MIMEDefang Start Guide

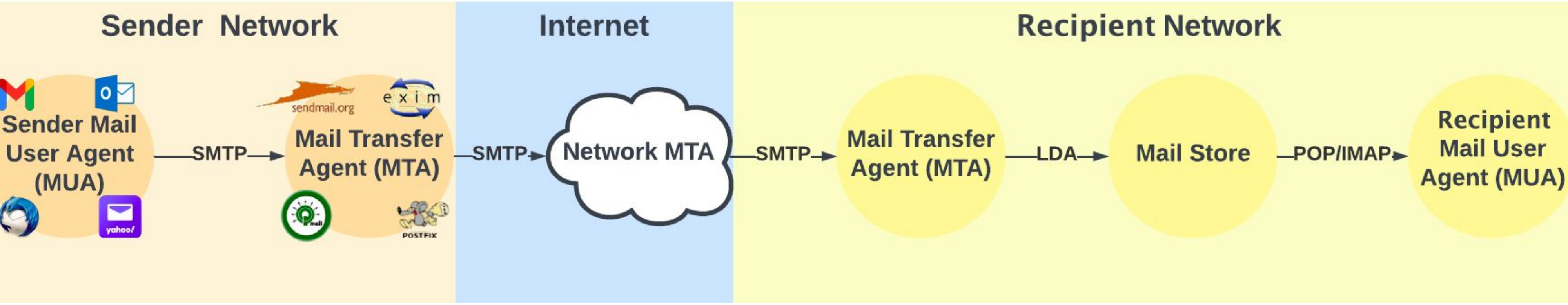
Made By Georgia Smith for The McGrail Foundation

What is MIMEDefang?

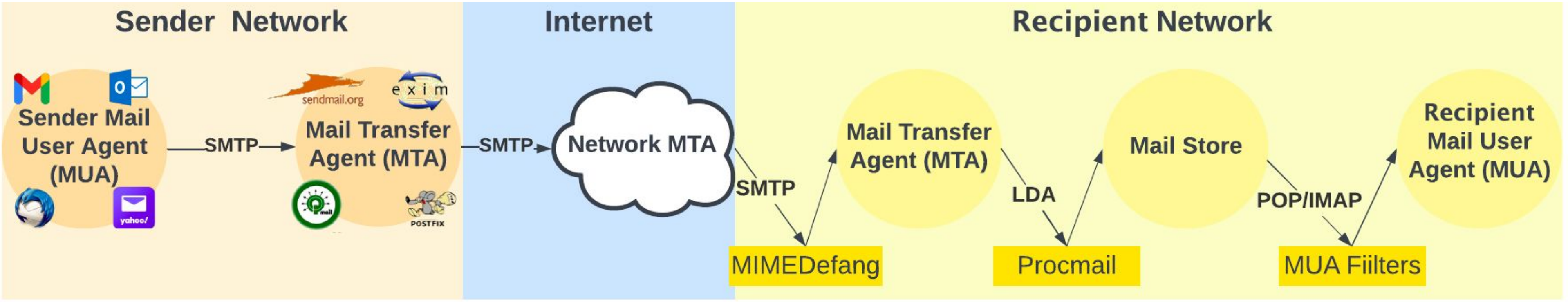
From MIMEDefang.org:

> MIMEDefang is an e-mail filtering tool that works with the Sendmail “Milter” library. MIMEDefang lets you express your filtering policies in Perl rather than C, making it quick and easy to filter or manipulate your mail.

The Email Process


Sender Network
Internet
Recipient Network
Sender Mail User Agent (MUA)
SMTP
sendmail.org
exim
Mail Transfer Agent (MTA)
POSTFIX
SMTP
Network MTA
SMTP
Mail Transfer Agent (MTA)
LDA
Mail Store
POP/IMAP
Recipient Mail User Agent (MUA)

Where MIMEDefang Fits In



Sendmail Overview

- A Milter (**M**ail f**ILTER**) is both the protocol & the Library
  - Invented by Sendmail
  - Can also be implemented in Postfix
  - Exim has similar features
- Milters are written in C, which is not the most user-friendly language
- A Sendmail Milter lets you interact at certain phases of a mail conversation
  - You can modify responses to SMTP commands and alter mail contents!

