Exim - MTA-Framework oder MTA?

Chemnitzer Linuxtage 2015

Heiko Schlittermann

schlittermann - internet & unix support, Dresden

21. März 2015

Inhalt

Exim

Entwicklung, Verbreitung

- **Ex**perimental **I**nternet **M**ailer
- seit 1995 Phil Hazel, seit ca. 2007 ca. 5. . . 8 aktive Entwickler
- aktuell stabil 4.85 von Januar 2015
- Releases ca. 1. . . 2x Jahr
- 12/2010 - großes Sicherheitsproblem für $\leq$ 4.72
- seit 2010 ca. 10 CVEs
- keine genauen Zahlen über Verbreitung
- Default MTA bei Debian, Appliances, Internetdienstleister

Exim

Position

Ja, das ist Religionskrieg :-)

- Lego vs. Playmobil (P. Heinlein)
- Anpassbarkeit
  - keine Annahmen über die Art der zu lösenden Herausforderungen
  - keine Annahmen über die Art Problemlösung
  - Bereitstellung von Werkzeugen
  - Router, Transports, ACL gleichen Funktionsblöcken
  - Intensive Expansion von Variablen zur Laufzeit
- gut verstandene Prozessstruktur $\rightarrow$ Stabilität
- extrem gutes Logging $\rightarrow$ Sicherheit
- exzellente Dokumentation (Referenz-Handbuch mit Beispielen: spec.txt (33k L), spec.pdf (ca. 500 Seiten))
- sehr gutes Debugging der Konfiguration möglich
- vorbildlichster Quelltext (C, kommentiert)
- sehr hilfreiche Community: <exim-users@exim.org>

Arbeitsweise und Anatomie

Überblick

- Binary ist ein ca 1 MB großer Universalklumpen
- Einfache Struktur der operativen Daten - 2 Text-Files je Message + ggf. 1 Messagelog, Spoolhierarchie in 16 Verzeichnissen
- Keine aufwändigen IPC - nichts, außer fork(2) oder exec(3)
- Wenig gemeinsam genutzte Daten - nur „Hint“-Files
- Ohoh - setuid 0!

Arbeitsweise und Anatomie

IN, OUT, Retry

Es gibt im wesentlichen 3 Phasen der Verarbeitung. In Bezug auf die Konfiguration sind das

1. Empfang

authenticators	Eventuell SMTP-Authentifizierung
acl	mit Ratelimit, Blacklists, Adressüberprüfungen, Inhaltsüberprüfung

2. Start des Sendeprozesses

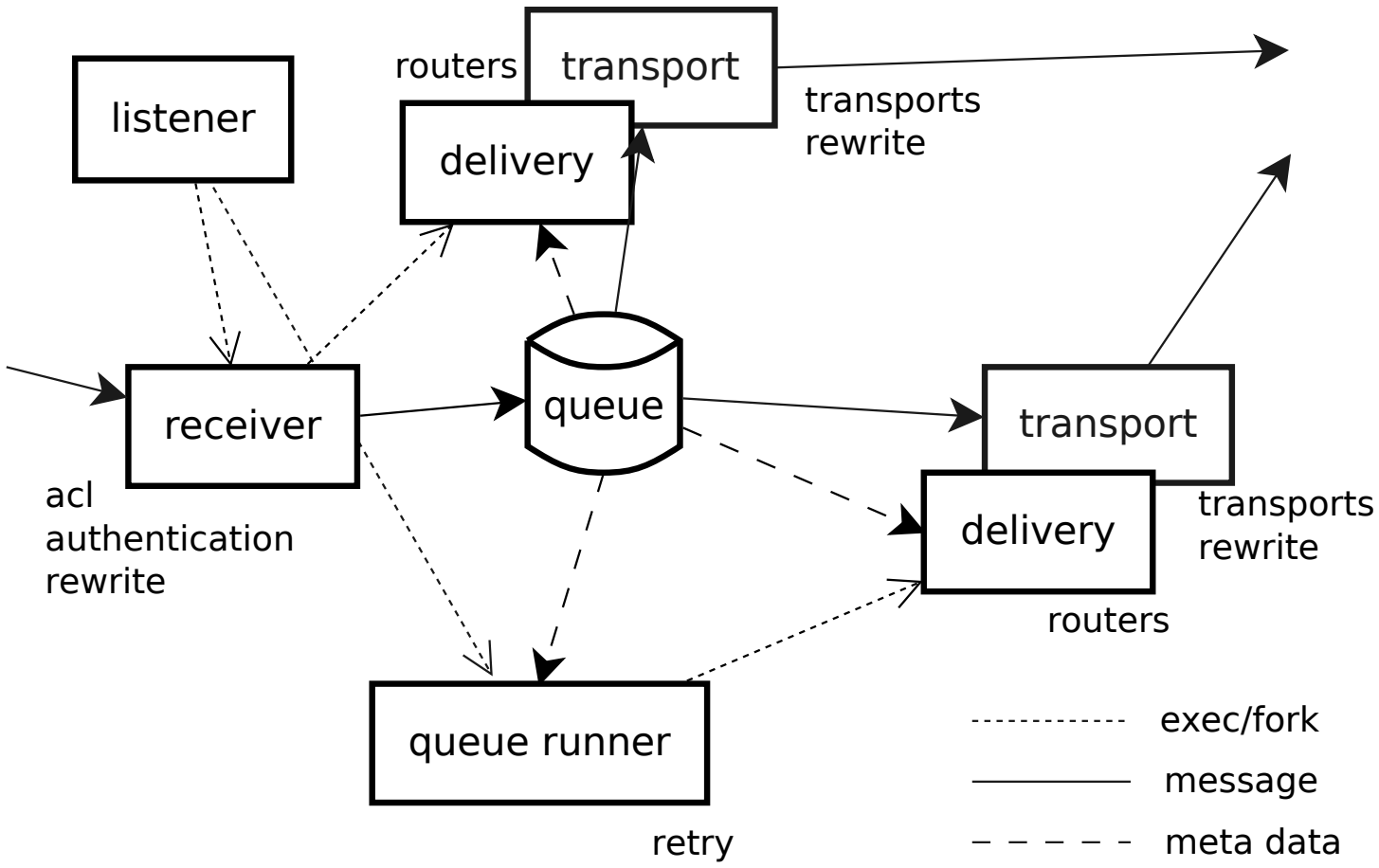
routers	Ermittlung Transportweg und -mechanismus
transports	Konfiguration der Transportmechanismen

