features eventually migrate their way into MIMEDefang.
- CanIt URL: http://www.canit.ca/

Conclusions

- Filtering at the SMTP server is the best place to filter mail.
- More efficient than LDA approaches.
- More flexible --- can modify SMTP return codes.
- With a bit of cleverness, you can still have per-recipient rules.
- Sendmail's Milter API is a pretty decent API for achieving robust server-based mail filtering.