Sendmail Overview: Continued

- The milter library invokes a callback to run your code, which then returns a reply message with the return value from your callback
- Assume each callback is autonomous, that is, use thread safe concepts and data storage techniques
    - **HINT**: Databases are your friend

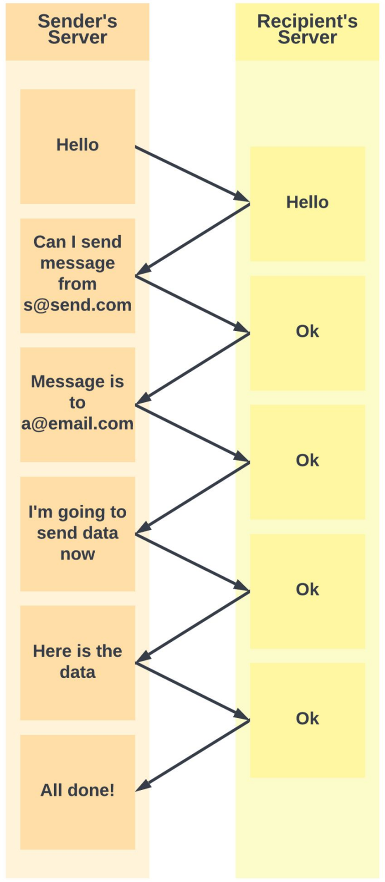
What is a callback?

A callback is a function that is passed as an argument into another function and returns a specific value. Often a callback is a function that is called when an event is triggered.

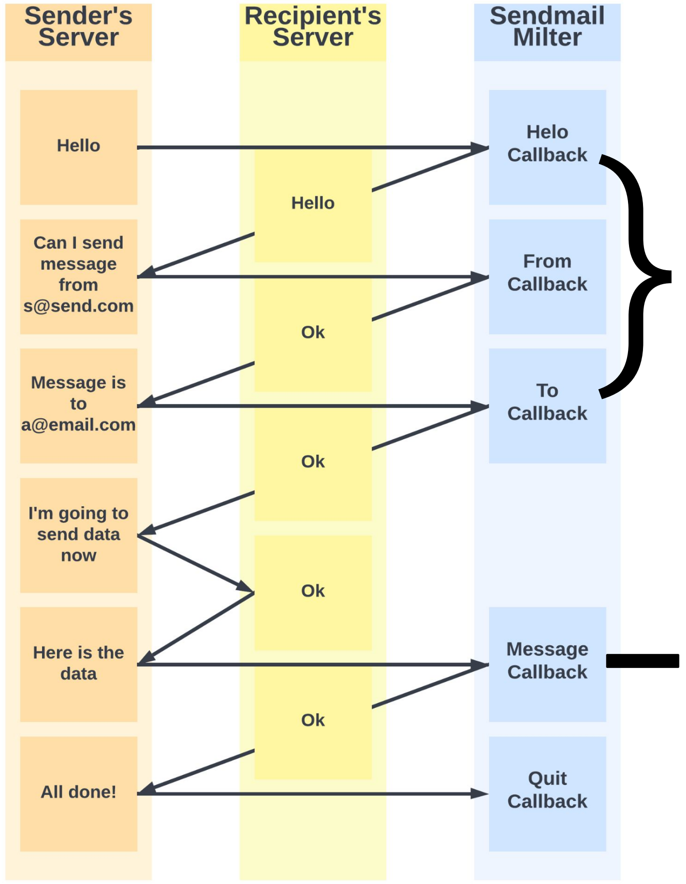
WARNING: callbacks are multi-threaded

When using callbacks assume each callback is autonomous and use data storage and thread safe best practices

A Simple SMTP Conversation


Sender's Server
Recipient's Server
Hello
Hello
Can I send message from s@send.com
Ok
Message is to a@email.com
Ok
I'm going to send data now
Ok
Here is the data
Ok
All done!

A Simple SMTP Conversation with Sendmail



These callbacks are chances to alter the *response* with Sendmail

The message callback is where we can filter the content within different parts of the email including the header and the body.

Where does MIMEDefang fit in?

- MIMEDefang is implemented in a Sendmail Milter which triggers the MIMEDefang protocol
    - MIMEDefang is linked against libmilter
- The MIMEDefang protocol triggers the mimedefang-multiplexor which then uses mimedefang.pl to filter the message
- Filtering, analysis, and Apache SpamAssassin checks are implemented at the Perl module level, that is, they are implemented in mimedefang.pl

