The Exim Mail Transfer Agent

Input Control and Access Control Lists

Philip Hazel

University of Cambridge
Computing Service

Topics

- Incoming message control features
- System filtering features
- No time for detailed discussion of
  - Configuration syntax
  - File and database lookups
  - String expansions
  - Domain, host, and address lists
- No coverage of routing and delivery or operational details, process structure, SMTP protocol, etc.
- The Exim release covered is 4.30

Incoming message control features

- SMTP authentication
- SMTP session encryption using TLS (SSL)
- Address verification (checking envelope addresses)
- Local policy is defined in *access control lists* (ACLs)
  - Rules for accepting messages for local delivery
  - Rules for accepting messages for relaying to other hosts
- Can also link into Exim a ***local_scan()*** function to do custom checks on incoming messages

Authentication

- SASL - Simple authentication and security layer
  - General framework for client-server authentication
  - Different authentication "mechanisms"
- Server advertises supported mechanisms
  - May be tailored for the client
- Client requests authentication by a specific mechanism
  - Data may be included with the request
- Server sends a "challenge" and the client responds
  - May be repeated any number of times
- Server accepts or rejects authentication
  - 235 Successful authentication
  - 435 Temporary problem with authentication
  - 535 Authentication failed

Authentication in SMTP

- Mechanisms advertised in response to EHLO

```
EHLO client.plc.ex
250-server.plc.ex Hello ph10 at client.plc.ex
250-SIZE 10485760
250-PIPELINING
250-AUTH PLAIN LOGIN
250 HELP
```

- Command is AUTH <*mechanism*> [*data*]
- Challenges use response code 334
- All data is base64 encoded

PLAIN authentication (RFC 2595)

- Client sends a single set of data, containing three items
  - Identity to login as (not relevant for SMTP)
  - Identity whose password is to be checked
  - The password
- Binary zeros separate the three data items

```
AUTH PLAIN AHBoMTAAc2VjcmV0
```

- Unencoded that is

```
AUTH PLAIN <nul>ph10<nul>secret
```

- The first field is usually empty in SMTP
- Server responds immediately with success/failure
  - Password is transmitted in cleartext if session not encrypted
  - No challenge is issued; only one exchange is needed
  - (Alternate usage has no data with AUTH and an empty prompt)

LOGIN authentication

- No formal definition; used by Pine and the c-client library
- Separate prompts for username and password

```
AUTH LOGIN
334 VXNlcm5hbWU6                (Username:)
cGxMA==                          (ph10)
334 UGVzc3dvcmQ6                 (Password:)
c2VjcmV0                         (secret)
235 Authentication successful
```

- Password is again passed in cleartext
  Three exchanges are required
- Some clients are picky about the exact text of the prompts

CRAM-MD5 authentication (RFC 2195)

- Server sends a challenge string which is different each time

```
AUTH CRAM-MD5
334 PDE4OTYuNjk3MTcwOTUyQHBvc3RvZmZpY2Uuc...
    <1896.697170952@postoffice.reston.mci.net>
```

- Client sends back a username and the MD5 digest of the challenge string concatenated with the password (in hex)

```
dGltIGI5MTNhNjAyYzdlZGE3YTQ5NWI0ZTZlNzMzNG...
tim b913a602c7eda7a495b4e6e7334d3890
235 Authentication successful
```

- Password does not traverse the network
  But must be stored in cleartext at both ends

Configuration file

- Exim uses a single runtime configuration file, divided into a number of sections
- The first section contains global option settings
- The other sections start with "begin *sectionname*"
  They are optional, and may appear in any order
- Comments, macros, if-then-else, and inclusions are available
- Option settings can refer to auxiliary data files, for example, a file of aliases (usually **/etc/aliases**)
- Many options are "expanded" whenever they are used

Default configuration file layout

Global option settings

`begin ACL`
Access control lists

`begin routers`
Router configuration

`begin transports`
Transport configuration

`begin retry`
Retry rules

`begin rewrite`
Rewriting rules

`begin authenticators`
Authenticator configuration

required for SMTP input

required for message delivery

Configuration file settings

- Option settings take the form *name* = *value*
- Many values are strings which are *expanded*, that is, changed each time they are used
  For example, the values of variables can be inserted
- Data can also be obtained from files and databases
- Lists of domains, hosts, and addresses are often used
  Can include wildcards, regular expressions, or use an indexed file or a database to make list searching efficient

