By the end of this session, you should be able to:

- **Illustrate how email flows from author to recipient.**
- **Configure a Postfix email server.**
- **Configure a Dovecot IMAP server.**

9.3. Email Overview

Email programs and daemons have multiple roles and utilize various protocols:

- **MUA**, or **M**ail **U**ser **A**gent, is the main role of your email client. A **MUA** is where you read your email and compose emails to others. The **MUA** uses **IMAP** (Internet **M**essage **A**ccess **P**rotocol) or **POP3** (**P**ost **O**ffice **P**rotocol) to download your email.
- **MSP**, or **M**ail **S**ubmission **P**rogram, is the role your email client has when you click 'Send'. The **MSP** submits your message to the **MTA** (**M**ail **T**ransfer **A**gent) using **SMTP** (**S**imple **M**essage **T**ransfer **P**rotocol).
- **MTA**, or **M**ail **T**ransfer **A**gent, is the main role of your email server. The **MTA** is responsible for starting the process of sending the message to the recipient. It does this by looking up the recipient and sending the message to their **MTA** using **SMTP**.
- **MDA**, or **M**ail **D**elivery **A**gent, is the role your email server takes when it receives an email for you. The **MDA** is responsible for storing the message for future retrieval. The **MTA** uses **SMTP**, **LMTP** (**L**ocal **M**ail **T**ransfer **P**rotocol), or another protocol to transfer the message to the **MDA**.

9.4. SMTP

SMTP (**S**imple **M**ail **T**ransfer **P**rotocol) is a **TCP/IP** protocol used as an **Internet** standard for electronic mail transmission.

It uses a plain "English" syntax such as **HELO**, **MAIL**, **RCPT**, **DATA**, or **QUIT**. **SMTP** is easily tested using **telnet**.

An example of an **SMTP** conversation is:

```
telnet localhost 25
220 localhost.localdomain ESMTP Postfix
HELO localhost
250 localhost.localdomain
MAIL from: root\@localhost
250 2.1.0 Ok
RCPT to: root\@localhost
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Dear root
Hi
.
```

9.5. POP3

The **P**ost **O**ffice **P**rotocol (**POP**) is one of the main protocols used by **MUA**s to fetch mail. By default, the protocol downloads the messages and deletes them from the server. This simpler, yet less flexible protocol is now on version 3.

9.6. IMAP

IMAP (**I**nternet **M**essage **A**ccess **P**rotocol) is the other main protocol used by **MUA** to fetch mail. When using **IMAP**, the messages are managed on the server and left there. Copies are downloaded to the **MUA**. This protocol is more complex and more flexible than **POP3**.

1. You compose an email using your **MUA**.
2. Your **MUA** connects to your outbound **MTA** via **SMTP**, and sends the message to be delivered.
3. Your outbound **MTA** connects to the inbound **MTA** of the recipient via **SMTP**, and sends the message along. (This step can happen more than once).
4. Once the message gets to the final destination **MTA**, it is delivered to the **MDA**. This can happen over **SMTP**, **LMTP** or other protocols.
5. The **MDA** stores the message (on disk as a file, or in a database, etc).
6. The recipient connects (via **IMAP**, **POP3** or a similar protocol) to their email server, and fetches the message. The **IMAP** or **POP** daemon fetches the message out of the storage and sends it to the **MUA**.
7. The message is then read by the recipient.

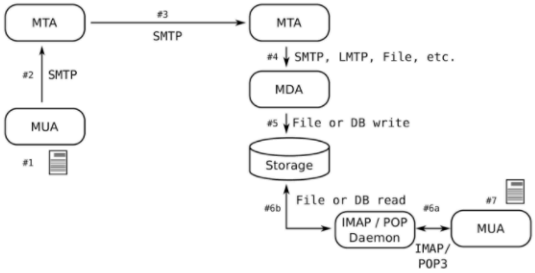

Figure 9.1: The Email Life Cycle

9.8. MTA Implementations

MTA (**M**ail **T**ransfer **A**gent) implementations use different software suites.

Sendmail was one of the first, if not the first **SMTP** implementation. **Sendmail** is difficult to configure with **m4** macros. Older versions had security problems.

Exim is an alternative to **Sendmail**.

Postfix has an architecture characterized by separate binaries and division of privileges. It is a drop-in replacement for **Sendmail**.