3. Queuerunner startet ggf. weitere Versuche

retry	Wiederholungsregeln

Arbeitsweise und Anatomie

Prozesse und Informationsfluss



Konfiguration

File

- Debian geht einen sonder(baren) Weg
- Beispiel-Konfig `example.conf.gz` als Ausgangspunkt
- `exim -bV` listet die verwendete Konfigurationsdatei und einkompilierte Features

```
Exim version 4.85_a466d09-XX #21 built 18-Mar-2015 15:36:46
Copyright (c) University of Cambridge, 1995 - 2014
(c) The Exim Maintainers and contributors in ACKNOWLEDGMENTS file, 2007 - 2014
Berkeley DB: Berkeley DB 5.3.28: (September  9, 2013)
Support for: crypteq iconv() IPv6 OpenSSL Content_Scanning
    DKIM Old_Demime
    PRDR OCSP Experimental_DANE Experimental_Event
Lookups (built-in): lsearch wildlsearch nwildlsearch iplsearch
    cdb dbm dbmjz dbmnz dnsdb dsearch
    ldap ldapdn ldapm
    nis nis0 nisplus passwd
    pgsql
Authenticators: cram_md5 dovecot plaintext spa
Routers: accept dnslookup ipliteral manualroute queryprogram redirect
Transports: appendfile/maildir/mailstore/mbx autoreply lmtp pipe smtp
Fixed never_users: 0
Size of off_t: 8
Configuration file is /usr/local/exim/etc/exim.conf
```

Konfiguration

Struktur

Strukturiertes Konfigurationsfile mit mehreren Abschnitten, teilweise miteinander verlinkt (Router referenziert Tranports, globaler Teil referenziert ACL, ACL nutzt Router)

```
...
begin acl
...
begin routers
...
```

(global)	etwa 250 allgemeine Direktiven
acl	Access Control Lists für SMTP
routers	Routing-Regeln (genutzt auch von ACL)
transports	Transport-Mechanismen
retry	Regeln für Wiederholungsversuche
rewrite	Adress-Manipulation in Envelope und Header
authenticators	SMTP-Authentifizierung

Macros, Kommentar, lange Zeilen

```
# Kommentar sind Zeichen für schlechte Konfiguration :)
USER_BASE = ou=users,BASE
BASE = dc=example,dc=com
received_header_text = Received: ${if def:sender_rcvhost \
  from $sender_rcvhost\n\t}{${if def:sender_ident {from \
  ...
  def:received_for {\n\tfor $received_for}}
```

Der Rest ist einfach :)

```
primary_hostname = foo.example.com
```

. . . solange keine $-Zeichen dazu kommen. Dann haben wir Expansion in allen Spielarten.

Konfiguration

Expansion

Etwa die Hälfte der Konfigurationsdirektiven erlaubt Variablensubstitution (Expansion) zur Laufzeit.

```
message_size_limit = ${if =={$received_port}{587} {200M}{50M}}
headers_add = X-Authenticated: ${if def:authenticated_id \
            {$authenticated_id}{unknown}}
```

Variablen	`$local_part`, `${local_part}`
Operatoren	`${md5:$local_part}`, `${uc:$domain}`
Funktionen	`${sg{$local_part}{.laus}{XXX}}`
Bedingungen	`${if eq{$local_part}{x}{~/mbox}{~/.mail}}`
Lookup (Key)	`${lookup{$local_part}lsearch{/etc/aliases}}`
Lookup (Query)	`${lookup psql{SELECT ...}}`

String-Expansion

Wenn nichts mehr geht

Als letzte Hilfe gibt es die Möglichkeit, Sockets auszulesen, Perl zu integrieren, oder externe Kommandos aufzurufen

```
${readsocket{<socket>}{<request>}}
${run{<command>[<arg>]...}}
${perl{<sub>}[{<arg>}...]}
```

Beispiel: Greylisting

```
# source: http://schlittermann.de/doc/grey
GREYKEY = $sender_address/$local_part@$domain
# sub unseen() from perl script
perl_startup = do '/etc/exim4/exim-exigrey.pm'
acl_smtp_rcpt = acl_check_rcpt
...
begin acl
  cl_check_rcpt:
       ...
       defer condition = ${perl{unseen}{GREYKEY}{1d}}
       ...
```