Data lookups (1)

lsearch	linear search
dbm	keyed file (choice of library)
cdb	keyed readonly file
nis	NIS lookup
dsearch	directory search
wildlsearch	linear search with wildcards

Single-key lookups

nisplus	NIS+ lookup
ldap	LDAP lookup (query in URL format)
mysql	MySQL lookup
pgsql	PostgreSQL lookup
oracle	Oracle lookup
passwd	Password data lookup
dnsdb	DNS lookup
whoson	Whoson lookup

Query-style lookups

String expansions

- Variable and header line substitutions
- String operations: substrings, hashing, IP address masking, character substitution, regex substitution (like Perl “s”), quoting for regex and lookup queries
- Conditional expansion: string matching, numerical tests, combining conditions with “and” and “or”
- Lookups are another form of conditional
- Password checking using *crypt()*, PAM, Radius, LDAP, Cyrus pwcheck or Cyrus saslauthd
- Calling programs, reading from files and sockets

SMTP authentication in Exim

- Different authenticator drivers for different mechanisms
  Can be configured for server or client or both
- On an Exim server
  AUTH is advertised if the client matches **auth_advertise_hosts**
  This is expanded, so can depend on circumstances
  For example, it can be empty unless the session is encrypted
- On an Exim client, authentication is attempted if
  The server is in **hosts_require_auth** or **hosts_try_auth**
  (options of the **smtp** transport)
  A client authenticator matches an advertised mechanism
- On failure, Exim delivers unauthenticated for **hosts_try_auth**

plaintext authenticator

```
plain:
  driver = plaintext
  public_name = PLAIN
  server_prompts = :
  server_condition = ${if and {{eq{$2}{ph10}}\
                   {eq{$3}{secret}}}{yes}{no}}
  server_set_id = $2
  client_send = ^ph10^secret
login:
  driver = plaintext
  public_name = LOGIN
  server_prompts = Username:: : Password::
  server_condition = ${if crypteq{$2}\
    {${lookup{$1}lsearch{/etc/passwd}\
    {${extract{1}{:}{$value}}}fail}}{yes}{no}}
  server_set_id = $1
  client_send = : ph10 : secret
```

cram_md5 authenticator

```
cram:
  driver = cram_md5
  public_name = CRAM-MD5
  server_secret = ${lookup{$1}dbm\
    {/md5/secrets}{$value}fail}
  server_set_id = $1
  client_secret = tanstaaf
```

- Authenticator conducts CRAM-MD5 dialogue
- Places user name in **$1**
- Expands **server_secret** and runs CRAM-MD5 check

Encrypted SMTP connections

- TLS (transport layer security) aka SSL (secure socket layer)
  Exim uses the OpenSSL or GnuTLS library for TLS support
- Server advertises support for STARTTLS command
  Client issues STARTTLS
  Server gives positive response
  An encryption key is then negotiated according to TLS rules
  Subsequent data is encrypted before transmission
  Session state is reset; a new EHLO is needed
- Message is not encrypted while in the hosts at either end
  TLS gives protection only against eavesdroppers
  In particular, protection for AUTH passwords
- Client certificates can be used for authentication

TLS on an Exim server

- Three options must be set for TLS to be used at all
  tls_certificate => the file containing the server's certificate
  tls_privatekey => the file containing the server's private key
  tls_advertise_hosts specifies which clients should be told
- The Exim user must be able to read the private key
- To verify client certificates
  tls_verify_certificates => the file containing the expected certificates
  tls_verify_hosts specifies clients that must be verified
  tls_try_verify_hosts specifies clients that may be verified

TLS on an Exim client

- Will try to use TLS by default if the server advertises it
  ... if Exim is built with TLS support!

