Firewall

Laboratorio del corso "Sicurezza dei sistemi informativi" (01UDUOV)
Politecnico di Torino – AA 2024/25
Prof. Cataldo Basile

preparata da:
Gabriele Gatti (gabriele.gatti@polito.it)

v. 3.7 (28/11/2023)

Indice

1 Scopo dell'esercitazione

Scopo di questa esercitazione è sperimentare configurazione ed uso degli elementi base di un sistema firewall usando i seguenti componenti software:

- Netfilter / IP Tables – fornisce funzionalità di manipolazione e filtraggio di pacchetti IP all'interno del kernel Linux;
- Dynamic port forwarding SSH – la funzionalità di "dynamic port forwarding" permette di configurare il server SSH come circuit-level gateway (`man sshd` per maggiori informazioni);

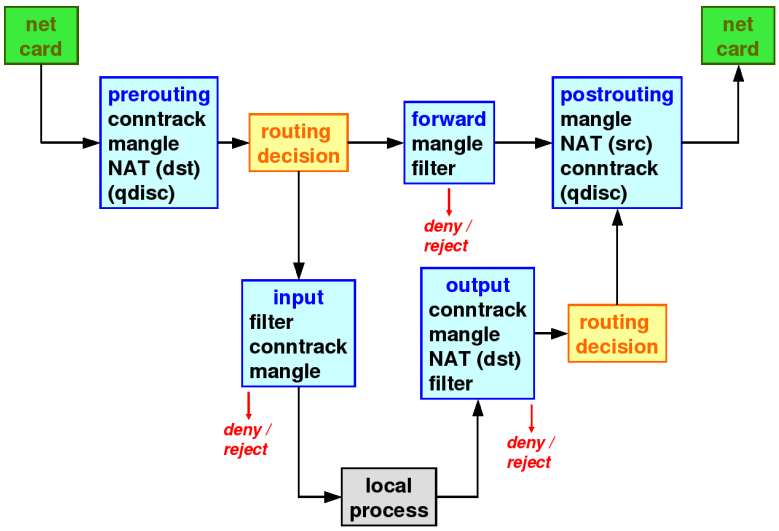

Figura 1: Architettura di Netfilter/IPtables

- Mod Proxy – il modulo `mod_proxy` permette di configurare un server Apache come un application gateway per applicazioni web;
- HTTPtunnel – crea tunnel HTTP per tentare di aggirare le politiche dei firewall;
- PTunnel – crea tunnel usando i pacchetti ICMP echo-request ed echo-reply per tentare di aggirare le politiche dei firewall.

IPtables

IPtables, la cui architettura è riassunta in Figura 1, utilizza il concetto di "chain", una lista ordinata di regole a cui è associata una politica di default. Esistono, per il filtraggio dei pacchetti, tre catene predefinite:

- `INPUT`, contiene le regole che devono essere applicate ai pacchetti diretti alla macchina su cui è installato IPtables che si sta configurando;
- `OUTPUT`, contiene le regole che devono essere applicate ai pacchetti in uscita dalla macchina su cui è installato IPtables che si sta configurando;
- `FORWARD`, contiene le regole che devono essere applicate ai pacchetti che non provengono né sono diretti alla macchina su cui è installato IPtables ma che, se l'IP forwarding è abilitato, verranno inoltrati verso una specifica interfaccia.

