

Writing Applications with Qpsmtpd

Fred Moyer

fred@redhotpenguin.com

Qpsmtpd

- Perl SMTP daemon written by Ask Bjørn Hansen
- “mod_perl” of smtp
- Plugin API makes writing extensions fun
- Works with Qmail, Postfix, and Exim
- Stock plugins include spam and virus protection
- If you've tried to extend a C based smtp daemon you've felt pain at some point

Setting it up

- Read the article by Matt Sergeant [1]
- Install using RPMs if you can
http://wiki.qpsmtpd.org/rpm-install_howto
- Installing from source is easy also
http://wiki.qpsmtpd.org/quick-install_howto
- Telnetting to port 25 works for testing...
- SWAKS is your friend [3]
- qpsmtpd@perl.org is also your friend

My config file

/home/smtpd/qpsmtpd/config/plugins

```
auth/auth_vpopmail_sql # set db auth in the plugin code
denysoft_greylist redhotpenguin.com taperfriendlymusic.org
quit_fortune
check_earlytalker
count_unrecognized_commands 4
check_relay
# require_resolvable_fromhost
# ^^ enable this to prevent zombies
check_delivery normal 1 127.0.0.1 vppmail dbuser dbpass
```

My config file

/home/smtpd/qpsmtpd/config/plugins

rhsbl <- these blacklist plugins blacklist

dnsbl <- yahoo.com et al. sometimes!

check_badmailfrom

check_badrcptto

check_spamhelo

sender_permitted_from

rcpt_ok # this is the last rcpt plugin

My config file

/home/smtpd/qpsmtpd/config/plugins

virus/klez_filter

spamassassin reject_threshold 20 munge_subject_threshold 5

virus/clamav

here is my first qpsmtpd application

craigslist recipient craig_qp@redhotpenguin.com site \

<http://www.craigslist.org> login fred@redhotpenguin.com \

password fredspass

queue/qmail-queue # leave enabled to forward the email

Many plugins available

- <http://wiki.qpsmtpd.org/plugins>
- Spam
- Virus
- Authentication – Vpopmail, PAM, text file, LDAP...
- Plugins are generally geared towards keeping the spam out, and letting the good mail through

Your first plugin

- Start with good examples
- Read Matt Sergeant's O'reilly article [1]
- Read the source for existing plugins
- `perldoc README.plugins ;)`

A simple subject filter

- Spam filters don't catch everything

```
block_subject [ all | [ domain1, domain2 ... ] ]
```

```
sub init {
```

```
  my ($self, $qp, @domains) = @_;
```

```
  unless (defined $domains[0]) {
```

```
    die "block_subject needs proper config\n";
```

```
  }
```

```
  $self->{_args}->{domains} = \@domains;
```

```
  my $content = <<"SUBJECTS";
```

```
YOUR REPRESENTATIVE ASSISTANCE IS NEEDED
```

```
Message From eBay Member
```

```
The Ultimate Online Pharmaceutical
```

```
SUBJECTS
```

A simple subject filter

```
my @subjects = split("\n", $block_content);  
$self->{_args}->{subjects} = \@subjects;
```

```
$self->log(LOGDEBUG,  
  "Blocking subjects $block_content");  
}
```

```
sub hook_data_post {  
  my ($self, $transaction) = @_;  
  if ($self->{_args}->{domains}->[0] eq 'all') {  
    if ($self->_blocked_subject($transaction)) {  
      return DENY;  
    }  
  }  
  else {
```

A simple subject filter

```
# Check each domain
```

```
my @recip_domains = map { $_->host }
```

```
  $transaction->recipients;
```

```
foreach my $domain (@recip_domains) {
```

```
  # Deny the email if the subject is in our blacklist
```

```
  if (grep { $_ =~ m/^[extract_itex]domain[/extract_itex]/ } @{$self->{_args}->{domains}}  
      && $self->blocked_subject($transaction))
```

```
  {
```

```
    return DENY;
```

```
  }
```

```
}
```

```
}
```

```
return DECLINED;
```

```
}
```

A simple subject filter

```
sub blocked_subject {
  my ($self, $transaction) = @_;
  my $subject = $transaction->header->get('Subject');
  chomp($subject);

  if (grep { $_ =~ m/$subject$/i } @{$self->{_args}->{subjects}})
  {
    $self->log(LOGINFO, "block_subject invoked for $subject");
    return 1;
  }
  return;
}
```

SMTP Applications?

- SMTP to HTTP Gateways
- Automate tedious processes
- Use emails to interact with applications
- Today's PDAs have IMAP based email but aren't very good at serving web pages

A Practical Application

- The problem at hand
 - A couple hundred things we couldn't sell
 - Throwing away is bad for the environment (and costs money)
 - <http://www.craigslist.org> - someone wants it
- But taking a picture of everything and posting an ad to Craigslist is a lot of clicking
- I don't like to click, I like to write Perl

A Practical Application

- Possible solutions
 - No internet connection
 - I could make a list of the items at hand
 - Call a friend and have him post
(no images though, and one less friend)
 - Or drive to local shop with wireless
and click away

A Practical Application

- My phone has a built in camera and email client
- I could take pictures and email them to myself with a description, and post from home
- That's repeating myself though, and too much work

A Practical Application

- A Qpsmtpd plugin to solve this problem?
- It came down to being Lazy
- Being able to post an item at will is preferable to batching transactions
- Bonus points for a per item picture, response rates for items with pictures are much higher than without

A Waste of Time?

- At 3 minutes overhead per item the transaction overhead is 7.5 hours for 150 items
- At 30 seconds overhead per item the transaction overhead is 75 minutes for 150 items
- Total time spent thinking, coding, testing and deploying was about 3 hours
- Bonus points for writing fun code!

