Lecture 15 — Email Services: MTA, POP/IMAP, Security, Mailing lists

Email is a vital service and must be as reliable as possible.

Review (show online resource):

- MUA (nail, mutt, alpine (was pine, but the developers wanted to use the Apache Commons license and couldn't with that name), hotmail, Outlook, ...)
- MTA, MDA (MSA discussed below)
- MAA (access agents)
- email envelope, mail headers, and mail body
- ***mail store*** mailbox file/folder (a.k.a. *mbox*), DBs, *maildirs*
- SMTP/ESMTP (demo with `mail -v pollock@acm.org`)
- POP, IMAP (POP is often referred to by version number POP3)
- MIME
- Notification: **`MAILCHECK`**, **`biff`**, `mailutil`, `frm`

Mail administrator = ***postmaster***. Other required name is ***abuse***.

Relay is the term used when an MTA accepts email that is not for local delivery. The email is *relayed* to another MTA for handling. Today it is very common to have an organization's MTA accept email from local client machines (and localhost users if any), and forward all this email to the ISP's MTA. A common problem today is the ***open relay***. This is an MTA that is willing to relay email for anyone. Don't do it! (*Refer to spammer article on website*.)

Your service must have a clear design (*architecture*) and well-documented policies: security, backup (user data, email stores, as well as fall-back off-site servers), quotas and email retention, privacy and monitoring, naming (users and mail folders). You must make sure someone is responsible for handling email sent to `postmaster` and `abuse`.

Your MTA must accept email from localhost users, other hosts on your network (known as ***local clients***), as well as incoming email from the Internet. However **your MTA should only relay email for localhost users and local clients**.

Notes: Popular MTAs include Sendmail, Postfix, Exim, and MS Exchange. Most of these include an MDA. A powerful stand-alone (can be used with any MTA) MDA is **`procmail`**. (Procmail is powerful and flexible, but it can be *much* slower than other, simpler MDAs.) There are commercial email servers that include *groupware* (shared folders, calendering, addressbooks) and MAA services in one package, such as atmail or Zimbra (supports MS MAPI protocol).

sSMTP (Simple SMTP) is a lightweight MTA to deliver mail from a computer to a mail hub, only. sSMTP is indeed simple, there are no daemons or anything hogging up CPU; Just the sSMTP utility. Unlike full MTAs, sSMTP does not receive mail, expand aliases, or manage a queue. (It is more of an MUA, but with the user interface of sendmail.) For this reason, it is often used as the MTA for non-mail servers (since occasionally a DNS, print, file, web, etc., server needs to send out email).

mSMTP is similar, but supports authentication, encryption, and other features.

RFCs to know: 821, 822 --> 2821, 2822 -->**5321, 5322** (ESMTP), 2076 and 4021 (headers), 2045 (MIME), 1939 (POP3; also 2449 and 5034), and 3501 (IMAP). Mbox format (a.k.a. *Berkeley mailbox format*) 1st mentioned in RFC-976 (UUCP mail) and sort-of defined in RFC-4155 (application/mbox MIME file type). Many others related to email too. (POP4 is not a standard, yet many parts of it are in current use.)

POP doesn't support mail folders. However, there are a variety of techniques for faking that. Sometimes IMAP folder names are appended to the user name (login as "user#folder"), and additional POP sessions are made for each folder. Sometimes the mailstore will add the IMAP foldername as a header when using POP, return all folders as the content of INBOX, and let the MUA sort it out. Some MUAs can use the server's webmail interface behind the scenes, and determine the IMAP folders that way.

No matter how you look at it, folders with POP are going to be unreliable. And while you can create folders in a POP account from most MUAs, they are only local to that MUA; there is no way to have the messages or those folders accessible from a different MUA.

(POP4 adds support for folders, but it isn't widely used.)

MSA is for *mail submission agent*. The obvious mail service architecture has the MTA do a lot of work besides routing email (which slows down the mail service):

- Address re-writing
- sanity and error checking
- header re-writing
- mailing-list management
- security checking (authentication and encryption)
- virus and spam filtering (including gray/white/black lists)
- handing errors (logging, dropping, bouncing, re-trying)
- removing or *sanitizing* email bodies (unsafe HTML, scripts, or graphics)
- adding disclaimers automatically to outgoing email.

A scalable design splits the workload, with the MSA doing all the checking and management, and the MTA just doing the mail routing. (All this is spelled out in **RFC-2476**.) **MSAs usually listen on port 587**; the MUA can send email to the MSA on this port, which will accept the mail from the MUA, do the work required, and if the mail isn't dropped send it to the MTA mail queue. A more common technique is to have the MSA listen on port 25, and forward email to the MTA via a socket.

Some ISPs are preventing encrypted email to port 25, except by premium-paying (enterprise) customers. To reduce spam (I think), they charge higher for direct MTA access, and restrict other customers to using their MSA.

Amavis

Considering all the tasks of the MSA, it is common to use a variety of software to do all that: Mailman, SpamAssassin, SpamPal, ClamAV, ..., rather than a single MSA server. To facilitate this, **Amavis** provides a pluggable framework that does most of the work required for an MSA. You install the other software and tell Amavis to use it. Amavis will handle the work of uncompressing email attachments into temp files for scanning, then handing each file to each enabled "plugin" service, one at a time, and collecting the results.

The original Amavis has been replaced with Amavis_new. You probably don't want to install the original Amavis.

Amavis doesn't handle actual SMTP conversations with clients however. Instead you will run one instance (process) of your MTA as the front-end of an MSA, that does nothing except hand-off the email to Amavis. Amavis in turn will send acceptable (possibly sanitized) email to the main MTA process. For Sendmail there are two config files, one for the submission instance and one for the MTA instance. For Postfix, you configure each instance with the one set of config files. (Postfix has a `postscreen` process you can enable, to block spam via blacklists, mx checks, etc. However, postscreen doesn't handle content filtering (for spam and malware.)

Milters

One problem with Amavis is that the mail must be received and queued (saved on disk) before it can be scanned and rejected or modified (adding headers for example). Also, it can't handle outgoing mail (to add legal notices, or digitally sign mail). The smart folk at Sendmail, Inc. (Now owned by ProofPoint, Inc.) invented a *mail filter* API, or "milter" interface. Incoming mail can be handed to one or more milters (one after another), before being queued. This allows you to not waste disk space or bandwidth downloading full emails before rejecting. (So you can scan the first attachment then cancel the receipt of the rest of the email.) Milters can also be used for out-going email, to process it just before sending.

Milters are separate programs (from 3rd party vendors) that use the mail filter API to communicate with an MTA. Generally, such communication is vai a socket; the milter runs as a daemon, waiting for work to arrive from the MTA.

The idea has proven very useful, and many, many milters are available. Furthermore, Postfix now implements the Sendmail milter API and can thus use the same milters as sendmail. (For other MTAs that don't support sendmail milters, you use Amavis.) Using milters, you probably don't need Amavis too. (However, if one of the mail processors you want to use doesn't support the milter API, you can always use Amavis.) See milter.org for a catalog of milters, and more information about how they work (see the developer section).

Email User Accounts

Managing email users is difficult. Most MTAs and MAAs can't read `/etc/shadow`, and in any case the list of users to accept email for is rarely the same as the local users.

Modern servers can be configured to use their own databases of users (i.e., files in `/etc`) or standard ones including **LDAP** and **Kerberos** (and thus Windows *active domains*) to obtain user names and passwords. **Using a central store is a good idea** in that a user's login password for the network is also their password for email. (Of course you may not think this is a good idea!)

Mail store — Mbox, Maildir, and Databases

Managing the mail store is also difficult. The traditional method of one mailbox file per user (Berkeley **mbox** format; see RFC-4155) doesn't allow simultaneous access to a mailbox (when two people try to send, or one sends and one is reading, the same mailbox). The file locking required leads to poor performance if there is a lot of email traffic.

Many systems today use a directory per user, and one mail message per file (**maildir** format). Maildir format was invented by Daniel J. Bernstein for his `qmail` package. Because of this, the format is sometimes called *qmail format*. This scheme also allows **per-user mail folders** (and with some extra configuration, **shared mail folders**). See cr.yp.to/proto/maildir.html for the only maildir standard document available.

Sam Varshavchik (author of the Courier Mail Server) wrote an extension to the maildir format called **maildir++**. This supports subfolders and mail quotas. Maildir++ directories contain subdirectories with names that start with a "." (dot) that are also maildir++ folders. This extension is a violation of the maildir specification, but it is a compatible violation and most maildir software supports maildir++. (The only definition of maildir++ format is found in the middle of http://www.inter7.com/courierimap/README.maildirquota.html.)

Each maildir contains three subdirectories: **`new`**, **`cur`**, and **`tmp`**. New mail is in `new`. Once read it is moved to `cur`. `tmp` is used during delivery only. Maildirs will contain additional files and/or subdirectories to support additional mail folders and management information (such as which IMAP folders you subscribe to).

Each message has a unique (file) name. The name can't contain any colons (or of course slashes or null bytes). When moved from `new` to `cur`, the name changes from *`name`* to *`name`*`:2,`*`status`*. The status indicates if the message has been read ("`S`" for seen), moved to trash ("`T`"), Draft ("`D`"), etc.

If you would like to have a **quota on your maildir mailboxes**, the best solution is to always use filesystem-based quotas: per-user usage quotas that are enforced by the operating system. This works well when the default maildir is located in each account's home directory. This solution will NOT work if maildirs are stored elsewhere (say `/var/mail`) since you'll need to set up quotas on that filesystem too.

You may have a virtual domain setup where a single userid represents different users (e.g. `joe@foo.com` and `joe@bar.com`). So one userid (on the mail server) is used for many individual maildirs, one for each virtual user. In this case, you can use MDAs and MAAs that support the **maildir++ quota system**.

