Electronic mail security

Diana Berbecaru
<diana.berbecaru@polito.it>

Politecnico di Torino
Dip. Automatica e Informatica

MHS (Message Handling System)

MTA (Message Transfer Agent)
MS (Message Store)
MUA (Message User Agent)

Basic functionality

- Email user (the client) makes an arrangement with a remote Mail Transfer Agent (MTA) server for the receipt and storage of the client’s emails.
- The MTA, using a suitable mail delivery agent (MDA), adds email messages to the client’s storage as they arrive
  - The remote mail storage (MS) is referred to as the user’s mailbox.
  - The default setting on many Unix systems is for the mail server to store formatted messages in mbox, within the user’s HOME directory. Users of the system can log-in and run a mail client on the same computer that hosts their mailboxes; in which case, the server is not actually remote, other than in a generic sense.
### RFC-822 messages
- only US-ASCII characters on 7 bits
- lines terminated by `<CR> <LF>`
- messages composed by header + body
- **header**
  - keywords at the beginning of the line
  - continuation lines start with a space
- **body**
  - separated from the header by an empty line
  - contains the message

### Header RFC-822
- `From:` sender (logical)
- `Sender:` sender (operational)
- `Organization:` organization of the sender
- `To:` destination
- `Subject:` subject
- `Date:` date and hour of sending
- `Received:` intermediate steps
- `Message-Id:` sending ID
- `CC:` copy to
- `Bcc:` copy (hidden) to
- `Return-Receipt-To:` return receipt to

### An SMTP / RFC-822 example
```
telnet duke.colorado.edu 25
  Trying ....
  Connected to duke.colorado.edu
  Escape character is '^]'
  220 duke.colorado.edu ...
HELO leonardo.polito.it
  250 Hello leonardo.polito.it ... Nice to meet you!
MAIL FROM: dianab
  250 cat ... Sender ok
RCPT TO: franz
  250 franz ... Recipient ok
DATA
  354 Enter mail, end with "." on a line by itself
```
From: dianab@polito.it (Diana Berbecaru)
To: franz@duke.colorado.edu
Subject: vacation

Hello Francesco,
This is a test mail message.
Diana

250 Ok
QUIT
221 duke.colorado.edu closing connection
connection closed by foreign host

ESMTP

- Extended SMTP, defined in RFC-1869 and subsequently incorporated (with SMTP) in RFC-2821
- the base protocol and the communication channel is the same
- the ESMTP servers must identify themselves to the communicating parties with:
  EHLO hostname
- if the receiving server speaks ESMTP, it must declare the extensions that it supports, one per line, in its response to EHLO

Standard ESMTP extensions

- 8BITMIME
  (RFC-1652) indicates that it accepts MIME messages that have in the body octet-aligned data
- SIZE dimension
  MAIL FROM: address SIZE=dimension
  (RFC-1870) declares the maximum dimension accepted by the server or the dimension of the message to be sent
- PIPELINING
  (RFC-1854) sending of more commands without the need to wait for the response for each one (exception: the ones that determine the status change)
Positive ESMTP examples

- ESMTP mailer without extensions:
  220 mail.polito.it - SMTP service ready
  EHLO mailer.x.com
  250 Hello mailer.x.com - nice to meet you!

- ESMTP mailer with extensions:
  220 mail.polito.it - SMTP service ready
  EHLO mailer.x.com
  250-Hello mailer.x.com - nice to meet you!
  250-EXPN
  250 8BITMIME

Negative ESMTP example

- The mailer does not know the ESMTP protocol:
  220 mail.polito.it - SMTP service ready
  EHLO mailer.x.com
  500 Command not recognized: EHLO

SMTP-Auth

- Extension of ESMTP defined in RFC-2554
- Command AUTH + options of MAIL FROM
- To authenticate a client …
- … before accepting the messages!!!
- Useful against spamming:
  - After the EHLO command the server sends the authentication mechanisms supported
  - The client chooses one
  - The authentication protocol is executed
  - If the authentication fails, the communication channel is closed
Negative AUTH example

- the mailer does not know (or does not accept) the authentication method proposed by the client:

```
220 example.polito.it - SMTP service ready
EHLO mailer.x.com
250-example.polito.it
250 AUTH LOGIN CRAM-MD5 DIGEST-MD5
AUTH PLAIN
504 Unrecognized authentication type
```

AUTH: LOGIN method

```
220 example.polito.it - SMTP service ready
EHLO mailer.x.com
250-example.polito.it
250 AUTH LOGIN CRAM-MD5 DIGEST-MD5
AUTH LOGIN
334 VXNlcm5hbWU6
bGlveQ==
334 UGFzc3dvcmQ6
YW50b25pbw==
235 authenticated
```

E-mail on multi-user systems
Multipurpose Internet Mail Extensions (MIME)

