Comunicazioni sicure tra server di posta elettronica

La sicurezza della posta elettronica è uno degli argomenti attualmente di maggiore interesse. Il problema risiede fondamentalmente nel fatto che i protocolli su cui si basa la distribuzione dei messaggi di posta elettronica non sono stati creati con sufficienti misure di sicurezza ed in particolare di autenticazione. La mancanza della certezza di chi sia il mittente di un messaggio, o anche solo da quale server il messaggio sia stato originato, rende possibili molti dei problemi attuali, dalla diffusione dei virus allo spam.

E' difficile pensare che in tempi brevi si possano adottare standard e protocolli completamente nuovi che risolvano questi problemi. Quello che invece sta lentamente succedendo è che vengono prese delle misure e fatte delle piccole modifiche o aggiunte ai protocolli esistenti per ridurre per quanto possibile i problemi.

Ripercorriamo velocemente il percorso di un messaggio di posta elettronica. Dal programma dell'utente, il Mail User Agent (MUA), il messaggio passa attraverso uno o più server SMTP aziendali,[1] poi il server aziendale connesso a internet invia il messaggio al corrispondente server dell'organizzazione destinataria (anche tramite server intermedi), che lo invia tramite altri server interni alla casella di posta del destinatario che vi accede tramite il proprio programma MUA. Nel passaggio da un server all'altro non vi è di solito alcuna protezione, ne di confidenzialità ne di autenticazione tra i due server.

Sottolineiamo un concetto importante. Vi sono due aspetti di sicurezza nel processo di trasmissione di un messaggio di posta elettronica:

1. l'identificazione del mittente e la confidenzialità del messaggio tra mittente e destinatario
2. l'identificazione tra i server che si scambiano il messaggio e la confidenzialità delle loro comunicazioni.

Il primo caso deve essere risolto (oggi) dall'utente utilizzando protocolli quali PGP o S/MIME. Il

1 I server SMTP sono di solito chiamati Mail Transfer Agent (MTA).

secondo caso, che consideriamo qui, è invece significativo per i problemi di spam, diffusione di virus eccetera, e deve essere affrontato da chi gestisce i servizi di posta elettronica. Un possibile miglioramento della situazione potrebbe essere ottenuto se ad esempio due organizzazioni che scambiano tra di loro molto traffico di posta elettronica, facessero autenticare tra di loro i server SMTP e trasferire i messaggi su di un canale cifrato. Questo potrebbe permettere ad entrambe le organizzazioni di far passare su di un canale privilegiato, ad esempio con minori controlli anti-spam od accesso a relay interni, i messaggi dell'altra organizzazione. In altre parole, è possibile introdurre metodi di autenticazione forte tra server SMTP tali da permettere un diverso trattamento dei messaggi provenienti od inviati a/da partner conosciuti e fidati.

Da un punto di vista tecnico, per raggiungere questo scopo si utilizza il protocollo SSL/TLS che permette di autenticare SERVER (chi riceve il messaggio) e CLIENT (chi invia il messaggio) con certificati digitali, e stabilire un canale cifrato di comunicazione. Illustreremo come questo possa essere fatto nel caso di due server SMTP *postfix*, solo per la semplicità della procedura di configurazione.

Sempre per semplicità assumiamo che tutti i certificati siano rilasciati dalla stessa CA, anche se la generalizzazione è molto semplice. Inoltre analizziamo solo la connessione in una direzione, ovvero consideriamo un server SMTP come mittente, ovvero CLIENT, e l'altro come ricevente dei messaggi di posta elettronica, il SERVER. La generalizzazione è di nuovo molto semplice.

Il SERVER

Per prima cosa abbiamo bisogno di procurarci tre file, il certificato digitale della CA,[2] `CA.pem`, il file contenente la chiave privata del server non cifrata con passphrase, `server-postfix-server.key`, ed il certificato digitale del server, `server-postfix-cert.pem`. Questi certificati possono essere creati con una CA interna oppure acquistati da una CA commerciale, l'unica richiesta di postfix è che siano in formato PEM. Nella Tabella 1 proponiamo un esempio di

2 Si veda ad esempio *Gestire Certificati Digitali con openssl*, ICTSecurity 18, Novembre 2003.

configurazione per il server. In questa configurazione il server offre il servizio TLS ma non lo richiede necessariamente, analogamente chiede il certificato del client ma non lo esige necessariamente. La selezione viene poi fatta con la configurazione `smtpd_recipient_restrictions` che permette di fare relay dei messaggi ricevuti solo dai client locali (`permit_mynetworks`) o dai client che si sono autenticati con un proprio certificato digitale (`permit_tls_clientcerts`). Per i client che scelgono la connessione con TLS, invocando il comando `STARTTLS` all'inizio del collegamento, il server verifica che il certificato digitale fornito sia firmato da una CA nota, ovvero quella indicata in `smtpd_tls_CAfile` oppure nei file presenti opzionalmente nella directory `smtpd_tls_CApath`, ed anche che la fingerprint del certificato[3] del client sia presente nel file indicato in `relay_clientcerts` di cui si da un esempio nella Tabella 2. E' anche possibile non verificare la fingerprint del certificato del client ma solo la firma di una CA nota, usando il comando `permit_tls_all_clientcerts`. Questo però è suggerito solo nel caso in cui si usi una CA privata solo per questo scopo, e nessuna CA pubblica.