Routing

Allgemein

- Routing - zentrale Rolle
- Routing bereits in den ACL für Adresstests
- Konfiguration enthält eine Kette von Router-Blöcken
- Vorbedingungen entscheiden, ob ein Block aktiv wird
- Routerblock liefert für eine gegebene Adresse:

accept	Zuordnung zu Transport oder Erzeugung neuer Adressen
decline	Verweigerung
fail	Bounce wird generiert
defer	falscher Augenblick

- Routerblock

```
remote:
  driver = dnslookup
  domains = ! +local_domains
  transport = remote_smtp
  ignore_target_hosts = 127.0.0.0/8
```

Routing

Treiber

Treiber legt das Verhalten des Routers fest, alle Treiber sind parametrisierbar (ca. 40 allgemeine Optionen, *dnslookup* ca. 15 spezifische Optionen)

dnslookup	Klassiker - MX, A/AAAA
manualroute	Tabelle `<domain> <next hops>`
queryprogram	Routing-Info über externes Programm
redirect	neue Adressen werden generiert
accept	Name ist Programm :)

Routing

Test

```
$ exim -bt hans@example.com
hans@example.com
  router = remote, transport = remote_smtp
  host example.com [2606:2800:220:6d:26bf:1447:1097:aa7]
  host example.com [93.184.216.119]
```

```
$ exim -d-all+dns+route -bt hans@example.com
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
routing hans@example.com
--------> remote router <--------
local_part=hans domain=example.com
checking domains
calling remote router for hans@example.com
  domain = example.com
DNS lookup of example.com (MX) gave NO_DATA
returning DNS_NODATA
DNS lookup of example.com (AAAA) succeeded
DNS lookup of example.com (A) succeeded
set transport remote_smtp
queued for remote_smtp transport: local_part = hans
routed by remote router
  envelope to: hans@example.com
  transport: remote_smtp
  host example.com [2606:2800:220:6d:26bf:1447:1097:aa7]
  host example.com [93.184.216.119]
>>>>>>>>>>>>>>>> Exim pid=24870 terminating with rc=0 >>>>>>>>>>>>>>>>
```

Transports

- Router referenzieren ggf. einen Transport-Block, der mindestens einen Treiber und ggf. noch weitere Optionen spezifiziert.

```
remote_smtp:
  driver = smtp

local_mailbox:
  driver = appendfile
  file = /var/mail/$local_part
  group = mail
```

- Treiber

smtp	SMTP, TLS, LMTP
appendfile	Mailbox, Maildir
pipe	Kommando-Pipeline (z.B. UUCP)

Beispiel: DANE für Arme

Aufgabenstellung

Es existiere ein JSON-File (`mxinfra.json`), in dem je MX-Host die SSL-Zertifikatsinformation liegt. Nun soll Exim, wenn er sich mit einem dieser Hosts verbindet, prüfen, ob das korrekte Zertifikat präsentiert wird.

Lösung

- Perl-Script generiert aus dem mxinfra-File eine Ordnerstruktur mit Zertifikaten `emig.d/certs/<hostname>`
- Transport prüft das Zertifikat zum aktuellen Ziel-Host

```
begin transports

  remote_smtp:
    driver = smtp
    hosts_require_tls = dsearch;/etc/exim4/emig.d/certs
    tls_verify_certificates = /etc/exim4/emig.d/certs/$host
```

- Bitte? Ja, ich glaube, das ist Very Poor Man's DANE.

Beispiel: Source based Routing

Vorversuche

Aufgabe

Wir haben mehrere Smarthosts und müssen je nach Sender-Adresse über einen anderen Smarthost versenden.

```
# file: smarthosts
# sender address       |submission-server[::port]|user-name              |password
# or *@domain, or *    |             default 587 |                        |
#----------------------+-------------------------+------------------------+----------
hans@example.com        mx.freenet.de::25         hans@example.com         xxx
*@example.com           ssl.schlittermann.de      heiko@schlittermann.de   Gheim
*                       smtp.km21.com             km433221                 zecrit
```

Lösung

Wir müssen beim Routing die Sender-Adresse als Kriterium verwenden, nicht die Zieladresse!

```
$ exim -be
> ${lookup{foo@example.org}lsearch*@{smarthosts}{$value}}
smtp.km21.com              km433221                  zecrit
> ${sg{smtp.km21.com              km433221 zecrit}{\\s+}{\t}}
smtp.km21.com km433221 secrit
> ${extract{1}{\t}{smtp.km21.com km433221 zecrit}}
smtp.km21.com
```

Beispiel: Source based Routing

Macros

Das kann jetzt schön in Macros verpackt werden, damit es übersichtlich wird:

```
ADDRESS_DATA = ${lookup{foo@example.org}\
                        lsearch*@{smarthosts}\
                        {${sg{$value}{\\s+}{\t}}}}
SMARTHOST    = ${extract{1}{\t}{$address_data}}
USER         = ${extract{2}{\t}{$address_data}}
PASS         = ${extract{3}{\t}{$address_data}}
```