1. An incoming email is passed to Sendmail and accepted, queued and routed as usual

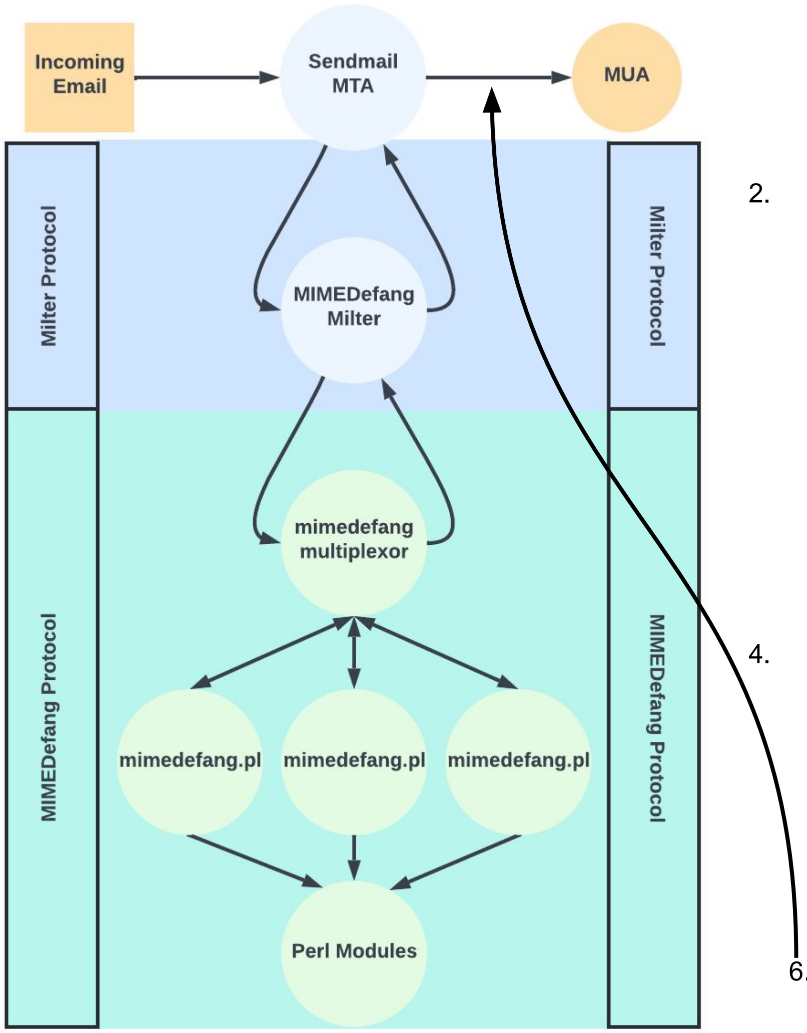
2. The MIMEDefang Milter is called and starts the MIMEDefang process

3. The MIMEDefang multiplexor manages the Perl processes to filter mail and communicates with the MIMEDefang Milter

4. The mimedefang.pl programs contain infrastructure to parse and manipulate MIME messages

5. Perl modules and subroutines filter the data and send data back to the multiplexor which communicates with Sendmail

6. Sendmail routes the email to your MTA



Why use MIMEDefang?

- Filters are in Perl rather than C
    - This is great because text manipulation is **MUCH** easier in Perl than in C
    - You also have access to the **free** Comprehensive Perl Archive Network (CPAN) which has many e-mail manipulation modules
- Filters are single threaded!
    - This mitigates any worries about thread-safety compared to Sendmail
- MIMEDefang has a **great** user base
    - There is a mailing list available here
    - MIMEDefang is maintained by The McGrail Foundation and is remains under active development
- Part of OpenSource.com's "Best Trio of 2017"

How MIMEDefang Works - Filters

- The filter is the policy
- Written in Perl
- Allows you to be a party to the SMTP conversation at various points using the flexibility of Perl
- Also let’s you use the Milter API to change email content

What to do with MIMEDefang

You’re limited only by your imagination. If you can think of it and code it in Perl, you can do it with MIMEDefang.

“

MIMEDefang provides *mechanism*; you provide the *policy*

-Dianne Skoll, Creator of MIMEDefang

What to do with MIMEDefang

People use MIMEDefang to:

- Block viruses
- Block or tag spam
- Remove HTML mail parts
- Add boilerplate disclaimers to outgoing mail
- Remove or alter attachments
- Replace attachments with URL’s
- Implement sophisticated access controls.
- And more! You’re only limited by your imagination when it comes to MIMEDefang!

Getting Started with MIMEDefang

The following software is required for MIMEDefang:

- A UNIX or UNIX-like operating system. MIMEDefang is known to run on Linux, FreeBSD, OpenBSD, NetBSD, Solaris, HP-UX, Tru64 UNIX, and AIX.
- Sendmail 8.12 or newer with Milter support, or Postfix with Milter support.
- Perl 5.8.9 or newer.
- The following Perl modules:
  - MIME::tools 5.420 or higher
  - MIME::Base64 2.11 or higher
  - MailTools 1.1401 or higher
  - Digest::SHA1 2.00 or higher
- (Optional) Other Perl modules like Mail::SpamAssassin
- A C compiler and C development environment

Install Dependencies

```
yum install perl

/usr/bin/perl -MCPAN -e 'install MIME::tools'

/usr/bin/perl -MCPAN -e 'install MIME::tools'

/usr/bin/perl -MCPAN -e 'install MIME::Base64’

/usr/bin/perl -MCPAN -e 'install MailTools’

/usr/bin/perl -MCPAN -e 'install Digest::SHA1’
```