Il CLIENT

La configurazione del client è molto simile a quella del server. Anch'esso ha bisogno dei tre file contenenti la chiave ed i certificati, quello proprio e quello della CA. Si può poi scegliere se il client deve usare SSL/TLS con tutti i server a cui si connette o solo con quelli che offrono il servizio. Quando un client si connette ad un server, verifica che il certificato inviato dal server nello scambio iniziale SSL/TLS sia valido e firmato da una CA nota, o quella indicata in `smtp_tls_CAfile` o nei file in `smtp_tls_CApath`. Inoltre il client verifica che il nome a dominio del server corrisponda al nome indicato nel certificato.

In realtà la politica di usare SSL/TLS solo se il server lo accetta è molto debole. Pertanto è stato introdotto un meccanismo più preciso in aggiunta. Nel file indicato in `smtp_tls_per_site` si

3 La fingerprint può essere estratta da un certificato con il comando `openssl x509 -fingerprint -in cert.pem`.

possono indicare i nomi di server corrispondenti sicuri con i quali si vuole o meno usare SSL/TLS. Dal punto di vista del client, devono essere indicati i server nexthop, ovvero quelli che il client contatta direttamente che non è detto siano i server di destinazione del messaggio di posta. Il formato di questo file è molto semplice ed indicato nella Tabella 4. Per ogni nome di server viene indicato se SSL/TLS non deve essere usato (`NONE`), se può essere usato se il server lo offre (`MAY`),[4] se deve essere usato sempre (`MUST`), o se deve essere usato sempre ma senza la verifica del nome a dominio del server nel certificato (`MUST_NOPEERMATCH`). Se il nome del server non compare in questa tabella, il default specificato precedentemente nella configurazione viene adottato.

Vi sono altri parametri nella configurazione di postfix che possono essere utili, quali quelli per selezionare gli algoritmi crittografici usati, che possono significativamente aumentare la sicurezza della configurazione.

Andrea Pasquinucci

pasquinucci@ucci.it

Riferimenti Bibliografici

[1] SSLv3.0: http://wp.netscape.com/eng/ssl3/

[2] TLSv1.0 descritto in RFC-2246

[3] openssl: http://www.openssl.org/

[4] RFC-2487 smtp-starttls

[5] postfix: http://www.postfix.org/

```
# TLS Server
smtpd_tls_key_file = /etc/postfix/server-postfix-server.key
smtpd_tls_cert_file = /etc/postfix/server-postfix-cert.pem
smtpd_tls_CAfile = /etc/postfix/CA.pem
smtpd_use_tls = yes
```

4 Si noti però che se il default è `smtp_enforce_tls=yes` il significato di `MAY` viene ad essere lo stesso di `MUST`.

```
#smtpd_enforce_tls = yes
smtpd_tls_ask_ccert = yes
#smtpd_tls_req_ccert = yes
smtpd_tls_loglevel = 2
smtpd_tls_received_header = yes
# clients CA certs in .pem form go in this dir
# remember to run c_rehash every time you add/change one
#smtpd_tls_CApath = /etc/postfix/certs.d
#
smtpd_recipient_restrictions = permit_mynetworks,
   permit_tls_clientcerts,
   reject_unauth_destination
# run postmap every time you change this file
relay_clientcerts = hash:/etc/postfix/relay_clients
```

Tabella 1. Configurazione TLS per postfix server in /etc/postfix/main.cf

```
60:8E:42:61:84:2C:29:AE:57:6B:47:A8:7B:E3:5B:FF client1.my.net
5D:35:D5:83:62:A5:57:B3:3C:C5:BC:F3:A4:D5:D7:A7 client2.my.net
```

Tabella 2. Configurazione dei client in /etc/postfix/relay_clients

```
# TLS client
smtp_tls_key_file = /etc/postfix/client1-postfix-server.key
smtp_tls_cert_file = /etc/postfix/client1-postfix-cert.pem
smtp_tls_CAfile = /etc/postfix/CA.pem
smtp_tls_loglevel = 2
smtp_use_tls = yes
#smtp_enforce_tls = yes
smtp_tls_enforce_peername = yes
smtp_tls_note_starttls_offer = yes
# server CA certs in .pem form go in this dir
# remember to run c_rehash every time you add/change one
#smtp_tls_CApath = /etc/postfix/certs.d
#
# use this to specify policies for different servers
# run postmap every time you change this file
#smtp_tls_per_site = hash:/etc/postfix/tls_per_site
```

Tabella 3. Configurazione TLS per postfix client in /etc/postfix/main.cf

```
unknown.dom.ain       NONE
host.dom.ain          MAY
important.host        MUST
some.host.dom.ain     MUST_NOPEERMATCH
```

Tabella 4. Esempio del file /etc/postfix/smtp_tls_per_site