9.9. MDA Implementations

Many **MTA** software suites have components which act as **MDA**s (**M**ail **D**elivery **A**gents) as well. Some examples of these software suites include:

- **Sendmail.**
- **Postfix.**
- **procmail.**
- **Sieve.**
- **Cyrus.**
- **Spam Assassin.**

9.10. MUA Implementations

MUA (**M**ail **U**ser **A**gent) is used to manage a user's email. Some of the software suites used include:

- **Thunderbird.**
- **Mutt.**
- **Evolution.**
- **Outlook/Outlook Express.**

Mail programs use **IMAP** and **POP** protocols to access mail stored on remote computers. Some of the servers used for implementing IMAP and POP are the following:

- **UW -** Reference implementation of the **IMAP** protocol.
- **Cyrus IMAP** - Enterprise-ready **IMAP**. It's more complex and more difficult to configure.
- **Dovecot -** Easy to use due to its powerful configuration options.
- **Courier -** Designed to be "plug-and-play" installable and does not require any site-specific configuration.

Postfix uses the `/etc/postfix` directory for configuration files. The two most important configuration files are:

- `/etc/postfix/master.cf`
- `/etc/postfix/main.cf`

Within **Postfix** there is a process called master that controls all other **Postfix** processes that are involved in moving mail. The defaults in the `/etc/postfix/master.cf` file are usually sufficient.

The `/etc/postfix/main.cf` file contains the configuration parameters for mail processing. There are several hundred options that may be applied. In most cases, only a few mail parameters need to be altered in `/etc/postfix/main.cf`. The following are the usual candidates for customization:

- The domain name to use for outbound mail (**`myorigin`**).
- The domains to receive mail for (**`mydestination`**).
- The clients to allow relaying of mail (**`mynetworks`**).
- The destinations to relay mail to (**`relay_domains`**).
- The delivery method, indirect or direct (**`relayhost`**).

- The domain name to use for outbound mail (**myorigin**).
- The domains to receive mail for (**mydestination**).
- The clients to allow relaying of mail (**mynetworks**).
- The destinations to relay mail to (**relay_domains**).
- The delivery method, indirect or direct (**relayhost**).

The **postconf** command can be used to customize **/etc/postfix/main.conf**:

```
# postconf -e 'inet_interfaces = all'
```

Some of the commonly changed configuration options for **Postfix** include the following:

- **`mynetworks`**: **SMTP** servers which are *trusted* to blindly forward email.
- **`mynetworks_style`**: Allows for dynamic creation of the mynetworks option:
  - **`mynetworks_style = class`**: Trust all hosts in the same A/B/C network class (not a good idea).
  - **`mynetworks_style = subnet`**: Trust all hosts in the same subnet as any interface on your host.
  - **`mynetworks_style = host`**: Trust only the local machine.
- **`inet_interfaces`**: **IP** addresses to listen on for incoming connections.

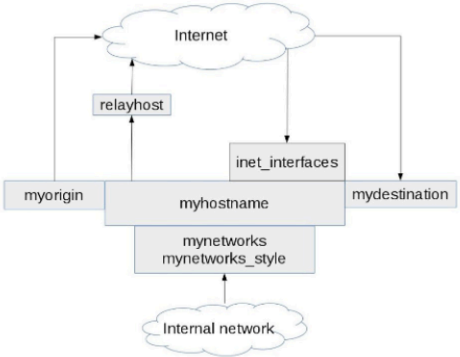

Figure 9.2: Postfix

mynetworks option:

- **`mynetworks_style = class`**: Trust all hosts in the same A/B/C network class (not a good idea).
- **`mynetworks_style = subnet`**: Trust all hosts in the same subnet as any interface on your host.
- **`mynetworks_style = host`**: Trust only the local machine.

- **`inet_interfaces`**: **IP** addresses to listen on for incoming connections.
- **`myorigin`**: Domain that outbound email should appear to come from.
- **`mydestination`**: Domains that are the final destination for messages.
- **`relayhost`**: **SMTP** server to forward outbound email.

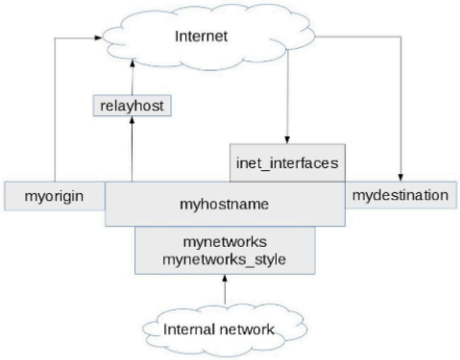

Figure 9.2: Postfix

9.14. Security Considerations

Since **SMTP** is a clear-text protocol, security is a concern. Network authentication relies by default on a trustworthy subnet.