With a typically huge number of saved mail message files and lots of directories, access can be slow on some filesystems with default settings. You should pick a filesystem type that handles large numbers of files well (and supports quotas). Newer versions of ext2/3/4 support directory indexing (to speed up access) but you may have to enable it manually. Make sure that your kernel is configured with `CONFIG_EXT3_INDEX=y`. If this variable isn't available, you need a new kernel. You can check if the indexing is already enabled with `tune2fs`:

```
tune2fs -l /dev/maildir-disk | grep features
```

Look for **`dir_index`**. If missing, add it using:

```
umount /dev/hda3
tune2fs -O dir_index /dev/hda3
e2fsck -fD /dev/hda3
mount /dev/hda3
```

Besides mbox and maildir, other mail stores can be used such as a full-scale database product such as MySQL, PostgreSQL, or Oracle.

Whatever the scheme for your mailstore, all MUAs, MDAs, and MAAs must be configured to use the same one! (Some may not support all mail store types.)

Additional Issues To Consider When Designing An Email Service

You will need a **static IP address for your mail server**, or you will need to use one of several ***dynamic DNS* (*dDNS*)** solutions out there to maintain your DNS information. None of these schemes work too well since many organizations use caching for DNS information. So when your IP changes, the people you get email from may not have updated the information. **Use static IPs for any mail (and other) servers**.

Consider which MAA services to offer. This is a business decision, not a technical one. Some factors in the decision include maintaining user email-only accounts on the mail server, security (authentication and encryption), and expected load. For example, **large ISPs usually offer POP and not IMAP**, because IMAP takes considerably more resources to run. The same is true for security (e.g., POPS/IMAPS/HTTPS).

Different mail servers offer a range of security features, configurability, and scalability. (Courier IMAP very scalable but not standard with Fedora, which as of v9 offers Sendmail, Postfix, and Cyrus IMAP for IMAP/POP). Dovecot MAA is easier to configure and is considered more secure than Cyrus. Also, Cyrus uses an internal mail store and can't be configured to use mbox or maildir. Courier offers a full implementation of IMAP but doesn't permit folders except as sub-folders of INBOX.

It is generally appropriate to offer at least two MAA methods, web mail plus POP or IMAP (or both). I prefer IMAP and webmail (both with encryption, and the MTA too), but the cost to support that for the expected workload may require webmail and POP only. If you setup IMAP, it is usually very little extra work to setup POP too, and make all your users happy!

IMAP is better than POP for roaming users (or those who check email from home and office). (Q: why?)

Microsoft Exchange is a groupware server designed to work with Microsoft Outlook, and providing features such as a messaging server, shared calendars, contact databases, public folders, notes and tasks. The proprietary Exchange protocol used is called MAPI. **MAPI** stands for *Messaging Application Programming Interface*. It is a popular MAA with Windows clients but not licensed on *nix systems.

MS Exchange servers can easily use IMAP, and so can Outlook and Outlook Express MUAs. So if your users can live without the extra groupware features just set up the standard Windows client configuration to use IMAP.

In the past there have been a few Open Source implementations but I've heard negative reports on all of them except "Zimbra" (which is open source but not free). However Red Hat is now (2008) backing a new one, **OpenChange**. OpenChange aims to provide a portable open source implementation of Microsoft Exchange Server and provides interoperability with the Exchange protocol, MAPI. The OpenChange implementation provides a client-side library that can be used in existing messaging clients, so they can offer native compatibility with MAPI. See `www.OpenChange.org` for more information.

Email Security and User Authentication

You will need to consider security: **How will you authenticate remote users?** Many different schemes are possible, and all are in common use. Note this must be done for sending (MTA) and receiving (MAA) servers, and the web mail MUA too. Also, **consider virus scanning, spam filtering, and HTML sanitizing**.

Other security issues are phishing protection, disabling images and scripts, a privacy policy, adding disclaimers, email monitoring, backup, and retention.

Various other authentication schemes are available. **An MTA will list the authentication schemes it accepts with an `AUTH` line** in the help (and sometimes initial) menu. The client picks one of those and exchanges security information in the agreed upon manner. While details on security will wait until another course here are some user authentication possibilities:

- **Plaintext username and password** sent and cached for a few minutes. (RoadRunner does this.)
- Use **POP/IMAP before SMTP**. The client must first authenticate using POP/IMAP. That credential (user's IP address) is cached for a minute or so. During that time, SMTP is available from that IP address. (This authentication method fools many folks into thinking that their MUA sends out-going email using POP or IMAP, which isn't possible.)
- Other methods include using public keys, challenge-response, etc. Proprietary protocols use these methods.

To encrypt the session, the menu displayed by the remote MTA will list the `STARTTLS` option. This uses standard PKI (SSL/TLS security using certificates, just like HTTPS). Often only `STARTTLS` will be listed; once the secure session is established a second menu appears with `AUTH` listed instead. Now using plain text passwords are okay!

POP and IMAP don't generally use `STARTTLS`. Instead, they use POPS and IMAPS on different ports than for POP and IMAP.

Because so many security schemes are available, most MTAs and MAAs use a standard library that provides these security services to the MTA and MAA on that server. The authentication library commonly used for email is called **SASL, the *Simple Authentication and Security Layer* (RFC-4422).** You configure SASL to provide the MTA and/or MAA with the selected (approved) authentication mechanisms, then configure the MTA/MAA to use SASL. The MTA and/or MAA will list the selected mechanisms on the `AUTH` line. Note the SASL utilities will need access to the username/password database you use.

A common SASL library was released with the Cyrus mail server project that can be used with Postfix or Sendmail too. However, some servers use an internal SASL library, such as the Dovecot MAA.

There are a number of other security measures you can take, such as adding SPF or DKIM, and other email filters. Most modern MTAs allow extensions to plug into the MTA, to support extra features. Sendmail created the framework for this, dubbed *mail filter*, or ***milter***. Other MTAs support these milters as well (including Postfix).

Design (of the mail service architecture) Scenario: A few users on a small LAN

Do you need to do anything? Most ISPs today provide POP and/or IMAP, plus an SMTP mail relay for their customers. You probably don't need to do anything!

If you have several mailboxes, your MUA can be configured to fetch mail from each (and possibly integrate a set of mailboxes). An alternative is to run **`fetchmail`** to grab email from many servers regularly, and send all such mail into a single local mailbox. (Note some email servers make money from displaying ads when you use their web-based MUA and don't allow any other access.)

Design Scenario: SOHO

Okay, so you want to set up a mail system for a SOHO. In a SOHO you should only run a single MTA, known as the **`mailhost`** or ***mailhub***. All email should be funneled through this server: local mail between users on a single host, or between hosts, or arriving email from the Internet, all should be sent to this MTA for processing and delivery. The MTA should be able to handle the expected load of email so there is no need for a separate MSA.

If you want to use spam and/or virus checking, you can configure the MDA to filter the email through SpamAssassin, SpamPal, rspamd, and/or ClamAV (or whatever software you're using for this). However, you could also setup an MSA/Amavis solution for this with only a little more work required, which will be more efficient, and allow you to use multiple scanners if desired.

It pays to **keep all user mailboxes on this same server**. That way it is easy to backup all user mailboxes regularly, and you minimize LAN bandwidth use.

Users on localhost (there shouldn't be any) can simply use any MUA to read their email. However, **users on other hosts will need to use some MAA: POP, IMAP, a web mail interface, or some combination of these**. Such services are usually run on the same host as the MTA.

For a SOHO, access via IMAP and web mail only will work, with security (user authentication) required only for Internet access from roaming users.

Most ISPs offer a (small) number of email accounts with their service, so if that seems secure enough, a SOHO won't need any mail service at all.

Design Scenario: Large Organization

In this scenario you may have many users (hundreds or thousands of clients), roaming users, possibly several ISPs, and possibly several branch offices connected by a WAN (or a VPN across the Internet).

See HEC Montréal: Deployment of a Large-Scale Mail Installation for a good example.

You will need MTAs on each branch (to reduce WAN traffic), a central MTA (mailhost) and a backup MTA.

You will need to decide what MAAs to support and how (per branch, or by central server access only). **In this case scalability is very important.** Roaming user access can be very important as well, requiring user authentication and strong security. It is often in this scenario that you find mail server farms (clusters/grids), and a separate MSA from the MTA. Since the processing of email is slow (seconds to minutes per message), the network bandwidth is usually not as important as fast processing.

Unlike with web application servers (where you have *state* such as a shopping cart to keep track of between HTTP messages) each email message is independent of others. **So effective load balancing can be done easily (cheaply) using DNS round-robin for some set of (otherwise identical) mailservers.** Even if some client or proxy server caches an IP address, this scheme will work well enough, most of the time. You generally just need to specify several MX records with the same preference value. (Bind sends the records in a different order every time. Sendmail will pick one randomly.) Otherwise you need a load balancer such as BigIP from F5.

You must decide if the MTAs at each branch should be allowed to accept or send email to any MTAs other than internal ones (or maybe just to the central MTA). Allowing branch MTAs to communicate directly reduces WAN traffic and load on the central MTA, possibly eliminating the need for a cluster there. However there are problems with allowing this: replies get sent (by default) to the branch and not the main MTA, email names and aliases may still need to be centrally managed so now you have a synchronization task when names or aliases change, and external clients will "see" the branch MTAs. If not centrally managed each branch will end up with its own domain name and email naming scheme and set of aliases. This confuses clients (who don't "see" one organization) and complicates employee migrations (email redirections will be needed from the old branch).

It is usually better to **have all email from one organization *masquerade* as a single domain name** even if branch MTAs are allowed to contact outside clients. This means all email addresses, both envelope and header and including the "from" address, must be re-written as *user@domain-name* before sending. (However some addresses should not be masqueraded, e.g. *root@host.domain-name*.)

In addition to this **spoke and hub design**, you have all the same issues as for SOHO. Managing enterprise email is no easy task!

Email can be DNS intensive. **Use a caching DNS server on each mail server.** In any case the maintenance tasks for the DNS records for email must not be forgotten when branches change system administrators.

