IPsec HOWTO

Ralf Spenneberg

ralf (at) spenneberg.net

Questo HowTo illustrerà i passaggi base ed avanzati per allestire una VPN utilizzando il supporto ad Ipsec dei kernels Linux 2.4 e 2.5/2.6. Poichè vi è molta documentazione disponibile sul kernel 2.4, questo HowTo si concentrerà soprattutto sulle nuove caratteristiche dei kernel attualmente in sviluppo. Una versione successiva comprenderà anche il kernel 2.4.

Tradotto in italiano da ginox

ginox (at) autistici.org

.

Table of Contents

Introduzione

L'ultima versione di questo documento si può sempre reperire presso The Linux Documentation Project[1] e sull'home page ufficiale http://www.ipsec-howto.org.

Perchè scrivere questo howto

Ho consultato molti HowTos in passato. La maggior parte di essi mi sono stati molto utili. Quando IPsec è stato introdotto come nuova funzionalità nel kernel ho iniziato a giocherellarci. Ho constatato che esiste pochissima documentazione a riguardo. Dunque ho iniziato a scrivere questo HowTo.

Formato del documento

Questo documento è diviso in 5 capitoli.

Capitolo 1: Introduzione

: Questa sezione

Capitolo 2: Teoria

: Teoria di IPsec. Il protocollo in breve.

Capitolo 3: Linux Kernel 2.2 e 2.4

: Come installare FreeS/WAN.

Capitolo 4: Linux Kernel 2.5/2.6

: Questa sezione descrive come installare una VPN basata su IPsec con i tools messi a disposizione da KAME, **setkey** e **racoon**, il demone IKE OpenBSD isakmpd e FreeS/WAN utilizzando la patch di Herbert Xu.

Capitolo 5: Configurazione avanzata

: Questa sezione introduce alcune configurazioni avanzate con l'utilizzo di DHCP-over-IPsec, NAT-Traversal etc.

Hanno collaborato alla stesura di questo documento:

- Fridtjof Busse
- Uwe Beck
- Juanjo Ciarlante
- Ervin Hegedus
- Barabara Kane

Aspetti Legali

Copyright

vengano inoltrati al maintainer. È possibile creare altri documenti a partire da questo e redistribuirli, a patto che:

- Si spedisca copia del lavoro che ne deriva (nei formati più utilizzati, come l'sgml) all'LDP (Linux Documentation Project) o a progetti simili presenti in rete. Si renda comunque noto all'LDP dove è possibile reperire il documento.
- Si mantenga intatta la licenza o si utilizzi la GPL. Si includano le note sul copyright o almeno un riferimento alla licenza utilizzata.
- Si mantengano i crediti ai precedenti autori ed ai principali collaboratori.

Nell'ipotesi che si desideri realizzare qualcosa di diverso da una semplice traduzione, si prega di contattare il maintainer attuale del documento per discutere il progetto.

Liberatoria

L'autore non si assume alcuna responsabilità circa gli utilizzi impropri di questo documento, nè fornisce alcuna garanzia, implicita o esplicita. Se il vostro cane muore, l'autore non potrà essere ritenuto responsabile!

Documenti inerenti

- Networking Overview HOWTO[3]
- Networking HOWTO[4]
- VPN-Masquerade HOWTO[5]
- VPN HOWTO[6]
- Advanced Routing & Traffic Control HOWTO[7]

Teoria

Cosa è IPsec?

IPsec è un estensione del protocollo IP che fornisce sicurezza a livello IP ed ai livelli superiori. È stato sviluppato prima all'interno dello standar IPv6 ed in seguito inserito in Ipv4. L'architettura di IPsec viene descritta nell'RFC2401. I seguenti paragrafi rappresentano una breve introduzione al protocollo.

IPsec usa due differenti protocolli - AH ed ESP - per fornire l'autenticazione, l'integrità e la confidenzialità della comunicazione. Può proteggere l'intero datagramma IP o solamente i protocolli di alto livello. I diversi modelli di funzionamento sono detti tunnel mode e transport mode. Nel tunnel mode il datagramma IP viene completamente incapsulato in un nuovo datagramma IP utilizzando IPsec. In transport mode solo il payload del datagramma IP viene trattato da IPsec che inserisce il proprio header tra l'header IP ed i livelli superiori (si veda Figure 1).

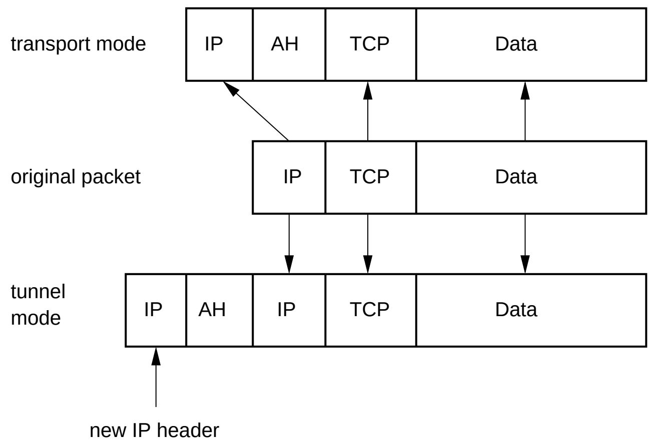

Figure 1. IPsec tunnel e transport mode

Per proteggere l'integrità di un datagramma IP il protocollo IPsec utilizza meccanismi di autenticazione basati su codici hash (HMAC).Per ottenere questo HMAC vengono usati algoritmi come MD5 o SHA per calcolare un hash basato su una chiave segreta e sul contenuto di un datagramma IP. L'HMAC viene quindi inserito nell'header IPsec ed il destinatario del pacchetto ne può verificare l'esattezza soltanto avendo accesso alla chiave segreta.

Per proteggere la confidenzialità di un datagramma IP il protocollo IPsec utilizza degli algoritmi di cifratura simmetrici. L'IPsec richiede l'implementazione di NULL (nessuna cifratura) e del DES. Attualmente vengono utilizzati algoritmi più robusti come 3DES, AES ed il Blowfish.

Per proteggersi contro un attacco di tipo denial of service, il protocollo IPsec utilizza un meccanismo di sliding window. Ciascun pacchetto possiede un numero di sequenza e viene accettato solo se tale numero rientra nella finestra o è più recente. I pacchetti più vecchi vengono immediatamente scartati. Questo protegge da attacchi di tipo replay, nei quali pacchetti originali vengono conservati e rispediti in seguito.

Perchè le due parti siano in grado di effettuare l'incapsulamento ed il recupero dei pacchetti IPsec è necessario che vi sia un luogo nel quale conservare le chiavi, gli algoritmi e gli indirizzi IP coinvolti nella comunicazione. Tutti questi parametri necessari per la protezione del datagramma IP, sono inseriti in una security association (SA). Le security association vengono conservate a turno in un database detto SAD (security association database).

Ciascuna security association definisce i seguenti parametri:

- IP sorgente e destinazione del corrispondente header IPsec, ovvero delle parti coinvolte.
- Il protocollo IPsec (AH or ESP), l'eventuale supporto per la compressione (IPCOMP).
- Gli algoritmi e le chiavi utilizzate.
- Security Parameter Index (SPI). Un numero di 32 bit che identifica la SA.

Alcune implementazioni di security association database permettono inoltre di immagazzinare:

- IPsec mode (tunnel o transport)
- Le dimensioni per la sliding window, per proteggersi contro i replay attacks.
- La durata della SA.

Poichè la security association contiene l'IP sorgente e destinazione, può proteggere una sola direzione in una connessione full duplex. Per proteggere entrambe le direzioni sono necessarie due security association.

Una security association specifica solo come proteggere il traffico. Sono necessarie ulteriori informazioni per definire quale tipo di traffico proteggere. Queste informazioni sono contenute in una security policy (SP) e immagazzinate a turno in un security policy database (SPD).