- Internet standard that extends the format of e-mail to support
  - text in character sets other than US-ASCII;
  - non-text attachments;
  - multi-part message bodies; and
  - header information in non-ASCII character sets
- virtually all human-written Internet e-mail and a fairly large proportion of automated e-mail is transmitted via SMTP in MIME format
- mapping messages into and out of MIME format is done automatically by an e-mail client or by mail servers when sending or receiving e-mail

Multipurpose Internet Mail Extensions (MIME)

- MIME defines
  - a number of content formats, standardizing representations that support multimedia mail, not just messages containing 7-bit ASCII characters
  - a collection of e-mail headers specifying information about the body, including content type
    - content type: declares general type of data
    - subtype: specifies a particular format for that type
  - a set of transfer encodings that enable the conversion of any content format into a form that is protected from alteration by the mail system
- MIME is extensible!
  - allows to register new content types and other MIME attribute values

MIME (Multipurpose Internet Mail Extensions)

- various data encodings
  - non-USA alphabets
  - "long" lines
  - binary data
- each part can be a multipart object
  - multipart format
  - distinct parts
  - parts of different type
- recursive format

Course Computer System Security - 2012
MIME example (text + attachment)

```plaintext
MIME-version: 1.0
Content-type: multipart/mixed; boundary="frontier"

This is a multi-part message in MIME format.
--frontier
Content-type: text/plain
This is the body of the message.
--frontier
Content-type: application/octet-stream
Content-transfer-encoding: base64
PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogICAgPC9ib2R5PgogIDxib2R5
PHN0b3AgPGFzdGJ5Om5vbmU+CiAgPC90cmFpbGU+CiAgPC90cmFpbGU+CiAgPC
```

Threats

- threats to the security of e-mail itself
  - loss of confidentiality
    - e-mails are sent in clear over open networks
    - e-mails stored on potentially insecure clients and mail servers
  - loss of integrity
    - no integrity protection on e-mails; body can be altered in transit or on mail server
  - lack of data origin authentication
  - lack of non-repudiation
  - lack of notification of receipt

Security services for e-mail messages

- integrity (without direct communication):
  - the message cannot be modified
- authentication
  - identifies the sender
- non repudiation
  - the sender cannot deny of having sent the mail
- confidentiality (optional):
  - messages are not readable both while in transit and while in stored in the e-mail box
What are the options

- secure the server to client connections (easy thing first)
  - POP, IMAP over ssh, SSL
  - https access to webmail
  - SMTP-Auth, SMTP over SSL
- secure the end-to-end email delivery
  - PGP
  - S/MIME

E-mail security – main ideas (I)

- no modification to the present MTA
  - messages encoded to avoid problems when passing through gateways (e.g. Internet-Notes) or MTA non 8BITMIME
- no modification to the present UA
  - inconvenient user interface
- with modification to the present UA
  - better user interface

E-mail security – main ideas (II)

- symmetric algorithms
  - for the encryption of messages
  - with message key
- asymmetric algorithms
  - to encrypt and exchange the symmetric key
  - for digital signature
- use public key certificates (e.g. X.509) for non-repudiation
- the message security is based only on the security of the MUA of the recipient, not on the security of MTA (not trusted)
Types of secure e-mail messages (I)

- **signed mail**
  - message digest of the content is encrypted with the private key of the signer = digital signature
  - content and signature are encoded in base 64
  - signed mail can be viewed only by recipients with S/MIME capability
- **clear signed mail**
  - digital signature of the content is formed
  - only the digital signature is encoded in base64, message content remains in clear
  - recipient without S/MIME capability can view the message content although they cannot verify the signature

Types of secure e-mail messages (II)

- **enveloped (encrypted) mail**
  - is generated using a recipient's public key.
  - the message is encrypted using a random symmetric key, which is encrypted using the recipient's public key and sent along with the message.
  - if message is sent to multiple recipients, the symmetric key is encrypted separately by every recipient's public key. The enveloped message and all encrypted symmetric keys are packaged together
- **signed & enveloped mail**
  - messages are first signed with the sender's private key and then enveloped using the recipients' public keys.