Note! One of the first steps in the following directions is to set up networking properly for a mail server. However this is a topic discussed in the *next* class, not this one. So don't worry if you don't fully understand the network setup instructions, and don't be afraid (or ashamed) to ask for help. There is a detailed walk-thru of the complete setup on-line you can consult if you get stuck.

Install Tasks Overview (Details follow)

- **System prep:**
  - **Configure networking**: Set static IP, hostname and domain name (`sysconfig/network` on RH). Configure resolver (default domain, DNS server IP addresses). Verify network connectivity (Static IP, gateway, all required firewall holes, TCP wrapper configuration). (RH files: `sysconfig/network`, `network-scripts/ifcfg-eth0`). Configure DNS: host and domain names, reverse IP lookup, MX records, and possibly SPF and other records in the primary nameserver. If that's a different host, set up a caching nameserver locally.
  - **Configure reliable system time**: NTP requires UDP/123, in and out. Use NTP service: `ntpd`; config files: `ntp.conf`, `ntp/*`.
  - **Configure `syslog` and `logrotate`** for `mail` facility (should be done automatically, but check).

    - **Configure security subsystems**: fix or turn off SELinux (the default policy as of F12 doesn't work with Postfix); edit /etc/selinux/config and set to permissive); make firewall holes for email ports: TCP/25, TCP/110 (POP3), TCP/143 (IMAP), TCP/993 (IMAPS), and TCP/995 (POPS3); configure TCP wrappers (hosts.allow) for your MTA, MSA, and MAA, if they use libwrap.so; and configure PAM and SASL (the defaults should work fine).
- **Configure MTA (`postfix` or `sendmail`):** Make sure your selected MTA is installed (see `alternatives` service for "`mta`"). For sendmail, you need both `sendmail` and `sendmail-cf` packages. If using authentication, make sure SASL (`cyrus-sasl`) is installed as well. **If installing multiple MTAs (e.g., both sendmail and postfix) you must first install "alternatives"** (`galternatives`; but part of `chkconfig` package as of F12**)**. Most of these packages will be installed by default but check to make sure!
- **Configure email aliases.**
- **Configure security**: **SASL**, TLS, STARTTLS, SMTP AUTH, SSL/TLS (web mail), blacklists, whitelists, greylists, SPF. (Done in security course.)
- **Configure MAA** (e.g., dovecot) for POP/POPS/IMAP/IMAPS.
- **Configure MAA (e.g., `dovecot`), MDA (e.g. `procmail`), and MUAs** (local ones, e.g. webmail, `mutt`, `alpine`, `mailx`) to use the chosen mail store. (And migrate any existing email to the new location.)
- **Configure Mail filtering (virus, spam, DoS attack countermeasures).** This often includes configuring Amavis (`amavisd-new`), some MSA, and one or more scanners. (Done in security class.)
- **Configure Web mail** MUA (e.g. `squirrelmail`). Note you will need to configure a web service (e.g. Apache, `httpd`) first.
- **Monitoring** for space, errors, security incidents; check log files for configuration errors, dropped mail, long delays, evidence of SPAM, or attacks.
- **Perform regular maintenance:**
    - configuring the daemons (e.g., `sendmail` and/or `postfix`)
    - update files as needed (such as user aliases or email redirects)
    - hiring/firing of employees
    - changing roles of employees, ...)
    - maintain mail lists
    - check for updates (especially security related patches)
    - set and document policies (e.g., email addresses)
    - maintain address books
    - web mail (and Apache) software maintenance
    - Regular backups of user's mailboxes
    - reply to queries (`abuse` and `postmaster`)

Detailed Directions:

Set hostname, domain name: *(Skip these steps for CTS-2322 email project.)*

1. Edit `/etc/sysconfig/network` to include your new host name:

```
HOSTNAME="wpserver"
```

2. Edit `/etc/hosts` to include your new FQDN:

```
127.0.0.1    localhost.localdomain  localhost
172.22.25.11 wpserver.gcaw.org      wpserver
```

3. Edit `/etc/sysconfig/network-scripts/ifcfg-eth0` to:

```
DEVICE=eth0
IPADDR=172.22.25.11    The static IP for the Instructor's host
NETMASK=255.255.255.0
GATEWAY=172.22.25.1    This may or may not work in DTEC-461
BOOTPROTO=none
ONBOOT=yes
```

(Note your NIC may not be "`eth0`", and the classroom IP addresses may be different.)

Setup cashing-only name server as forward-only: *(Skip this step for CTS-2322 email project.)*

The last problem is to get your resolver to resolve the names assigned to our class. This is harder than it appears as we are using a fake domain name, "gcaw.org".

You have several choices: Configure your `named` (BIND) to be authoritative for `gcaw.org`. (This is what I have done on the instructor's host.) An easier choice would be to set your resolver to only use the instructor's host DNS server, by removing all other nameserver directives from `resolv.conf`. Or you can configure your caching nameserver to forward all requests to the instructor's host. This is easy if you've already setup named as a caching DNS server. Add the following lines to your `named.conf` file in the `options` section:

```
forwarders { 172.22.25.11; };
forward only;
```

You must reboot to activate changes to your hostname (or, use `hostname` command and then restart every network service (since they cache that name). Rebooting is easier!)

Verify network connectivity:

I will provide you with a static IP address you can use. To obtain one edit the hostname page on the class wiki. Note that others won't be able to send you email as they don't know your name or IP address. And **without MX records, *user@domain* won't work** at all. Setup your resolver to use files before DNS, and add all the names and IP addresses from the wiki. Also, I will set up the instructor's workstation as an authoritative DNS server for `gcaw.org`, so you can point your resolver to use that. (The instructor's firewall must allow incoming port UDP/53.) ***Make sure the instructor's workstation is booted before working on your own server in the classroom!***

Configure Reliable Time, Time Zones, And Locales:

The short answer is to configure NTP, `/etc/localtime` (and/or `TZ`), and set the proper default locale (so timestamps in email messages show up correctly). This is probably done already as part of the default install. These topics will be discussed in detail later in this course.

Configure Syslog and Logrotate for Mail Facility:

```
# Log anything (not mail, ...) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none;cron.none \
          /var/log/messages
# Log all the mail messages in one place.
    mail.*                    /var/log/maillog
```

Configure *resolver* and DNS with Mail Exchanger (MX) & SPF (& DKIM) records:
(Skip these steps for CTS-2322 email project.)

For testing on a single host away from HCC you can avoid using DNS by configuring your `hosts` file with the IP address you get from DHCP (and your chosen hostname). As long as you don't reboot (and even if you do, as long as your IP address remains the same) and the *resolver* uses files before DNS, you will be able to use that host and domain name. You should also add your new (fake) domain name to the `search` directive in `resolv.conf`. Multiple domains may be listed but the first becomes the *default domain*. Make sure that is `gcaw.org`:

```
search gcaw.org hccfl.edu
```

MX records in the DNS DB tell people the IP address to send email to for a given domain name. If the mail server which will serve your new domain will have a full-time connection to the Internet, it should be the primary MX host for your domain. In this configuration, your MX records would look like this:

```
yourdomain.com.        IN  MX  10    yourmailserver.yourdomain.com.
```

In the real-world, you need to find another machine to ***queue*** mail for your domain while your machine is down for any reason. You point your MX records at that *backup mail server*, with a lower priority (higher number). For example:

```
yourdomain.com.        IN  MX  10    yourmailserver.yourdomain.com.
yourdomain.com.        IN  MX  20    othermailserver.otherdomain.com.
```

("othermailserver*"* is an off-site backup mail server so mail never bounces.) When an MTA attempts to delivery email it will look up the MX records and use the highest priority (lowest numbered) one that it can connect with. **If several have the same priority it may pick the same one every time, round-robin, or randomly pick one** (sendmail does that).

Always use the canonical domain name in MX records, or routing loops (or other errors) can occur!

For large organizations, you might want your own mail servers directly connected to the Internet and add an offsite backup mail server. Additionally you will want to verify your A and PTR records so the server name resolves correctly.

Finally, you will want to add the security features. SPF (***Sender Policy Framework*** `spf.pobox.com/`) uses an SPF (older: TXT) record that is used to indicate the authorized MTA IP address for some domain, as shown below. (So when my MTA receives email from some mail server at 1.2.3.4 claiming the email is from `wpollock.com`, a simple DNS lookup will show the email is coming from a fake address and hence is spam.) Another popular security scheme is **DomainKeysIdentfiedMail** (`dkim.org`), which is a bit more complex.

(*TODO: Show DKIM setup.*)

A complete set of records might look like this:

```
; DNS zone file for: gcaw.org
; Generated by Wayne Pollock, 3-7-02
$TTL    86400
$ORIGIN gcaw.org.
@    IN   SOA   ns1.gcaw.org. hostmaster.gcaw.org. (
                  2004081900   ; serial
                  360000       ; refresh, seconds
                  7200         ; retry, seconds
                  3600000      ; expire, seconds
                  360000 )     ; minimum, seconds

; Nameserver(s):
                IN      NS      ns1.gcaw.org.
                IN      NS      ns2.gcaw.org.

; MX: "user@gcaw.org" is sent to "user@mail.gcaw.org":
                IN      MX      10 mail.gcaw.org.
                IN      SPF     "v=spf1 a mx ptr -all"

; Records for this host (the primary nameserver):
                IN      A       10.41.223.253

; Aliases:
www             IN      CNAME   @
mail            IN      CNAME   @
ftp             IN      CNAME   @
```

Configure Security Sub-systems:

Edit the firewall rules. On on many Red Hat systems, these are in the files `/etc/sysconfig/ip*tables`. You can copy a line such as the one that allows SSH (port TCP/22), and paste it, and change the port number on the copy. You will need one such rule for each firewall hole: ports 25, 110, 143, 993, and 995 (all TCP). If you don't want to support plain, un-encrypted POP and IMAP, you don't need ports 110 or 143. On Fedora, you can easily reload the modified rules file with "`service iptables restart`".