A note on MIME::tools

- MIME::tools is a Perl module that provides **robust** tools for parsing, decoding, and generating MIME messages
- It is one of the best tools in your belt for customizing your MIMEDefang filters and modifying emails
- Learn more about MIME::tools at https://metacpan.org/pod/MIME::Tools

Installing MIMEDefang with EPEL

Step 1: Install EPEL:

```
yum install epel-release
```

Step 2: Refresh the repositories:

```
yum repolist
```

Step 3: Remove Postfix and use Sendmail:

```
yum remove postfix
```

NOTE: MIMEDefang works with Postfix too but requires sendmail-milter for the Sendmail Milter Library. If you'd like to use Postfix please go to step 8a. Otherwise, remove Postfix and install Sendmail.

Step 4: Install Sendmail:

```
yum install sendmail sendmail-milter sendmail-cf
```

Installing MIMEDefang with EPEL

Step 6: Install MIMEDefang:

yum install mimedefang

Step 7: Enable MIMEDefang as a milter with Sendmail:

```
echo "INPUT_MAIL_FILTER(\`mimedefang',
\`S=unix:/var/spool/MIMEDefang/mimedefang.sock, F=T, T=C:15m;S:4m;R:4m;E:10m')" >> /etc/mail/sendmail.mc
```

NOTE: Cut and paste this exactly. The quotes, ticks, and backticks are important and it's all one command line.

Step 8: Compile the sendmail configuration files:

```
make -C /etc/mail
```

Step 8a: if you want to use Postfix instead you should run the following commands:

```
postconf -e “smtpd_milters = unix:/var/spool/MIMEDefang/mimedefang.sock”
```

```
postconf -e “non_smtpd_milters = $smtpd_milters”
```

Installing MIMEDefang with EPEL

Step 9: Change a few default settings in /etc/sysconfig/mimedefang

```
LOG_FILTER_TIME=yes
MX_RELAY_CHECK=yes
MX_HELO_CHECK=yes
MX_SENDER_CHECK=yes
MX_RECIPIENT_CHECK=yes
MX_TICK_REQUEST=60
MX_TICK_PARALLEL=3
```

Installing MIMEDefang with EPEL

Step 10: Enable and Start the MIMEDefang Service:

```
systemctl enable mimedefang
```

```
systemctl start mimedefang
```

Step 11: Enable and Start the Sendmail Service if you decided not to use Postfix:

```
systemctl enable sendmail
```

```
systemctl restart sendmail
```

Step 11a: Enable and Start the Postfix Service if you decided not to use Sendmail:

```
systemctl enable postfix
```

```
systemctl restart postfix
```

Installing MIMEDefang with EPEL

Step 12: Download the KAM Demo filter

NOTE: This demo filter does not change anything in an email, it just provides debug statements so you can watch an email as it goes through the MIMEDefang processing.

```
wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1yu6cnEN_22A07_9ApvBxrvLs4iiAeeW1' -O kamdemo.zip
```

Step 13: Install the mimedefang-filter.kamdemo replacing /etc/mail/mimedefang-filter

Step 14: Test that the filter is syntactically correct:

```
mimedefang.pl -test
```

Step 15: Read the new filter:

```
md-mx-ctrl reread
```

NOTE: If you end up with an SELinux error, try confirming with

```
setenforce 0
```

Step 16: Monitor the mail log:

```
tail -f /var/log/maillog
```

Installing MIMEDefang with EPEL

Step 17: Inject an email manually and watch the mail log:

```
telnet localhost 25

helo test.com
mail from: kevin@test.com
rcpt to: root
data
Subject: This is a test
Date: jasjdkasjdksajd

This is a message
.
```

NOTE: That last line has just a period on it. And if you need:

```
yum install telnet
```

Apache SpamAssassin - Quick Aside

- Apache SpamAssassin is an open-source email filtering software designed to identify and classify spam emails.
  - It is widely used as an email filtering solution for businesses, organizations, and individuals who need to manage large volumes of email.
- It uses a set of rules and algorithms to analyze email content and identify patterns and characteristics that are commonly associated with spam emails. The software then assigns a score to each email based on the results of this analysis. The score helps determine whether an email is likely to be spam or not.
- Some of the key features of Apache SpamAssassin include its ability to recognize and flag phishing attempts, email scams, and malicious content. It also offers support for multiple languages, can be customized using plugins, and offers a flexible rule-writing system.

