Electronic mail security

Antonio Lioy
<lioypolito.it>

Politecnico di Torino
Dip. Automatica e Informatica

MHS (Message Handling System)

- MUA (Message User Agent)
- MSA (Message Submission Agent)
- MTA (Message Transfer Agent)
- MS (Message Store)

E-mail on multi-user systems

E-mail in client-server mode

Webmail

Protocols and ports

- SMTP (Simple Mail Transfer Protocol)
  - 25/tcp (MTA)
  - 587/tcp (MSA)
- POP (Post Office Protocol)
  - 110/tcp
- IMAP (Internet Message Access Protocol)
  - 143/tcp
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: cat
250 cat ... Sender ok
RCPT TO: franz
250 franz ... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself

Problems in securing e-mail
- connectionless system (store-and-forward, also because of MX records)
- MTA not trusted
- security of MS
- mailing-list
- compatibility with what is already installed
- concurrent solutions:
  - Internet = PGP, PEM, MOSS, S/MIME
  - OSI = X.400

Mail spamming
- also named UBE (Unsolicited Bulk Email)
- sending of unauthorised advertising (publicity) messages
- common habits:
  - hide the real sender
  - delegate the work to do to an MTA that operates as an “open mail relay”, that is one that accepts mail also from / to other users (not from his domain)
- heavy load on the servers and on the communication channels
- (usually) bothers the users
- risk to end up in ORB, RBL or similar
Mail relay

![Mail relay diagram]

How to fight spamming

- do not configure your own MTA as an "open relay" but restrict its use only to authorized users
- authentication strategies:
  - IP address of the MUA
  - problem with the mobile users and IP spoofing
  - value of the field From
    - can be easily tricked with a fake mail
  - SMTP authentication
    - secure authentication?
  - SMTP on SSL with client authentication

Anti-spamming initiatives

- MAPS (Mail Abuse Prevention System)
  - http://maps.vix.org/
- RBL (Realtime Blackhole List)
- RSS (Relay Spam Stopper)
- not easy to be removed once you've been inserted in such a system: it becomes a 'MUST' to configure correctly your own MTA
- activate/use the address abuse@domain, as required by RFC-2142

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 clients 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 in the DATA part 8-bit characters are accepted and not mangled
- 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 several commands with no need to wait for the response to each one (exception: those that provoke a status change)

DSN extension

(Delivery Status Notification)

- extends the RCPT command with:
  - NOTIFY=notify-list possible values: NEVER, SUCCESS, FAILURE, DELAY
  - ORCPT=original-recipient specifies the original recipient
- extends the MAIL command with:
  - RET=returned-message possible values: FULL, HDRS
  - ENVID=sender-id identifier created by the sender
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-4954
- command AUTH + options of MAIL FROM
- to authenticate a client …
- … before accepting messages from it!!!
- 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

- syntax (RFC-2595):
  AUTH PLAIN id_pwd

- id_pwd is defined as:
  [ authorize_id ] 0 authentication_id 0 pwd

- example:
  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

- example:
  220 example.polito.it - SMTP service ready
  EHLO mailer.x.com
  250-example.polito.it
  250 AUTH LOGIN PLAIN
  AUTH PLAIN
  235 authenticated

AUTH: PLAIN method

- example:
  220 example.polito.it - SMTP service ready
  EHLO mailer.x.com
  250-example.polito.it
  250 AUTH LOGIN PLAIN
  AUTH PLAIN
  235 authenticated

© Antonio Lioy - Politecnico di Torino (1995-2009)
### MAIL FROM with authentication
- the optional parameter AUTH of MAIL FROM indicates who sends the message
- it is used < > to indicate an unknown identity or not sufficiently authenticated
- allows to communicate the identity of the sender among cooperating MTA in a trusted environment
- each MTA must propagate the identity when sending (forwarding) the message
- possible use for authorisation policies

### MAIL FROM with authentication: example
```
220 example.polito.it - SMTP service ready
EHLO mailer.x.com
250-example.polito.it
250 8BITMIME
MAIL FROM:<rettore@polito.it> AUTH=profumo
250 OK
```

### Protection of SMTP with TLS
- RFC-2487 “SMTP Service Extension for Secure SMTP over TLS”
- STARTTLS = option of EHLO and command
- if the negotiation is succesful, the protocol status is reset (starts again from EHLO and the extensions supported can be different)
- if the negotiated security level is insufficient:
  - the client sends immediately QUIT and closes the connection
  - the server responds to each command with code 554 (refused due to low security)

### Protection of SMTP with TLS: example
```
220 example.polito.it - SMTP service ready
EHLO mailer.x.com
250-example.polito.it
250-8BITMIME
250-STARTTLS
250 DSN
STARTTLS
220 Go ahead
... TLS negotiation is started between client and server
```