- All the following options are set on the **smtp** transport
  Expansion allows for different values for different servers

- Set **hosts_avoid_tls** to suppress encryption for specific hosts
- Set **hosts_require_tls** to insist on encryption
  Otherwise, Exim will send in clear if STARTTLS is rejected

- Set **tls_verify_certificates** to verify the server's certificate
- Set **tls_require_ciphers** to restrict which ciphers are used

- Set **tls_certificate** and **tls_privatekey** for client certificate
  Used only if the server requests a certificate

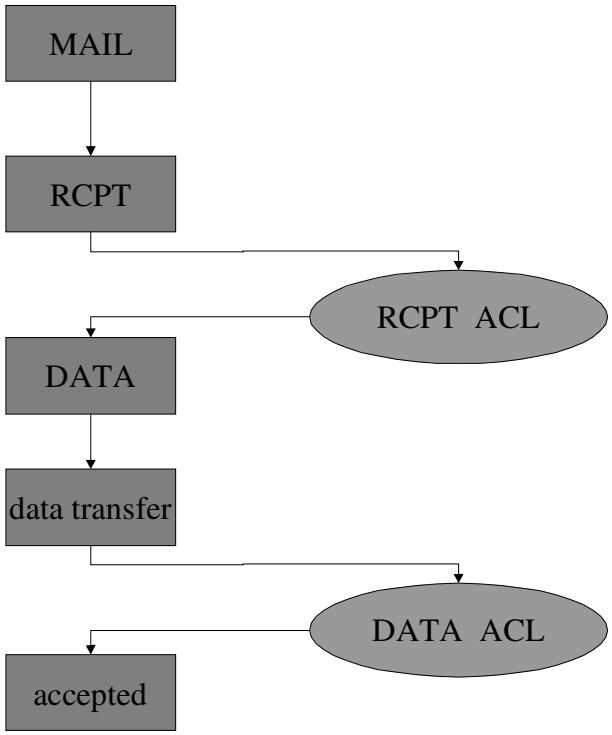
Access control lists

- Most ACLs are relevant for SMTP input
  They do apply to local SMTP (**-bs**)
  An ACL is available for non-SMTP input

- Main ACLs for incoming SMTP messages
  acl_smtp_rcpt defines the ACL to be run for each RCPT
    Default is "deny"
  acl_smtp_data defines the ACL to be run after DATA
    Default is "accept"

- Tests on message content can be done only after DATA or in a non-SMTP ACL

- Other ACLs can be used for AUTH, ETRN, EXPN, EHLO, MAIL, STARTTLS, VRFY, and at start of an SMTP session



Can check host, sender, recipient

Can reject individual recipients

Can check header lines and body

Can only reject entire message

A simple ACL

```
domainlist my_domains = a.b.c:*.e.f:...
acl_smtp_rcpt = acl_check_rcpt

begin acl

acl_check_rcpt:
  accept   local_parts = postmaster
           domains     = +my_domains

  require  verify      = sender

  accept   domains     = +my_domains
           verify      = recipient
```

- Implicit “deny” at the end

Finding an ACL

- **acl_smtp_rcpt** etc. are expanded, and can then be:
  An absolute file name (the file contains the ACL)
  The name of an ACL in the configuration (previous example)
  The text of an ACL itself

```
acl_smtp_vrfy = accept
```

- Choice of ACL can be made to depend on client host, sender address, recipient address, day of the week, or anything else that Exim knows about

```
acl_smtp_rcpt = ${if eq \
  ${mask:$sender_host_address/24}{10.1.2.0/24}\
  {acl_local}{acl_remote}}
```

ACL statements

- Each statement contains a verb and a list of conditions

  verb *condition 1* (one per line)
  condition 2
  ...

