QMAIL & SMTP:
A Secure Application
for an Unsecure Protocol

Orr Dunkelman

orrd@vipe.technion.ac.il

SMTP, MUA and MTA – Speak English

Whenever we deal with protocols we (=Internet geeks) like to use abbreviations. However, this makes the understanding of what we say hard if you are not used to those abv. or if you lack the intuition to guess what those initials mean.
So here is a small dictionary that will be used throughout this lecture:

IP	Internet Protocol
TCP	Transport Control Protocol
UDP	User Datagram Protocol
DNS	Domain Name Service
SMTP	Simple Mail Transfer Protocol
MTA	Message Transfer Agent
MUA	Message User Agent
MDA	Mail Delivery Agent

SMTP, MUA and MTA – Speak English

Whenever we deal with protocols we (=Internet geeks) like to use abbreviations. However, this makes the understanding of what we say hard if you are not used to those abv. or if you lack the intuition to guess what those initials mean.

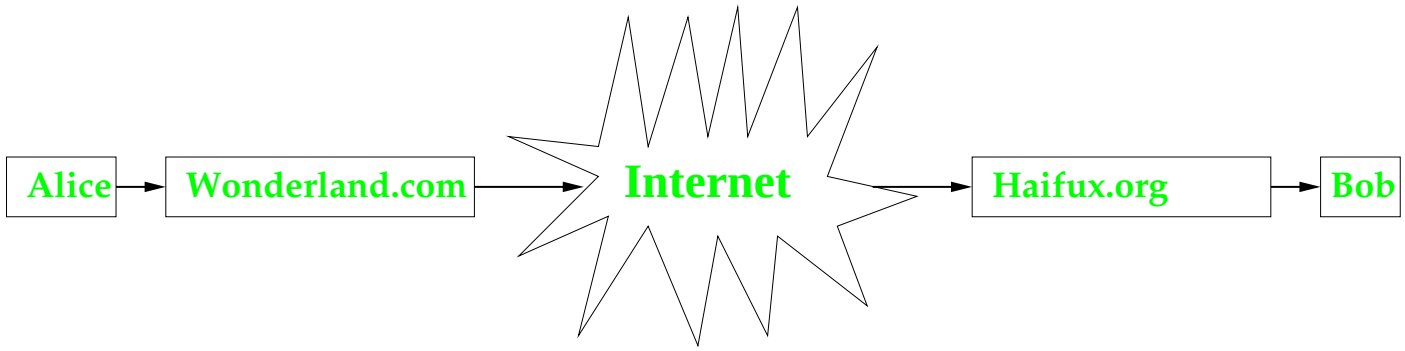
So here is a small dictionary that will be used throughout this lecture:

IP	Internet Protocol
TCP	Transport Control Protocol
UDP	User Datagram Protocol
DNS	Domain Name Service
SMTP	Simple Mail Transfer Protocol
MTA	Message Transfer Agent
MUA	Message User Agent
MDA	Mail Delivery Agent

Sending Mail – The Never-ending Adventure

When Alice sends Bob a message she is likely to use many protocols. First she types the mail on her machine, then she sends the mail to her "out-going" mail server. Then her mail server uses some Internet magic to deliver her message to Bob's "in-going" mail server. Bob's server accepts the message, and asks the MDA to put it into Bob's mail box.

Later Bob will read his mail box using some other application.



What Protocols are Involved?

There are 3 logical connections: Between Alice and Wonderland.com, Between Wonderland.com and Haifux.org and between Haifux.org and Bob.

Alice and Wonderland.com communicate using an MUA-MTA protocol like POP3 or IMAP. Similarly, Bob and Haifux.org communicate using such protocol. Those MUA-MTA protocols are usually local, and in some cases they are actually synchronized reading and writing from a file.

The part of the logical connection between Wonderland.com and Haifux.org (the MTA-MTA connection) is done using SMTP.

SMTP – "Simple" Mail Transfer Protocol

RFC 821, dated August 1982, defines SMTP. The idea of using a reliable, order-preserving transport protocol to transfer mail electronically is just a "simple" implementation of data transfer.

The RFC addresses only one problem: How to transfer mail from one SMTP Server to another.

The RFC does not tell you anything of the form: How to know what SMTP server to connect to, what to do when it is not accessible, how to transfer binary data, etc.

SMTP for the Lazy

The minimal SMTP server must support the following commands:

HELO	Starting an SMTP connection
MAIL	A new email from sender
RCPT	The recipient is ...
DATA	Here comes the mail
RSET	Reset connection (the session is aborted)
NOOP	No Operation
QUIT	Disconnect

SMTPing with 7 Commands

When server A connects to server B, it starts a transport connection (e.g., a TCP connection). Then B sends to A:

220 B Service ready

For any SMTP operation there is a response. The responses are indexed by some 3-digit code, but usually the response contains also text readable by geeks.