Remember to add the same holes for IPv6 (unless you don't plan on using that yet.)

Newer versions of Fedora use `firewalld` to manage the rules. That tool has its own files for storing rules between reboots, so editing the `iptables` files directly won't work. You must either use the `firewalld` configuration files and tools, or disable that and enable the old `iptables` service.

Make sure SELinux will allow your services. The *targeted* policy used by default on Fedora 14 works for Sendmail but not Postfix. Hopefully this isn't an issue anymore, but I suggest you set SELinux to *permissive* mode for now. (In a production setting, you'd want to get the policy fixed.) Do this by editing `/etc/selinux/config` and reading the comments.) To change without rebooting, use `getenforce` and `setenforce` commands.

Some MTAs, MSAs, and MAAs use TCP Wrappers. If so, you need to allow them by editing the `/etc/hosts.allow` file. Dovecot uses programs from `/usr/libexec/dovecot/*`. You may have to list those programs too in the `.allow` file. (On Fedora 13, just `/usr/sbin/dovecotpw` and `/usr/libexec/dovecot/dovecot-auth` use PAM, and none of the programs use TCP Wrappers or SASL.)

MAAs (MTAs also if configured for it) use PAM and/or SASL for authentication. You may need to configure these services.

To see if some program uses PAM or TCP Wrappers, use "`ldd` *`path-to-program`*" and see if it was compiled with `libwrap.so` or `libpam.so`.

Configure MTA (Postfix): (*Skip if using sendmail MTA.*)

A note on names: Sendmail has been *the* MTA for 25 years. Many program have hard-coded the names of the sendmail utilities and command line options. This is why sendmail config is in `/etc/mail` not `/etc/sendmail`, and why the binaries are usually called `sendmail`, `mailq`, etc., even if using Postfix (or another) MTA.

Postfix is unlike Sendmail in some respects. Instead of one huge program, it is composed of over a half-dozen programs, each doing a single job. These programs don't need special IPC either; one program processes a mail message saved in a file, then the next program can deal with it. Sometimes the programs communicate with simple pipes. These programs can run in parallel (on different messages), and several instances of each can be running at once. Finally, these programs don't need root permissions, and can even run inside chroot jails. This design makes Postfix responsive, and by keeping each piece simple, helps ensure there were no security flaws.

For example, arriving mail connection attempts are handled by `smtpd`, one process per connection. The resulting mail is then passed to `cleanup`, which adds `Received` headers (and any missing important headers). The message is then saved in a directory known as the incoming mail queue. `qmgr` watches that queue and hands the message to `local` or some other software for handling or relaying the message. (See postfix.org/OVERVIEW.html for more details.)

The flow of email from one program to another is controlled by `master`. For example, if you wanted to run an MSA to screen out some spam, you configure master to hand connections to `postscreen` instead of `smtpd`, and have `postscreen` sent the message to `smtpd`. `Master` is the only program that needs to run as root, so it can listen on port 25. The remaining programs don't run as root (actually, there are two unprivileged user accounts used, to keep things even more secure). The `master` program also defines how the other programs will communicate, what options are passed to those programs when started, and how many instances to allow running simultaneously. Once configured for your architecture, you rarely need to edit `master.cf`. All remaining configuration is done by editing `main.cf`. (There are other files used in some cases, as well.)

Postfix allows one to continue long lines in these config files; if any line starts with white-space, it is assumed a continuation of the last non-comment line. This can catch the unwary: if you un-comment out some line by removing a leading "#", and that is followed by a space, you need to remember to remove the space as well! The error message (if any) will be about the previous line, which could be dozens of lines earlier in the file.

To configure Postfix, first make sure it is installed. Most servers setup *sendmail*. (The Solaris default setup forwards all mail to ***mailhost.domain***, where the central MTA (hub) should be. So be sure the canonical name of your MTA host in the DNS records is `mailhost`.)

Some systems allow several MTAs to be installed at the same time, using the **`alternatives --config mta`** command. Make sure all MTAs are stopped before switching from one to another! Without the alternatives system of symlinks, the last installed MTA will overwrite some of the files and commands of the one installed earlier.

In any case, **make sure all mail servers are off now, and only Postfix is configured to start at boot time.**

Postfix accepts email from localhost (the loopback interface) and other interfaces if it is configured to, optionally checks email for problems (and then drops, logs, or bounces the email), and delivers it to local users (by default using an internal MDA). Email is also examined for some changes to make: add missing headers, fix some errors, and rewrite sender/recipient addresses (e.g., change "`wayne`" to "`wayne@domain`") to *masquerade* the domain.

The only problem with this setup is, mail sent to system accounts and then forwarded to some human will be from "`root@domain`" and you won't know which host sent the email. To take care of `root` and other system accounts, you need to use the ***address rewriting*** features of Postfix (see the *readme* for details).

For a minimal SOHO setup, you only need to **make sure TCP port 25 gets through your firewall (and possibly TCP Wrappers)**, and **edit `/etc/postfix/main.cf`** with these lines:

```
myhostname = localhost  # or a real name:wpserver.gcaw.org
mydomain = localdomain  # or a real one: gcaw.org
myorigin = $mydomain
mydestination = $myhostname localhost localhost.$mydomain
➥ $mydomain
inet_interfaces = all  # and comment out the loalhost line!
```

(The symbol "➥" means the line was wrapped for readability; it should be entered as a single long line.) Check it with: **`postfix -v check`**.

The setting for `myorigin` may need to be different. Fedora and other Red Hat based systems incorrectly set the hostname, and swap the name and the FQDN. This confuses Postfix; the error shows up as mail to or from `user@localdomain.localdomain`. If you have one computer only, or are using the default network setup (as you will with the CTS-2322 email project), **you should set `myorgin` to `$myhostname`** instead. (The `main.cf-diff` resource show this both ways.)

Once your MTA is installed, you must start it. For postfix use:

```
# postfix start or postfix reload
```

Once your MTA is installed and started, you must test it:

```
# echo "it works" |sendmail -f root root & \
    tail -f /var/log/maillog
```

Note the command is `sendmail` no matter what your MTA is!

By default, the Postfix MDA (`local`) will deliver mail to `/var/mail/`*`username`* in MBOX format.

To test your MTA from a different machine, use the email address syntax of "*`username`*`@[`*`ip-address`*`]`"; the square braces are required!

Configure MTA (Sendmail): (*Skip if using Postfix*)

Sendmail is the oldest MTA and still one of the most popular. Early versions had some security issues, but that isn't true anymore. Sendmail is more configurable than most MTAs, and includes more features than most. It is also very efficient (if configured correctly). Configuring sendmail can be difficult, and there is commercial support available to help with that.

Generally, sendmail runs two instances, one listening for incoming mail (placed in `/var/spool/mqueue` until delivered/deleted), and another for locally-originated, outgoing mail (placed in `/var/spool/clientmqueue` until sent). Each instance has a separate configuration file, `/etc/mail/sendmail.mc` and .../`submit.mc`. These `.mc` files are actually macro files; after editing them you must run make to build the actual configuration files, `*.cf`. Although text, the `.cf` files are best thought of as binaries (that is, don't try to edit them directly).

The configuration directives in the sendmail configuration files enable various features, but those only work if the corresponding features have been compiled into the binary. To see what features are compiled into a sendmail binary, run:

```
    $ sendmail -bt -d0.13 </dev/null
```

In addition to declaring what features to enable at runtime, the config files may contain option settings. In most cases, the order of the macros in the `.mc` file matters.

Older versions of sendmail ran SetUID root. This is insecure and newer versions don't. However, any MTA must still be started as root in order to listen for incoming mail on the *well-known* port TCP/25. To have access to secure mail queue files and directories, those files and directories are accessible by group "**`smmsp`**" (**s**end**m**ail **m**ail **s**ubmission **p**rogram). The sendmail binary is run SetGID to this group to allow access.

As with some other MTAs, sendmail runs as a number of separate processes. One (the MTA) must be run as root to listen on port 25. The other (the MSA) handles locally submitted email and thus doesn't need to run as root. Unlike Postfix both parts are built into the same program, so if you run "`ps -ef`" you will see two `sendmail` processes running. The difference is the command line arguments used.

The first instance listens on port 25, runs as root, and uses **`sendmail.cf`** configuration file. The second instance uses a different configuration file, **`submit.cf`**. This instance is used (if the file exists) when a local MUA "submits" an email to `sendmail`. (The `sendmail` startup script starts both daemons.) Generally, the default `submit.cf` is fine, accepting email only from local MUAs.

To start sendmail use `-bd` (run as daemon), `-q15m` (check the mail queue every 15 minutes), `-v` *`envelope_address`* <*`msg_including_headers`*.

With Solaris, `sendmail` needs some additional configuration to work. Make an entry in `/etc/host` for `mailhost` (which should be the IP of your mail hub; this might be the current host on a SOHO or stand-alone workstation configuration. Next, modify the sendmail m4 config file `/usr/lib/mail/cf/mail.cf` with the following entries:

```
OSTYPE(`solaris2')
MASQUERADE_AS(`gcaw.org')
FEATURE(`nullclient', `mailhost')
```

Next rebuild `sendmail.cf` with:

```
    m4 ../m4/cf.m4 mail.cf >/etc/mail/sendmail.cf
```

As of Solaris 9, you need to configure sendmail as your message submission agent (MSA) as well as your message transfer agent (MTA). See Internet RFC 2476, and see `/usr/lib/mail/cf/submit.mc` for a prototype.

Sendmail Configuration Files (in `/etc/mail`)

Sendmail has the most complex configuration file of any program ever. This file, `sendmail.cf` (and `submit.cf`) is written in a custom language! Because it has proven impossible to edit this file (for the most part), a much simpler configuration file is used, `sendmail.mc` (and `submit.mc`). This file is written using "`m4`" macros. It is used to generate the `sendmail.cf` file. The easiest way is to just use the `make` command from the `/etc/mail` directory.