- If all the conditions are satisfied

accept	Accepts SMTP command or non-SMTP message (else may pass or reject - see later)
defer	Give a temporary rejection (= **deny** for non-SMTP)
deny	Rejects (else passes)
discard	Like **accept** but discards addresses
drop	Like **deny** but drops an SMTP connection
require	Passes (else rejects)
warn	Takes some warning action (e.g. logs or adds header) Always passes

ACL conditions and modifiers

- When a condition is false, no subsequent ones are evaluated
- Modifiers can appear among the conditions

  message is used when access is denied

```
require  message = sender must verify
         verify = sender
         message = recipient must verify
         verify = recipient
```

- Modifiers that follow a false condition are not seen

  This example does not work

```
require  verify = sender
         message = sender must verify
```

ACL modifiers (1)

- **message** defines a custom message for a denial or warning

```
deny     message  = You are black listed at \
                    $dnslist_domain
         dnslists = rbl.mail-abuse.org : ...
```

- **log_message** defines a custom log message

```
require log_message = Recipient verify failed
        verify   = recipient
```

- **endpass** is used with the **accept** verb for a 3-way outcome

```
accept  domains  = +local_domains
        endpass
        verify   = recipient
```

  Above **endpass**, failure causes the next statement to be run

  Below **endpass**, failure causes rejection

ACL modifiers (2)

- **control** can specify message freezing or queueing

```
accept hosts = ...
       control = queue_only
```

- **delay** causes Exim to wait before continuing

```
deny  !verify = recipient
       delay = 60s
```

- **logwrite** writes to the Exim log unconditionally

```
accept sender_domains  = +special_domains
       !verify  = sender
       logwrite = Accepted unverified sender \
                  from $sender_address_domain
```

ACL modifiers (3)

- **set** sets ACL variables

```
warn   condition = ...
       set acl_m4 = value
```

  acl_m*x* variables remain set for the message
  acl_c*x* variables remain set for the connection
  Available for use in delivery (from release 4.23)

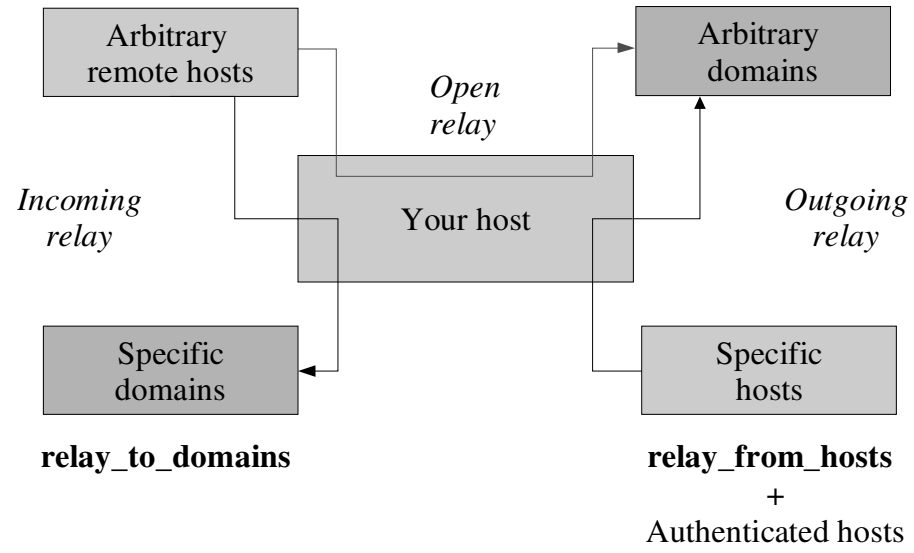
The default ACL (1)

```
acl_check_rcpt:

  accept  hosts       = :

  deny    domains     = +local_domains
          local_parts = ^[.] : ^.*[@%!/|]

  deny    domains     = !+local_domains
          local_parts = ^[./|] : ^.*[@%!] :\
                        ^.*/\\.\\./

  accept  local_parts = postmaster
          domains     = +local_domains

  require verify      = sender
```

The default ACL (2)

```
  accept  domains     = +local_domains
          endpass
          message     = unknown user
          verify      = recipient

  accept  domains     = +relay_to_domains
          endpass
          message     = unrouteable address
          verify      = recipient

  accept  hosts       = +relay_from_hosts

  accept  authenticated = *

  deny    message     = relay not permitted
```

Good and bad relaying



DNS black lists (1)