Una security policy specifica normalmente i seguenti parametri:

- Sorgente e destinazione del pacchetto da proteggere. In transport mode sono i medesimi della corrispondente SA. Ma in tunnel mode possono cambiare.
- Il protocollo (e la porta) da proteggere. Alcune implementazioni non permettono di definire un protocollo in particolare. In questo caso tutto il traffico tra i due IP specificati viene protetto.
- La security association da utilizzare per la protezione dei pacchetti.

Il setup manuale di una security association è passibile di errore e non molto sicuro. La chiave segreta e gli algoritmi di crittografia devono essere condivisi dalle due parti di una virtual private network. È soprattutto lo scambio di chiavi a mettere in crisi gli amministratori di sistema: come scambiarsi una chiave simmetrica senza un canale cifrato già attivo?

Per risolvere questo problema è stato sviluppato l'internet key exchange protocol (IKE). Nella prima fase quest'ultimo autentica le due parti. Nella seconda la security association viene negoziata e le chiavi simmetriche vengono scambiate attraverso il metodo Diffie Hellmann. Il protocollo IKE rinegozia periodicamente le chiavi per garantire la confidenzialità.

Il protocollo IPsec

La famiglia di protocolli IPsec è composta da: Authentication Header (AH) e Encapsulated Security Payload (ESP). Entrambi sono indipendenti dai protocolli IP. AH è il protocollo IP 51 ed ESP è il 50 (si veda `/etc/protocols`). Le due sezioni seguenti presentano brevemente le loro caratteristiche.

AH - Authentication Header

Il protocollo AH protegge l'integrità del datagramma IP, calcola un HMAC del pacchetto in base ad una chiave segreta, al payload e le parti del header IP che non possono cambiare, ad esempio i campi con gli indirizzi IP. Quindi aggiunge l'header AH all'header del pacchetto.

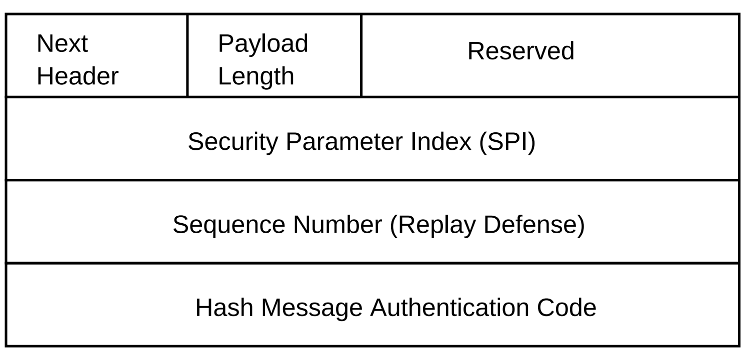

Figure 2. L'Header AH protegge l'integrità del pacchetto

L'header AH è lungo 24 bytes. Il primo byte è detto *Next Header*, contiene il protocollo dell'header seguente. In tunnel mode viene incapsulato un intero datagramma IP: il valore di questo campo è dunque 4. Quando invece in transport mode viene incapsulato un pacchetto TCP, il valore è 6. Il byte seguente indica la lunghezza del payload. Quindi vi sono 2 byte riservati. Seguono 32 bit contenenti il *Security Parameter Index* (SPI), che indicano quale SPI utilizzare per interpretare correttamente il pacchetto al momento della ricezione. I successivi 32 bit sono per il *Sequence Number*, per proteggersi da attacchi di tipo replay. Infine gli ultimi 96 bit contengono l'*hash message authentication code* (HMAC). Quest'ultimo protegge l'integrità del pacchetto poichè solo chi conosce la chiave segreta può crearlo e verificarne l'esattezza.

Poichè il protocollo AH calcola l'HMAC in base ad alcune parti non mutabili del datagramma IP, tra le quali gli indirizzi, il NAT non si sposa bene con IPsec. Il Network address translation (NAT) sostituisce infatti un IP (di solito il sorgente) con uno differente. Quindi l'HMAC viene invalidato subito. Un'estensione al protocollo detta NAT-Traversal extension riesce però a superare questa limitazione.

ESP - Encapsulated Security Payload

Il protocollo ESP può garantire sia l'integrita di un pacchetto utilizzando HMAC sia la confidenzialità della trasmissione utilizzando la cifratura. Dopo aver cifrato il pacchetto e calcolato l'HMAC viene generato ed aggiunto l'header ESP. L'header ESP si compone di due parti, mostrate in Figure 3.

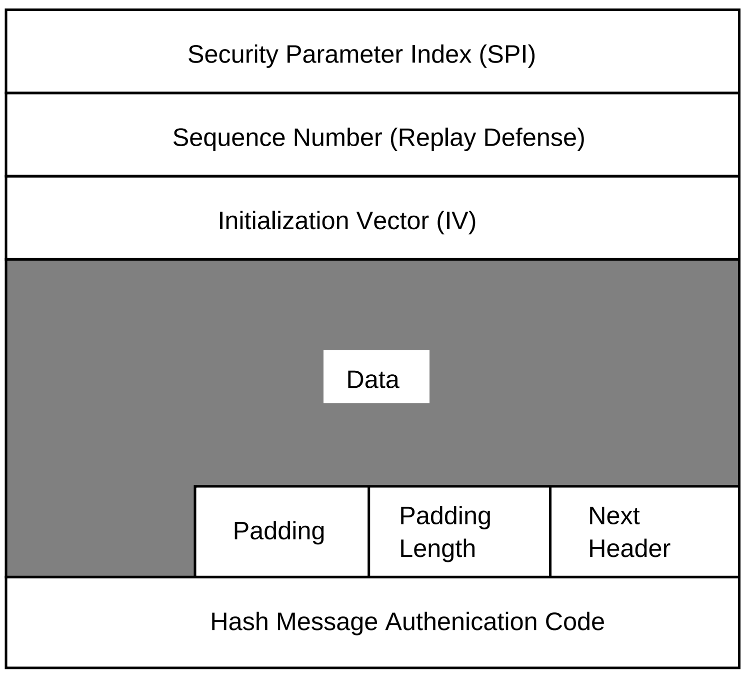

Figure 3. Header ESP

I primi 32 bit sono il *Security Parameter Index* (SPI), indicano quale SA utilizzare per interpretare correttamente il pacchetto ESP al momento della ricezione. I successivi 32 bit contengono il *Sequence Number*, per proteggersi da attacchi di tipo replay. Seguono 32 bit per l'*Initialization Vector* (IV), utilizzato durante le operazioni di cifratura. Gli algoritmi di crittografia simmmetrica sono vulnerabili ad attacchi statistici se non viene utilizzato un IV. Quest'ultimo infatti assicura che due payload in chiaro identici producano due versioni cifrate differenti.

IPsec utilizza cifrari a blocchi per il processo di cifratura. Dunque il payload deve essere completato (padding) nel caso in cui non risulti un multiplo della dimensione del blocco. Quindi la lunghezza dell'eventuale pad viene inserita nell'header. Subito dopo 2 byte indicano quale sia l'header del protocollo sottostante, *Next Header*. Infine vengono aggiunti 96 bit con l'HMAC per garantire l'integrità del pacchetto. L'HMAC è calcolato sul payload, l'header IP non viene considerato.

L'utilizzo del NAT non rende inutilizzabile il protocollo ESP. Ma nonostante questo il NAT nella maggior parte dei casi non può essere utilizzato in combinazione con IPsec. Il NAT-Traversal è una possibile soluzione al problema incapsulando i pacchetti all'interno di pacchetti UDP.

Protocollo IKE

Il protocollo IKE risolve la maggior parte dei problemi a cui abbiamo accennato nell'instaurazione di una connessione sicura: l'autenticazione delle parti e lo scambio di chiavi simmetriche. Crea inoltre le security association e popola il SAD. IKE utilizza di solito un demone in user space e non viene inclusa nel sistema operativo. IKE utilizza la porta 500/udp per la comunicazione tra le parti.