The Source

- Initialization

```
use WWW::Mechanize;  
use Data::Dumper;
```

```
sub init {  
    my ($self, $qp, %args) = @_;  
    $self->{_args} = \%args;  
    $self->{_mech} = WWW::Mechanize->new;  
    $self->log(LOGDEBUG, "Craigslit plugin init: \n" .  
        Dumper(\%args));  
}
```

The Source

- Start processing after all data is received
- Check authorization first

```
sub hook_data_post {  
  my ($self, $transaction) = @_;
```

```
  return DECLINED unless  
    (($transaction->recipients->[0]->address eq  
     $self->{_args}->{recipient})  
     && $self->qp->connection->relay_client);
```

The Source

- Get the subject and body

```
my $subject = $transaction->header->get('Subject');
my $body;
while ( my $line = $transaction->body_getline ) {
    $body .= $line;
}
use Email::MIME;
my $em = Email::MIME->new($subject . $body);
```

The Source

- Get the attached images and save them

```
use Email::MIME::Attachment::Stripper;
my $stripper =
  Email::MIME::Attachment::Stripper->new($em);
my $attachments = $stripper->attachments;
if (@attachments) {
  foreach my $at (@attachments) {
    my $fh;
    open($fh, ">", "/tmp/" . $at->{filename}) || die $!;
    print $fh $at->{payload};
    close($fh);
  }
}
```

The Source

- Login to <http://www.craigslist.org>

```
$self->{_mech}->get($self->{_args}->{site});
$self->{_mech}->submit_form(
  form_name => "login",
  fields => {
    inputEmailHandle => $self->{_args}->{login},
    inputPassword    => $self->{_args}->{password}
  });
if (! $self->{_mech}->success ) {
  $transaction->body_write("Error during login");
  return DECLINED;
}
```

The Source

- Login to www.craigslist.org

```
$self->{_mech}->get($self->{_args}->{site});
$self->{_mech}->submit_form(
  form_name => "login",
  fields => {
    inputEmailHandle => $self->{_args}->{login},
    inputPassword    => $self->{_args}->{password}
  });
if (! $self->{_mech}->success ) {
  $transaction->body_write("Error during login");
  return DECLINED;
}
```

The Source

- Walk the website

```
$self->{_mech}->follow_link(  
  text_regex => qr/wanted/ );  
if (! $self->{_mech}->success ) {  
  $transaction->body_write("Error following wanted link");  
  return DECLINED;  
}
```

... click through a few more links to get to ad posting

The Source

- Post the ad

```
$self->{_mech}->submit_form(  
  form_number => 1,  
  button => 'imagesForm',  
  fields => {  
    PostingTitle => $subject,  
    PostingBody => $body,  
    Ask          => 0,  
  });  
if (! $self->{_mech}->success) {  
  $transaction->body_write("Error, could not post ad");  
  return DECLINED;  
}
```

The Source

- Post the images

```
foreach my $at (@attachments) {
    $self->{_mech}->submit_form(
        form_number => 1,
        fields => { file1 => '/tmp/' . $at->{filename} }
    );
    if (! $self->{_mech}->success) {
        $transaction->write_body(
            "Error adding image " . $at->{filename});
        return DECLINED;
    }
    unlink('/tmp/' . $at->{filename});
}
```

We're done

- Grab the PDA
- Take a picture
- Compose an email
- Receive a flood of responses
(still need to write that part of the app)

Perl did the work

- No clicking (except taking the photo)
- No dealing with a cumbersome web based application
- Our energy goes to where it's needed most
- All we had to do was code up a defined process

Recommended API Practices

- Run your application last after all other plugins, before the queue plugin
- Forward the email to the sender as a receipt
- Use a separate To: address for each app, one that isn't vulnerable to dictionary attacks

Giving feedback to the User

- Indicate success or failure
- Let them know why it failed -
`$transaction->write_body("Why it failed");`
- Be strict in what you accept – the spammers will find your openings eventually

Recommended AAA Approaches

- Require users to authenticate via SMTP auth
 - as secure as normal SMTP transactions

```
$self->qp->connection->relay_client == 1
```
- Combine username with To: address and match that against sender address, fairly secure but not unbreakable
 - To: fred_craigslis@domain.com,
 - From: fred@fredsdomain.com
- Greylisting can help here but remember that zombies can retry with dictionary attacks

Recommended AAA Approaches

- Require the user register their email address and verify they are a real person
- Don't use a password in the email - most users have one password and will blissfully use it here as well
- Use good judgement in weighing ease of use vs. security in authentication – err on the side of security.

Recommended AAA Approaches

- Don't use this approach to build secure apps unless you really know what you are doing. Better yet, just don't do it.
- Don't send sensitive information in plain text email
- This approach is about making automated tasks easy from insecure clients (PDAs). Proceed as such.

More Ideas

- Reply to those annoying “DO NOT REPLY” emails from Bugzilla, et al.
- Check the weather
- Check the traffic report
- Next bus
- Post to a picture blog from your PDA
- Any process that is repeatable and boring

Credits

- The Qpsmtpd community
- Ask Bjørn Hansen for authoring Qpsmtpd
- Matt Sergeant and Robert Spier for listening to my crazy ideas on #qpsmtpd

References

[1] – Using Qpsmtpd, Matt Sergeant

<http://www.oreillynet.com/pub/a/sysadmin/2005/09/15/qpsmtpd.html>

[2] – The Qpsmtpd Wiki

<http://wiki.qpsmtpd.org>

[3] – SWAKS

<http://jetmore.org/john/code/#swaks>

Slides

These slides are freely available at

<http://www.redhotpenguin.com/talks>

Thank you NPW 2006!