Creare connessioni cifrate con stunnel

Capita, e purtroppo anche frequentemente, di dover offrire servizi molto insicuri, utilizzando ad esempio protocolli che trasmettono password e dati sensibili in chiaro. In alcuni casi è possibile incapsulare queste connessioni all'interno di canali cifrati. Da una parte è possibile realizzare dei circuiti privati cifrati, detti genericamente VPN (sicure), sui quali transita tutto il traffico tra due destinazioni: tipici esempi di ciò sono le VPN create con IPSEC o le più nuove VPN create con SSL/TSL.[1] Un altro approccio è quello di creare una connessione sicura che incapsula solamente il protocollo di interesse. Questo può essere fatto in modi differente, ad esempio usando la possibilità di port-forwarding di ssh, oppure usando un tool molto semplice come stunnel che descriviamo in questo articolo.

COME FUNZIONA

Stunnel permette di creare dei tunnel cifrati sfruttando il protocollo SSLv3, quello normalmente utilizzato dai siti web per la navigazione protetta.[2] Stunnel funziona solo con protocolli che utilizzano solo TCP ed una porta unica e fissa. Ad esempio stunnel può essere adoperato con http (tcp-80), pop3 (tcp-110), imap (tcp-143), smtp (tcp-25) ecc. ma non ad esempio con ftp che utilizza tcp-21 ed anche un'altra porta scelta dinamicamente, oppure con dns che utilizza UDP.

Stunnel è di facile installazione e configurazione, a seconda delle necessità può essere usato in tre modi distinti. Su di una macchina si attiva un processo stunnel che funge da cifratore/decifratore di una connessione. In altre parole, cifra quello che riceve in chiaro e decifra quello che riceve cifrato. Nella modalità **server**, stunnel ascolta per connessioni cifrate su di una porta tcp specificata e le invia in chiaro verso una porta non cifrata. In questa modalità stunnel permette di rendere sicuro un protocollo non sicuro. Ad esempio, supponiamo di avere un server http/pop3/imap normale, possiamo renderlo sicuro, ovvero cifrato, mettendo stunnel in ascolto per connessioni cifrate sulle porte 443/995/993 od altre porte a scelta, e ri-inviando la connessione decifrata al vero server sia sulla stessa macchina che su di una altra macchina (locale). Nella modalità **client**, stunnel funziona al contrario, ascolta per connessioni in chiaro su di una porta tcp specificata e le invia cifrate verso una porta remota. In questa modalità stunnel può rendere sicuri dei clienti locali che non sono in grado di cifrare le connessioni verso un server remoto. Mettendo insieme le due modalità con due istanze di stunnel su due macchine diverse, è possibile creare un tunnel cifrato per un particolare protocollo, tra un client ed un server che non supportano protocolli cifrati. Bisogna notare che stunnel comunque funziona sempre in modalità client-server, ovvero vi

1 In un futuro articolo ci occuperemo di OpenVPN (http://openvpn.sourceforge.net/).
2 Molti siti ora utilizzano TLS, che in pratica è una nuova versione di SSL.

deve essere un client che inizia la connessione al server, il server non può stabilire connessioni ai client.

La cifratura adottata da stunnel si basa sulle librerie di openssl, e quindi ha bisogno di certificati digitali per stabilire l'identità del server, e se richiesto anche del client.

INSTALLAZIONE

L'installazione è molto semplice sia su piattaforme unix che windows. Di norma vi sono pacchetti già compilati, ma la compilazione ad esempio in unix consiste nei soliti tre comandi: `configure; make; make install`. Al comando `configure` si possono passare opzioni quali `–without-tcp-wrappers` per disabilitare l'utilizzo dei tcp-wrappers.

Per utilizzare stunnel nella modalità **server** è necessario avere un certificato digitale. La cosa migliore è avere sia un certificato creato per stunnel che il certificato della CA che lo ha firmato. Il primo è necessario nella modalità **server**, il secondo è fortemente consigliato nella modalità **client**. Il certificato per stunnel deve essere in formato PEM ed avere la chiave privata RSA non cifrata. Inoltre il file in cui esso viene messo deve avere un formato particolare, vedi Tabella 1,[3] e leggibile solo dall'amministratore.

Consideriamo qui solo la versione 4 di stunnel, bisogna notare che la configurazione è cambiata completamente nella versione 4 rispetto alle versioni precedenti. Per attivare stunnel è sufficiente dare il comando:

```
stunnel /etc/stunnel/stunnel.conf
```

ovvero passandogli come parametro il nome del file di configurazione. Specificando diversi file di configurazione è possibile avere più di una istanza di stunnel attiva sulla stessa macchina.