Apache SpamAssassin - Quick Aside

- When integrated with Apache SpamAssassin, MIMEDefang can use the latter's scoring system to help identify and filter spam emails.
- Apache SpamAssassin has **great** documentation available at https://spamassassin.apache.org/

Installing Apache SpamAssassin and Integrating with MIMEDefang

1. Install Apache SpamAssassin

```
yum install spamassassin
```

2. Once you have installed SpamAssassin, you will need to configure it to work with MIMEDefang. We want to edit the SpamAssassin configuration file, which is located at /etc/spamassassin/local.cf.
   a. Add a spam score threshold to the file, the below threshold is set at 5 - that is, anything scoring 5 or higher will be marked as spam

```
required_score 5.0
```

3. In the MIMEDefang filter file, you may need to add the following line to tell MIMEDefang to use SpamAssassin:

```
$Features{"SpamAssassin"} = 1;
```

Installing Apache SpamAssassin and Integrating with MIMEDefang

4. Restart MIMEDefang with systemd or your local system and service manager

```
systemctl restart mimedefang
```

5. Ta-da! Now MIMEDefang should work with Apache SpamAssassin if enabled in your filter, as the KAM Demo Filter shows.
   a. To test the integration of Apache SpamAssassin and MIMEDefang, you can send a test email to your server and check the headers of the email to see if the X-Spam-Flag header has been added. If the header has been added, it means that SpamAssassin has been successfully integrated with MIMEDefang.

KAM Demo Walkthrough - Setup

```
use strict;

#**************

# Set administrator's e-mail address here.  The
# administrator receives quarantine messages and is listed
# as the contact for site-wide MIMEDefang policy.  A good
# example would be 'defang-admin@mydomain.com'

#*******************************$AdminAddress =
'postmaster@localhost';

$AdminName = "MIMEDefang Administrator's Full Name";
```

KAM Demo Walkthrough - Setup

```
#*****************************************

# Set the e-mail address from which MIMEDefang quarantine
# warnings and user notifications appear to come.  A good
# example would be 'mimedefang@mydomain.com'.  Make sure to
# have an alias for this address if you want replies to it
# to work.

#********************************************

$DaemonAddress = 'mimedefang@localhost';
```

KAM Demo Walkthrough - Setup

```
#********************************************************

# If you set $AddWarningsInline to 1, then MIMEDefang tries
# *very* hard to add warnings directly in the message body
# (text or html) rather than adding a separate "WARNING.TXT"
# MIME part.  If the message has no text or html part, then
# a separate MIME part is still used.

#********************************************************

$AddWarningsInline = 0;
```

KAM Demo Walkthrough - Setup

```
# Set the next one if your mail client cannot handle
# multiple "inline" parts.

$Stupidity{"NoMultipleInlines"} = 0;


# Detect and load Perl modules

#detect_and_load_perl_modules();


#Disable SpamAssassin for the Demo

$Features{"SpamAssassin"} = 0;
```

KAM Demo Walkthrough - Setup

```
# This routine can be used to setup SQL connections, Redis,
# EventReporter, etc.

sub filter_initialize {

  md_syslog('warning', "DEBUG: filter_initialize");


  use POSIX;

}
```

KAM Demo Walkthrough - Setup

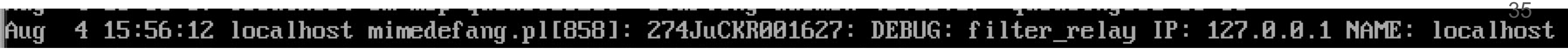
```
# Called when a child process exits to provide clean-up like
# disconnecting from SQL

sub filter_cleanup {

  md_syslog('warning', "DEBUG: filter_cleanup");


  return 0;

}
```

KAM Demo Walkthrough - Setup

```
# Called Periodically by free children - Used for things
# like keepalive on sql connections

sub filter_tick {

  md_syslog('warning', "DEBUG: filter_tick");

}
```

KAM Demo Walkthrough - Setup

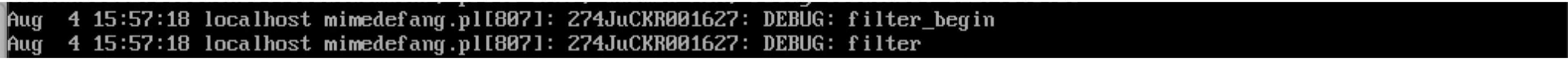
```
#This lets you reject  SMTP  connection  attempts early  on
# in  the  SMTP dialog, rather than waiting until the whole
# message has been sent.

sub filter_relay {

  my ($ip, $name) = @_;

  md_syslog('warning', "DEBUG: filter_relay IP: $ip NAME:
$name");

  return ('CONTINUE', "ok");

}
```



```
Aug  4 15:56:12 localhost mimedefang.pl[858]: 274JuCKR001627: DEBUG: filter_relay IP: 127.0.0.1 NAME: localhost
```

KAM Demo Walkthrough - Setup

```
# Called just before e-mail parts are processed
sub filter_begin {
  my($entity) = @_;


  md_syslog('warning', "DEBUG: filter_begin");
}
```