Beispiel: Source based Routing

Routers + Transports

```
begin routers

  smarthosts:
    driver = manualroute
    address_data = ADDRESS_DATA
    route_data = SMARTHOST
    transport = smtpa
    no_more

begin transports

  smtpa:
    driver = smtp
    port = submission
    hosts_require_auth = *

begin authenticators

  plain:
    driver = plaintext
    public_name = PLAIN
    client_send = ^USER^PASS
```

Beispiel

Beispiel: Source based Routing

Vorversuche Das Routing können wir wieder relativ einfach testen:

```
$ exim -f hans@example.com -t nobody@nowhere
nobody@nowhere
  router = smarthosts, transport = smtpa
  host mx.freenet.de [2001:748:100:40::8:112]  port=25
  host mx.freenet.de [195.4.92.212]            port=25

$ exim -f fred@example.com -t ...
nobody@nowhere
  router = smarthosts, transport = smtpa
  host ssl.schlittermann.de [212.80.235.130]

$ exim -f fred@foobar.com -t ...
nobody@nowhere
  router = smarthosts, transport = smtpa
  host smtp.km21.com [54.209.129.218]
```

Und natürlich haben wir die ganzen Debug-Optionen noch, für Expansion, DNS, ...

Acess Control Lists

Für jede Phase der SMTP-Kommunikation gibt es einen ACL-Block mit Regeln. Abarbeitung der Regeln erfolgt bis zur Entscheidung.

accept	alles gut, weitermachen
deny	permanenter Fehler
require	„deny“ oder weitermachen
defer	temporärer Fehler

Die Zuordnung der SMTP-Phasen zu den ACL-Blöcken ist frei.

```
acl_smtp_connect = ...
...
acl_smtp_rcpt = acl_check_rcpt
acl_smtp_data = ...
...
```

Access Control Lists

Konfiguration

```
begin acl

  acl_check_connect:
    ...

  acl_check_rcpt:

    accept  domains       = +local_domains
            local_parts   = postmaster
    ...
    require message       = relaying denied
            domains       = +local_domains

    require message       = unknown recipient
            verify        = recipient/callout=use_sender,defer_ok
    ...
    accept

  acl_check_data:

   accept   hosts         = +relay_from_hosts
   deny     message       = sorry, size matters
            condition     = ${if >{$message_size}{20M}
   accept
```

Access Control Lists

Features

Zugriff auf **alles**, was an Information verfügbar ist, u.a.:

- Ratelimit mit beliebigen Keys:
  `ratelimit = 10/2h/$sender_host_address`
- Überprüfung von Adressen (Routing): `verify = recipient`
- Überprüfung von Adressen (Callout):
  `verify = recipient/callout=use_sender,defer_ok`
- DNS-Blacklists: `dnslists = sbl.spamhaus.org`
- Authentifizierte Verbindung: `authenticated = *`
- Verschlüsselte Verbindung: `encrypted = *`
- Content-Scan: `malware = *`, `spam = ...`
- Header-Syntax: `verify = header_syntax`
- Header-Absender: `verify = header_sender`
- SSL: `verify = certificate`
- Reverse-DNS: `verify = reverse_host_lookup`
- Generische Bedingung: `condition = ...`

Aufgabe

Alle Empfänger müssen der selben Domain angehören (z.B. weil wir domainspezifische Spam-Policies haben)

Lösung

```
begin acl

  acl_check_rcpt:
    ...

    defer   message = multiple recipients with same domain only
            !acl = same_domain
    accept

  same_domain:
    accept   condition = ${if !def:acl_m_domain}
             set acl_m_domain = $domain

    accept   domains = $acl_m_domains

    deny
```

Access Control Lists

Test 1

```
$ swaks --pipe 'exim -bh 8.8.8.8' -f ... -t info@example.org,office@example.org -q rcpt
<-  **** SMTP testing session as if from host 8.8.8.8
<-  **** This is not for real!
...
 -> RCPT TO:<info@example.org>
>>> using ACL "acl_check_recipient"
>>> processing "defer"
>>> check !acl = same_domain
>>> using ACL "same_domain"
>>> processing "accept"
>>> check condition = ${if !def:acl_m_domain}
>>>                 = true
>>> check set acl_m_domain = $domain
>>>                        = example.org
>>> accept: condition test succeeded in ACL "same_domain"
...
<-  250 Accepted
 -> RCPT TO:<office@example.org>
>>> using ACL "acl_check_recipient"
>>> processing "defer"
>>> check !acl = same_domain
>>> using ACL "same_domain"
>>> processing "accept"
>>> check condition = ${if !def:acl_m_domain}
>>>                 =
>>> accept: condition test failed in ACL "same_domain"
>>> processing "accept"
>>> check domains = $acl_m_domain
>>> example.org in "example.org"? yes (matched "example.org")
>>> accept: condition test succeeded in ACL "same_domain"
...
<-  250 Accepted
```