### 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 in transit and when stored in the mailbox

### 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 UA of the recipient, not on the security of MTA (not trusted)

Types of secure messages

- **clear-signed**
  - msg in clear (so that anybody is able to read it) + digital signature
  - only who has a secure MUA can verify the signature

- **signed**
  - [ msg + dsig ] 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) \} K_{pri} _sender

- **encryption**
  - confidentiality
  - typically: \{ msg \} K_{M} + \{ K_{U} \} K_{pub} _receiver

- **encoding**
  - to avoid modification by the MTA
  - typically: base64, uuencode, binhex

Secure electronic mail formats

- IETF
- underground
- DOD + EC

- PEM
- PGP
- X.400

- MOSS
- MIME-PGP
- X.421

- S/MIME

PGP (Pretty Good Privacy)

- authentication, integrity and confidentiality for electronic mail or private files

- same objectives as PEM and similar structure but less structured

- peculiar way of public-key certification (trusted "friends" and trust propagation algebra)

- RFC:
  - RFC-1991 (informational)
  - RFC-4880 (OpenPGP)

- versions for UNIX, VMS, MS-DOS, Mac, Amiga, ... 

- the author (Phil Zimmerman) and the program have become a symbol of the freedom in Internet

Phil Zimmermann

- releases PGP as freeware in 1991
- jailed, released on bail and investigated until 1996, when accusations are dropped and he creates PGP Inc. later acquired by NAI
- august 2002 leaves NAI and creates PGP Co.
PGP - algorithms (until v. 2.6)

- fixed
- symmetric encryption:
  - IDEA
- digest:
  - MD5
- asymmetric encryption (for digital signature and symmetric key exchange):
  - RSA
- all free of charge for non-commercial purposes

PGP 2.6 example: signature + encryption

```
message M
MD5 -> RSA
ZIP -> IDEA
(M+S) + (K_M) -> B64
```

sender's private key
receiver's public key

PGP - certification

- each certificate has several signatures (those of all persons that trust the key owner)
- trust is propagated transitively with some approximation:
  - completely
  - partially
  - untrusted
  - unknown

PGP web of trust

PGP – key distribution

- public-keys stored individually by each user (in its key-ring)
- keys distributed directly by the owner (at a PGP party!) or by a key-server (http, smtp, finger)
- projects for key distribution via X.500 or DNS (pgp.net):
  - www.pgp.net
  - keys.pgp.net
  - ftp.pgp.net

PGP & NAI

- rights of PGP acquired in december 1997 by NAI (Network Associates Inc.)
- new version, based on DSA, DH, 3DES
- integration with several MUAs
- attempted penetration of the corporate market:
  - pseudo-CA (=super-signer)
  - acceptance of the X.509 format (sep’98)
- august 2002: rights given to PGP Co.
Gnu Privacy Guard (GPG)
- PGP is no more freeware (!) and it doesn't exist any more for Linux (!!) but only for Windows (!!!)
- GPG = PGP rewriting under GPL licence and without any patented algorithm
- Interoperable with PGP 2.x (with some problems) and with OpenPGP (RFC-2440)
- DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPEMD-160 e TIGER
- several graphical front-ends
- for Linux, FreeBSD, OpenBSD, Windows (95/98/NT/2000/ME), ...

PGP – encrypted message

-----BEGIN PGP MESSAGE-----
Version: 2.6.1

hAItqN4Z4suw/3i8e/8anP3QJkJv4i9tjg2WxI7GfVvWJq4Zi/6sMADy601xhS9PEU
A4ng8/nt255/2r6s/jkTYwje2W9dD9h6L6PTEd990zAv9gB59hItzjdJ
C7400/5bLx3w4q7Aq+ggE57D3sKj3WkUxVkJK8RqH6j3c0m3w3ek8w3X
BUbQp7yv5p93tL5RGXeQam6oerN7d3As0s7q7q7q7777777777777
-----END PGP MESSAGE-----

PGP – authenticated message

-----BEGIN PGP SIGNED MESSAGE-----
Text of the message
-----BEGIN PGP SIGNATURE-----
Version: 2.6.1

iQCRWw6t6LPsyuas6a58hL3TAg/KdnoaAq+Cw6yjy1zxFKCy8Oj4QKcI76QWw
3Gbh9jR69h5Hpeo54a09jy5fFdc+P+j9KIT81JhNkEAAor197/n4PM
-----END PGP SIGNATURE-----

Security of multimedia electronic mail

MOSS (MIME Objects Security Services)
- standard Internet
- RFC-1847/1848
S/MIME (Secure MIME)
- standard de-facto
- RSA
MIME-PGP
- RFC-2015 (PGP), RFC-3156 (OpenPGP)
X.421
- multimedia extension of X.400

MIME (Multipurpose Internet Mail Extensions)

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

Secure multimedia electronic mail (MOSS or S-MIME)
- digital signature/encryption with X.509 certificates
- protection of MIME messages

signed
signed and encrypted
encrypted

- text
- table Excel
docum. Word
- digital signature
in S/MIME format
- encrypted envelope
in S/MIME format
RFC-1847

- MIME extensions for message security
- for digital signature:
  - Content-Type: multipart/signed;
    protocol="TYPE/STYPE";
micalg="...";
boundary="..."