The default **Postfix** configuration is only marginally secure:

- There is no **SSL/TLS** encryption.
- The **`mynetworks_style`** directive allows for all hosts in the subnet to forward. Luckily, the default setting of the **`inet_interfaces`** directive allows only localhosts to connect.
- No spam filtering is enabled by default.

9.15. Postfix Authentication

Postfix does not provide a **S**imple **A**uthentication and **S**ecurity **L**ayer (**SASL**) mechanism itself, but requires a properly set up **SASL** implementation. The **Postfix daemon** can support the **Cyrus SASL** or the **Dovecot SASL** mechanisms.

To configure the **SASL** mechanism for **Postfix** using the **Dovecot SASL** system, you would do the following:

1. Configure **Dovecot** and the **Dovecot SASL** system.
2. Configure **Postfix** to use the **Dovecot SASL** system for authentication.
3. Configure **Postfix** to trust users who have properly authenticated.
4. Configure **Postfix** to enable trusted users to change their outbound envelope (From) address.

The **Postfix** options you will need to change in order to enable **Dovecot SASL** are the following:

```
smtpd_sasl_type = dovecot
```

```
smtpd_sasl_path = private/auth
```

The **Postfix** options which enable the **SMTP SASL** authentication are:

```
smtpd_sasl_auth_enable = yes
```

```
broken_sasl_auth_clients = yes
```

9.28.a. Troubleshooting Dovecot

By default, **Dovecot** logs using the **syslog** facility **mail**, the configuration of your **syslog** server determines the location.

IMAP and **POP3** are plain-text protocols. Therefore, using **telnet** or the **openssl** client for troubleshooting is possible.

```
$ telnet localhost imap
a1 LOGIN student student
a2 LIST "" "*"
a3 EXAMINE INBOX
a4 FETCH 1 BODY[]
a5 LOGOUT
```

A working email client is the easiest way to troubleshoot and test an **IMAP** or **POP3** server. The **mutt** email client works well.

```
mutt -f imaps://user@mail.example.com:993/
```

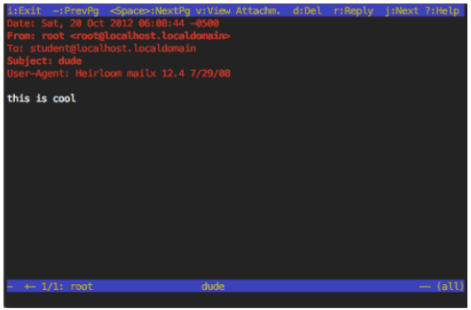

Figure 9.4: Mutt Screen

Postfix SASL authentication with **Dovecot** can be enabled over a **UNIX Domain Socket** and over **TCP**.

To enable **UNIX** accounts to authenticate over a **UNIX Domain Socket**, edit the `/etc/dovecot/conf.d/10-master.conf` file.

Make these changes in the **`service auth`** section:

```
service auth {
unix_listener auth-userdb { }
unix_listener /var/spool/postfix/private/auth { }
}
```

When testing plain-text **SASL** authentication, you will need to submit a **`base64`** encoded username and password. This string can be generated using the **`echo`** and **`base64`** commands:

```
$ echo -en "\0user\0password" | base64
```

```
AHVzZXIAcGFzc3dvcmQ=
```

Copy that string and use this syntax to test the **SMTP** authentication:

```
AUTH PLAIN AHVzZXIAcGFzc3dvcmQ=
```

9.17. Postfix Security

Postfix security requires **SSL** or **StartTLS**.

To enable optional **TLS** encryption, use the following options:

```
smtpd_tls_cert_file = /etc/postfix/server.pem
smtpd_tls_key_file = $smtpd_tls_cert_file
smtpd_tls_security_level = may
```

To enable authentication only after a **TLS** session has been established, use the following option:

```
smtpd_tls_auth_only = yes
```

Some of the tools used to monitor a **Postfix** server include:

- **pflogsumm** - A **Perl** script which translates **Postfix** log files into a human-readable summary.
- **qshape** - Prints the domains and ages of items in the **Postfix** queue.
- **mailgraph** - A package which creates **RRD** graphics of mail logs.