After receiving 220, A send to B the following command:

HELO A

B responds with "250 OK" (or sometimes with "250 welcome A, pleased to meet you").

SMTPing with 7 Commands (cont.)

After the exchange of the greetings A starts a mail:

1. MAIL FROM: <reverse-path>
   <reverse-path> contains the trail the email did, when the leftmost value is A (the mail originated from the rightmost value).
2. RCPT TO: <forward-path>
   <forward-path> is usually the final address of the message. However, sometimes we would like it to be something else.
3. After the second OK (or more precisely 250), A can send the mail, by issuing the command – "DATA". DATA is answered by 354 ("send the mail data, end with .")

SMTPing with 7 Commands (cont.)

After sending one email, you can either quit, perform nothing, send a new mail (repeat steps 1–3), or issue a reset command.

Note that those commands are enough to send emails from one server to another.

SMTPing with More than 7 Commands

There are more than 7 SMTP commands, and there are many replies (sometimes the answer is not OK, but it is not an error). The commands are:

- TURN – The recipient server (B) decides it wants to send mail to A, and asks them to switch the roles.
- HELP – A reveals the fact that he is a lamer and doesn't know the entire syntax of the commands by heart.
- SEND – Sends the mail to terminal.
- SOML – Send Or MaiL. If the recipient is active and accepts messages on her terminal, she gets it on her terminal. Otherwise, it is kept in her mailbox.
- SAML – Send And MaiL.

SMTPing with More than 7 Commands

- EXPN – EXPaNd the mail addresses related to a mailing list that is managed by the server B.
- VRFY – VeRiFY mail address of a user. A response can be either OK (250, user is here, use this mail address), User no local (251, I know a system with that user, use this mail address), Don't know (550), There is some ambiguity (553).

There is a defined set of answers to those queries. Some of them are OK, some are error messages, but some can be of the form "send this to this address" or "send this to that server".

SMTP has many extensions. Some define how to transfer attachments, some define how to announce successful deliveries, and some just add functionality.

Routing of Mails

It is important to understand how a mailer at your system knows who to approach in order to send an email to some other server. RFCs 882,883,973,974 (all now obsolete) define how we know who accepts messages for a given server.

First there was a distinction between the MF (mail forwarder) and MD (mail destination). The MF was a server that is accessed by the public to send messages to the final destination – the MD servers (where the recipient's mailbox is stored).

Today, we use relays and Mail EXchangers. An MX entry is the name of the server handling the mails of the domain.

It is possible to have more than one MX entry for a given server, and in that case the sender must decide what to do.

Routing of Mails (cont.)

Each DNS record containing an MX record has two values: priority and server name.

The sending server performs a DNS query to get the MX records of the destination server. If the answer is empty it can either err, or try and assume that the same domain name handles the mail services.

Once there are answers, the sending server approaches the servers in non-decreasing priority and tries to send them the message. If the sending operation fails, they can either err, or continue to the next server on the list.

When all fails, the server can decide to retry later, or to err immediately.

Routing of Mails (cont.)

Relays are the servers with the non-minimal priority. Those servers are contacted when the "real" mail server(s) are down, unaccessible, malfunction, bombed by a nuclear bomb, etc. They need to be configured to know that they perform as relays.

For example, Haifux.org has the following MX records:

```
haifux.org.      IN MX 10 haifux.org.
haifux.org.      IN MX 20 hamakor.org.il.
```

This means that when you send a mail to haifux@haifux.org, your SMTP server tries to approach haifux.org (132.69.253.254). If this server is unaccessible, then your mailer might try hamakor.org.il (192.117.122.104).

Loose Ends

We shall not cover the following topics:

- How attachments are attached to mails (see RFC 2045).
- How a relay saves mails, and how it transfers them back when the server is back.
- How DNSes are updated.
- MUA-MTA protocols.
- How mails are kept in the user mail box.

SMTP Servers – History of Security Holes

In the beginning there was Sendmail. And Sendmail was widely used. But badly written.
And many bugs existed. Some used to break systems.
And the Sendmail was fixed. And re-broken. And fixed. And re-broken.
And the DJB came to see the SMTP and Sendmail, which the children of men builded. And the DJB said: Behold, the people is one, and they have all one SMTP server; and this they begin to do: and now nothing will be protected at their servers, which they have imagined to be secure.
So the DJB said: "Let there be a secure SMTP server." and for days DJB sat and coded until there was Qmail.
And Qmail was secure, but hard to use.

SMTP Servers

There are three leading SMTP servers: Sendmail, POSTFIX and Qmail.