Il protocollo IKE consta di due fasi: la prima realizza una *Internet Security Association Key Management Security Association* (ISAKMP SA). Nella seconda l'ISAKMP SA viene utilizzata per la negoziazione e l'instaurazione delle IPsec SAs.

L'autenticazione delle parti nella prima fase può di solito essere basata su pre-shared keys (PSK), RSA keys e certificati X.509 (**racoon** supporta anche Kerberos).

La prima fase può utilizzare di solito due differenti modalità: main mode ed aggressive mode. Entrambe autenticano le parti e stabiliscono una ISAKMP SA, ma l'aggressive mode è in grado di farlo utilizzando la metà dei messaggi. Il prezzo da pagare per la maggior velocità è l'assenza del supporto per l'identificazione dei partecipanti e quindi la possibilità di attacchi di tipo man-in-the-middle nel caso di utilizzo di pre-shared keys. Questo però è anche il solo scopo dell'aggressive mode. Il main mode infatti è strutturato per non accettare preshared keys differenti da macchine sconosciute. L'aggressive mode non supporta alcuna forma di identificazione delle due parti e trasferisce l'identità del client in chiaro. Le due parti si devono conoscere prima che l'autenticazione avvenga e chiavi differenti possono venire utilizzate per partecipanti differenti.

Nella seconda fase vengono negoziate le security association utilizzando l'ISAKMP SA appena instaurata. Quest'ultima fornisce i meccanismi di autenticazione necessari a proteggersi contro attacchi man-in-the-middle. Questa seconda fase utilizza il quick mode.

Di norma le due parti negoziano una sola ISAKMP SA, che utilizzano per creare diverse (almeno due) IPsec SA unidirezionali.

Linux Kernel 2.2 and 2.4 -- FreeS/WAN

ToDo

Linux Kernel 2.5/2.6 utilizzando i KAME-tools

Questo capitolo spiega l'utilizzo dello stack IPsec nativo in Linux dalla versione $\geq$2.5.47 e nei kernel 2.6.*. L'installazione e la configurazione dello stack IPsec è molto differente da FreeS/WAN e molto più simile alle diverse varianti di *BSD come FreeBSD, NetBSD e OpenBSD.

Illustrerò prima la configurazione e l'installazione del kernel e dei tools in user space. Quindi mostrerò come effettuare una connessione con i parametri impostati manualmente in transport mode ed in tunnel mode. Infine sarà presentata una connessione con negoziazione automatica dei parametri utilizzando preshared keys e certificati X.509. Per ultimo verrà trattato il supporto per i roadwarriors.

Installazione

L'installazione richiede un kernel 2.5.47 o successivi oppure un 2.6.*. I sorgenti del kernel possono essere prelevati da http://www.kernel.org. Dopo averli scaricati vanno decompressi, configurati e compilati.

```
cd /usr/local/src
tar xvjf /path-to-source/linux-<version>.tar.bz2
cd linux-<version>
make xconfig
make bzImage
make modules
make modules_install
make install
```

Questi sono i comandi più comuni per configurare e compilare un kernel Linux. In caso si necessiti di un setup particolare è opportuno fare riferimento al Kernel-Howto.

Durante la configurazione del kernel è importante attivare le seguenti opzioni:

```
Networking support (NET) [Y/n/?] y
  *
  * Networking options
  *
  PF_KEY sockets (NET_KEY) [Y/n/m/?] y
  IP: AH transformation (INET_AH) [Y/n/m/?] y
  IP: ESP transformation (INET_ESP) [Y/n/m/?] y
  IP: IPsec user configuration interface (XFRM_USER) [Y/n/m/?] y

Cryptographic API (CRYPTO) [Y/n/?] y
  HMAC support (CRYPTO_HMAC) [Y/n/?] y
  Null algorithms (CRYPTO_NULL) [Y/n/m/?] y
  MD5 digest algorithm (CRYPTO_MD5) [Y/n/m/?] y
  SHA1 digest algorithm (CRYPTO_SHA1) [Y/n/m/?] y
  DES and Triple DES EDE cipher algorithms (CRYPTO_DES) [Y/n/m/?] y
  AES cipher algorithms (CRYPTO_AES) [Y/n/m/?] y
```

A seconda di quale versione si stia utilizzando potrebbe essere necessario abilitare anche il supporto per IPv6.

Compilato ed installato il kernel è necessario procurarsi i tools in user space. Attualmente sono disponibili presso http://ipsec-tools.sourceforge.net/[9].Compilando a mano il pacchetto è necessario specificare la posizione degli header del kernel. Sono necessari gli header della versione 2.5.47 o successive.

```
./configure --with-kernel-headers=/lib/modules/2.5.47/build/include
make
make install
```

Ora tutto dovrebbe essere pronto per funzionare.

Connessione manuale usando setkey

Con connessione manuale (manual keyed connection) si intende che tutti i parametri necessari vengono forniti dall'amministratore. Il protocollo IKE non viene utilizzato per l'autenticazione delle due parti e la negoziazione di questi parametri. L'amministratore decide quale protocollo, algoritmo e chiave utilizzare per la creazione delle security association e popola di conseguenza il security association database (SAD).

Transport Mode

Si illustrerà prima una connessione manuale in transport mode. Probabilmente è il modo più facile per iniziare poichè è la connessione più semplice da realizzare. Poniamo che vi siano due macchine con indirizzi IP 192.168.1.100 e 192.168.2.100 che comunicano attraverso IPsec.

Tutti i parametri contenuti nel SAD e nel SPD possono essere modificati attravreso il comando **setkey**. Esiste una man page piuttosto esaustiva su questo comando. Dunque solo le opzioni necessarie per realizzare una connessione in transport mode sono mostrate qui di seguito. **setkey** legge i propri parametri da un file quando invocato con l'opzione -f **setkey -f /etc/ipsec.conf**. Un modello di `/etc/ipsec.conf` piuttosto realistico:

```
#!/usr/sbin/setkey -f

# Configurazione per 192.168.1.100

# Svuoto il SAD e SPD
flush;
spdflush;

# Attenzione: utilizzate queste chiavi solo per i test!
# Generate delle chiavi vostre!

# Le AH SAs utilizzano chiavi di 128 bit
add 192.168.1.100 192.168.2.100 ah 0x200 -A hmac-md5 \
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.2.100 192.168.1.100 ah 0x300 -A hmac-md5 \
0x96358c90783bbfa3d7b196ceabe0536b;

# Le ESP SAs utilizzano chiavi di 192 bit (168 + 24 parity)
add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 192.168.1.100 192.168.2.100 any -P out ipsec
           esp/transport//require
           ah/transport//require;

spdadd 192.168.2.100 192.168.1.100 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

Saranno necessarie delle chiavi per rimpiazzare quelle dello script di esempio in caso si desideri utilizzare una connessione manuale in transport mode in un ambiente di produzione. Per generare le proprie chiavi è possibile utilizzare il seguente metodo:

```
$ # chiavi a 128 Bit
$ dd if=/dev/random count=16 bs=1| xxd -ps
16+0 Records ein
16+0 Records aus
cd0456eff95c5529ea9e918043e19cbe
```