Access Control Lists

Test 2

```
$ swaks --pipe 'exim -bh 8.8.8.8' -f ... -t info@example.org,office@example.org -q rcpt
<-  **** SMTP testing session as if from host 8.8.8.8
<-  **** This is not for real!
...
 -> RCPT TO:<info@example.org>
>>> using ACL "acl_check_recipient"
>>> processing "defer"
>>> check !acl = same_domain
>>> using ACL "same_domain"
>>> processing "accept"
>>> check condition = ${if !def:acl_m_domain}
>>>                 = true
>>> check set acl_m_domain = $domain
>>>                        = example.org
>>> accept: condition test succeeded in ACL "same_domain"
...
<-  250 Accepted
 -> RCPT TO:<office@example.com>
>>> using ACL "acl_check_recipient"
>>> processing "defer"
>>> using ACL "same_domain"
>>> check condition = ${if !def:acl_m_domain}
>>>                 =
>>> accept: condition test failed in ACL "same_domain"
>>> processing "accept"
>>> check domains = $acl_m_domain
>>> example.com in "example.org"? no (end of list)
>>> accept: condition test failed in ACL "same_domain"
>>> processing "deny"
>>> deny: condition test succeeded in ACL "same_domain"
>>> defer: condition test succeeded in ACL "acl_check_recipient"
...
<** 451 multiple recipient with same domain only
```

Beispiel: Empfängerüberprüfung

Callforward

Aufgabenstellung

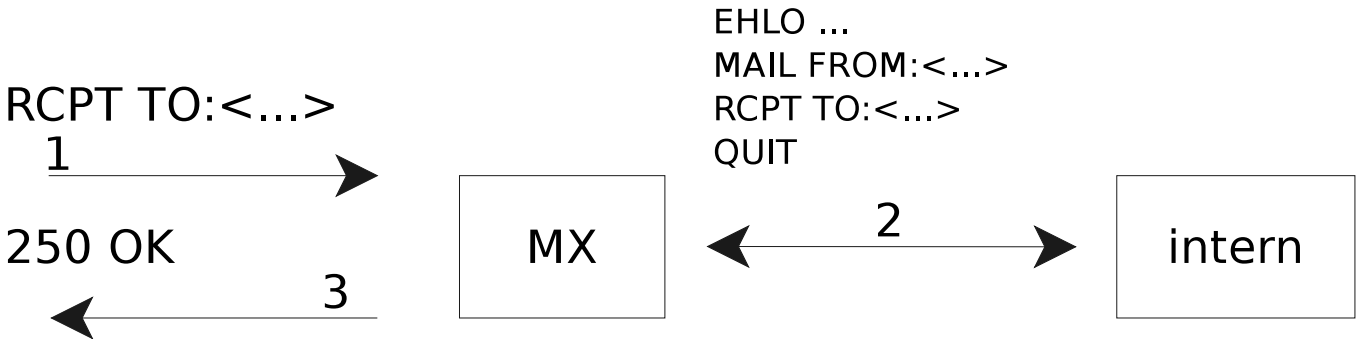
Exim sei MX und soll Nachrichten an einen internen Server weiterleiten, aber nur, wenn der Empfänger wirklich existiert.

Lösung?

- Nutzerdatenbank duplizieren
- Nutzerdatenbank (LDAP, AD, . . .) anzapfen?

Lösung!

Die Entscheidung an den internen MTA zu delegieren: Callforward - Überprüfung des Empfängers per SMTP



Beispiel: Empfängerüberprüfung

Callforward

```
begin acl

  acl_check_recipient:

    ...

    deny   domains = +internal_domains
           !verify = recipient/callout=use_sender,defer_ok

    accept
```

„Probleme“

- Exchange 2010(?): Unknown User erst nach DATA
- Content-Scan auf dem internen Server

Test Callforward

existierender Empfänger

```
$ swaks -f hans@example.com -t hs@schlittermann.de --pipe 'exim -bhc 8.8.4.4' -q rcpt
=== Trying pipe to exim -bhc 8.8.4.4...
<-  **** SMTP testing session as if from host 8.8.4.4
<-  **** This is not for real!
<-  220 hs12.schlittermann.de ESMTP Exim 4.85_RC1-53-a466d09-XX Fri, 20 Mar 2015 12:33:18 +0100
...
 -> RCPT TO:<hs@schlittermann.de>
>>> using ACL "acl_check_rcpt"
>>> check !verify = recipient/callout=use_sender,defer_ok
>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>> routing hs@schlittermann.de
>>> schlittermann.de in "! +local_domains"? yes (end of list)
>>> calling dnslookup router
...
>>> routed by dnslookup router
>>> Attempting full verification using callout
>>> Connecting to ssl.schlittermann.de [212.80.235.130]:25 ... connected
>>>   SMTP<< 220 ssl.schlittermann.de ESMTP Exim (mx) 4.80 Fri, 20 Mar 2015 12:33:20 +0100
>>>   SMTP>> STARTTLS
>>>   SMTP<< 220 TLS go ahead
>>>   SMTP>> MAIL FROM:<hans@example.com>
>>>   SMTP<< 250 OK
>>>   SMTP>> RCPT TO:<hs@schlittermann.de>
>>>   SMTP<< 250 Accepted
>>>   SMTP>> QUIT
>>> ----------- end verify ------------
<-  250 Accepted
 -> QUIT
```

