IKEV2 TESTING

DAVIDE DE ZUANE & RAHMI EL MECHRI

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1 INTRODUCTION

L'aumento della connettività e n

La sicurezza negli ultimi anni sta diventando sempre più importante per le comunicazioni,

c'è sempre più bisogno di realizzare collegamenti sicuri, uno strumento molto utile a questo fine sono le VPN.

Tutte le informazion su IKE sono contenute all'interno della RFC7296[1].

1.1 IPsec

IPsec (Internet Protocol Security) è una suite di protocolli che fornisce sicurezza alle comunicazioni Internet a livello IP. L'uso attuale più comune di IPsec è quello di fornire una rete privata virtuale (VPN). IKE (Internet Key Exchange) Internet Key Exchange) è il protocollo di negoziazione e gestione delle chiavi più comunemente munemente utilizzato per fornire chiavi negoziate e aggiornate dinamicamente per IPsec. Introduzione a quello che è ipsec e suo funzionamento

1.2 IKE

Importanza di IKE per effettuare lo scambio di chiavi per stabilire la SA.

Concetti su cui si basa IKE

1.3 Strongswan

Parlare rapidamente di quello che è quello che è anche l'architettura di strongswan charon.

2 SETUP

Andiamo a vedere nel dettaglio l'ambiente e la configurazione che abbiamo utilizzato per realizzare i test. Per verificare le capacità di IKE abbiamo previsto:

- 3 modalità di autenticazione;
- 2 chiper suite differenti da utlizzare.

Nella fase di sperimentazione abbiamo utilizzato le seguenti convenzioni:

- **Initiator**: l'host che invia la richiesta di stabilire una SA;
- **Responder**: l'host che risponde alle richieste.

2.1 Environment

Per simulare i due host della comunicazione abbiamo creato due macchine virtuali tramite l'utilizzo di qemu/kvm, questo per avere delle performance il più possibile simili a quelle reali. Le due macchine virtuali sono state create in modalità bridge, questo per evitare problemi con la modalità NAT. In questo modo le due macchine appartengono ala stessa rete e questo ci fa facilita la configurazione.
Le macchine virtuali utilizzato hanno le seguenti specifiche:

- *Processore*: 2 core
- *Memoria*: 2048MB
- *OS*: Debian 11
- *Network*: Bridge

Setup delle macchine virtuali tramite qemu cli. Dopo le macchine virtuali la lista dei vari requirements da installare

2.2 Configuration

I file e le directory coinvolte nel processo di configurazione sono i seguenti. Dato che una delle principali modifiche di IKEv2 rispetto alla versione precedente è la possibilità di autenticazione tramite certificati. I file e le directory interessate dalla configurazione sono le seguenti.

```
/etc
├── ipsec.conf
├── ipsec.secrets
└── ipsec.d
    ├── cacerts
    ├── certs
    └── private
```

- Il file `ipsec.conf`[1] specifica la maggior parte delle configurazioni e le informazioni di controllo per il sottosistema IPsec (ulteriori specifiche e sintassi sono disponibili al seguente link).
- Il file `ipsec.secrets`[1] continene i segreti che poi verrranno utilizzati nella fase di autenticazione (ulteriori specifiche al seguenti link).

1 Le configurazioni utilizzate si trovano in appendice.

Certificati

Una delle principali novità che introduce `IKEv2` è la possibilità di eseguire l'autenticazione tra certificati X.509. In fase di testing abbiamo preso in considerazione due tipi di certificati:

- Certificati RSA
- Certificati ECDSA

A partire da una chiave pubblica è necessario realizzare un certificato di chiave pubblica e questo richiede la chiave privata di una CA. Nel nostro caso ci siamo creati dei certificati da CA e li abbiamo ditribuiti manualmente tra i due host.