- Default is to lookup the client host's IP address

```
deny  message = rejected because \
        $sender_host_address is in a black \
        list at $dnslist_domain\n$dnslist_text
      dnslists = blackholes.mail-abuse.org

warn  message = X-Warning: $sender_host_address\
        is in a black list at $dnslist_domain
      log_message = found in $dnslist_domain
      dnslists = dialups.mail-abuse.org
```

- Can also lookup mail domains

```
deny  message = sender's domain is listed at \
        $dnslist.domain
      dnslists = dsn.rfc-ignorant.org/\
                 $sender_address_domain
```

DNS black lists (2)

- The RHS value can be specified

```
deny dnslists = rblplus.mail-abuse.org=127.0.0.2
```

- A comma-separated list of values is allowed
- Negated and bit-mask tests are also available
- The value is in **$dnslist_value** during message expansion
- DNS list lookups are cached for each incoming message
  (Not repeated for each recipient)

Address verification (1)

- Verification asks:

  Could we deliver a bounce to this address?

- Check by running the address through the routers
- "Verify mode" is set; routers can behave differently
  Skip this router if **no_verify** is set
  Use this router only for verification if **verify_only** is set
  Fail instead of accept if verifying and **fail_verify** is set

Address verification (2)

- Default verifies only the domain for remote addresses
- Can use “callout” for checking the local part

```
require verify = sender/callout
```

- Makes an SMTP call to the routed host and checks with a RCPT command (this is expensive, but a cache is used)
  - Can be used for recipients as well as senders
  - “Random” and postmaster checks can be requested
- Verification defers can be allowed to pass

```
require verify = sender/defer_ok
        verify = recipient/callout=defer_ok
```

Verifying header syntax

```
require  verify = header_syntax
```

- Checks those headers that contain addresses
  From: To: Cc: Bcc: Reply-To: Sender:
- Can be used only after DATA or in non-SMTP ACL
- Catches syntactic junk (often seen in spam)

```
To: @
To: Undisclosed recipients
To: abc@x.y.z <abc@x.y.z>
To: <>
```

- Rejects unqualified addresses by default
  Set **sender_unqualified_hosts / recipient_unqualified_hosts**

Verifying a header sender address

```
require  verify = header_sender[/options]
```

- Ensures that there is a valid sender in at least one header line
  Checks **Sender:, Reply-To:,** and **From:**
- Can be restricted to bounce messages only

```
deny   senders = :
       message = Need valid header sender
      !verify  = header_sender
```

- **senders** checks the envelope sender address
  Empty item checks for empty sender (bounce message)

Requiring encryption

- Can check for specific ciphers

```
deny   message   = wrong cipher
       encrypted = DES-CBC3-SHA
```

- Use ***** to check for any cipher
- Can check a client's certificate

```
accept  verify = certificate
```

  Requires **tls_verify_hosts** or **tls_try_verify_hosts** to be set

- Insisting on encryption for authentication

```
auth_advertise_hosts = ${if eq{$tls_cipher}{}\
     (main option)           {}{*}}

accept encrypted = *     (in ACL for AUTH)
```

More complex ACL for AUTH

- Suppose you want to allow all authentication mechanisms on encrypted connections, but only CRAM-MD5 when the session is not encrypted

```
acl_check_auth:
  accept  encrypted = *
  accept  condition = ${if eq \
            {${uc:$smtp_command_argument}}\
            {CRAM-MD5}{yes}{no}}
  deny    message = Require CRAM-MD5 or \
                    TLS encrypted connection
```

- **$smtp_command_argument** is set in non-message ACLs

Nested ACLs

- Calling a nested ACL

```
accept  verify = recipient
        acl    = ${lookup{$local_part}dbm;\
          {/etc/per-user/acls}{$value}fail}
```

- Forced **fail** in a condition expansion ignores the condition
  Above example accepts if lookup forces failure

- An empty ACL causes the condition to fail
  Without **fail**, above example denies if lookup fails

The Exiscan patch

- Exiscan is a patch that is maintained by Tom Kistner
- It adds conditions to the DATA ACL

demime	sanity checks on MIME structure
malware	detects viruses and other malware using 3rd party scanners such as Sophos
spam	uses results from SpamAssassin
regex	does regex matches on a message

