E-mail security

Antonio Lioy
< lioy @ polito.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 in client-server mode

- MUA (e.g. Thunderbird, Outlook Express)
- Mailserver (MSA)
- Post Office (MS)
- MTA

Webmail

- web browser
- web server
- HTTP engine
- virtual MUA
- Mailserver (MSA)
- Post Office (MS)
- MTA
Protocols and standard 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
From: cat@athena.polito.it (Antonio Lioy)  
To: franz@duke.colorado.edu  
Subject: vacation  

Hello Francesco,  
I renew my invitation to come to my place during your vacation in Italy. Let me know when you arrive.  
Antonio

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

Problems in securing e-mail

- connectionless system (store-and-forward, also because of MX records)
- untrusted MTA's
- security of MS
- mailing-list encryption
- 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) or UCE (Unsolicited Commercial E-mail)
- sending of unwanted messages:
  - unauthorised advertisement
  - attacks (malware, phishing, …)
- today it is nearly 88% of the total e-mail traffic
  - heavy load on servers and network channels
  - heavy annoyance to the users
- canned pork meat and Monty Python
- the opposite of “spam” is “ham” (term used by identification and filtering applications)

Spamming strategies

- hide the real sender
  - … but use a valid sender
- send spam via special MTA
  - open mail relay
  - zombie or botnet
  - with variable or phantom IP address
- content obfuscation
  - deliberate mistakes (e.g. Vi@gr@)
  - image rather than text
  - Bayesian poisoning (e.g. text from a book)
  - inside an error message
(Open) mail relay

“open mail relay” = MTA accepting mail also not from/to its users

Anti-spam for MSA

- do not configure your own MSA as an “open relay” but restrict its use only to authorized users
- authenticate the users of our MSA:
  - IP address of the MUA
    - problem with mobile nodes, IP spoofing and malware (at valid nodes)
  - value of the field From
    - can be easily tricked with a fake mail
  - SMTP authentication
    - secure authentication methods?
Anti-spam for incoming MTA (I)

- reject or accept mail from an MTA, after checking a blacklist or whitelist
- DNSBL (DNS-based BlackList)
  - the address A.B.C.D is used to send spam?
  - nslookup –q=A.D.C.B.A.dnsbl.antispam.net
  - if NXDOMAIN then it is not a spammer
  - else the query returns:
    - an address 127.0.0.X (where X is a code providing the reason for being black-listed)
    - a TXT record TXT with more information
- RFC-5782 “DNS blacklists and whitelists”

Anti-spam for incoming MTA (II)

- URI DNSBL (~URI reputation data)
  - delay in identifying new spammer MTA (and their short life)
  - honeypot / spamtrap for capturing spam and classifying the URI found in the messages
  - lookup of the URI found in the body of an (incoming) message against those in spam messages
DNSBL lists

- several lists (free/commercial, anonymous or not):
  - MAPS RBL (Realtime Blackhole List)
  - Spamhaus SBL (Spamhaus Block List)
  - SORBS (Spam and Open Relay Blocking System)
  - APEWS (Anonymous Postmaster Early Warning System)
- not easy to be removed once inserted: it’s strongly suggested to correctly configure your own MTA
- activate/use the address abuse@domain, as required by RFC-2142

Anti-spam for incoming MTA (III)

- greylisting
  - spammers have scarce time
  - temporary error (“try later”)
  - OK if the same MTA comes back after T (e.g. 5’)
  - ham delayed + server load (history of the contacts)
- nolisting (poor man’s greylisting)
  - spammers have scarce time, do not contact all the MX nodes and/or contact only the highest MX
  - primary MX provides no answer, secondary MX is OK, tertiary MX provides no answer
  - ham is delayed as well
Anti-spam for incoming MTA (IV)

- **DKIM (DomainKeys Identified Mail)** – various RFC
  - a mail domain guarantees:
    - the identity of the sender
    - the (partial) integrity of the message
  - … via a digital signature
    - created by the MSA or outgoing MTA
    - which covers some headers and part of the body
    - verifiable via a public key (e.g. in the DNS)
  - increasing use (e.g. Gmail, Yahoo)
  - permits to discard messages with fake sender and hence supports anti-spam and anti-phishing

Anti-spam for incoming MTA (V)