Test Callforward

nicht-existierender Empfänger

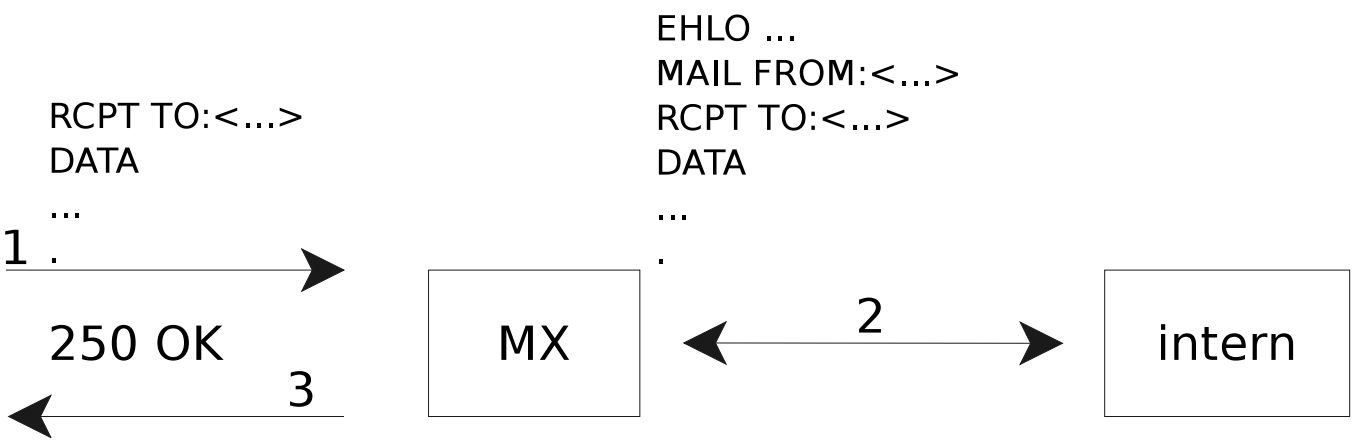
```
$ swaks -f hans@example.com -t foo@schlittermann.de --pipe 'exim -bhc 8.8.4.4' -q rcpt
=== Trying pipe to exim -bhc 8.8.4.4...
<-  **** SMTP testing session as if from host 8.8.4.4
<-  **** This is not for real!
<-  220 hs12.schlittermann.de ESMTP Exim 4.85_RC1-53-a466d09-XX Fri, 20 Mar 2015 12:37:17 +0100
...
 -> RCPT TO:<foobar@schlittermann.de>
>>> using ACL "acl_check_rcpt"
>>> check !verify = recipient/callout=use_sender,defer_ok
>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>> routing foobar@schlittermann.de
>>> schlittermann.de in "! +local_domains"? yes (end of list)
>>> calling dnslookup router
...
>>> routed by dnslookup router
>>> Attempting full verification using callout
>>> Connecting to ssl.schlittermann.de [212.80.235.130]:25 ... connected
>>>   SMTP<< 220 ssl.schlittermann.de ESMTP Exim (mx) 4.80 Fri, 20 Mar 2015 12:37:20 +0100
>>>   SMTP>> STARTTLS
>>>   SMTP<< 220 TLS go ahead
>>>   SMTP>> MAIL FROM:<hans@example.com>
>>>   SMTP<< 250 OK
>>>   SMTP>> RCPT TO:<foobar@schlittermann.de>
>>>   SMTP<< 550-recipient verify failed
>>>          550 Server time: Fri, 20 Mar 2015 12:37:20 +0100
>>>   SMTP>> QUIT
>>> ----------- end verify ------------
<** 550-Callout verification failed:
<** 550-550-recipient verify failed
<** 550 550 Server time: Fri, 20 Mar 2015 12:37:20 +0100
 -> QUIT
```

Beispiel: Empfängerüberprüfung

Callforward + Cutthrough

```
begin acl

  acl_check_recipient:
    ...
    deny   domains = +internal_domains
           control = cutthrough_delivery
           !verify = recipient/callout=use_sender,defer_ok
    accept
```



- Delivery nach intern noch während der externen Session
- Resultat DATA wird nach außen weitergegeben

Test Callforward

cutthrough

```
swaks --attach eicar.com -f hs@schlittermann.de -t hs@schlittermann.de --pipe 'exim -bhc 8.8.4.4'
=== Trying pipe to exim -bhc 8.8.4.4...
<-  **** SMTP testing session as if from host 8.8.4.4
<-  **** This is not for real!
<-  220 hs12.schlittermann.de ESMTP Exim 4.85_RC1-53-a466d09-XX Fri, 20 Mar 2015 14:04:10 +0100
...
>>> using ACL "acl_check_rcpt"
>>> check control = cutthrough_delivery
>>> check !verify = recipient/callout=use_sender,defer_ok
>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>> routing hs@schlittermann.de
...
>>>   SMTP>> EHLO hs12.schlittermann.de
>>>   SMTP<< 250-ssl.schlittermann.de Hello hs12.schlittermann.de [84.19.194.10]
>>>   SMTP>> MAIL FROM:<hs@schlittermann.de>
>>>   SMTP<< 250 OK
>>>   SMTP>> RCPT TO:<hs@schlittermann.de>
>>>   SMTP<< 250 Accepted
 -> DATA
>>> processing "accept"
 ...
 -> WDVPIVAlQEFQWzRcUFpYNTQoUF4pN0NDKTd9JEVJQ0FSLVNUQU5EQVJELUFOVElWSVJVUy1URVNU
 -> LUZJTEUhJEgrSCogCg==
>>>   SMTP>> .
>>>   SMTP<< 550-Classified by clamav as malware (Eicar-Test-Signature).
>>>          550-Server time: Fri, 20 Mar 2015 14:04:14 +0100
>>>          550 Server spool id: 1YYwaz-0007uU-UZ
>>>   SMTP>> QUIT
>>> ----------- cutthrough shutdown (target timeout on read) ------------
<** 550-Classified by clamav as malware (Eicar-Test-Signature).
<** 550-Server time: Fri, 20 Mar 2015 14:04:14 +0100
<** 550 Server spool id: 1YYwaz-0007uU-UZ
 -> QUIT
```