SEVER

La modalità server può essere utilizzata in vari modi. Ad esempio stunnel invia la connessione cifrata ricevuta verso un server attivo sulla stessa macchina o su di un'altra macchina, in pratica fa una specie di port-forwarding. In un'altra modalità, stunnel è attivato ad esempio da inetd al posto del programma, tipicamente imapd, ed a sua volta attiva il programma localmente, oppure stunnel ascolta per le connessioni e per ognuna di essa attiva il programma destinatario. In Tabella 2 diamo un esempio di configurazione in cui stunnel riceve una connessione cifrata sulla porta 443 della propria macchina, come se fosse un server web sicuro, e la invia al vero server web 192.168.1.22 sulla porta 80. E' da notare che stunnel può opzionalmente girare in una directory chroot (solo in ambiente unix) che deve essere scrivibile dall'utente nobody. Il certificato contenuto in `stunnel.pem` è quello che viene inviato al cliente che si connette per una eventuale verifica. In

3 Si veda l'articolo sul numero 18 di ICT Security per come generare certificati digitali usando openssl.

Tabella 3 diamo un esempio di configurazione in cui per ogni connessione viene attivato imapd.

CLIENT

Questa configurazione permette ad un client che non supporta connessioni cifrate di connettersi ad un server che le richiede. Nel semplice esempio in Tabella 4 stunnel finge di essere un server web non cifrato ascoltando sulla porta 80. Tutte le connessioni ricevute vengono cifrate ed inviate al server web sicuro all'indirizzo 10.11.1.33:443. E' da notare il parametro `verify`: se esso è 0, non viene effettuata nessuna verifica del certificato digitale inviato dal server. Se `verify=1` viene verificato il certificato inviato dal server con il certificato della CA indicato dal parametro `CAfile`, se il certificato della CA necessaria è presente. In caso di assenza del certificato della CA che ha emesso il certificato del server la connessione viene stabilita ugualmente. Se `verify=2` viene sempre verificato il certificato inviato dal server con i certificati e le CRL delle CA presenti. Se `verify=3` non solo viene verificato il certificato come nel caso precedente, ma il certificato del server deve essere presente nel file `CAfile` in coda ai certificati delle CA, in questo modo si verifica sia che il certificato è stato emesso da una CA riconosciuta ed anche che il certificato sia esattamente quello noto.

TUNNEL

Un esempio di tunnel è fatto facilmente ponendo ad esempio sul **client** la configurazione della Tabella 4 e sul **server** quella della Tabella 2. Si ottiene in questo modo che un web browser si collega ad un sito tramite un tunnel SSL in realtà senza che né il browser né il sito utilizzino SSL. In altre parole le macchine di una rete si collegano a quello che ritengono un sito web normale in porta 80, ma invece è stunnel con la configurazione 4 che rimanda la connessione questa volta cifrata ad un'altra macchina, anche lontanissima, con la configurazione 2 che a sua volta la invia al sito destinatario all'indirizzo 192.168.1.22:80.

Stunnel ha vari altri parametri di configurazione che permettono di gestirlo e configurarlo in modo più adatto alle proprie esigenze. Rimane comunque un tool leggero e di facile utilizzo per realizzare connessioni cifrate con SSLv3.

Andrea Pasquinucci
Libero Professionista in Sicurezza Informatica
pasquinucci@ucci.it

Riferimenti Bibliografici

[1] http://www.stunnel.org/

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

[3] Specifiche di SSLv3: http://wp.netscape.com/eng/ssl3/

```
-----BEGIN RSA PRIVATE KEY-----
[encoded key without password]
-----END RSA PRIVATE KEY-----
[empty line]
-----BEGIN CERTIFICATE-----
[encoded certificate]
-----END CERTIFICATE-----
[empty line]
```

Tabella 1. Formato del certificato digitale per il server stunnel

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/stunnel/
# PID is created inside chroot jail
pid = /stunnelA.pid
# remember that nobody must write in chroot
setuid = nobody
setgid = nobody
#debug = 6

[server-https]
accept  = 443
connect = 192.168.1.22:80
TIMEOUTclose = 0
```

Tabella 2. Configurazione server port-forwarding

```
cert = /etc/stunnel/stunnel.pem
pid = /var/run/stunnelB.pid
setuid = nobody
setgid = nobody
#debug = 6

[imapd]
accept = 993
exec = /usr/sbin/imapd
execargs = imapd
```

Tabella 3. Configurazione server per imapd

```
chroot = /var/stunnel/
pid = /stunnelC.pid
setuid = nobody
setgid = nobody
#debug = 6

client = yes
verify = 3
CAfile = /etc/stunnel/CA+cert.pem
CRLfile = /etc/stunnel/CA.crl

[client-https]
accept = 80
connect = 10.11.1.33:443
```

Tabella 4. Configurazione client