```
$ # chiavi a 192 Bit
$ dd if=/dev/random count=24 bs=1| xxd -ps
24+0 Records ein
24+0 Records aus
9d6c4a8275ab12fbfdcaf01f0ba9dcfb5f424c878e97f888
```

Utilizzate sempre il device **/dev/random** quando generate delle chiavi poichè garantisce un buon grado di casualità.

Per prima cosa lo script effettua il flush del security association database (SAD) e del security policy database (SPD). Quindi crea le AH SAs e le ESP SAs. Il comando **add** aggiunge una security association al SAD e richiede un IP sorgente ed uno di destinazione, il protocollo IPsec (**ah**), l'SPI (**0x200**) e l'algortimo. L'algoritmo di autenticazione viene specificato con l'opzione **-A** (la cifratura utilizza **-E**, la compressione **-C**; la compressione di IP non è ancora supportata però). Dopo l'algoritmo deve essere inserita la chiave. Quest'ultima deve essere formattata in double-quoted "ASCII" o in esadecimale con prefisso **0x**.

Linux supporta i seguenti algoritmi per AH ed ESP: hmac-md5 and hmac-sha, des-cbc and 3des-cbc. Tra poco sarà possibile anche utilizzare: simple (senza cifratura), blowfish-cbc, aes-cbc, hmac-sha2-256 and hmac-sha2-512.

spdadd aggiunge una security policy ad un SPD. Con queste è possibile specificare quali protocolli vadano protetti con IPsec e quali chiavi utilizzare. Il comando richiede l'indirizzo IP sorgente e destinazione del pacchetto da proteggere, il protocollo (la porta, nel nostro esempio any) e la policy da utilizzare (-P). La policy indica la direzione (in/out), l'azione da intraprendere (ipsec/discard/none), il protocollo (ah/esp/ipcomp), la modalità (transport) ed il livello (use/require).

Un file di configurazione simile va creato su entrambe le parti interessate alla comunicazione IPsec. Il nostro esempio funziona senza alcun cambiamento su 192.168.1.100, ma sono necessarie alcune modifiche per 192.168.2.100, per riflettere la diversa direzione dei pacchetti. La maniera più semplice per adattare la configurazione è scambiare le direzioni nelle security policies: sostituire **-P in** con **-P out** e vice versa. Qui di seguito un esempio:

```
#!/usr/sbin/setkey -f

# Configurazione per 192.168.2.100

# Flush SAD e SPD
flush;
spdflush;

# Attenzione: utilizzate queste chiavi solo per i test!
# Generate delle chiavi vostre!

# Le AH SAs utilizzano chiavi di 128 bit
add 192.168.1.100 192.168.2.100 ah 0x200 -A hmac-md5 \
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.2.100 192.168.1.100 ah 0x300 -A hmac-md5 \
0x96358c90783bbfa3d7b196ceabe0536b;

# Le ESP SAs utilizzano chiavi di 192 bit (168 + 24 parity)
add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

# Security policies
spdadd 192.168.1.100 192.168.2.100 any -P in ipsec
           esp/transport//require
           ah/transport//require;
```

```
spdadd 192.168.2.100 192.168.1.100 any -P out ipsec
           esp/transport//require
           ah/transport//require;
```

Quando il file di configurazione è correttamente posizionato per entrambe le parti, è possibile renderlo operativo con il comando **setkey -f /etc/ipsec.conf**. Il corretto caricamento può essere verificato visualizzando il contenuto dei due database, SAD ed SPD:

```
# setkey -D
# setkey -DP
```

Il setup attuale ricalca lo scenario in Figure 4.

Figure 4. Due macchine in transport mode utilizzando AH ed ESP

Se si effettua un ping da una macchina all'altra il traffico viene cifrato e tcpdump mostra il seguente risultato:

```
12:45:39.373005 192.168.1.100 > 192.168.2.100: AH(spi=0x00000200,seq=0x1):
ESP(spi=0x00000201,seq=0x1) (DF)
12:45:39.448636 192.168.2.100 > 192.168.1.100: AH(spi=0x00000300,seq=0x1):
ESP(spi=0x00000301,seq=0x1)
12:45:40.542430 192.168.1.100 > 192.168.2.100: AH(spi=0x00000200,seq=0x2):
ESP(spi=0x00000201,seq=0x2) (DF)
12:45:40.569414 192.168.2.100 > 192.168.1.100: AH(spi=0x00000300,seq=0x2):
ESP(spi=0x00000301,seq=0x2)
```

Tunnel Mode

Il tunnel mode viene utilizzato quando le due parti sono anche dei gateway ed è in grado di proteggere il traffico tra le due reti (Figure 5). Il pacchetto IP originale viene cifrato ed incapsulato da un gateway e spedito all'altra parte. Quest'ultima estrae il pacchetto originale e lo consegna.

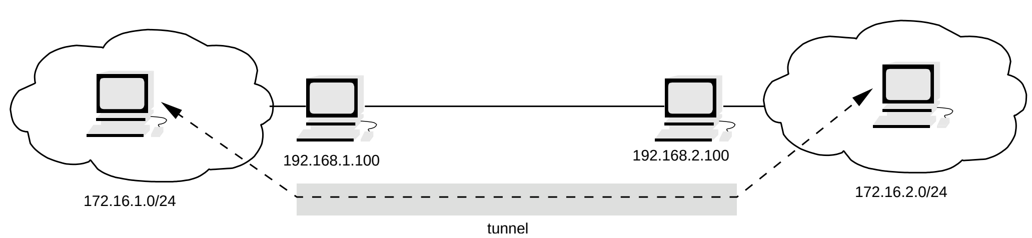

Figure 5. I due gateway proteggono il traffico tra le due reti.

La configurazione delle security association e delle policies per il tunnel mode è simile al transport mode ed illustrata qui di seguito.

```
#!/usr/sbin/setkey -f

# Flush the SAD and SPD
flush;
spdflush;

# ESP SAs cifra utilizzando chiavi a 192 bit (168 + 24 parity)
# e chiavi a 128 per l'autenticazione
add 192.168.1.100 192.168.2.100 esp 0x201 -m tunnel -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831 \
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add 192.168.2.100 192.168.1.100 esp 0x301 -m tunnel -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df \
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Security policies
spdadd 172.16.1.0/24 172.16.2.0/24 any -P out ipsec
           esp/tunnel/192.168.1.100-192.168.2.100/require;

spdadd 172.16.2.0/24 172.16.1.0/24 any -P in ipsec
           esp/tunnel/192.168.2.100-192.168.1.100/require;
```

Questo esempio utilizza il solo protocollo ESP. L'ESP è in grado di garantire integrità e confidenzialità. In questo caso l'ordine degli algoritmi ESP è importante. Prima si indica l'algoritmo di cifratura e la rispettiva chiave, quindi l'algoritmo di autenticazione e la chiave di quest'ultimo.

A differenza della realizzazione di IPsec su BSD una security association in Linux può essere utilizzata sia per il transport che per il tunnel mode. Il primo è il default, dunque qualora sia necessario il secondo, la security association deve essere definita con l'opzione **-m tunnel**.

Le security policies specificano ora gli IP delle reti protette: i pacchetti con quella sorgente o destinazione saranno protetti da IPsec. Nel tunnel mode è sempre necessario definire il tunnel e gli indirizzi IP delle due parti. Queste informazioni sono necessarie per riconoscere l'IPsec SA appropriata.

Connessione automatizzata utilizzando racoon

Il demone IKE di KAME, **racoon**, è stato portato su Linux. È in grado di instaurare automaticamente le SA necessarie al funzionamento del protocollo IPsec. Racoon supporta l'autenticazione con preshared keys, con certificati X.509 e con Kerberos. Il demone può utilizzare il main mode o l'aggressive mode nella prima fase dell'IKE. In questo capitolo mostreremo la configurazione di **racoon** in main mode utilizzando preshared keys e certificati X.509 (ToDo: Kerberos). Infine verrà introdotto brevemente uno scenario relativo ad una configurazione di tipo roadwarrior.

Preshared Keys

La forma più semplice di autenticazione utilizzando **racoon** sono le preshared keys. Devono essere definite nel file `/etc/psk.txt`, che non deve essere leggibile da utenti non privilegiati (**chmod 400 /etc/psk.txt**) ed utilizza la seguente sintassi:

```
# IPv4 Adressen
192.168.2.100          simple psk
5.0.0.1                0xe10bd52b0529b54aac97db63462850f3
# USER_FQDN
ralf@spenneberg.net    This is a psk for an email address
# FQDN
www.spenneberg.net     This is a psk
```

È organizzato in colonne. La prima contiene un identificativo della parte autenticata dalla psk. Tutto ciò che è scritto nella seconda colonna è la PSK.

La configurazione di **racoon** è piuttosto veloce. Una configurazione tipica è riportata qui di seguito, `/etc/racoon.conf`:

```
path pre_shared_key "/etc/psk.txt";

remote 192.168.2.100 {
        exchange_mode main;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method pre_shared_key;
                dh_group modp1024;
        }
}

sainfo address 172.16.1.0/24 any address 172.16.2.0/24 any {
        pfs_group modp768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Per prima cosa viene indicato dove **racoon** troverà le preshared keys. Quindi viene definita la configurazione per 192.168.2.100 ed i parametri da utilizzare per la prima fase della negoziazione con IKE. Il secondo paragrafo specifica quali parametri utilizzare per la security association. Questa parte potrebbe interessare un solo indirizzo IP o servirsi del generico comando **anonymous** al posto di un IP. Sempre qui vengono definiti gli algoritmi di cifratura, autenticazione e compressione per la SA. Devono essere specificati tutti e tre per non incorrere in errori all'avvio di **racoon**.

Il demone IKE **racoon** non inizia la negoziazione del tunnel appena avviato. Piuttosto attende fino a che il tunnel si rende necessario. Perche' questa segnalazione ci sia, il kernel deve sapere quando notificare **racoon** . Per ottenere ciò e' necessario specificare una security policy senza la corrispondente security association. Ogni qualvolta Linux dovrà proteggere un pacchetto in accordo con le security policies e quando nessuna security association e' disponibile richiamerà **racoon** e gli domanderà di instaurare le necessarie security association. **Racoon** inizierà le negoziazioni IKE e creerà le SA al termine. Linux potrà quindi spedire i pacchetti.

Per lo scenario descritto sono necessarie le seguenti policies su 192.168.1.100:

```
#!/usr/sbin/setkey -f
#
# Svuota SAD e SPD
flush;
spdflush;

# Crea le policies per racoon
spdadd 172.16.1.0/24 172.16.2.0/24 any -P out ipsec
           esp/tunnel/192.168.1.100-192.168.2.100/require;

spdadd 172.16.2.0/24 172.16.1.0/24 any -P in ipsec
           esp/tunnel/192.168.2.100-192.168.1.100/require;
```

Dopo che le policies sono state caricate con **setkey -f /etc/ipsec.conf racoon** può essere lanciato. A scopo di test si può usare **racoon -F -c /etc/racoon.conf**. Di nuovo è necessario modificare la configurazione dell'altra parte perchè rifletta le differenti direzioni del traffico. Gli indirizzi IP nei files `/etc/psk.txt`, `/etc/ipsec.conf` e `/etc/racoon.conf` devono essere cambiati.

L'instaurazione del tunnel può essere seguita attraverso i log:

```
2003-02-21 18:11:17: INFO: main.c:170:main(): @(#)racoon 20001216 20001216
 sakane@kame.net
2003-02-21 18:11:17: INFO: main.c:171:main(): @(#)This product linked Open
SSL 0.9.6b [engine] 9 Jul 2001 (http://www.openssl.org/)
2003-02-21 18:11:17: INFO: isakmp.c:1365:isakmp_open(): 127.0.0.1[500] use
d as isakmp port (fd=7)
2003-02-21 18:11:17: INFO: isakmp.c:1365:isakmp_open(): 192.168.1.100[500]
 used as isakmp port (fd=9)
2003-02-21 18:11:37: INFO: isakmp.c:1689:isakmp_post_acquire(): IPsec-SA r
equest for 192.168.2.100 queued due to no phase1 found.
2003-02-21 18:11:37: INFO: isakmp.c:794:isakmp_ph1begin_i(): initiate new
phase 1 negotiation: 192.168.1.100[500]<=>192.168.2.100[500]
2003-02-21 18:11:37: INFO: isakmp.c:799:isakmp_ph1begin_i(): begin Identit
y Protection mode.
2003-02-21 18:11:37: INFO: vendorid.c:128:check_vendorid(): received Vendor
 ID: KAME/racoon
2003-02-21 18:11:37: INFO: vendorid.c:128:check_vendorid(): received Vendor
 ID: KAME/racoon
2003-02-21 18:11:38: INFO: isakmp.c:2417:log_ph1established(): ISAKMP-SA es
tablished 192.168.1.100[500]-192.168.2.100[500] spi:6a01ea039be7bac2:bd288f
f60eed54d0
2003-02-21 18:11:39: INFO: isakmp.c:938:isakmp_ph2begin_i(): initiate new p
hase 2 negotiation: 192.168.1.100[0]<=>192.168.2.100[0]
2003-02-21 18:11:39: INFO: pfkey.c:1106:pk_recvupdate(): IPsec-SA establish
ed: ESP/Tunnel 192.168.2.100->192.168.1.100 spi=68291959(0x4120d77)
2003-02-21 18:11:39: INFO: pfkey.c:1318:pk_recvadd(): IPsec-SA established:
 ESP/Tunnel 192.168.1.100->192.168.2.100 spi=223693870(0xd554c2e)
```

Certificati X.509

Racoon supporta l'utilizzo di certificati X.509 per l'autenticazione. Dovrebbero essere verificati attraverso una certificate authority (CA). La configurazione è simile alla PSK e differisce per le sole fasi di autenticazione:

```
path certificate "/etc/certs";

remote 192.168.2.100 {
        exchange_mode main;
        certificate_type x509 "my_certificate.pem" "my_private_key.pem";
 verify_cert on
        my_identifier asn1dn;
 peers_identifier asn1dn;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}

sainfo address 172.16.1.0/24 any address 172.16.2.0/24 any {
        pfs_group modp768;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

Il certificato e la chiave privata sono conservati all'interno della directory `/etc/certs`. Il percorso viene specificato attraverso l'opzione **path certificate**, nel file di configurazione. I certificati e le revoche sono in formato PEM, così come sono stati generati da **openssl**. Per la creazione dei certificati si veda il capitolo dedicato qui di seguito. Se per la verifica si utilizza una CA ((**verify_cert on** è l'azione compiuta per default), il certificato di quest'ultima deve essere posto in questa stessa directory. È necesario rinominare il certificato o creare il link simbolico in modo che il file col certificato sia raggiungibile nel file system attraverso l'hash:

```
ln -s CAfile.pem `openssl x509 -noout -hash < CAfile.pem`.0
```

Se il certificato dovesse inoltre essere verificato attraverso un certificate revocation file (CRL), quest'ultimo dovrebbe risiedere nella stessa directory del certificato ed essere anch'esso raggiungibile attraverso l'hash:

```
ln -s CRLfile.pem `openssl crl -noout -hash < CAfile.pem`.r0
```

Quando si mantengono certificati e chiavi private bisogna tenere presente che **racoon** non è in grado di decifrare una chiave privata cifrata. Quindi la chiave deve essere conservata in chiaro. Se si è generata una chiave cifrata sarà necessario decifrarla:

```
# openssl rsa -in my_private_key.pem -out my_private_key.pem
read RSA key
Enter PEM pass phrase: password
writing RSA key
```

Roadwarrior

I Roadwarrior sono client che utilizzano IP dinamici per connettersi al gateway della VPN. In combinazione con **racoon** sorgono due diversi problemi:

- L'IP è sconosciuto e non può essere indicato nel file di configurazione o in `/etc/psk.txt`. È necessario trovare un altro modo per determinare l'identità del client. Nel caso delle pre-shared keys diviene necessario l'aggressive mode! La soluzione migliore è utilizzare i certificati X.509.
- Non è possibile creare una security policy per attivare **racoon** sino a quando l'indirizzo IP non sia noto. **racoon** dovrà dunque impostare quest'ultima e la security association a connessione già stabilita.

È necessario quindi effettuare diverse modifiche al file `/etc/racoon.conf` per rendere possibile quanto illustrato sopra:

```
path certificate "/etc/certs";

remote anonymous {
        exchange_mode main;
        generate_policy on;
        passive on;
        certificate_type x509 "my_certificate.pem" "my_private_key.pem";
        my_identifier asn1dn;
        peers_identifier asn1dn;
        proposal {
                encryption_algorithm 3des;
                hash_algorithm md5;
                authentication_method rsasig;
                dh_group modp1024;
        }
}
```

```
sainfo anonymous {
        pfs_group modp1024;
        encryption_algorithm 3des;
        authentication_algorithm hmac_md5;
        compression_algorithm deflate;
}
```

L'opzione **generate_policy on** fa in modo che **racoon** crei la policy appropriata soltanto a connessione già stabilita. L'opzione **passive on** indica a **racoon** di attendere che la connessione inizi dall'esterno. **Racoon** non potrà iniziarne.

I cambiamenti più importanti sono comunque il parametro **anonymous** nella voce **remote** e in **sainfo**. Questo fa in modo che **racoon** accetti connessioni da qualsiasi host.

Linux Kernel 2.5/2.6 utilizzando isakmpd di OpenBSD

Thomas Walpuski ha effettuato il porting per Linux del demone IKE di OpenBSD (http://bender.thinknerd.de/~thomas/IPsec/isakmpd-linux.html). L'isakmpd può ora venir utilizzato con Linux 2.5.47+ e 2.6.x. Questo capitolo descrive l'installazione e la configurazione di isakmpd.

Installazione

Se si utilizzano distribuzioni basate su RPM o Debian l'installazione può avvenire attraverso gli appositi tools per gestire i pacchetti. L'autore di questo documento ha compilato un pacchetto RPM di isakmpd per il kernel 2.6.0 (http://www.spenneberg.org/VPN/Kernel-2_6_IPsec). Si tenga presente che il pacchetto potrebbe non funzionare con altre versioni, poichè l'ABI nel kernel ha subito diverse modifiche. Il progetto Debian possiede un pacchetto installabile con **apt-get install isakmpd**.

Quando si installa dai sorgenti è necessario procurarsi il pacchetto keynode (http://www1.cs.columbia.edu/~angelos/keynote.html) se si intende utilizzare certificati X.509. Inoltre e' necessario un kernel 2.5.47+ o 2.6.x.

Per ottenere i sorgenti si seguano le indicazioni presenti nell'home page di Thomas Walpuski. Quindi si editi il GNUmakefile attivando la linea **OS=linux**. Se i sorgenti del kernel non si trovano in `/usr/src/linux` è necessario inoltre modificare il file `sysdep/linux/GNUmakefile.sysdep` indicando il percorso corretto.

La compilazione avviene con il comando **make**.

Isakmpd è accompagnato da due comandi: **keyconf** e **certpatch**. Questi tools si trovano nella subdirectory `apps` e dovranno essere compilati a mano (Sono parte del mio pacchetto RPM). **Certpatch** è in grado di aggiungere un SubjectAltName ad un certificato già esistente, **keyconv** converte chiavi DNSSEC in formato openssl e vice-versa.

Sono riuscito a compilare con successo i due programmi in questo modo (i vostri parametri potrebbero essere diversi):

```
gcc -DMP_FLAVOUR=MP_FLAVOUR_GMP -I../.. -I../../sysdep/linux   -I /usr/src/linux-2.6.0 -lcrypto -lgmp certpatch.c -o certpatch
gcc  -I../.. -I../../sysdep/linux   -I /usr/src/linux-2.6.0 -lcrypto -lgmp base64.c keyconv.c -o keyconv
```

Un ultimo avvertimento: tutte le manpages sono in formato Latin1. Red Hat 9 non visualizza questo formato. È necessario convertirle per poterle leggere (nel pacchetto

RPM lo sono già): **iconv --from-code LATIN1 --to-code UTF-8 --output isakmpd2.8 isakmpd.8**

Una volta compilato, si creino le seguenti directory:

```
mkdir /etc/isakmpd
mkdir /etc/isakmpd/ca
mkdir /etc/isakmpd/certs
mkdir /etc/isakmpd/keynote
mkdir /etc/isakmpd/crls
mkdir /etc/isakmpd/private
mkdir /etc/isakmpd/pubkeys
```

Utilizzare preshared keys (PSK)

Isakmpd utilizza un file di configurazione e un file per le policy. Rispettivamente `/etc/isakmpd/isakmpd.conf` e `/etc/isakmpd/isakmpd.policy`. Il primo utilizza il ben noto formato in stile .INI. Ciascuna sezione inizia con una linea del tipo:

```
[section]
```

All'interno di una sezione è possibile assegnare un valore ad un tag:

```
tag=valore
```

Se l'assegnamento è più lungo di una riga è necessario terminare la riga con un Backslash per riprendere sulla successiva. I commenti possono essere ovunque e sono introdotti dal simbolo #.

Per iniziare ci concentreremo su di una configurazione semplice che utilizza dei preshared secret per l'autenticazione. Si consulti anche Figure 5 per maggiori informazioni sul setup.

```
[General]
Listen-on=              192.168.1.100

[Phase 1]
192.168.2.100=                ISAKMP-peer-west

[Phase 2]
Connections=            IPsec-east-west

[ISAKMP-peer-west]
Phase=                  1
Local-address=          192.168.1.100
Address=                192.168.2.100
Authentication=         ThisIsThePassphrase

[IPsec-east-west]
Phase=                  2
ISAKMP-peer=            ISAKMP-peer-west
Configuration=          Default-quick-mode
Local-ID=               Net-east
Remote-ID=              Net-west

[Net-west]
ID-type=                IPV4_ADDR_SUBNET
Network=                172.16.2.0
Netmask=                255.255.255.0

[Net-east]
ID-type=                IPV4_ADDR_SUBNET
```

```
Network=                172.16.1.0
Netmask=                255.255.255.0

[Default-quick-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-3DES-MD5-PFS-SUITE
```

Questo file di configurazione descrive un tunnel tra i due gateways 192.168.1.100 e 192.168.2.100 e si riferisce al primo di questi. Il tunnel viene utilizzato dalle due sottoreti 172.16.1.0/24 e 172.16.2.0/24.

Esaminiamo ora ciascuna sezione. La prima, [General], contiene il setup generale. Qui viene specificato se isakmpd debba stare in ascolto su un determinato IP, opzione importante nel caso in cui sul gateway VPN siano in uso diversi indirizzi IP.

La sezione [Phase 1] illustra quale configurazione utilizzare per instaurare una connessione con l'IP 192.168.2.100. Se il client è un roadwarrior è necessario utilizzare default, al posto dell'IP.

La sezione [Phase 2] descrive i tunnels da realizzare dopo che la Phase 1 di autenticazione ha avuto luogo. Se isakmpd non deve iniziare la connessione, ma attendere passiva, si utilizzi l'opzione **Passive-connections**.

A questo punto è necessario definire i nomi utilizzati come riferimenti per la Phase 1 e la Phase 2. Incominciamo da ISAKMP-peer-west: viene utilizzata nella **Phase** 1, sono noti **Local-address** ed il remote **Address**. Se quest'ultimo è sconosciuto è sufficiente rimuovere il tag. **Authentication** nel nostro caso contiene una preshared key, che inseriamo direttamente come testo in chiaro.

Proseguiamo con il tunnel IPsec-east-west: viene utilizzato in **Phase** 2 e deve essere attivato dall'opzione **ISAKMP-peer** ISAKMP-peer-west. In questa sezione definiamo il tipo di configurazione per la connessione con **Configuration**, e gli ID addizionali per i tunnels (**Local-ID** e **Remote-ID**).

Poichè questi ID sono ancora riferimenti, dobbiamo definirli. **ID-type** può assumere i seguenti valori: IPV4_ADDR, IPV6_ADDR, IPV4_ADDR_SUBNET e IPV6_ADDR_SUBNET.

Ultimo ma non meno importante, dobbiamo definire la configurazione quick-mode , alla quale facciamo riferimento nella descrizione del tunnel. Sono necessarie le opzioni: **DOI** (default: IPSEC), **EXCHANGE_TYPE** (default: QUICK_MODE) e **Suites**.Quest'ultima nel nostro caso assume il valore, QuickMode-Encapsulated-Security-Payload-3DES-Encryption-MD5-HMAC-Perfect-Forward-Secrecy. È possibile specificare valori diversi separati da virgole. Si consulti la man-page per un elenco.

`isakmpd.policy` è molto più stringato. Qui di seguito un esempio:

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "passphrase:ThisIsThePassphrase"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg == "3des" &&
            esp_auth_alg == "hmac-md5" -> "true";
```