Logging

Sicherheit heißt auch Logging. Auskunft über das Verarbeiten der Nachricht. Gesteuert wird über `log_selector`, `log_write`, `debug_print`. Kein Logging bedeutet Fehler 4xx!

mainlog	alle relevanten Transaktionen, *dokumentiertes* menschen- und maschinenlesbares Format
rejectlog	Details zu abgewiesenen Nachrichten
paniclog	Konfigurationsfehler, schwere Probleme
syslog	Fallback, wenn nicht mal mehr paniclog geht
messagelog	Transaktionen zu einer spezifischen Nachricht bis zur „completion“

```
$ exim -Mvl 1Whwqz-00019E-Hg
2014-05-07 10:05:25 Received from a.bohl@example.com H=mout.foobar.com⟵
    (wotan.wgnd.lokal) [12.8.252.26]⟵
    I=[84.19.194.3]:587 P=esmtps X=TLS1.2:DHE_RSA_AES_128_CBC_SHA1:128⟵
    S=3995 id=5369E8CA.1030704@foobar.com T="Monte Timaro"
2014-05-07 10:06:28 gmail.de [173.194.70.18] Connection timed out
...
2014-05-07 10:09:38 hans@gmail.de R=dnslookup T=smtp defer (110): Connection timed out
```

Logging

mainlog

```
14:13:04 1Wi0ie-0005e8-Q7 <= wwwrun@emarsys.net H=mx.net.schlittermann.de [84.19.194.2]↩
    I=[84.19.194.3]:587↩
    P=esmtps X=TLS1.2:DHE_RSA_AES_128_CBC_SHA1:128↩
    S=51433 id=0.1.B1.FAE.1CF69EDB12C0806.0@pmta40192.emarsys.net
14:13:05 1Wi0ie-0005e8-Q7 => raabe@example.com↩
    F=<wwwrun@emarsys.net>↩
    R=domain_forward T=smtp↩
    H=mail.example.com [71.81.118.92] X=TLS1.0:DHE_RSA_AES_128_CBC_SHA1:128↩
    C="250 OK id=1Wi0if-0008Kc-Et"↩
    QT=1s DT=1s
14:13:05 1Wi0ia-0005dq-Ha => cwinkler@example.org F=<agent@ukrs939471.pur3.net>↩
    R=domain_forward T=smtp↩
    H=diw.vpn.schlittermann.de [10.10.10.18] X=TLS1.2:DHE_RSA_AES_128_CBC_SHA1:128
    C="250 OK id=1Wi0ig-00035h-Iq" QT=7s DT=7s
14:13:05 1Wi0ie-0005e8-Q7 Completed QT=1s
14:13:07 1Wi0ia-0005dq-Ha Completed QT=7s
14:13:07 1Wi0ih-0005ew-Lw <= agent@ukrs394971.pur3.net H=mx.net.schlittermann.de [84.19.194.2]↩
    I=[84.19.194.3]:587↩
    P=esmtps X=TLS1.2:DHE_RSA_AES_128_CBC_SHA1:128↩
    S=17836 id=0.0.C9.E5D.1CF69EDAA039062.0@mta20135.pur3.net
14:13:13 1Wi0ih-0005ew-Lw => info@diw-bau.de F=<agent@ukrs394971.pur3.net>
    R=domain_forward T=smtp↩
    H=diw.vpn.schlittermann.de [10.10.10.18] X=TLS1.2:DHE_RSA_AES_128_CBC_SHA1:128↩
    C="250 OK id=1Wi0in-00035n-Ht" QT=6s DT=6s
14:13:13 1Wi0ih-0005ew-Lw Completed QT=6s
```

Logging

rejectlog

```
06:30:13 1WhtSh-0004KX-Ta H=(ete4g.com) [174.36.30.154] I=[84.19.194.2]:25⏎
    F=<ete39@ete4g.com> rejected after DATA: spam 9
Envelope-from: <ete39@ete4g.com>
Envelope-to: <info@example-dresden.de>
P Received: from [174.36.30.154] (helo=ete4g.com)
        by mx.net.schlittermann.de with esmtps (TLS1.0:DHE_RSA_AES_256_CBC_SHA1:256)
        (Exim 4.80)
        (envelope-from <ete39@ete4g.com>)
        id 1WhtSh-0004KX-Ta
        for info@example-dresden.de; Wed, 07 May 2014 06:28:08 +0200
P Received: from [58.61.157.29] (port=55782 helo=WIN-5JAQ6OH8L0P)
        by spoon.arvixe.com with esmtpa (Exim 4.80.1)
        (envelope-from <ete39@ete4g.com>)
        id 1WhtSa-0002vh-7B
        for info@example-dresden.de; Tue, 06 May 2014 21:28:01 -0700
  Disposition-Notification-To: winni@ttm-group.com.cn
  MIME-Version: 1.0
F From: winni <winni@ttm-group.com.cn>
S Sender: ete39@ete4g.com
T To: info@example-dresden.de
R Reply-To: winni@ttm-group.com.cn
  Date: 7 May 2014 12:27:48 +0800
  Subject: Fw: cooperation for plastic and Metal parts
  Content-Type: text/html; charset=utf-8
  Content-Transfer-Encoding: base64
```