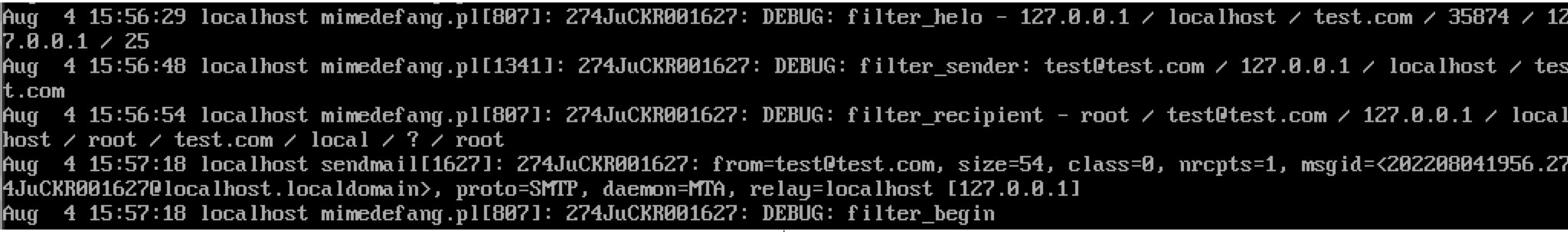


```
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_begin
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter
```

KAM Demo Walkthrough - Setup

```
# This function is called once for each part of a MIME message.There are many
# action_*() routines which can decide the fate of each part; see the
# mimedefang-filter man page.
sub filter {

  my($entity, $fname, $ext, $type) = @_;

  if (&message_rejected()) {

     md_syslog('warning', "DEBUG: filter - message already rejected");
     return; # Avoid unnecessary work

     } else {

     md_syslog('warning', "DEBUG: filter");

     }
  return action_accept();

}
```

KAM Demo Walkthough



```
Aug  4 15:56:29 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_helo - 127.0.0.1 / localhost / test.com / 35874 / 12
7.0.0.1 / 25
Aug  4 15:56:48 localhost mimedefang.pl[1341]: 274JuCKR001627: DEBUG: filter_sender: test@test.com / 127.0.0.1 / localhost / tes
t.com
Aug  4 15:56:54 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_recipient - root / test@test.com / 127.0.0.1 / local
host / root / test.com / local / ? / root
Aug  4 15:57:18 localhost sendmail[1627]: 274JuCKR001627: from=test@test.com, size=54, class=0, nrcpts=1, msgid=<202208041956.27
4JuCKR001627@localhost.localdomain>, proto=SMTP, daemon=MTA, relay=localhost [127.0.0.1]
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_begin
```

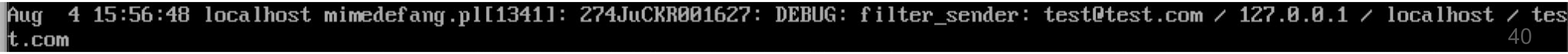
filter_begin is called when the data comes through AFTER filter_helo, filter_sender, and filter_recipient

KAM Demo Walkthrough

```
#  This is called for multipart "container" parts such as message/rfc822.
#  You cannot replace the body (because multipart parts have no body),
#  but you should check for bad filenames.
sub filter_multipart {
  my($entity, $fname, $ext, $type) = @_;

  md_syslog('warning', "DEBUG: filter_multipart");
  return action_accept();
}
```

KAM Demo Walkthrough

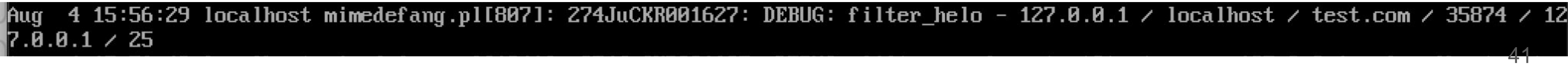
```
#This lets you filter based on the sender of an email

sub filter_sender {

  my ($sender, $ip, $hostname, $helo) = @_;

  md_syslog('warning', "DEBUG: filter_sender: $sender / $ip / $hostname / $helo");

  #if ($sender =~ /drew/i) {

  #  return ('ERROR', "You are not welcome here");

  #}

  return ('CONTINUE', "ok");

}
```



```
Aug  4 15:56:48 localhost mimedefang.pl[1341]: 274JuCKR001627: DEBUG: filter_sender: test@test.com / 127.0.0.1 / localhost / tes
t.com
```

