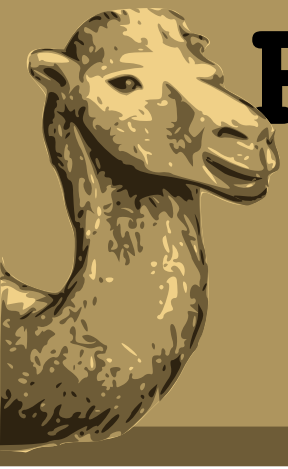
Perl Mongers im Ruhrgebiet



Ruhr.pm

qpsmtpd
Modifizierbarer SMTP-Daemon in Perl

Autor: Michael Holzt

EMail: holzt @ linuxmanufaktur.de

Datum: 13. Mai 2008

http://ruhr.pm.org/

Template Version 0.2

The use of a camel image in association with Perl is a trademark of O'Reilly Media, Inc. Used with permission.


Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-nc-nd/3.0/

Dieses Dokument wird unter den Bedingungen der CreativeCommons Lizenz des Types CC-BY-NC-ND 3.0 zur Verfügung gestellt. Es gelten die Bedingungen des vollständigen Lizenztextes in der englischen Originalfassung der unter der genannten URL zu finden ist.

Die Bedingungen in unverbindlicher Kurzform:

- Es ist erlaubt das Dokument für private Zwecke zu verbreiten und zu kopieren.
- Eine kommerzielle Nutzung des Dokuments wird ausgeschlossen.
- Eine Bearbeitung und Veränderung des Dokuments ist untersagt.
- Bei Verbreitung ist der Autor des Dokuments zu nennen. Es darf dabei nicht der Eindruck entstehen, Sie oder der Autor würden für die Verbreitung entlohnt.
- Sie müssen bei Verbreitung anderen die zugrundeliegende Lizenz mitteilen.

Ruhr.pm

Vorstellung

- Michael 'kju' Holzt, Gelsenkirchen
- Linux seit 1996 (Erstkontakt 1994), Perl seit etwa 1997
- Entwickler beim Debian/GNU Linux Projekt, Eigene Projekte: gwhois, cramfsswap, Interessenschwerpunkte Netze und Embedded Linux
- Selbständig (Die Linux Manufaktur)

Ruhr.pm

Gliederung des Vortrags

1. Einführung – Was ist qpsmtpd, wie ist es entstanden, wer verwendet ist?
2. Nutzung – Was benötigt qpsmtpd, wie wird es installiert und konfiguriert?
3. Anpassung – Wie flexibel ist qpsmtpd, wie schwierig ist die Anpassung an meine eigenen Vorstellungen und Ideen?

Einführung

Was ist qpsmtpd, wie ist es entstanden
und wer nutzt es?

Ruhr.pm

Einführung: Was ist qpsmtpd?

- Ursprünglich: “qmail perl smtp daemon”
  - SMTP-Daemon, ursprünglich Ersatz für qmail-smtpd
  - Wie der Name sagt: In Perl geschrieben
  - Weiterverarbeitung durch MTA, z.B. qmail, Postfix, exim, ... also nicht mehr auf qmail beschränkt
  - Namensänderung vorgeschlagen aber bislang kein Konsens
  - MIT-Lizenz (entspricht der 2 Clause BSD License)

Ruhr.pm

Einführung: Was ist qpsmtpd? (2)

- SMTP-Kern mit Plugins
  - 31 Hooks für in Perl geschriebene Plugins
  - Dabei sind 61 Plugins, weitere gibts im Netz
    - Filter: 10 Virenscanner, RBL, Greylisting, SPF, DKIM, Spamassassin, Earlytalker, Pflichtheader; Filterung auf SMTP-Ebene erspart Bounces und ist rechtlich besser
    - Authentifizierung, Identifizierung (GeoIP, p0f), Logging, verschiedene Queues, sogar TLS ist ein Plugin
  - Eigene Plugins sind in Windeseile geschrieben, dabei natürlich auch Nutzung von CPAN möglich

Ruhr.pm

Einführung: Geschichte, Nutzer

- Entwickler: Ask Bjoern Hansen
  - Administrator von perl.org
  - vermisste Features in qmail-smtpd, schrieb erste Version in 2001
  - Modularität seit Version 0.10 in 2002
  - Aktuell: Version 0.40 (Juni 2007), aber lange stabil
- Große und kleine Nutzer
  - läuft z.B. auf perl.org (klar!) und auch auf cpan.org, apache.org, lists.mysql.com, ...

Nutzung

Was benötigt qpsmtpd, wie wird es installiert und konfiguriert?

Ruhr.pm

Nutzung: Was benötigt qpsmtpd?