- with N body parts:
  - the first N-1 ones are those to be protected
    (content-type: 
  - the last one contains the digital signature
    (content-type: TYPE/STYPE)

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"

RFC-2634

- Enhanced Security Services for S/MIME
- addresses the following subjects:
  - signature on the return receipt of a mail
  - security labels
  - secure mailing-list
  - signature of certificate attributes

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

Architecturally 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-Hellman (obbligatorio)
  - 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 for the signature: .p7s
- **application/pkcs10**
  - used to send a certification request to a CA

### S/MIME: signature example

```
Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1;
boundary="-----aaaaa"

-----aaaaa
Content-Type: text/plain
Content-Transfer-Encoding: 7bit
Hello!

-----aaaaa
Content-Type: application/pkcs7-signature
Content-Transfer-Encoding: base64
MIIN2QasDDSdw/625dXeghdhdsf76xHfrJ65a4f
fV5NQ1aD+sFsDs543Sdwe6+25d8fGDE0e0srs5

-----aaaaa
```

### Naming in S/MIME

- **used for:**
  - selecting the certificate
  - verifying the sender's address
- S/MIMEv2 uses the **Email=** or **E=** fields in the DN of the X.509 certificate, but it is possible to use the extension `subjectAltName` with rfc822 encoding
- S/MIMEv3 mandates the use of the `subjectAltName` extension with rfc822 encoding

### Naming and MUA

- **NS Messenger and MS Outlook Express check that the sender is the same as the value of e-mail (in the DN) or with the first rfc822 field (in the subjectAltName)**
- **typical behaviour of S/MIMEv2**
- **MS Outlook 2000 makes no verification among the sender and the certified e-mail address**
- **typical behaviour of S/MIMEv3**
Client-server e-mail services

- **Authentication of the user**
- **Authentication of the server**
- **Confidentiality/integrity of mail messages**
  - On the server
  - While in transit

**Post Office**

**MUA**

- 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

```plaintext
telnet pop.polito.it 110
+OK POP3 server ready <7831.84549@pop.polito.it>
USER lioy
+OK password required for lioy
PASS antonio
+OK lioy 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 = MD5( challenge + password )
- Response encoded in hexadecimal
- Supported by Eudora

---

### APOP example

```plaintext
telnet pop.polito.it 110
+OK POP3 server ready <7831.84549@pop.polito.it>
APOP lioy 36a0b36131b82474300846abd6a041ff
+OK lioy mailbox locked and ready
STAT
+OK 2 320
..........
QUIT
+OK POP3 server signing off
```

### POP: general considerations

- POP is acceptable only on a secure channel (e.g. on SSL)
- Server APOP freeware by Qualcomm
- Use a POP / APOP password different from the one for login because the post office must know it in clear
- The mail is transmitted however in clear
- There is no server authentication
**IMAP security**

- by default weak authentication
  
  `LOGIN user password`

- strong authentication:
  
  `AUTHENTICATE KERBEROS_V4`
  `AUTHENTICATE GSSAPI`
  `AUTHENTICATE SKEY`

- mutual authentication only if Kerberos is used

- no protection of the transmission of messages

- recent versions of Netscape and MS mailer can use IMAP on SSL

---

**RFC-2595 (TLS per POP / IMAP)**

- RFC-2595
  
  “Using TLS with IMAP, POP3 and ACAP”

- first the communication channel is opened then the security characteristics are negotiated by means of a dedicated command:
  
  `STARTTLS` for IMAP and ACAP
  `STLS` for POP3

- client and server must allow to be configured to reject `user` and `password`

- client compares the identity in the certificate with the identity of the server

---

**Separate ports for SSL/TLS?**

- discouraged by IETF due to the following reasons:
  
  - involve different URLs (e.g. http and https)
  - involve an incorrect secure / insecure model (e.g. is 40-bit SSL secure SSL? is insecure an application without SSL but with SASL?)
  - not easy to implement “use SSL if available”
  - doubles the number of necessary ports

- but present some advantages:
  
  - simple to filter traffic on packet-filter firewalls
  - SSL with client-authentication allows not to expose the applications to attacks