Types of secure e-mail messages (III)

- **clear-signed**
  - msg in clear (because anybody should be able to read it) + signature
  - only who has a secure MUA can verify the signature
- **signed**
  - [msg + signature] encoded (e.g. base64, uuencode)
  - only who has a secure MUA (or performs operations manually) can decode and verify the signature
- **encrypted / enveloped**
  - [encrypted msg + encrypted keys] encoded
  - only who has a secure MUA (and the keys!) can decrypt the message
- **signed and enveloped**
Secure messages: creation

- transform in canonical form
  - standard format, independent from OS / host / net
- MIC (Message Integrity Code)
  - integrity and authentication
  - typically: msg + { h(msg) } Kpri_sender
- encryption
  - confidentiality
  - typically: { msg } K_M + { K_M } Kpub_receiver
- encoding
  - to avoid modification by the MTA
  - typically: base64, uuencode, binhex

Secure electronic mail formats

- IETF
  - PEM
  - MOSS
  - S/MIME
- underground
  - PGP
  - MIME-PGP
- DOD + EU
  - X.400
  - X.421

S/MIME (Secure/Multipurpose Internet Mail Extensions)

- security enhancement to MIME email
  - original Internet RFC822 email was text only
  - MIME provided support for varying content types and multi-part messages
  - with encoding of binary data to textual form
  - S/MIME added security enhancements
- have S/MIME support in various modern mail agents: MS Outlook, Netscape etc
Secure e-mail on multi-user systems

Mail User Agent

Mail Transfer Agent

SMTP

Secure multimedia electronic mail

(MOSS o S-MIME)

- digital signature / encryption with X.509v3 certificates
- protects MIME messages

S/MIME

security of MIME messages

promoted by RSA

v2 published as a series of informational RFC:
  - RFC-2311 “S/MIME v2 message specification”
  - RFC-2312 “S/MIME v2 certificate handling”
  - RFC-2313 “PKCS-1: RSA encryption v.1-5”
  - RFC-2314 “PKCS-10: certification request syntax v.1-5”
  - RFC-2315 “PKCS-7: cryptographic message syntax v.1-5”
S/MIMEv3

- proposed standard IETF
- RFC-2633
  “S/MIME v3 message specification”
- RFC-2632
  “S/MIME v3 certificate handling”
- RFC-2634
  “Enhanced Security Services for S/MIME”
- RFC-2314 “PKCS-10: certification request syntax v.1-5”
- RFC-2630
  “CMS (Cryptographic Message Syntax)”

S/MIME architecture

From the architecture point of view it is based on:

- **PKCS-7** (S/MIME v2)
  CMS (S/MIME v3)
  specifies the cryptographic characteristics and the message types (equivalent to PEM)
- **PKCS-10**
  format of certificate request
- **X.509**
  format of public key certificates

S/MIME: algorithms

- message digest:
  - SHA-1 (preferred), MD5
- digital signature:
  - DSS (mandatory)
  - digest + RSA
- key exchange:
  - Diffie-Helmann (mandatory)
  - key encrypted with RSA
- encryption of message:
  - 3DES with 3 keys
  - RC2/40
MIME type
- application/pkcs7-mime, used for:
  - msg. encrypted (envelopedData)
  - msg. signed (signedData) addressed only to S/MIME users because are encoded in base64
  - msg. that contain only a public key (= certificate, in signedData)
  - standard extension: .p7m

MIME type
- multipart/signed
  - signed messages addressed also to users not supporting S/MIME
  - the message is in clear
  - the last MIME part is the signature
  - standard extension: .p7s
- application/pkcs10
  - used to send a certification request to a CA

S/MIME: signature example
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
mcalg=sha1;
boundary="-----aaaaa"

-----aaaaa
Content-Type: text/plain
Content-Transfer-Encoding: 7bit
Ciao!
-----aaaaa
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: base64
MIIM2qaDDgDwe/625dBxgbdhds76r8fcrJe65a4f
fvVSW2Qle+SFDs543Dwe6+25dBxR0eDars5
-----aaaaa-
client - server e-mail services

- **POP (Post-Office Protocol)**
  - POP-2 (RFC-937), POP-3 (RFC-1939)
    - user authentication by means of a password in clear (!!!)
  - APOP
    - user authentication by means of a challenge
  - K-POP
    - mutual authentication by means of tickets
- **IMAP (Internet Mail Access Protocol)**
  - username and password in clear
  - can use OTP, Kerberos or GSS-API

---

POP-3 example

telnet pop.polito.it 110
+OK POP3 server ready <7831.84549@pop.polito.it>
USER berbecaru
+OK password required for berbecaru
PASS diana
+OK berbecaru mailbox locked and ready
STAT
+OK 2 320
..........
QUIT
+OK POP3 server signing off

---

APOP

- APOP command replaces the set of commands
  - USER + PASS
- the *challenge* is the part of the hello line contained among the parentheses < ... > (including the parentheses)
- syntax:
  - APOP user response-to-challenge
- response = MDS( challenge + password )
- response encoded in hexadecimal
- supported by Eudora
APOP example

S: <wait for connection on TCP port 110>
C: <open connection>

telnet pop.polito.it 110

S: +OK POP3 server ready <7831.84549@pop.polito.it>
C: APOP berbecaru 36a036131b8247430846ab6a041ff
S: +OK berbecaru mailbox locked and ready
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
        
C: QUIT
S: +OK POP3 server signing off