KAM Demo Filter

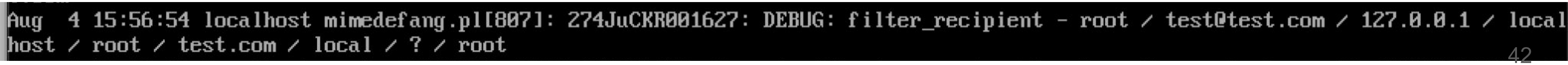
```
# This lets you reject connections after the HELO/EHLO SMTP command.

sub filter_helo {

  my ($ip, $hostname, $helo, $port, $myip, $myport) = @_;

  # $ip and $name are the IP address and name of the sending relay, $helo, is the argument
  # supplied in the HELO/EHLO command.

  # $port, $myip and $myport which contain the client's TCP port, the Sendmail daemon's listening
  # IP address and the Sendmail daemon's listening port.

  md_syslog('warning', "DEBUG: filter_helo - $ip / $hostname / $helo / $port / $myip / $myport");

  return ('CONTINUE', "ok");

}
```



```
Aug  4 15:56:29 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_helo - 127.0.0.1 / localhost / test.com / 35874 / 12
7.0.0.1 / 25
```

KAM Demo Filter

```
# This lets you reject messages to certain recipients, rather than waiting
# until the whole message has been sent.

sub filter_recipient {

  my($recip, $sender, $ip, $host, $first, $helo, $rcpt_mailer, $rcpt_host,
$rcpt_addr) = @_;

  my($answer, $explanation);

  md_syslog('warning', "DEBUG: filter_recipient - $recip / $sender / $ip /
$host / $first / $helo / $rcpt_mailer / $rcpt_host / $rcpt_addr");

  return ('CONTINUE', "ok");  return ('CONTINUE', "ok");

}
```



```
Aug  4 15:56:54 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_recipient - root / test@test.com / 127.0.0.1 / local
host / root / test.com / local / ? / root
```

KAM Demo Filter

```
#  This function customizes the warning message when an attachment
#  is defanged.
sub defang_warning {
  my($oldfname, $fname) = @_;
  return
    "An attachment named '$oldfname' was converted to '$fname'.\n" .
    "To recover the file, right-click on the attachment and Save As\n" .
    "'$oldfname'\n";
}
```

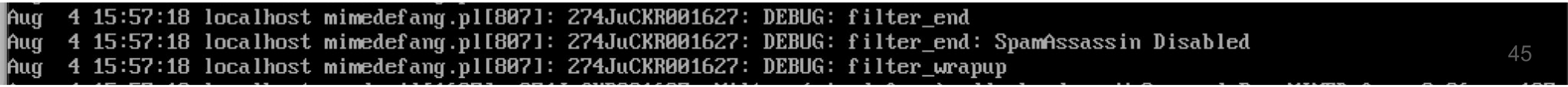
KAM Demo Filter

```
# If SpamAssassin found SPAM, append report.  We do it as a separate
# attachment of type text/plain
sub filter_end {
  my($entity) = @_;
  md_syslog('warning', "DEBUG: filter_end");
  # No sense doing any extra work
  return if message_rejected();
  # Get a RFC 2822 formatted date string for now
  #my ($date_2822);
  #$date_2822 = POSIX::strftime "\%a, \%_d \%b \%Y \%H:\%M:\%S \%Z", localtime(time);
  #action_change_header('Date', $date_2822);
  # Spam checks if SpamAssassin is installed
```

KAM Demo Filter

Only scan messages smaller than 512kB. Larger messages are unlikely to be spam..

```
if ($Features{"SpamAssassin"}) {

   if (-s "./INPUTMSG" < 512*1024) {

   md_syslog('warning', "DEBUG: filter_end: SpamAssassin Started");

   ##### Content Here #####

else {

   md_syslog('warning', "DEBUG: filter_end: SpamAssassin Skipped - Message too large");

   }

} else {

   md_syslog('warning', "DEBUG: filter_end: SpamAssassin Disabled");

}
```



```
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_end
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_end: SpamAssassin Disabled
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_wrapup
```