- **SPF (Sender Policy Framework)** – RFC 4408
  - a mail domain declares which are its outgoing MTA, via a specific record in the DNS
  - examples:
    ```
    $ nslookup -q=txt polito.it.
polito.it text = "v=spf1 ptr ~all"
$ nslookup -q=txt gmail.com.
gmail.com text = "v=spf1 redirect=_spf.google.com"
$ nslookup -q=txt _spf.google.com.
_spf.google.com text = "v=spf1 ip4:216.239.32.0/19
 ip4:64.233.160.0/19 ip4:66.249.80.0/20 ip4:72.14.192.0/18
 ip4:209.85.128.0/17 ip4:66.102.0.0/20 ip4:74.125.0.0/16
 ip4:64.18.0.0/20 ip4:207.126.144.0/20 ip4:173.194.0.0/16
 ?all"
    ```
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

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-SIZE 26214400
  - 250 8BITMIME
Negative ESMTP example

- the mailer does not know the ESMTP protocol:

```plaintext
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

```
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
```

- Username: lioy
- Password: antonio
AUTH: PLAIN method

- syntax (RFC-2595):
  AUTH PLAIN id_pwd

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

220 example.polito.it - SMTP service ready
EHLO mailer.x.com
250-example.polito.it
250 AUTH LOGIN PLAIN
AUTH PLAIN bGlveQBsaW95AGFudG9uaW8=
235 authenticated

AUTH: CRAM-MD5 method

220 x.polito.it - SMTP service ready
EHLO mailer.x.com
250-x.polito.it
250 AUTH CRAM-MD5 DIGEST-MD5
AUTH CRAM-MD5
334 PDY5LjIwMTIwMTAzMjAxMDU4MDdAeC5wb2xpdG8uaXQ+
bGlveSA1MGUxNjZiZDc5NGZjNDNjZmMIZjk1MzQ1NDI3MjA5Nw==
235 Authentication successful

lioy hmac(antonio,<69.2012010320105807@x.polito.it>)hex
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 (as an attachment or inside the msg)
  - 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: \( \text{msg} + \{ h(\text{msg}) \} K_{\text{pri\_sender}} \)
- **encryption**
  - confidentiality
  - typically: \( \{ \text{msg} \} K_M + \{ K_M \} K_{\text{pub\_receiver}} \)
- **encoding**
  - to avoid modification by the MTA
  - typically: base64, uuencode, binhex

## Secure electronic mail formats

<table>
<thead>
<tr>
<th>IETF</th>
<th>underground</th>
<th>DOD + EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEM</td>
<td>PGP</td>
<td>X.400</td>
</tr>
<tr>
<td>MOSS</td>
<td>MIME-PGP</td>
<td>X.421</td>
</tr>
<tr>
<td>S/MIME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Antonio Lioy - Politecnico di Torino (1995-2014)
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 + M+S → ZIP → IDEA + {M+S}+{KM} → B64

sender's private key

K_M message 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 web of trust diagram]

- X signs Y
- Y signs X
- Completely trusted
- Partially trusted
- Untrusted
- Unknown
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
  - due to legal issues with RSA
- 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), ...

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 o S-MIME)

- digital signature/encryption with X.509 certificates
- protection of MIME messages

<table>
<thead>
<tr>
<th>signed</th>
<th>signed and encrypted</th>
<th>encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>table Excel</td>
<td>table Excel</td>
<td>table Excel</td>
</tr>
<tr>
<td>docum. Word</td>
<td>docum. Word</td>
<td>docum. Word</td>
</tr>
<tr>
<td></td>
<td>encrypted envelope in S/MIME format</td>
<td>encrypted envelope in S/MIME format</td>
</tr>
</tbody>
</table>

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”

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)”
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/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 it is encoded in base64
  - msg. that contain only a public key (= certificate, in a degenerate signedData body)
  - standard extension: .p7m
  - always base64-encoded
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 (per RFC-1847) and its base64-encoded
  - standard extension for the signature: `.p7s`

- **application/pkcs10**
  - used to send a certification request to a CA
  - base64-encoded

S/MIME: signature example

```plaintext
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

MIIN2QasDSSdwe/625dBxgdhdksf76rHfrJe65a4f
  fvVSW2QleD+SfDs543Sdwe6+25dBxfdER0eDsrs5
-----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

Client-server e-mail services

- authentication of the user
- authentication of the server
- confidentiality/integrity of mail messages
  - on the server
  - while in transit
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 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

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
IMAP security

- by default weak authentication
  
  \texttt{LOGIN user password}

- strong authentication:
  \texttt{AUTHENTICATE KERBEROS\_V4}
  \texttt{AUTHENTICATE GSSAPI}
  \texttt{AUTHENTICATE SKEY}

- mutual authentication only if Kerberos is used

---

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 \texttt{user} and \texttt{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