Per provare la connessione si avvii isakmpd:

```
isakmpd -d -4 -DA=90
```

isakmpd partirà in foreground (-d) utilizzzando ipv4 (-4) e un debuglevel pari a 90.

Una volta instaurata la connessione sarà possibile effettuare dei ping da una subnet all'altra. Se sono stati installati gli ipsec-tools utilizzando il comando **setkey** si potranno visualizzare le policies e le security association aggiunte da isakmpd. Se si usa ctrl-c per fermare isakmpd, non verranno cancellati anche i due database, SAD e SPD. Sarà necessario provvedere manualmente ad essa. Se si utilizza invece **kill -TERM** saranno svuotati entrambi.

Utilizzare certificati X.509

L'isakmpd può utilizzare certificati X.509 per il processo di autenticazione. È possibile creare i certificati utilizzando i normali strumenti a disposizione ed è necessario che per ogni macchina partecipante alla VPN, esistano i seguenti files:

- `/etc/isakmpd/private/local.key` La chiave privata della macchina in .pem format. Con permessi 600.
- `/etc/isakmpd/ca/ca.crt` Il certificato della certification authority.
- `/etc/isakmpd/certs/ip-address.crt` Il certificato della macchina locale.

Perchè isakmpd possa trovare ed utilizzare un certificato quest'ultimo deve possedere un SubjectAltName. Questa estensione del X.509v3 può essere definita durante la generazione del certificato o aggiunta in seguito utilizzando il comando **certpatch**. Quest'ultimo ha bisogno della chiave privata della CA: estrae il certificato, aggiunge l'estensione e lo rifirma.

```
certpatch -i ip-address -k ca.key  originalcert.crt newcert.crt
```

Certpatch può aggiungere un indirizzo IP, un FQDN o un UFQDN al certificato.

Quando questi files sono stati creati e riposti nelle directory appropriate con i giusti permessi è possibile procedere con la creazione del file di configurazione e delle policy. Nel primo è necessario rimuovere l'opzione Authentication ed aggiungere la linea **ID=East** nella sezione ISAKMP-peer-west. Quindi bisogna definire East. Inoltre è necessario specificare i percorsi per raggiungere i certificati e quanto in relazione con essi. Un file di configurazione completo:

```
[General]
Listen-on=              192.168.1.100

[Phase 1]
192.168.2.100=                ISAKMP-peer-west

[Phase 2]
Connections=            IPsec-east-west

[ISAKMP-peer-west]
Phase=                  1
Local-address=          192.168.1.100
Address=                192.168.2.100
ID=                     East

[East]
ID-type=                IPV4_ADDR
Address=                192.168.1.100

[IPsec-east-west]
Phase=                  2
ISAKMP-peer=            ISAKMP-peer-west
Configuration=          Default-quick-mode
Local-ID=               Net-east
Remote-ID=              Net-west
```

```
[Net-west]
ID-type=                IPV4_ADDR_SUBNET
Network=                172.16.1.0
Netmask=                255.255.255.0

[Net-east]
ID-type=                IPV4_ADDR_SUBNET
Network=                172.16..2.0
Netmask=                255.255.255.0

[Default-quick-mode]
DOI=                    IPSEC
EXCHANGE_TYPE=          QUICK_MODE
Suites=                 QM-ESP-AES-SHA-PFS-SUITE

[X509-certificates]
CA-directory=           /etc/isakmpd/ca/
Cert-directory=         /etc/isakmpd/certs/
Private-key=            /etc/isakmpd/private/local.key
```

Anche il file contenente la policy deve essere modificato. Dal momento che intendiamo accettare soltanto host muniti di certificato firmato dalla CA di cui ci fidiamo, aggiungiamo alcune righe dopo l'opzione Authorizer. Un file di policy completo:

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "DN:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.Com/OU=VPN/CN=RootCA"
Conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg == "3des" &&
            esp_auth_alg == "hmac-md5" -> "true";
```

Il testo che segue l'opzione **DN:** deve coincidere con il subject del certificato della CA:

```
openssl x509 -in ca/ca.crt -noout -subject
```

Ora è possibile avviare isakmpd come al solito e provare la connessione.

Generare Certificati X.509

La maggior parte del software per realizzare VPN utilizza certificati X.509 per l'autenticazione delle parti. Sono gli stessi certificati utilizzati dal Secure Socket Layer (SSL) nel protocollo HTTP.

Questo capitolo illustra brevemente come creare un certificato.

Utilizzare OpenSSL

Il modo più semplice per creare un certificato X.509 con Linux consiste nel servirsi del comando **openssl** e di alcuni comandi correlati. Quando il pacchetto OpenSSL viene installato fornisce di solito anche il comando **CA** o **CA.pl**, che utilizzeremo per la generazione dei certificati.

Per prima cosa si verifichi che il comando sia realmente disponibile. Di solito non è presente nel path. Nella Red Hat si trova in `/usr/share/ssl/misc/CA`.

Quindi si proceda con la creazione di una certification authority.

```
$ mkdir certs
```

```
$ cd certs
$ /usr/share/ssl/misc/CA -newca
CA certificate filename (or enter to create) <enter>

Making CA certificate ...
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
...............++++++
.............++++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase: capassword
Verifying password - Enter PEM pass phrase: capassword
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [NRW]:
Locality Name (eg, city) [Steinfurt]:
Organization Name (eg, company) [Spenneberg.com]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:RootCA 2003
Email Address []:ralf@spenneberg.net
```

Si inseriscano i valori opportuni alla richiesta del Country Name, etc. Se si desidera che vengano proposti valori di default corretti (come nel nostro caso) si modifichi di conseguenza il file `openssl.cnf`. Nella Red Hat si trova di solito in `/usr/share/ssl/openssl.cnf`.

La certification authority è valida per un anno. Normalmente si desidera però che il certificato della propria CA abbia una vita più lunga. È possibile cambiare questo valore al volo senza modificare il file `openssl.cnf`:

```
$ cd demoCA/
$ openssl x509 -in cacert.pem -days 3650 -out cacert.pem
-signkey ./private/cakey.pem
Getting Private key
Enter PEM pass phrase: capassword
$ cd ..
```

La CA è ora pronta. Procediamo con la creazione di una richiesta di firma per un certificato:

```
$ /usr/share/ssl/misc/CA -newreq
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
..............................++++++
..................................++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase: certpassword
Verifying password - Enter PEM pass phrase: certpassword
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
```

```
State or Province Name (full name) [NRW]:
Locality Name (eg, city) [Steinfurt]:
Organization Name (eg, company) [Spenneberg.com]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:VPN-Gateway
Email Address []:ralf@spenneberg.net

Please enter the following 'extrà attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
```

Il file `newreq.pem` contiene la rihiesta e la chiave privata cifrata. Si potrà riutilizzare questo file come chiave privata per FreeS/WAN o Racoon. Ora firmiamo la richiesta utilizzando la certificate authority.