- Each condition passes back expansion variables that contain useful information
- Get Exiscan from **http://duncanthrax.net/exiscan/**

Testing policy controls

The **-bh** option runs a fake SMTP session

```
exim -bh 192.203.178.4
>>> host in host_lookup? yes (matched "*")
>>> looking up host name for 192.203.178.4
>>> IP address lookup yielded dul.crynwr.com
>>> checking addresses for dul.crynwr.com
>>>   192.203.178.4
>>> host in host_reject_connection? no (option
                                          unset)
...
LOG: SMTP connection from dul.crynwr.com [192...]
220 your.host.name ESMTP Exim 4.20 Wed, 20 Mar...
enter SMTP commands here
```

Message filtering

- Exim supports three kinds of filtering
  - User filter: run while routing ("***.forward*** *with conditions*")
  - System filter: run once per message per delivery attempt
  - Transport filter: external program added to transport
- User and system filters are run for each delivery attempt
  - If delivery is deferred, filters run more than once
- User and system filters use the same syntax
  - System filter has some additional commands (**fail**, **freeze**)
  - They can be enabled for redirection filters
- Exim also supports a ***local_scan()*** function
  - Local C code can inspect a message at the point of arrival

The system filter (1)

- Runs once per message, at every delivery start
  - Use **first_delivery** to detect very first time
  - Can see all recipients in **$recipients**
- Can add to recipients or completely replace recipients
  - Non-significant delivery adds, significant delivery replaces
- Can add header lines that are visible to the routers, transports, and user filters
- Can remove header lines
- Can freeze message or bounce it
- Set up by

```
system_filter = /etc/exim/sysfilter
system_filter_file_transport = address_file
system_filter_pipe_transport = address_pipe
system_filter_user = exim
```

The system filter (2)

- Not powerful enough to do detailed spam checking
- Useful for per-message logging or archiving tasks

```
#Exim filter
if first_delivery and
  ${mask:$sender_host_address/24}
  is 192.168.34.0/24
then
  unseen save
    /var/mail/archive/${substr_0_10:$tod_log}
endif
```

- Cannot use for per-recipient tasks, but can see all recipients

The system filter (3)

- Example: autoreply for an obsolete domain

```
if $recipients contains obs.domain.example
then
  mail
    from postmaster@your.domain
    subject "Warning: obsolete domain used"
    file /var/mail/obs-warn-message
    once /var/mail/obs-warn-oncelog
    once_repeat 3d
endif
```

- This sends just one message, even if there are several recipients in the obsolete domain

Filter commands

- **deliver** does "true" forwarding (sender does not change)
- **save** delivers to a named file
- **pipe** delivers via a pipe to a given command
- **mail** generates a new mail message
- **logwrite** writes to a log file

- **deliver**, **save**, and **pipe** are significant by default
  Can be made not significant by **unseen**
- **logwrite** happens during filtering
- The others are just set up during filtering and happen later
  The result of **pipe** is not available during filtering

- Can lock out a number of facilities in user filters
  save, **pipe**, **mail**, and **logwrite** commands
  existence tests, lookups, Perl, **readfile**, **run** in expansions

local_scan() function

- An installation can supply its own ***local_scan()*** function
  Written in C and linked into the Exim binary

- Called just before a message is accepted, after all other tests

- Can inspect header lines (in main memory) and body (on disk)

- Can reject the message with custom error message
  Permanent or temporary rejection

- Can accept the message
  Add or remove header lines
  Modify the recipients list (no recipients => discard)
  Supply a string for **$local_scan_data**

Exim is available from

ftp://ftp.csx.cam.ac.uk/pub/software/email/exim/...

.../exim4/exim-4.*xx*.tar.gz (or **.bz2**) is the latest release

- GNU General Public Licence
- ASCII documentation included
- PostScript, PDF, Texinfo, HTML are also available
- FAQ in ASCII and HTML with keyword-in-context index
- See also **http://www.exim.org**
- Discussion list: **exim-users@exim.org**
- Announce list: **exim-announce@exim.org**