M4 is an old Unix facility for macro processing text files, and has an unusual syntax: quotes come in pairs: `` `this is quoted' ``. In addition, the string `dnl` starts a comment. The order of statements in this file matters:

> divert(-1)
> comments go here
> divert(0)dnl - *the dnl means "discard thru newline", saves space in sm.cf*
> VERSIONID(`@(#) filename.mc 1.0 (sendmail) 6/20/02')
> OSTYPE(`Linux')
> *defines and FEATURE macros go here*
> MAILER(local) *Use an MDA (allows local delivery)*
> MAILER(smtp) *Allows SMTP (MTA)*

The most important line to change is:

`` DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA') ``
The default config is for a *client* configuration only (outgoing email but sendmail will not listen for incoming email except from localhost), which is fine if you use POP or IMAP only for email. To fix, remove the "`,Addr=127.0.0.1`" option.

There are other files to edit in this directory. After any changes run `make`:

`local-host-names` **(cw)** - A list of name for which this server will accept email for local delivery. (Basically a list of aliases for the hostname.)

`/etc/alias` - The general use of aliases is to make aliases for *roles* then assign people to the roles (as is sales, cathy above). The `.REDIRECT` feature will bounce the mail to the sender, with a message containing the new email address. (When this file is updated you must run the `newaliases` command, not `make`.)

`virtualusertable` - For more complex aliases such as when you are hosting another domain (or in some cases when relaying), you must use the `virtualusertable`, which will allow aliases that are not allowed in the `alias` file, such as "`postmaster@other.com: postmaster`".

`access` - Control spam by defining which hosts (if any) are allowed to *relay* mail thru this host. Basically this table is a list of email address, domain names, or IP network numbers. For each an action can be specified: `Discard`, `OK`, `RELAY`, `REJECT`, or `550` *`some error message`*.

`trusted-users (ct)` - A list of *trusted* users. Only trusted users can forge the `From` headers. (Useful when a service such as Apache or sendmail itself generates email automatically and you which the email to appear to come from user or some standard alias. (For example `webmaster@mydomain` instead of `root@mydomain` or `nobody@mydomain`.)

`genericstable` - Convert outgoing email names to full names: `wpollock` to "`Wayne Pollock`". Not often needed since many MUAs use both email and full names in the `To:` header.

Some macros to consider changing/adding include:

`FEATURE(`promiscous_relay')`
: Relay mail from anywhere. Bad idea.

`FEATURE(`relay_entire_domain')`
: Relay mail from any local domain name.

`FEATURE(`relay_based_on_MX')`
: Relay mail from any host that lists the local host as the mail server in its MX DNS record.

`FEATURE(`relay_local_from')`
: Relay mail that uses the local host in the FROM: header. Easily forged so don't use.

`FEATURE(`relay_hosts_only')`
: Relay only from hosts (w/o this, domains are listed) in `/etc/mail/relay-domains`.

`FEATURE(`accept_unresolvable_domains')`
: Accept mail from unknown domains (i.e., DNS lookup fails).

`FEATURE(`nocanonify')`
: Skip the DNS lookup altogether. ???

`FEATURE(`access_db')`
: Uses /etc/mail/access to allow mail to listed hosts/domains.

`FEATURE(`blacklist_recipients')`
: Allows `access` DB to block senders too.

`FEATURE(`rbl')`
: Real-time black list (of spammers), see `www.mail-abuse.org`.

Sendmail Satellite Configuration

Use `LOCAL_RELAY(`*`hub`*`)` to have unqualified names ("`joe`") sent to `hub`. Use `MAIL_HUB(`*`hub`*`)` to have local-qualified names ("`joe@local.host`") go to `hub`. Use `SMART_HOST(hub)` to have all other names go through `hub`. (Of course, `hub` must allow relaying from `local.host` or this won't work.)

Other sendmail m4 configuration options

`FEATURE(`redirect')`
: Allows aliases of *`old new`*`.REDIRECT`, and mail arriving for *old* will bounce with a message giving *new*.

`FEATURE(`use_cw_file')`

Use `sendmail.cw` file to list aliases for the localhost, e.g., mail.foo.org, foo.org, bar.org, mail.bar.org.

`FEATURE(`use_ct_file')`

Use sendmail.ct file to list *trusted* users who are allowed to forge FROM headers (root, www, ...).

`MASQUARADE_AS(`foo.org')`

Use foo.org in outbound email headers in lieu of the real local hostname.

`FEATURE(`masquerade_envelope')`

Masquerade addesses in the envelope too.

```
define(`PROCMAIL_MAILER_PATH′, `/usr/bin/procmail′)dnl
FEATURE(local_procmail, `/usr/bin/procmail')
```

Use procmail as local MDA. Procmail is smart with SPAM filters you can set. Use `/etc/procmailrc` and `~/.procmailrc`.

`define(`MAIL_HUB', `mail.hcc.com')`

Incoming mail from mail hub ???

`define(`SMART_HOST', `mail.hcc.com')`

Out-going mail to the hub) ???

Testing the Email MTA Service:

There should be some useful log entries, and some email in root's mailbox. The next test is "`mail root`" command. Next you can try it from another host (*some.other.server*) using **`mail`** command, or **`telnet`** ***`yourhost`*** **`25`**:

```
HELO some.other.server       or EHLO ... for ESMTP
MAIL From:<user@some.other.server> SIZE=35
➥    AUTH=user@some.other.server
RCPT TO: <root@your.server>
DATA
Subject: test
Hey, this works too!
.
QUIT
```

Another way to test is to send a piece of email to one or more *auto-responders*:

```
sa-test@sendmail.net
check-auth@verifier.port25.com
autorespond+dkim@dk.elandsys.com
test@dkimtest.jason.long.name
dktest@exhalus.net
dkim-test@altn.com
dktest@blackops.org
```

The email should be returned to you showing your message in the body of a new message, including all of the header changes that were made in transit.

Another alternative is to send a message to an email service provider such as Gmail or Yahoo!, and view the full text of the message you receive there.

Mail Aliases:

`/etc/aliases` (or `/etc/mail/aliases` or **`/etc/postfix/aliases`**, but location is configurable so make sure you edit the right one!) is used when email arrives from any source (local MUA, remote MTA) to examine and possibly modify the `TO` addresses in the email. Use **`newaliases`** command (or **`postalias aliases`**) after any changes to this file. (The `postmap` command works but issues warnings about the colons in the file. I recommend just using the semi-standard **`newaliases`** command.) This file allows one name in the local domain to be sent to a different email address. (So does the `.forward` file in the user's home directory.) Examples:

```
postmaster: root
abuse: postmaster
virusalert: /dev/null
sales: cathy
billing: jsmith@foo.com
sysadmins: wayne, bob, mike
root: sysadmins
joe: joe@hisNewJob.com.REDIRECT
```
sendmail only feature!

The general use of aliases is to make aliases for *roles* then assign people to the roles (as is "`sales`: `cathy`" above). The `.REDIRECT` feature will bounce the mail to the sender, with a message containing the new email address. (***Only for Sendmail, not Postfix!***) Using Postfix you do this by creating a *relocation map* like this: **`relocated_maps=hash:/etc/postfix/relocated`**. Then edit that file with lines like this: **`joe    joe@hisNewJob.com`**

Like all Postfix *map* files, the human readable file must be converted to a database form. In general use **`postmap`** ***`file`*** but for aliases use **`postalias`** ***`file`*** instead.

> After changes to any map file such as `/etc/aliases`, the MTA daemon must be reloaded or restarted!

For more complex aliases such as when you are hosting another domain (or in some cases when relaying), you must use the `virtual table`, which will allow aliases such as "`postmaster@other.com: postmaster`", which isn't allowed in the alias file. Such an alias will funnel all email for the various postmasters (the ones with virtual domains on your server) to a single account.

POP / IMAP

You might allow POP/IMAP access only to email for all members of your organization. Or you might have several "branch" offices each with its own email server, but configured to send and receive all email through a central mail server. Such **a central server is referred to as a *mail hub*** (or mailhost). Sendmail refers to it as a *smart_host*. **The hub must allow *relaying* to the branch office hubs**.

A *firewall* should be used to prevent other hosts from sending/receiving email. **You will need to open TCP ports (*not* UDP) for `pop3s` (995), `imaps` (993), and maybe `pop3` (110) and `imap` (143) too**.

```
grep -i 'pop3|imap' /etc/services.
```

Edit `/etc/sysconfig/iptables`, copy some TCP line 4 times and change the port numbers, then restart `iptables`. In addition, you will need to add the following to **`/etc/hosts.allow`**:

```
imap imaps pop3 pop3s: 127. .gcaw.org
```

Since your MAA doesn't generally use the default system mailboxes in `/var/mail`, mail put there by the MDA won't be found by your MAA. The MDA must deliver mail where the MAA expects to find it.

Using Dovecot MAA

To configure Dovecot as the MAA, you need to verify that `/etc/pam.d/dovecot` is there and correct (usually just "`auth pam-stack.so service=system-auth`"). Next, make any needed changes to **`dovecot.conf`**. (On Fedora 14 and newer, this one large file is now split into many smaller files. You need to make changes to `/etc/dovecot/conf.d/10-mail.conf`.) The defaults on Fedora seem fine (uses mbox, IMAP, and POP), but you can tweak this if desired; I made these changes (first two lines are for default mbox operation):

```
mail_location = mbox:~/mail:INBOX=/var/mail/%u
mbox_very_dirty_syncs = yes
mail_privileged_group = mail
maildir_stat_dirs = yes # slower but safer
auth_verbose = yes  # for debugging auth problems
mail_debug = yes  # to debug why mail isn't found
```

After setting up your firewall / TCP wrappers access, turn on Dovecot:

```
chkconfig dovecot on; /etc/init.d/dovecot start
```

(`tail maillog` for any problems and fix.) After a basic setup is working, you can go back and make more elaborate changes to suit.