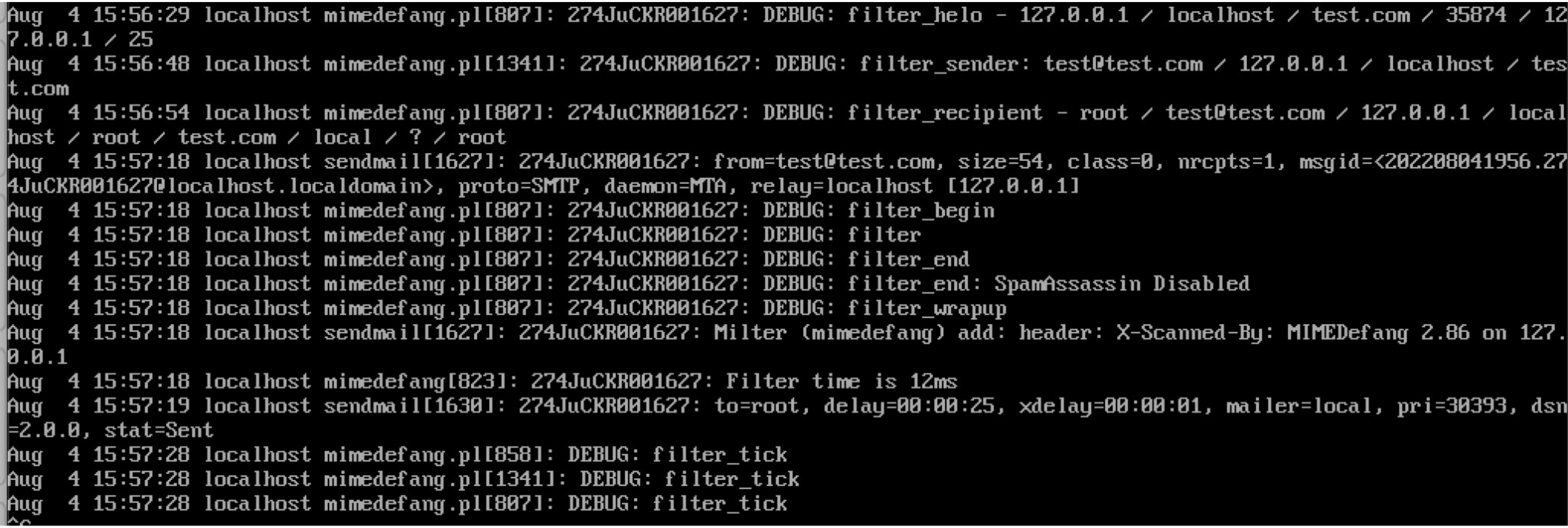
KAM Demo Filter - Apache SpamAssassin Content

```
my($hits, $req, $names, $report) = spam_assassin_check();

my($score);

if ($hits < 40) {

$score = "*" x int($hits);

} else {

$score = "*" x 40;

}

# We add a header which looks like this: X-Spam-Score: 6.8 (******)
# NAME_OF_TEST,NAME_OF_TEST - The number of asterisks in parens is the integer
# part of the spam score clamped to a maximum of 40. MUA filters can easily be
# written to trigger on a minimum number of asterisks...
```

KAM Demo Filter - Apache SpamAssassin Content

```
    # If you find the SA report useful, add it, I guess...
        action_add_part($entity, "text/plain", "-suggest",
"$report\n", "SpamAssassinReport.txt", "inline");
    } else {
        # Delete any existing X-Spam-Score header?
        action_delete_header("X-Spam-Score");
    }
    md_syslog('warning', "DEBUG: filter_end: SpamAssassin Ended");
```

KAM Demo Filter - Putting it All Together



```
Aug  4 15:56:29 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_helo - 127.0.0.1 / localhost / test.com / 35874 / 12
7.0.0.1 / 25
Aug  4 15:56:48 localhost mimedefang.pl[1341]: 274JuCKR001627: DEBUG: filter_sender: test@test.com / 127.0.0.1 / localhost / tes
t.com
Aug  4 15:56:54 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_recipient - root / test@test.com / 127.0.0.1 / local
host / root / test.com / local / ? / root
Aug  4 15:57:18 localhost sendmail[1627]: 274JuCKR001627: from=test@test.com, size=54, class=0, nrcpts=1, msgid=<202208041956.27
4JuCKR001627@localhost.localdomain>, proto=SMTP, daemon=MTA, relay=localhost [127.0.0.1]
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_begin
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_end
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_end: SpamAssassin Disabled
Aug  4 15:57:18 localhost mimedefang.pl[807]: 274JuCKR001627: DEBUG: filter_wrapup
Aug  4 15:57:18 localhost sendmail[1627]: 274JuCKR001627: Milter (mimedefang) add: header: X-Scanned-By: MIMEDefang 2.86 on 127.
0.0.1
Aug  4 15:57:18 localhost mimedefang[823]: 274JuCKR001627: Filter time is 12ms
Aug  4 15:57:19 localhost sendmail[1630]: 274JuCKR001627: to=root, delay=00:00:25, xdelay=00:00:01, mailer=local, pri=30393, dsn
=2.0.0, stat=Sent
Aug  4 15:57:28 localhost mimedefang.pl[858]: DEBUG: filter_tick
Aug  4 15:57:28 localhost mimedefang.pl[1341]: DEBUG: filter_tick
Aug  4 15:57:28 localhost mimedefang.pl[807]: DEBUG: filter_tick
^C
```

KAM Demo Filter - Putting it All Together

- This Demo Filter provides the skeleton for you to be able to customize MIMEDefang to your needs - whether that be blocking emails from certain senders, ensuring that spam does not reach its destination, or scan files to ensure they are not dangerous to your machines
- Try different functions in the filters to test MIMEDefang and get a feel for how to best use it for yourself!