Per la generazione abbiamo utilizzato il tool `pki`

CA Certificate

Partiamo con la generazione dei certificati da Certification Authority, di seguito sono riportati i due comandi da utilizzare. Ne occorrono due poichè per firmare i certificati ECDSA occorre una chiave con lo stesso schema.

```
$ pki --gen --type rsa --size 2048 --outform pem > 'ca.rsa.key.pem'
$ pki --gen --type ecdsa --size 256 --outform pem > 'ca.ecdsa.key.pem'
```

Ora utilizziamo la chiave privata per firmare il certificato di chiave pubblica.

```
$ pki --self --ca --lifetime 3650 --in 'ca.<type>.key.pem' --type <type> \
    --dn "CN=CA" --outform pem > ca.<type>.cert.pem
```

Occorre poi distribuire questi due certificati ai due host, vanno messi all'intenro della directory `cacerts`.

Host Certificate

Passiamo ora a generare i certificati che gli host andranno ad utilizzare nella fase di autenticazione, occorre generare la coppia chiave privata, chiave pubblica.

```
$ pki --gen --type ecdsa --size 256 --outform pem > 'host.ecdsa.key'
$ pki --gen --type rsa --size 2048 --outform pem > 'host.rsa.key'
```

E' buona norma salvare le chiavi all'interno della directory `private`. Ora andiamo ad estrarre la chiave pubblica da quella appena genrata e la firmiamo con la chiave delle CA del passo precedente.

```
$ pki --pub --in 'host.rsa.key' --type rsa | pki --issue --lifetime 1825 \
   --cacert 'ca.rsa.cert.pem' --cakey 'ca.rsa.key.pem'                   \
   --dn "CN=<Host_IP>" --san @<Host_IP> --san <Host_IP>                  \
   -- flag serverAuth --outform pem > 'host.rsa.cert.pem'
```

Si procede in maniera analoga con le opportune modifiche anche per il certificato ECDSA. Questi vanno poi posiizonati all'intenro della directory `certs`.

2.2.1 Mschap

Il riassunto della configurazione è mostrato in tabella, per l'initiator e il responder sono riportate le modalità della loro autenticazione.

Configuration	
Initiator	EAP-Mschapv2
Responder	RSA Certificate 2048
Chiper Suite	AES_CBC_128_HMAC_SHA2_256_128_DH_ECP_256

Esaminando gli scambi di IKE AUTH osserviamo che questa modalità richiede in totale 4 exchange.

2.2.2 RSA

Configuration	
Initiator	RSA Certificate 2048
Responder	RSA Certificate 2048
Chiper Suite	AES_CBC_128_HMAC_SHA2_256_128_DH_ECP_256

2.2.3 ECDSA

Configuration	
Initiator	ECDSA Certificate 256
Responder	ECDSA Certificate 256
Chiper Suite	AES_CBC_128_HMAC_SHA2_256_128_DH_ECP_256

3 TESTING

Per misurare i cicli macchina abbiamo utlizzato perf

per installarlo apt-get intstall linux-perf se da problemi con workload failed è a causa dei permessi e per risolverlo basta sovrascrivere il contenuto di

/proc/sys/kernel/perf$_{e}$vent$_{p}$aranoid per fare il report di tutto l'ambiente utilizzare pef report

3.1 Time

3.2 Performance

3.3 Results

4 CONSLUSIONI

A CONFIGURATION FILE

Di seguito riportiamo i file di configurazione `ipsec.conf` e `ipsec.secrets` rispettivamente di initiator e di responder. Una possibile modifica ai file potrebbe essere quella di rendere il tutto simmetrico, allo stato attuale i due non possono scambiarsi di ruolo. Alcune note:

- la connessione **default** definisce la configurazione comune a tutte le altre.
- la connessione **secure** è quella con cui specifichiamo la chiper_suite sicura.
- **also** permette di realizzare l'erditarietà multipla tra le connessioni.
- il parametro **auto** specifica quale operazione effettuare con la connessioni all'avvio di IPsec; il valore *add* la aggiunge alle possibile conessioni ma non cerca di stabilirla