To test your setup, send yourself an email so `/var/mail/`*`user`* MBOX exists and isn't empty. Next, try mailx or mutt, telling the MUA to use IMAP (or POP):

```
MAIL="imap://$USER@localhost/INBOX" mailx
```

This may fail; check the maillog to see why. In my case, there was a Dovecot error message about failing to `chown /home/wpollock/mail/.imap`. The trick for me was to set up that folder for maildirs, including a `.imap` directory, for Dovcot:

```
mkdir -p ~/mail/{new,cur,tmp,.imap/INBOX}
# chgrp -R mail ~/mail/  # May not be needed?
```

(You may not need all those, but it can't hurt, and you will need them later, when you set up maildirs.) You should all those folders to `/etc/skel`, so new accounts get them automatically.

You can also connect to your POP or IMAP port using `telnet` or `nc`. More details on testing can be found below in TestingPopAndImap.

Configuration for Maildirs or IMAP

At this point, no any MUA or MDA is set to use maildirs. Older MUAs such as `mail` don't understand IMAP/POP or maildir mailboxes, so you read your mail using some modern MUA: `alpine` (with the maildir patch), **`mutt`**, nail/mailx, or some GUI MUA such as Thunderbird.

In most cases, to indicate mbox you specify a pathname to a file, and to indicate maildirs you specify a pathname to a directory *including a trailing slash*.

Some MUAs (and some MDAs, and some additional utilities) recognize both `MAIL` and `MAILDIR` environment variables. It can't hurt to set both when using Maildirs.

There is a Perl script **`mb2md`** to convert existing mbox files to maildirs (see `batleth.sapienti-sat.org/projects/mb2md/`). There is a command `maildirmake` to create the (empty) folder structure), but most MDAs will create it as needed.

Dovecot: To have Dovecot use maildirs in the user's home directory, make the following changes to `/etc/dovecot/conf.d/10-mail.conf`:

```
mail_location = maildir:~/Maildir  #trailing slash not needed
# mail_location = mbox:~/mail:INBOX=/var/mail/%u
```

Procmail MDA: To use maildirs for delivery, you need to configure the MDA configured in the MTA. Often this is `procmail`. The sendmail `sendmail.mc` configuration file has a command line to use to invoke `procmail`, and this has one option on Fedora by default that must be removed: "`-Y`" says to assume mbox format. Make the following change in `sendmail.mc`:

< FEATURE(local_procmail,`',`procmail -t **-Y** -a $h -d $u')dnl
----
> FEATURE(local_procmail,`',`procmail -t -a $h -d $u')dnl

Then rebuild the `sendmail.cf` file using `make`, and restart `sendmail`.

Next configure `procmail`. On most systems (including Fedora), `procmail` has no global configuration file by default. You must create one (or use the `~/.procmailrc` files). Add the following to a new **`/etc/procmailrc`** (or any new file in `/etc/procmailrcs/`, depending on your system setup):

```
ORGMAIL=$HOME/Maildir/    Trailing slash says to use maildir
DEFAULT=$HOME/Maildir/
MAILDIR=$HOME/Maildir/
```

Postfix MDA: Postfix uses an internal MDA, "`local(8)`" by default. To have Postfix's built-in MDA will deliver email to a user's home directory in maildir format:

`home_mailbox = Maildir/` *Trailing slash says to use maildir*

(Remember to reload Postfix. Use: `ls -laR ~/Maildir` to see the affect.) To instead have Postfix use another MDA such as `procmail` (which you still need to configure), you change the "`local`" transport in the Postfix `main.cf` file:

```
    mailbox_command = /path/to/procmail
```

You may be able to have all the procmail config info passed on the command line, and avoid editing /etc/procmail*, like this:

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION"
 DEFAULT=$HOME/Maildir/ MAILDIR=$HOME/Maildir
```

`mailx`: To configure nail/mailx to use the new maildirs, add the following to `/etc/mail.rc` or to your `~/.mailrc` (or see the `MAIL` environment variable, below):

```
    set newfolders=maildir
    set folder=Maildir/        Trailing slash says to use maildir
    set MAIL=Maildir/
```

(and comment out the `set newmail=nopoll` line).

To use IMAP instead with nail/mailx, use (or see the `MAIL` environment variable, below) the following configuration:

```
    set folder=imap://user@localhost
```

Mutt: In addition to `~/.muttrc`, you can use the global config in `/etc/Muttrc` and `Muttrc.local` (note the annoying capital "M"!). Add the following to `Muttrc.local` (only the first 4 lines are required, I just like the others) to make it use local maildirs:

```
    set mbox_type=Maildir
    set spoolfile="~/Maildir/"
    set folder="~/Maildir/"
    set maildir_trash=yes
    set sleep_time=3
    color indicator white red
    set allow_ansi = yes
```

If you want `mutt` or `nail` use your IMAP server instead of directly accessing mailboxes (mbox or maildir), add the following in `~/.bash_profile` or`/etc/profile` or `/etc/profile.d/imap.sh`:

```
   export MAIL="imap://$USER@localhost/INBOX"
```

or for pop:

```
    export MAIL="pop://$USER@localhost/"
```

(or use "pops" or "imaps" instead of "pop" and "imap" if you wish.)

Alpine: You can **configure `alpine`** (or `pine`) to use maildirs by rebuilding the source and adding the required patch. However Pine does work with IMAP out of the box. Under `Setup-->config`, configure your `inbox-path` as:

```
inbox-path={wphome1.gcaw.org/novalidate-cert/user=wpollock/ssl}INBOX
user-domain=wphome1.gcaw.org              optional
smtp-server=wphome1.gcaw.org              optional
```

Authenticating Users *(Skip these steps for CTS-2322 email project.)*

To allow MTAs (and some MAAs) to authenticate users, **SASL** is generally used. (Some will use PAM instead.) SASL provides a number of different mechanisms for authentication. You need to configure your MAA and MTA to use one of them (the Fedora defaults are fine for localhost users).

By default dovecot (and other MAAs) will use plain text usernames and passwords to authenticate users against `/etc/passwd` and `/etc/shadow`. This is dangerous! However, using email *from* localhost *to* localhost means the passwords will not traverse your network, and thus is safe.

Dovecot will use a generic SSL cert with `localhost.localdomain` as the server name. You can make your own certificates using the following command:

```
openssl req -new -x509 -nodes -out /tmp/public.pem \
      -keyout /tmp/private.pem -days 3650
cp /tmp/public.pem /usr/share/ssl/certs/dovecot.pem
cp /tmp/private.pem /usr/share/ssl/private/dovecot.pem
service dovecot restart
```

Cyrus imapd MAA: Using with Postfix and SASL *(Skip this if using Dovecot.)*

The simple setup is to use your MAA as your MDA (if it supports it). If using **Cyrus `imapd`**, just make sure it is configured to run at boot time and change postfix to use `cyrus` as the MDA by un-commenting the lines in `main.cf`:

```
mailbox_transport = cyrus
local_recipient_maps =
```

(The second line re-enables alias checking, disabled when changing the MDA.) Then check (and change if needed) the path for the Cyrus-imapd `deliver` MDA program (`locate deliver`) in the `cyrus` transport entry in `master.cf`:

```
/usr/lib/cyrus-imapd/deliver
```

Generally MAAs require knowledge of when new mail arrives or is deleted. It is possible to *poll* for this information but modern systems provide a ***file alteration monitor*** (**FAM**). Programs such as MAAs can register with the FAM and get notifications of changes without any polling. (Not all systems provide FAMs and those that do may provide it as a kernel API, a DLL, or a daemon.)

To allow your MAA to authenticate users **SASL** must be configured. SASL provides a number of different mechanisms for authentication. For some of them you must **make sure the `saslauthd` is started and configured to run at boot time**. You need to configure your MAA to use the chosen authentication method. (The Cyrus defaults are fine for localhost users; examine `imapd.conf`.)

Unlike the normal MDA, Cyrus (and some other MAAs) won't automatically create mailboxes when the first message arrives. Instead you manually create the mailboxes (as user `root` or `cyrus`; set a password) when creating a new email user account:

```
# /etc/init.d/cyrus-imapd start
# /usr/lib/cyrus-imapd/cyradm localhost
IMAP Password: *****
localhost.localdomain> cm user.wpollock
localhost.localdomain> cm user.root
localhost.localdomain> exit
```

Note: Cyrus-imap comes with a certificate for `localhost.localdomain`. So you should get a domain name mismatch warning message with your MUA when you use `imaps` or `pop3s`. To fix this requires generating a new X.509 certificate. You can do this yourself but the certificate will still give a warning about being untrusted. You can get a free valid certificate from LetsEncrypt.org or StartSSL.com (or cacert.org, but you will need to install their public key in your MUA to avoid the warning).

Possible problems:

```
cd /var/lib/imap
rm mailboxes.db
chmod -R g+r .
find . -type d |xargs chmod g+x
chown cyrus.mail /var/spool/imap/stage./
vi /etc/hosts.{allow,deny}  # check maillog
```

Test MAA POP and IMAP:

After reloading the MTA and MAA, use `mail` to send yourself some email. Examine the `maillog` file to see if it worked, and check the mail folders, to make sure the mail was delivered to the mail store correctly. (Use ls and cat, not an MUA.)

To test pop3: `telnet localhost pop3`: (you can use "`nc`" instead of `telnet`, and 110 instead of `pop3`.) Enter the following to test:

```
user accoutname
pass password
retr 1
quit
```

To test imap: `telnet localhost imap`: (or use `nc`, and 143 instead of `imap`.) Then enter the following (including the line numbers) to test it:

```
a01 capability
a02 login accoutname password
a03 list "" "*"
a04 select INBOX
a05 fetch 1:1 ALL
a06 logout
```

Not very secure! You can test the SSL versions the same way, using `s_client`, if you have set up imaps and/or pops:

```
  openssl s_client -connect localhost:imaps # or pop3s
```

To use a different set of users than the passwd/shadow files, one way is to create a database of POP/IMAP users and their passwords in `/etc/cram-md5.pwd`. Set the mode to 0400 (owner root). The syntax is:

```
# comments and blank lines allowed
username<TAB>password
# example:
# joe    foobar
```

(A challenge string is sent from server to client, who encrypts with MD5 password and sends result back, server also encrypts and compares results.) Of course you will need to configure the MAA and MTA to use these user names.

Mail filtering for spam, viruses

Although the mail filtering facilities of postfix are impressive, it is usually better to have separate, dedicated spam and virus scanners. In this configuration, postfix accepts email as normal, and then relays it to **`amavisd`** on port 10024 rather than adding it to the mail queue.

Amavis is not a virus scanner by itself. Rather it is the glue between the mail server and a command-line virus scanning tool. Amavis intercepts an e-mail message and rips it apart, uncompressing, decoding, and storing any attachments in separate files. These are then scanned by the external scanners such as SpamAssassin, SpamPal, and ClamAV. (Amavis has a *plug-in* design.)

Amavis then relays the email back to postfix via port 10025 which then processes as normal. (This is a second instance of postfix's `smtpd`. If amavis sent the mail back to the original `smtpd` on port 25, it would loop it back to Amavis!)

Amavis adds headers to the message that the MTA/MSA uses to deliver/bounce/drop the email. (Note the email flows through `qmgr` twice, and two different `smptd` daemons!)

An interesting perspective is that Postfix is split into an MSA which sends email to Amavis (for virus and spam scanning), which then sends the email to a second Postfix process, this time acting only as an MTA.

To avoid the hassles of piping email to such MTAs and passing envelope addresses on the command-line, a new protocol **LMTP (*local mail transport protocol*)** is used between such MTAs. LMTP is based on ESMTP. Postfix and Amavis communicate using either TCP/IP or Unix Sockets.

Each email messaged is fully expanded out for processing, resulting in an *expansion factor* of around 2 (depending on the amount of compression). This is offset by the *decoding reduction factor* or approximately 7/8. **The formula to estimate required space is:**

*max_email_size *(1+expansion_factor)*max_concurrent_amavis_instances * 7/8*

If your max email size is 10MB (say) and you allow 5 amavis instances concurrently, you need approximately 10MB * 5 * (1+2) * 7/8 = 132MB of temporary disk space. **A significant performance increase is possible if you use `tmpfs` (a RAM disk) mounted at $TEMPBASE (`/var/amavis`).**

Webmail

Setting up web mail is discussed below (after *Apache Setup*).

Fetchmail

`fetchmail` grabs the remote mail using POP3 (port110) from any number of sites, and then hands off the mail to sendmail for local delivery. Add this line:

```
    localhost   RELAY
```

to your `/etc/mail/access` file. The reason this might prove necessary is that sometimes sendmail may even reject relay from the localhost if it's not explicitly authorized.

Mailing lists

The `aliases` file can also be used to create and maintain simple *mailing lists*. For more complex mailing lists use **`majordomo`**, **`SmartList`**, or **`mailman`** instead (avoid the older list-serv and list-proc software).

There are three email addresses associated with any mailing list: *listname*, owner-*listname*, and *listname*-request. The first is where you sent mail so all list member receive it, the second is usually used to report errors with the list, and the last is for requests such as "add/drop me to the list". With sendmail version >= 8, if a mail alias contains a corresponding alias with an "owner-" prefix, bounces and errors for that list will automatically be sent to the owner- address. Other mailing list software may use additional email addresses; for example Mailman uses "*listname*-bounces" and several other email addresses.

A ***list owner*** is the person responsible for the list policies. The owner can change settings such as content filtering options, privacy options, archiving settings, etc. The owner can also add, invite, or remove subscribers to their list.

A ***list moderator*** is the person who can approve or deny postings to the list and/or manage a list of approved posters. (Normally all subscribers can post, but some can be given *read-only* access, and some non-subscribers can be given permission to post.) Not all lists have a human moderator. The owner can simply set up some content filters and allow all subscribers (or anyone at all) to post. The owner is often the moderator as well (and in any case can do whatever the moderator can). A busy list can also have a separate (or even multiple) moderators, who use a moderator password to access some of the admin pages that the owner can access.

Mailman is Gnu software to manage email discussion lists. Mailman gives each mailing list a web page and allows users to subscribe, unsubscribe, etc. over the web. Even the list manager can administer his or her list entirely from the web. Mailman also integrates most things people want to do with mailing lists, including archiving, mail-to-news gateways, integrated bounce handling, spam prevention, email-based admin commands, direct SMTP delivery (with fast bulk mailing), support for virtual domains, and more. Mailman's website (including the install and setup documentation) is at `www.list.org`.

To install and use Mailman you will need Python, a working web server, and a working MTA. To get everything running once you've installed the Mailman software, you need to hook Mailman up to both your web server and your mail system. If you plan on running your mail and web servers on different machines (sharing Mailman installations via NFS) be sure that the clocks on those two machines are synchronized closely. The RPM package for mailman takes care of the Apache setup for you, as well as mailman's log file rotation. Most files are installed in `/usr/lib/mailman`, with lots of HTML documentation in `/usr/share/doc/mailman*/admin/doc/`.

Mailman sets up a web page for each list for list members (subscribers) to use to subscribe, manage their options, browse the list archives, or unsubscribe. As usual users can also use special email messages to do this.

Warning! Most mailing list software uses the envelope *from* address to determine the user's identity, not the contents of any header or the body. Whenever you subscribe to some list be sure to save the complete email someplace. If your email address changes, you may need to "fake" the envelope from address to unsubscribe! Using a web interface (which allows you to enter your email address and a password) will be easier.

In addition to a page for list users Mailman sets up pages for the owner (or administrator), the moderator(s), and users. The URLs for the web pages for *mylist* would be:

http://hostname/mailman/admin/mylist/	*the admin page*
http://hostname/mailman/admindb/mylist/	*the moderator page*
http://hostname/mailman/listinfo/mylist/	*the user's page*
http://hostname/pipermail/mylist/	*the list's public archives*
http://hostname/mailman/private/mylist/	*the list's private archives*

(A little work with Apache and URL rewrites, and you can use URLs such as "http://mylist.domainname/" (listinfo page), ".../admin", ".../archive", and so on.)

Once installed, you must set a site (admin) password with `/usr/lib/mailman/bin/mmsitepass` *secret*. Next run `.../update`. If your site hosts many lists by different owners, create a site-wide list "mailman" so you can contact them easily (may be done automatically). Do this using the web interface.

Finally start the mailman daemon and make sure it is set to run at boot time. (This won't start until you've created the "mailman" list!)

The web interface doesn't appear to have a way to delete a list. Use .../`bin/rmlist [--archives]` *`nameOfList`* and possibly edit the mail `aliases` file.

Mail Server Maintenance

Mail servers use several files to control operation. You can edit these files (then restart/reload the service). MTAs don't read the plain-text files you edit, you must convert them to a more efficient form known as a ***map*** first. (This is really just an ISAM file.) How this is done depends on the file you've updated. (Fail-safe: make sure all files are updated at boot time, from `rc.local` if necessary.)

Use `cpio`/`tar`/`pax` via `cron` to backup mail store nightly, after MTA and MAA are shutdown:

```
#!/bin/sh
cd /
find ./var/spool/imap ./var/lib/imap ./etc -depth \
   | cpio -oaBv >/dev/st0  2>/var/log/cpio.log
```

The log file that is created basically just lists the files were backed up, which is not real convenient for notifying the SA of the status of the backup.

Mail Control

Have an *Acceptable Use Policy* (AUP) available from your public web server (and anonymous FTP server). Samples: `www.us.uu.net/support/usepolicy/`, `www.sprint.net/acceptableuse.htm`.

Run `identd`. This daemon listens at port `113` for ID requests from remote sites. (So open the firewall hole for this port.) `identd` will tell remote sites the username of the user(s) currently connected to a given Src-IP/Dest-IP,Src-port/Dest-port connection. Verify the correct entry for this is in `/etc/services` (`grep ^auth`), and enabled in `xinetd` or `init.d`.

Configure MTA to control relaying. You should pick the most restrictive option that will work for your situation.

Lecture 16 — Apache Web Server Setup and Web Mail Configuration

A *web server* accepts requests from clients (known as *web browsers*) for specific documents, often called *web pages*. Usually these are text documents with **HTML** (*HyperText Markup Language*) formatting, but may be any type of document. In addition, a web server can generate documents dynamically, as the result of running separate (external) programs, or performing database lookups.

The requests include a **URL** (*Uniform Resource Locator*), which uniquely identifies a document on the Internet. A URL is actually a type of URI (*Uniform Resource Identifier*), which in turn is a type of IRI (*International Resource Identifier*; mention *punycode*). A URL has several parts, including a *protocol* (such as `http://`, `https://`, `ftp://`, etc.), a *web server* (either an IP address or a DNS name such as `www.example.com`, `mail.example.com`, etc.), an optional *port number* (by default "`:80`" for HTTP, "`:443`" for HTTPS), a *pathname*, and optional other data. A typical example might be `http://wpollock.example.com/somedocument.html`.

A URL can also point to a directory rather than a document. In this case, it is up to the web server to determine what document to return to the client. Some possibilities include a (nicely formatted) directory listing, an error message, or some default document. For Apache a default document is named "**`index.html`**" (or some variation such as "`index.htm`" or "`index.php`").

A default web page for the top directory of the web server is called the server's ***homepage***. This is the page you get with a URL similar to "`http://servername/`".) By default Apache ships with a default "test" homepage. You will probably need to change that!

The requests and responses are sent via TCP using **HTTP (*HyperText Transmission Protocol*)**. A request packet may include **form data** in addition to required headers that the user entered on some web page that allows user input (a form that includes a submit button). The response from the web server will include some headers and the document's contents. The most important header in the response is the Status header, which is a 3 digit number. **A value of 200 means no problems, 403 means permission denied (usually because you set the wrong permissions), 404 means document not found**, and so on.

> To troubleshoot HTTP and web server configuration problems, use the Firefox extensions called "Live HTTP Headers" and "Modify Headers".

Apache is the world's most popular web server by a wide margin. (NginX is closing in, however.) Windows and other versions are available, but these may be limited in what features are provided (due to limitations of the operating system). The Apache foundation (apache.org) hosts many projects besides a web server (httpd.apache.org), but when we say "Apache", we generally mean the `httpd` web server.

There are two major versions of apache. Version 1 has a reputation for rock-solid performance and is still used for that reason. Version 2 is better only it hasn't been used for as long. Version 2 is highly modular. This allows you to easily add or remove functionality as needed for your system. Even the configuration file is now modular.

Apache Configuration

To set up the Apache web server is easy, the default configuration just works. Most of the configuration is listed in the file `/etc/httpd/conf/httpd.conf`. However that file has an "`Include conf.d/*.conf`" to include the snippets in those files, as if they were part of the main file.

The configuration consists of *directives*. These can be applied to the whole web server (such as the `User` and `Group` to run `httpd` as), specific directories, filenames, or URLs. Each block of directives is contained in HTML-like tags, such as `<Directory foo>` ... `</Directory>`, `<File *.txt>` ... `</File>`, or `<Location bar>` ... `</Location>` (for URLs). The tags are generally not case-sensitive.

Specific directives are made available by loading some Apache *module*. For example, to use any of the SSL directives to enable and configure HTTPS, you must load `ssl_module` first. You can easily tune Apache by not loading modules you don't need.

> The documentation for Apache is well organized. You can lookup directives alphabetically, by type, or by module. There are plenty of examples, and the config file(s) include many comments to explain.

To start with, go through `httpd.conf` and examine the global directive defaults. Some of the ones I generally consider changing are:

```
ServerTokens Prod
MaxSpare{Servers,Threads}
ExtendedStatus On
ServerAdmin webmaster@localhost    set email alias for this
ServerName
DocumentRoot
DirectoryIndex   add index.htm, index.php, ...
HostnameLookups Off
ServerSignature EMail
```

You can run a command to check the syntax of your configuration; on current versions, that is "`httpd -t`". If Apache fails to start, disable files in `conf.d` one at a time until it starts. (I often have a problem when a DB is not configured correctly, for say SquirrelMail, RT3, Wiki, etc.) Often the error messages in httpd's log, or the output of `journalctl` can provide a clue what caused the failure.

Apache include special directives for security. For any file, directory, or location, you can limit access in many ways: limit HTTP methods, limit source IP addresses, even password protection. These (and some other) directives can also be placed in special per-directory files (default name is "`.htaccess`"). This allows webauthors to tweak access without providing them access to `httpd.conf`, however using this feature greatly slows Apache down.

Finally, Apache can be configured to serve up multiple websites. This can be done by name (only requires a single IP address, with many DNS names) or by IP address. Using SSL (HTTPS) requires using one IP address per virtual website.

Webmail Configuration: Apache Configuration

Step one: turn on Apache. Do a "`ps -ef | grep [h]ttpd`". Qu: why so many processes? Ans: it takes time to start a process so a master httpd process starts of a bunch of spares. When a HTTP request is received, the master process hands it off to one of the spares.

Now view the server's *homepage*. If this works, your server is working fine! If not, you need to examine the `/var/log/httpd/error.log` file to see the error messages.

Best advice is to configure apache for SSL so users can authenticate and send/read email securely. However, this requires a certificate, so we won't do this now. Note this also requires firewall holes for ports 80 (and 443 when HTTPS is used).

You will need to enable PHP if not on by default. (**Run through `httpd.conf`, show `conf.d/*`, and discuss admin using GUI.**)

You can setup Apache for Squirrelmail webmail with

```
    Alias /webmail /usr/share/squirrelmail
```

An alternative is to configure a *virtual host* with a name such as `webmail.gcaw.org`. In Apache, a given server can server multiple websites, each is a virtual host. There are two techniques for this:

- IP-based virtual hosts require one IP address per website. This can be done using IP alias (multiple IP addresses per NIC) or (rarely) multiple NICs. This method works well with SSL, (HTTPS, since the host name is included in a security certificate, and that name must map to the IP address used.
- Name-based virtual hosts relies on HTTP 1.1 request headers, which identify the server name the client is trying to reach. This method scales up well. Note that even with name-based virtual hosting, the first step is to match the virtual host with an IP address.

  If you have multiple virtual hosts using the same IP (commonly, using "*" for the IP address), the `ServerName` (or `ServerAlias`) directive within each `VirtualHost` block is used. (Note, if you forget to include such a directive, the global one is inherited!) Use `ServerAlias` when a given website has multiple names.

Any HTTP request that doesn't match any virtual host define, will use the global configuration. (See httpd.apache.org/docs/2.4/vhosts/ for details.)

A `CNAME` DNS record can be added for `http://webmail.gcaw.org/`. (In real life, you'll want to use `https` instead.) Or use URL re-writing. Reload DNS and check the logs for errors.

Since this web server is only for webmail, you can avoid any virtual domains. Simply set the directives below in the global section of `httpd.conf`.

You then configure apache to serve this virtual domain. The first way uses HTTP and the second uses HTTPS (only use one, if either is used):

```
<VirtualHost *:80>
    ServerAdmin webmaster@gcaw.org
    ServerName wpserver.gcaw.org
    DocumentRoot /usr/share/squirrelmail
</VirtualHost>

<VirtualHost webmail.gcaw.org:443>
    ServerAdmin webmaster@gcaw.org
    ServerName webmail.gcaw.org
    DocumentRoot /usr/share/squirrelmail
</VirtualHost>
```

Squirrelmail

This is a set of PHP scripts that are run by Apache when a user visits the `http://`*`hostname`*`/webmail/` URL. Thus the Apache user must access the squirrelmail directories: the config directory, the mailstore (but since Dovecot can't access maildirs, configure squirrelmail to use IMAP), and the attachments directory. Note that uploaded but unsent attachments stay forever, so a `cron` job is used to clean the attachment directory (via `tmpwatch(8)`).

Using HTTPS requires some changes to the apache setup made by the RPM, to force (or at least allow) SSL access to `/webmail`.

Squirrelmail presents HTML 4.0 forms you fill out and submit to log in, which send your username and password in plain text. These are passed to your IMAP or IMAPS server. IMAP should be safe since both the Squirrelmail/HTTP[S] MUA and the IMAP server run on the same host, so sensitive data doesn't traverse any network.

Each page loaded in your browser makes a new IMAP/IMAPS connection, which could be a performance problem. It is possible to cache IMAP sessions to reuse them (requires cookies to store the authentication data.) See `www.imapproxy.org/` for details.

You should pick the two directories to be outside of your website but with appropriate ownership and permission so Apache processes can access them. You also need to make sure the web server can access the IMAP/IMAPS server. This is no problem if both are on the same server (but may require firewall/TCP Wrappers holes if different servers.)

Webmail Configuration Steps:

`cd` to `/usr/share/squirrelmail`, and from there **run the `config/conf.pl`** program. (Turn off color (hit 'C') if text doesn't show up!)

Run through the configuration menus, especially remember to use the "**`D`**" choice to setup defaults for your IMAP server ("D" for Dovecot). You can configure web-based administration and add additional feature modules as well. Go to the **`SquirrelMail.org`** website for documentation on these.

Next, check the Apache configuration for Squirrelmail; it should be in the file `/etc/httpd/conf.d/squirrelmail.conf`. The default in that file may need to be changed. For example, on the version I used, the configuration uses "`/webmail/`" as the alias for Squirrelmail. This is fine. But the next part automatically redirects the user from `HTTP://` to `HTTPS://`. Since that isn't setup (for now), you need to comment out that part. After making any changes, check the result using "`httpd -S`". If okay, reload/restart Apache to use the newly added/modified configuration. Save a `diff` listing of your changes in your system journal as well.

```
#
# SquirrelMail is a webmail package written in PHP.
#

Alias /webmail /usr/share/squirrelmail

<Directory
"/usr/share/squirrelmail/plugins/squirrelspell/modules">
  Deny from all
</Directory>

# this section makes squirrelmail use https connections
only, for
#   this you
# need to have mod_ssl installed. If you want to use
unsecure http
# connections, just remove this section:

#<Directory /usr/share/squirrelmail>
#  RewriteEngine  on
#  RewriteCond    %{HTTPS} !=on
#  RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

#</Directory>
```

(Instead of commenting out that section, you can consider surrounding that part with:

```
<IfModule ...>
   ...
</IfModule>
```

Then if you later enable SSL, you won't need to remember to update this configuration.)

Test the setup by restarting/reloading apache (`httpd`) and point your web browser to **`http://`*`hostname`*`/webmail/src/configtest.php`**. If that works, try **`http://`*`hostname`*`/webmail/`**. (Or http://webmail.*domain*/ if you configured that.) (If it doesn't work, edit /etc/httpd/conf.d/squirrelmail.conf.)

One alternative to Squirrelmail (there are several) is *sqwebmail*, the webmail component of Courrier mail. It bypasses IMAP by reading maildirs directly. Of course, this means that the web server and IMAP server are on the same host. Sqwebmail uses a root setuid program to access the maildirs directly (in the user's home directories). See www.courier-mta.org/sqwebmail/.

Nginx

While fast, Apache is designed for flexibility. On the other hand, Nginx is designed for speed. It does handle most standard web server tasks, including webmail.

Some sites use Nginx as a proxy (or front-end) for Apache. Using a cluster of Nginx webservers, outfitted with Memcached distributed memory cache, allows very short response times and many concurrent sessions. More complex queries (say for dynamic content) are forwarded to the Apache server(s).

On Fedora installing Nginx is simple: `yum install nginx`. Then start it. The default configuration serves content from `/usr/share/nginx/html`. You can edit the configuration file in `/etc/nginx/nginx.conf`. The syntax is:

```
section-name {
    directive;
    ...
}
...
```