Natürlich Beobachtung des Logfiles, oder aber `exiwhat` und `eximqsumm`, `exipick`

Prozesse

```
$ exiwhat
 7489 handling incoming connection from www-2.whonagorf.org (mx.www-2.whonagorf.org) [192.255.135.183]↩
    I=[84.19.194.2]:25
 7955 handling incoming connection from (henriromano.com) [69.158.123.187] I=[84.19.194.2]:25
 7957 handling incoming connection from [69.158.123.187] I=[84.19.194.2]:25
 7994 handling incoming connection from (ifo.nl) [69.158.123.187] I=[84.19.194.2]:25
 7995 handling incoming connection from (immo-centrale.be) [69.158.123.187] I=[84.19.194.2]:25
 8165 handling TLS incoming connection from mail-ve0-f179.google.com [209.85.128.179] ↩
    I=[84.19.194.2]:25
 8268 delivering 1Wjb2G-00027h-SM: waiting for a remote delivery subprocess to finish
 8270 delivering 1Wjb2G-00027h-SM to pop.net.schlittermann.de [84.19.194.3]↩
    (******@************-dresden.de)
 8606 handling incoming connection from (localhost) [94.101.224.93] I=[84.19.194.2]:25
 9207 handling incoming connection from (vipmta198.vipmarketingonline.info) [103.249.102.198] ↩
    I=[84.19.194.2]:25
 9608 handling incoming connection from www-2.whonagorf.org (mx.www-2.whonagorf.org) [192.255.135.183] ↩
    I=[84.19.194.2]:25
 9633 handling incoming connection from static.165.4.4.46.clients.your-server.de (server1.tof61.com) [46.
    I=[84.19.194.2]:25
 9634 handling incoming connection from pointelite.net [5.39.17.162] I=[84.19.194.2]:25
```

Queue-Zusammenfassung

```
$ mailq | exiqsumm

Count  Volume  Oldest  Newest  Domain
-----  ------  ------  ------  ------

    9    13MB     66h     62h  ele.pku.edu.cn
    1    25KB      0m      0m  email.cz
    2    90KB     31h     31h  gmail.de
    1    37KB      6h      6h  kbb-****.de
    1    45KB     31h     31h  kpng.com
    2    3481     34h     32h  ********.*****.de
-------------------------------------------------------------
   16    13MB     66h      0m  TOTAL
```

Queue-Details

```
$ exipick
66h  1.4M 1WibFu-0005iS-C1 <****zhu@****.com>
        D ***eng@263.net
          ***ian@ele.pku.edu.cn

35h  1.6K 1Wj4sy-0000hU-Bv <> *** frozen ***
          www.********.**@********.******.de

31h   45K 1Wj8C4-0002Ba-43 <l*******@********-schuhe.de>
          ch.*.****@gmail.de
        D ****@aol.com
```

Sicherheit

setuid 0

- lokale Filetransports – MDA
- Port 25 – Linux Capabilities oder Port-Forwarding

Unix-Rechte

- Schreiben ausschließlich unterhalb des Spool- und Log-Verzeichnisses
- Lesen der Konfiguration, TLS-Key ohne Sonderrechte
- Konzept von „trusted configs“, „trusted users“
- Chroot ist leicht möglich - wenig Interaktion mit Systemkomponenten

Selbstbau

- keine „automake“-Hölle, lediglich *lookup*-devel
- Beschränkung auf notwendige Treiber
- GnuTLS oder OpenSSL

Leistung

Fehlende Performance lässt sich oft leicht ingenieurtechnisch ausgleichen (Skalierung), fehlende Flexibilität nicht so einfach.

- Schnelles IO-System für Spool
- Messagelog ist verzichtbar
- Shared Storage für das Spool - `localhost_number`
- Fallback-Hosts (transports option) für Problemkinder
- Alternative Queue-Runner
- Geschickte Konfiguration (Lookup-Caching)
- Locking der Hintfiles beachten (Retry, Ratelimit)

DANE in 1 Minute

Exim 3.85 kann DANE.

- DNSSec enablen
- Hosts festelegen, mit denen DANE notwendig ist

```
begin routers
    dnslookup:
      driver = dnslookup
      transport = remote_smtp
      dnssec_request_domains = *

begin transports
    remote_smtp:
      driver = smtp
      hosts_try_dane = *
```

Was fehlt

Noch einige Dinge vergessen?

- TLS - geht einfach so
- Header-Rewriting `*@*.example.com $1@example.com Ff`
- Retry-Rules `*.example.com rcpt_4xx F,2h,5m;G,2d,15m`
- SMTP-Authentifizierung (Client/Server)
- PRDR (ok), DNSSEC (ok), DANE (ok), Enhanced Status Codes (??)

DANKE

```
2015-03-21 15:32:26 [2858] 1WjPOs-0000k4-B5 Completed
```

schlittermann.de
hs@schlittermann.de

Linux
Mail
Exim
DNSSec
Perl