Name	**Sendmail**	**POSTFIX**	**qmail**
Current Version	8.10.12	2.0.16	1.03
Release Date	17/SEP/03	13/SEP/03	15/JUN/1998
Security Holes (2003)	213	19	0
Percentage of Market	38.78%	2.53%	9.32%
Code lines	118000	110000	16600

Microsoft's programs (Exchange, IIS) have 19.79% of the market.

QMAIL's Security

As noted by the observant reader, qmail-1.03 has no known security holes since it was released in 1998.

This fact is **very unique**. Few programs (that are used) have no known security holes in them. All but two are new. (The other one is djbnds, DJBs DNS server application).

Also the software has a guarantee. 500$ were suggested to the person who compromises the server's security. The prize was enlarged to 1000$, and never claimed.

QMAIL's Security – Reasons

The main ideas behind qmail's security are (taken from qmail-1.03/SECURITY):

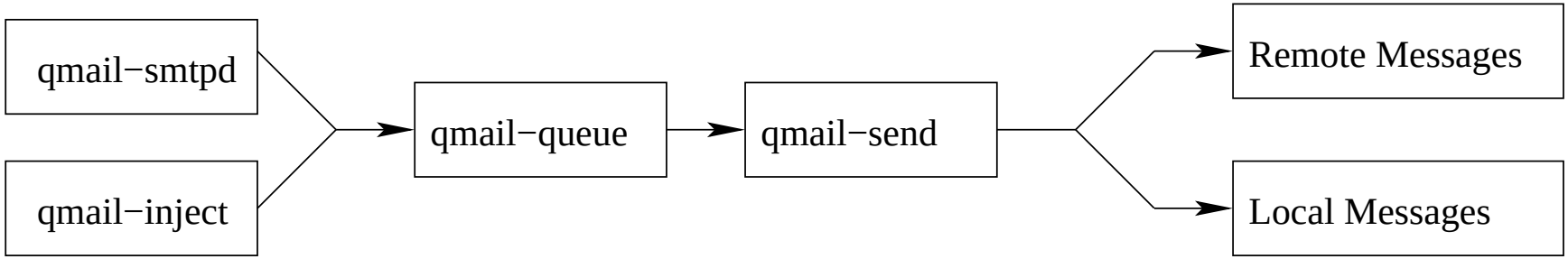
- Do as little as possible in setuid programs.
- Do as little as possible as root.
- Move separate functions into mutually untrusting programs.
- Keep it simple, stupid.

And the best idea, which gives security:

- Write bug-free code.

How Qmail Works

Qmail has a chain of handling messages:



A message enters the queue (either from the network or from the local server). Then it is being processed by qmail-send. According to the destination the qmail-send decides which of the chains would handle the message – the remote chain or the local chain.

The Path of the Message

When I run pine on VIPE and send a message, it is traditionally delivered to Sendmail to deliver it to the recipient. When Sendmail is not there, there is the qmail-inject – a local program that parses the message, put it in an envelope, and transfer the envelope to the queue.

Envelopes are actually a logical wrap of the message. They contain the origin address, destination address, what kind of a destination is?

Take a Number – Enter the Queue

Each message that enters the queue passes the following "Via Dolorosa":

1. qmail-queue creates a new file under pid/ with a unique name and inode number. For example, pid/1234.
2. After its creation qmail-queue passes the file from pid/ to mess/.
3. qmail-queue writes the message into the file (mess/1234).
4. qmail-queue builds the message's envelope in intd/ (intd/1234).
5. qmail-queue links todo/1234 to intd/1234. This is actually the act of inserting the mail into the queue.

Each of these files has 24 hours to live. Once those 24 hours pass – the files are cleaned, and removed.

Order in My Queue

Each message is in one out of 5 states. The state can be determined be the existence (or non-existence) of the various files qmail uses:

- mess/1234: the message.
- todo/1234: the envelope: where the message came from, where it's going.
- intd/1234: the envelope, under construction by qmail-queue.
- info/1234: the envelope sender address, after preprocessing.
- local/1234: local envelope recipient addresses, after preprocessing.
- remote/1234: remote envelope recipient addresses, after preprocessing.
- bounce/1234: permanent delivery errors.

What State I am in?

State	mess	intd	todo	info	local	remote	bounce	
S1	-	-	-	-	-	-	-	No message
S2	+	-	-	-	-	-	-	Message is introduced
S3	+	+	-	-	-	-	-	Building an Envelope
S4	+	?	+	?	?	?	-	Message is queued
S5	+	-	-	+	?	?	?	Preprocessing

'+' means that the file exist, '-' means the file does not exist, and '?' means the file might exist.

When a message enters state S4 qmail-send detects todo/1234, it removes info/1234, local/1234 and remote/1234 if they existed before. After analyzing todo/1234, qmail-send creates info/1234 and if necessary local/1234 and/or remote/1234.