A.1 Initiator

ipsec.conf

```
########################################################
# ipsec.conf - strongSwan IPsec configuration file
########################################################
conn %default
    leftsourceip=%config
    right=<ip_responder>
    rightsubnet=0.0.0.0/0
    auto=add

conn secure
    ike=aes256-sha384-ecp384!

conn base-mschap
    leftauth=eap-mschapv2
    eap_identity="<identity>"
    rightauth=pubkey

conn base-rsa
    rightauth=pubkey-rsa-2048
    leftauth=pubkey-rsa-2048
    leftcert=<path_to_cert>

conn base-ecdsa
    rightauth=pubkey-ecdsa-256
    leftauth=pubkey-ecdsa-2048
    leftcert=<path_to_cert>

conn secure-rsa
    also=base-rsa
    also=secure

conn secure-ecdsa
    also=base-ecdsa
    also=secure

conn ipsec-ike
    also=secure
    also=base-mschap
```

ipsec.secrets

```
##########################################################
# ipsec.secrets - strongSwan IPsec configuration file
##########################################################
<identity> : EAP "<password>"

: ECDSA "/etc/ipsec.d/private/<key>.pem"
: RSA "/etc/ipsec.d/private/<key>.pem"
```

A.2 Responder

ipsec.conf

```
##########################################################
# ipsec.conf - strongSwan IPsec configuration file
##########################################################
conn %default
    keyexchange=ikev2
    left=<ip_host>
    leftsubnet=0.0.0.0/0
    forceencaps=yes
    compress=no
    type=tunnel
    fragmentation=yes
    rekey=no
    right=<ip_initiator>
    rightid=%any
    rightsourceip=0.0.0.0/0
    rightdns=8.8.8.8,4.4.4.4
    auto=add

conn mschap
    rightauth=eap-mschapv2
    eap_identity=%identity
    leftcert=<path_to_cert>
    leftsendcert=always

conn rsa
    leftcert=<path_to_cert>
    leftauth=pubkey-rsa-2048
    rightauth=pubkey-rsa-2048

conn ecdsa
    leftcert=<path_to_cert>
    leftauth=ecdsa-256
    rightauth=ecdsa-256
```

ipsec.secrets

```
##########################################################
# ipsec.secrets - strongSwan IPsec configuration file
##########################################################
<identity> : EAP "<password>"

: RSA "/etc/ipsec.d/private/<key>.pem"
```

```
: ECDSA "/etc/ipsec.d/private/<key>.pem"
```

B TOOLS

Per instaurare la connessione IPsec si utilizza il seguente comando.

```
$ ipsec up <conn_name>
```

Per verificare che la SA sia stata correttamente instaurata è possibile utilizzare il seguente tool `ip xfrm`, il quale consente di effettuare la trasformazione dei pacchetti. Questo fornisce un interfaccia ai due database:

- SAD: Security Association Database, tramite l'oggetto `state`.
- SPD: Security Policy Database, tramite l'oggetto `policy`.

L'esecuzione del seguente comando fornisce una vista delle entry presenti nel SAD, possiamo poi utilizzare queste informazioni in wireahrk per poter vedere il traffico tra i due host in chiaro.

```
$ ip xfrm state list
src <initiator> dst <responder>
    proto esp spi 0xc49d3a6d reqid 1 mode tunnel
    replay-window 0 flag af-unspec
    auth-trunc hmac(sha256) <skey> 128
    enc cbc(aes) <skey>
    anti-replay context: seq 0x0, oseq 0xc, bitmap 0x00000000
src <responder> dst <initiator>
    proto esp spi 0xca382e6d reqid 1 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha256) <skey> 128
    enc cbc(aes) <skey>
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
```

Wireshark

Per vedere il traffico sniffato in chiaro occorre configurare il protocollo ISAKMP all'interno di wireshark, andiamo a specificare quelle che sono le chiavi negoziate per l'autenticazione di messaggio e di cifratura.

- Andare su `Edit->Preferences->Protocols->ISAKMP`.
- Aggiungere all'interno della tabella le varie entry riportate tramite `ip xfrm`

C CERTIFICATI

Spiegare quella che è una root CA
Come funziona la catena di certificati
Spiegare quelli che sono i campi all'interno di un certificato
Cofronto tra certificato RSA e ECDSA
Per verificare le informazioni contenute all'interno di un certificato utlilizzare il seguente comando:

```
$ openssl x509 --in <certificate>
```

REFERENCES

[1] Charlie Kaufman, Paul E. Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, October 2014.