- Perl
- CPAN-Module
  - Net::DNS
  - MIME::Base64
  - Mail::Header
- Bei Perl-Versionen vor 5.8 zusätzlich:
  - Data::Dumper
  - File::Temp
  - Time::HiRes

Ruhr.pm

Nutzung: Installation

- adduser smtpd
- mkdir /home/smtpd/qpsmtpd
- cd /home/smtpd/qpsmtpd
- tar zxvf ~/qpsmtpd-0.40.tar.gz

- Natürlich ist auch ein anderer Benutzername und ein anderer Pfad möglich

Ruhr.pm

Nutzung: Installation (2)

- Verschiedene Wege qpsmtpd zu starten
  - qpsmtpd – über inetd, tcpserver etc.
  - qpsmtpd-forkserver – je Verbindung ein neuer Prozess
  - qpsmtpd-prefork – Prozesse auf Reserve halten
  - qpsmtpd-server – Server auf select-Basis
  - qpsmtpd-async – Asynchron auf Basis von Danga
  - Apache::Qpsmtpd – über Apache und mod-perl2

Ruhr.pm

Nutzung: Konfiguration

- In '/var/qmail/control' sowie '<qpsmtpd>/config'.
- Notwendig: 'plugins'
  - Listet zu ladenden Plugins, u.U. auch Parameter (häufiger: eigene Configs)
  - Werden entsprechend der Reihenfolge in der Config geladen und in die Hooks gehängt
  - Mehrfaches Laden mit verschiedenen Parametern möglich: mein_plugin, mein_plugin:0 foo, mein_plugin:1 bar
- Unter Umständen nötig: 'spool_dir'

Ruhr.pm

Nutzung: Konfiguration (2)

- 'plugins'-Beispiel:

```
logging/file loglevel LOGINFO /var/log/qpsmtpd.log
rcpt_ok
queue/qmail_queue
```

- Variationen:

```
logging/syslog loglevel LOGINFO priority LOG_NOTICE
queue/postfix_queue [socketpfad]
queue/exim-bsmtp [exim-rsmtp binary]
```

Anpassung

Wie flexibel ist qpsmtpd, wie schwierig ist die Anpassung an meine eigenen Vorstellungen und Ideen?

Ruhr.pm

Anpassung: Hooks

- qpsmtpd kennt derzeit 31 verschiedene Hooks:
  - logging
  - config
  - pre-connection connect ehlo_parse ehlo helo_parse helo auth_parse auth auth-plain auth-login auth-cram-md5 rcpt_parse rcpt_pre rcpt mail_parse mail mail_pre data data_post queue_pre queue queue_post quit reset_transaction disconnect post-connection
  - unrecognized_command deny ok received_line

Ruhr.pm

Anpassung: Pluginstruktur

- Grundstruktur:

```
sub hook_<hookname> {
  my ($self, $transaction) = @_;

  [...]
  return(DECLINED);
  return(OK, “Accepted”);
  return(DENY, “Not allowed”);
  return(DENYSOFT, “Currently not available”);
}
```

Ruhr.pm

Anpassung: Rückgabewerte

- DECLINED – weiter mit nächstem Plugin
- OK – akzeptiert
- DENY – permanenter Fehler
- DENYSOFT – temporärer Fehler
- DENY_DISCONNECT,
  DENYSOFT_DISCONNECT
- DONE – bereits Rückmeldung ausgegeben

Ruhr.pm

Anpassung: Mein erstes Plugin

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

  if ( lc($rcpt) eq '<hans@meine.do.main>' ) {
    return(DENY, “Mail to $rcpt not accepted here”);
  }

  return(DECLINED);
}
```

Ruhr.pm

Anpassung: Headerprüfung

```
sub hook_data_post {

  my ($self, $transaction) = @_;

  return(DENY, “Kein Mailinhalt”) if $transaction->data_size
   == 0;

  return(DENY, “Kein Absender”) if ! $transaction->header
   ->get('From');

  my $date = $transaction->header->get('Date');

  return(DENY, “Kein Datum”) if ! $date;

  return(DENY, “Mail zu alt”) if ! &check_date($date);


  return(DECLINED);

}
```

Ruhr.pm

Anpassung: Eigene Meldung

```
sub hook_quit {
  my $qp = shift->qp;
  my $fortune = '/usr/games/fortune';
  return DECLINED unless -e $fortune;

  my @fortune = `$fortune -s`;
  @fortune = map { chop; s/^/ \/ /; $_ } @fortune;

  $qp->respond(221, $qp->config('me')
  . " closing connection.", @fortune);
  return DONE;
}
```

Ruhr.pm

Anpassung: check_earlytalker

```
sub register {
  my ($self, $qp) = @_;
  $self->register_hook('connect', 'connect_handler');
}

sub connect_handler {
  my ($self, $transaction) = @_;
  my $in = new IO::Select;
  $in->add(\*STDIN) || return(DECLINED);

(Fortsetzung ...)
```

Ruhr.pm

Anpassung: check_earlytalker (2)

```
  if ($in->can_read(1)) {
    $self->log(LOGDEBUG, “Client sprach vor uns!”);
    return(DENYSOFT, “SMTP-Protokollverletzung”);
  }

  return(DECLINED);
}
```

Ruhr.pm

Anpassung: Nervige Werbung

```
my $footer;


sub register {
  my ($self, $qp) = @_;
  @now = localtime();
  $footer = "\nSupermail, Mail im Jahr " . ($now[5]+1900) .
   "\n";


  $self->register_hook('data_post', 'add_footer');
}
```

Ruhr.pm

Anpassung: Nervige Werbung (2)

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

  if ( $self->qp->connection->remote_ip eq '127.0.0.1') {
    $transaction->body_write( $footer );

  return(DECLINED);
}
```

Ruhr.pm

Anpassung: Queuing (für arme)

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

  open(my $mta, "|/opt/mta/inject");
  print {$mta} $transaction->header->as_string, "\n";
  print {$mta} $transaction->body_as_string;
  close($mta);

  return(OK, "Queued!");
}
```

Ruhr.pm

Das wars!

- http://www.qpsmtpd.org
- http://wiki.qpsmtpd.org
- Fragen?
- Danke!