Preprocessing of a Message

Messages at the preprocessing state (S5), are handled as follows: Each address in remote/1234 and in local/1234 is marked either as "DONE" or (surprisingly) "NOT DONE".

In case of a permanent error trying to send the message, qmail-send create bounce/1234, and mark the address as DONE. At its own time, qmail-send may decide to send a bounce message, using the information in bounce/1234 and mess/1234.

When all address are DONE the local/1234 (and similarly remote/1234) is deleted. Then the bounce/1234 is handled. Once it is deleted, qmail deletes info/1234 (moving back to S2), and finally removes mess/1234 (back to state S1).

A Bit About Process Communication

Note that qmail-queue needs to inform qmail-send that there is a new message in the queue. This is done by using a named pipe — lock/trigger.

Before scanning todo/ qmail-send opens lock/trigger O_NDELAY for reading. It then selects for readability on lock/trigger. qmail-queue pulls the trigger by writing a byte O_NDELAY to lock/trigger. This makes lock/trigger readable and wakes up qmail-send. Before scanning todo/ again, qmail-send closes and reopens lock/trigger.

Remote Messages Handling

Qmail has a mechanism to handle remote destinated messages. There are agents that are responsible for sending those messages to other servers — the qmail-remote program. That program is controlled by the qmail-rspawn which spawns those remote agents.

The qmail-remote program initiates a TCP connection with the destination and negotiate SMTP connection with it.

Despite what I have said in the lecture — the messages are not returned to the smtp daemon for handling.

Local Messages Handling

The mechanism to handle local messages is similar in nature — the qmail-local program is responsible to put the new message into the user's mailbox, while qmail-lspawn makes sure nothing really gets stuck.

The main difference between the two (remote and local delivery) is the fact that for remote delivery you do not need to have root access. As the local agents need to write into user's accounts, they better have enough permissions for doing so.

The Journey of a Local Message

First qmail-lspawn verifies that such a user or alias exist on the machine. If this is a user, qmail-local verifies that there is account to the user. Then it checks that the uid of that user is not 0. **qmail will not deliver messages to users with uid 0**.

Then qmail checks that the home directory of the user is visible to the qmail group, and that it is owned by the user (e.g., that ∼orrd belongs to user orrd).

Then the permissions of the message file is changed to be owned by the user.

After this has been done, a copy of qmail-local is being created, with the right paramaters to assure delivery to the user. As in this moment all the data is accessible by the user (both the message and the mailbox), the qmail-local is created with the uid of the user.

Configuring qmail

Unlike Sendmail, to configure qmail, you do not require to master the art of regular expressions. There are several important configuration files that need to be configured (all are under the control directory):

- me – the name of localhost.
- defualtdomain – this is the suffix added automatically to any mail address that has no dots in it after the sign.
- plusdomain – this is the suffix added automatically to any address that has + at the end (a nice trick of qmail).
- locals – logical names that this machine accepts mail for. For example, vipe has a line of haifux.org in its locals file.
- virtualdomains – a list of virtual domains the server handles, and who deals with them internally.

Configuring qmail (cont.)

- rcpthosts – a list of domains the SMTP server accepts messages to. Those might be remote domains also.

Local Deliveries and .qmail

Local deliveries are put in the user's mailbox.

Before this is done, the user's .qmail file is parsed. Here you can call for procmail and other mail-related processes.

An interesting feature by qmail is the ability to bind user-anything@machine to user@machine. In case an address is in the form of user-suffix, and this email is not a valid email on the machine, qmail checks whether user@machine would like to get it.

This is done by trying to access .qmail-suffix in user.

So on qmail system, if my user is orrd, I can have orrd-www-haifux-org@machine and define in ∼orrd/.qmail-www-haifux-org or in ∼orrd/.qmail-www-haifux or in ... the way to handle such mails.

Handling Aliases and Virtual Domains

Without entering too much into qmail's witchcraft, for good aliasing and virtual domains support, you should use qmsmac (another program).

qmsmac lets you define aliases and handle virtual domains (even though you can handle virtual domains without qmsmac). The idea is to define in the alias environment the configuration of the virtual domain.

References

[1] Jonathan B. Postel, *Simple Mail Transfer Protocol*, RFC 821.

[2] RFCs 822, 882, 883, 973, 974, 1123, 1651, 1652, 1854, 1893, 1939, 2045, 2821.

[3] David Schweikert, *MTA, MUA, MDA, SMTP and why I sleep well knowing that we use Postfix.*

[4] *E-Mail Server Survey Results for April 2003*, available at http://www.credentia.cc/surveys/smtp/200304/.

[5] *Security Focus*, *http://www.securityfocus.com/.*

[6] Daniel J. Bernstein, qmail-1.03.

[7] Daniel J. Bernstein, qmsmac-0.71.