```
$ /usr/share/ssl/misc/CA -sign
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DÈ
stateOrProvinceName   :PRINTABLE:'NRW'
localityName          :PRINTABLE:'Steinfurt'
organizationName      :PRINTABLE:'Spenneberg.com'
commonName            :PRINTABLE:'VPN-Gateway'
emailAddress          :IA5STRING:'ralf@spenneberg.net'
Certificate is to be certified until Apr 29 06:08:56 2004 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

A seconda della versione del comando **CA** il certificato potrebbe essere visualizzato sullo stdout. Assomiglierà al seguente:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=RootCA 2003/Email=ralf@spenneberg.net
        Validity
            Not Before: Apr 30 06:08:56 2003 GMT
            Not After : Apr 29 06:08:56 2004 GMT
        Subject: C=DE, ST=NRW, L=Steinfurt, O=Spenneberg.com,
CN=VPN-Gateway/Email=ralf@spenneberg.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c5:3b:9c:36:3a:19:6c:a9:f2:ba:e9:d2:ed:84:
                    33:36:48:07:b2:a3:2d:59:92:b0:86:4c:81:2c:ea:
                    5c:ed:f3:ba:eb:17:4e:b3:3a:cc:b7:5b:5d:ca:b3:
                    04:ed:fb:59:3c:c5:25:3e:f3:ff:b0:22:10:fb:de:
                    72:0a:ee:42:4b:9a:d3:27:d3:b6:fb:e9:88:10:c8:
                    47:b7:26:4f:71:40:e4:75:c4:c0:ee:6b:87:b8:6f:
                    c9:5e:66:cf:bb:e7:ad:72:68:b8:6d:fd:8f:4c:1f:
                    3a:a2:0d:43:25:06:b9:92:e7:20:6c:86:15:a0:eb:
```

```
                    7f:f7:0b:9a:99:5d:14:88:9b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CB:5C:19:9B:E6:8A:8A:FE:0E:C4:FD:5E:DF:F7:BF:3D:A8:
18:7C:08
            X509v3 Authority Key Identifier:
                keyid:01:BB:C6:33:BE:F5:9A:5E:B0:0C:5D:BD:41:E9:78:
6C:54:AD:66:8E
                DirName:/C=DE/ST=NRW/L=Steinfurt/O=Spenneberg.com/
CN=RootCA 2003/Email=ralf@spenneberg.net
                serial:00

    Signature Algorithm: md5WithRSAEncryption
        6f:89:2b:95:af:f1:8d:4d:b7:df:e8:6d:f7:92:fb:48:8c:c4:
        1a:43:68:65:97:01:87:a6:84:b5:a1:38:bd:62:74:70:db:9e:
        78:19:d9:0c:af:18:ad:13:77:56:7d:3f:19:61:da:ba:74:30:
        8e:c5:50:0e:e3:eb:ff:95:cd:8d:d6:7e:c3:0e:ab:5b:34:94:
        bc:16:0f:ef:dc:de:40:bb:7d:ba:a2:b8:5d:f9:74:e7:28:58:
        75:a0:66:d2:8d:85:ba:38:82:08:10:33:ef:be:29:c9:31:9d:
        63:a9:f7:e0:99:ea:a7:ed:b6:b5:33:1b:1c:4a:a4:05:40:6e:
        40:7b
-----BEGIN CERTIFICATE-----
MIIDjDCCAvWgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBgjELMAkGA1UEBhMCREUx
DDAKBgNVBAgTA05SVzESMBAGA1UEBxMJU3RlaW5mdXJ0MRcwFQYDVQQKEw5TcGVu
bmViZXJnLmNvbTEUMBIGA1UEAxMLUm9vdENBIDIwMDMxIjAgBgkqhkiG9w0BCQEW
E3JhbGZAc3Blbm5lYmVyZy5uZXQwHhcNMDMwNDMwMDYwODU2WhcNMDQwNDI5MDYw
ODU2WjCBgjELMAkGA1UEBhMCREUxDDAKBgNVBAgTA05SVzESMBAGA1UEBxMJU3Rl
aW5mdXJ0MRcwFQYDVQQKEw5TcGVubmViZXJnLmNvbTEUMBIGA1UEAxMLVlBOLUdh
dGV3YXkxIjAgBgkqhkiG9w0BCQEWE3JhbGZAc3Blbm5lYmVyZy5uZXQwgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBAMU7nDY6GWyp8rrp0u2EMzZIB7KjLVmSsIZM
gSzqXO3zuusXTrM6zLdbXcqzBO37WTzFJT7z/7AiEPvecgruQkua0yfTtvvpiBDI
R7cmT3FA5HXEwO5rh7hvyV5mz7vnrXJouG39j0wfOqINQyUGuZLnIGyGFaDrf/cL
mpldFIibAgMBAAGjggEOMIIBCjAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1P
cGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUy1wZm+aKiv4O
xP1e3/e/PagYfAgwga8GA1UdIwSBpzCBpIAUAbvGM771ml6wDF29Qel4bFStZo6h
gYikgYUwgYIxCzAJBgNVBAYTAkRFMQwwCgYDVQQIEwNOUlcxEjAQBgNVBAcTCVN0
ZWluZnVydDEXMBUGA1UEChMOU3Blbm5lYmVyZy5jb20xFDASBgNVBAMTC1Jvb3RD
QSAyMDAzMSIwIAYJKoZIhvcNAQkBFhNyYWxmQHNwZW5uZWJlcmcubmV0ggEAMA0G
CSqGSIb3DQEBBAUAA4GBAG+JK5Wv8Y1Nt9/obfeS+0iMxBpDaGWXAYemhLWhOL1i
dHDbnngZ2QyvGK0Td1Z9Pxlh2rp0MI7FUA7j6/+VzY3WfsMOq1s0lLwWD+/c3kC7
fbqiuF35dOcoWHWgZtKNhbo4gggQM+++KckxnWOp9+CZ6qfttrUzGxxKpAVAbkB7
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
```

È opportuno rinominare i files `newreq.pem` e `newcert.pem` in qualcosa di più significativo.

```
$ mv newcert.pem vpngateway_cert.pem
$ mv newreq.pem vpngateway_key.pem
```

Ora potrete divertirvi creando certificati per ogni peer nella VPN.

Nel caso in cui una chiave privata venga persa o compromessa, è necessario revocarla poichè il certificato corrispondente è ancora valido. Le chiavi di revoca sono contenute nella certificate revocation list (CRL). Per prima cosa creiamo una lista vuota:

```
$ openssl ca -gencrl -out crl.pem
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
```

Per revocare un certificato è necessario possedere il file che lo contiene. Quest'ultimo è conservato anche in `demoCA/newcerts/`. Il nome del certificato può essere letto da `demoCA/index.txt`. Quindi si utilizzi il seguente comando:

```
$ openssl ca -revoke compromised_cert.pem
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase: capassword
Revoking Certificate 01.
Data Base Updated
```

Quando il certificato è stato revocato è necessario rigenerare la certificate revocation list riutilizzando il comando di creazione.

Notes

1. http://www.tldp.org
2. http://www.ipsec-howto.org
3. http://www.tldp.org/HOWTO/Networking-Overview-HOWTO.html
4. http://www.tldp.org/HOWTO/Net-HOWTO/index.html
5. http://www.tldp.org/HOWTO/VPN-Masquerade-HOWTO.html
6. http://www.tldp.org/HOWTO/VPN-HOWTO/
7. http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/
8. http://www.kernel.org
9. http://ipsec-tools.sourceforge.net
10. http://bender.thinknerd.de/~thomas/IPsec/isakmpd-linux.html
11. http://www.spenneberg.org/VPN/Kernel-2_6_IPsec
12. http://www1.cs.columbia.edu/~angelos/keynote.html