The **qshape** command is useful for determining where emails are getting stuck in queues. The output of the command displays the distribution of the items in the **Postfix** queues by recipient domain. A command line option "**-s**" will list by sender domain:

```
$ qshape | head
```

The output from this command shows:

```
<recipient name> T <total queued> 5 <0 to 5 mins> 10 <6 to 10 mins>
```

The columns in the output are:

- Recipient domain.
- Total (T) items in the queue.
- A time distribution in minutes of items in the queue. Time "buckets" are ranges: 0 to 5, 6-10, each bucket being twice as large as the prior bucket.

qshape is bundled with **Postfix** version 2.1 or later, but may be contained in a separate package, like the **Fedora** package **postfix-perl-scripts**.

qshape is bundled with **Postfix** version 2.1 or later, but may be contained in a separate package, like the **Fedora** package **postfix-perl-scripts**.

To learn more, please see **man qshape**. You can also find more information on the **Internet**: http://www.postfix.org/QSHAPE_README.html.

The open source tool **SpamAssassin** can be used as a "milter" (Mail filter: tied into the **MTA**, http://en.wikipedia.org/wiki/Milter), or a standalone **MDA**. **SpamAssassin** uses tests and other criteria to determine if a message is spam. This makes it more difficult for a spammer to circumvent the filter.

Make sure your **MTA** is not an **open relay** and is configured to only send outbound email for the proper hosts or authenticated users. This can be done by blocking or rejecting badly formed email, as well as by not accepting email from known spam hosts (blacklists).

Other tools that have been used to counteract spam are the **Sender Policy Framework** (**SPF**) and **DomainKeys**.

9.22. Email Aliases and Forwarding

Another common configuration change made on email servers are aliases and forwarding.

Email address aliases are managed by the `/etc/aliases` file and the changes are applied by using the **`newaliases`** command.

Email can also be sent to a different **MTA** using forwarding maps or rules.

9.23. Advanced Email Software

In addition to having just **MTA** software, there are full-suite messaging systems. These full-suite systems provide webmail, mobile mail, and function as **MTA**s:

- **HORDE**: http://www.horde.org/
- **Zimbra**: http://www.zimbra.com/
- **Open Xchange**: http://www.open-xchange.com/

9.24. Dovecot

Dovecot is an open-source **IMAP/POP3** server. **Dovecot** is secure, easy to configure and is standards compliant.

The author of **Dovecot** has a 1000 EUR bounty for any remote bug found.

To learn more, see: http://www.dovecot.org/security.html.

The **doveconf** utility parses the configuration file for both the **Dovecot** daemons and for debugging purposes. It can not edit running configuration files, but you can check the syntax of your configuration files with:

```
doveconf -n
```

Examples of configuration files include:

- `/etc/dovecot/dovecot.conf`
- `/etc/dovecot/conf.d/`

The common **Dovecot** setup binds to a specific **IP** address, defines the protocols to handle, applies password restrictions, and points to user and password databases:

- **`listen = 10.20.34.111`**
  IP address to bind.
- **`protocols = imap pop3 lmtp`**
  Protocols to serve.
- **`disable_plaintext_auth = yes`**
  Default setting which disallows plaintext passwords.
- **`imap_client_workarounds pop3_client_workarounds`**
  Change these settings to support 'broken' client implementations of the **IMAP** or **POP3** protocols.
- Password databases: http://wiki2.dovecot.org/PasswordDatabase.
- User databases: http://wiki2.dovecot.org/UserDatabase.

Dovecot security includes an **SASL** authentication. This can act as an **SASL** provider for other systems.

SSL related configuration options include:

- **`ssl`** – Can be either **`yes`** or **`required.`**
- **`disable_plaintext_auth=yes.`**

The difference between these two options is that the **`ssl=required`** option forces encryption for non-plain text authentication as well.

- **`ssl_cert = <some/file>`** – Public key.
- **`ssl_key = <some/file>`** – Private key.

Because they are **SSL** keys, you can use the same certificates for **Apache** and **Dovecot**. Optionally, you can use different certificates for each protocol (**IMAP**,**POP3**).

The **SSL/TLS** configuration includes:

- **`conf.d/10-ssl.conf.`**
- Share certificate between daemons.