Naturalmente, altre chain possono essere create per rendere più semplice ed organizzata la gestione delle regole.
IPtables può essere configurato con il comando `iptables` (`man iptables` per maggiori informazioni). Di seguito una lista dei principali comandi:

- l'opzione `-P chain target` permette di modificare la politica di autorizzazione di default della catena `chain` (INPUT, OUTPUT, FORWARD o personalizzate) impostandola al valore `target` (ulteriori spiegazioni sul significato di target sono riportate alla fine di questo paragrafo);
- `-A chain` e `-I chain` permettono di aggiunge una nuova regola rispettivamente in coda e in testa alla catena `chain`;
- `-D chain` e `-R chain` permettono di cancellare o sostituire una regola in `chain` (tramite l'ordinale, se è noto, o tramite un indirizzo IP);

- `-F [chain]` cancella tutte le regole presenti in `chain` o tutte le catene, se non viene specificato il valore di `chain` (attenzione: la politica di default non viene però modificata).
- `-N chain` crea una nuova catena di nome `chain`;
- `-X chain` cancella la catena `chain`, creata dall'utente.

Si mostra di seguito l'utilizzo del comando necessario a re-inizializzare tutte le catene:

```
iptables -F
```

oppure solo una, quella passata come parametro (in questo caso quella di OUTPUT):

```
iptables -F OUTPUT
```

Quando si specifica una regola di filtraggio, è necessario fare riferimento a diverse porzioni del pacchetto IP. Per questo IPtables mette a disposizione una serie di comandi (la lista completa è disponibile tramite `man`):

- `-s IP_source`, indirizzo IP sorgente;
- `-d IP_dest`, indirizzo IP destinazione;
- `--sport porta`, porta sorgente;
- `--dport porta`, porta destinazione;
- `-i interfaccia`, interfaccia di rete di invio/ricezione del pacchetto (catena di INPUT);
- `-o interfaccia`, interfaccia di rete di invio/ricezione del pacchetto (catena di OUTPUT);
- `-p proto`, protocollo (es. `tcp`, `udp`);
- `-j azione`, azione da intraprendere (il *target* in terminologia IPtables);
- `-y` o `--syn`, se viene usata anche l'opzione `-p tcp`, identifica un pacchetto con (solo) il flag SYN abilitato;
- `--icmp-type tipo`, equivalente a `-p icmp`, `tipo` specifica il tipo di pacchetto ICMP
- `-l`, abilita il log del risultato della valutazione di una regola in tramite syslog in `/var/log/messages`

Le azioni da intraprendere, in terminologia IPtables, i "target", sono i seguenti:

- `ACCEPT`, accetta il pacchetto;
- `DROP`, scarta il pacchetto e non invia notifica;
- `REJECT`, scarta il pacchetto ma invia una notifica, sconsigliato nella maggior parte dei casi perché accelera le procedure di scanning. Di default, viene inviato un pacchetto di tipo "ICMP port unreachable", tuttavia questo può essere cambiato impostando l'opzione `--reject-with`;
- `RETURN`, impone l'applicazione della default action anche se non è stata esaminata tutta la catena.

Altri target validi sono tutte le catene definite dall'utente.
I file di configurazione necessari per svolgere alcuni esercizi (`04lab-materiale.zip`) sono disponibili sul portale della didattica.

Comandi accessori

Nel corso dell'esercitazione dovrete scambiarvi dati da un computer all'altro col comando `scp`.
Prima di avviare il server per la prima volta generate le chiavi dell'host con il comando:

```
ssh-keygen -A
```

Abilitate quindi il server `ssh`:

```
systemctl start ssh
```

e ricordate che per copiare un file (es. `prova`) con `ssh` da una macchina client alla directory `root` su di una macchina con un server `ssh` attivo, potete usare il seguente comando (inserendo la password di root quando richiesto):

```
scp prova root@IP_address_ssh_server:/root
```

Verrà inoltre usato il server web Apache; per avviarlo usate il comando:

```
systemctl start apache2
```

2 Packet filter

2.1 Personal firewall

In questo primo esercizio introdurremo l'uso del comando `iptables` e vedremo come configurare un packet filter locale, per proteggere una singola macchina.

Svuotate tutte le chain di IpTables con il comando:

```
iptables -F
```

Verificate la configurazione di IPtables con il comando:

```
iptables -L -v -n
```

Qual è la politica di autorizzazione configurata al momento? (Ignorate le chain che fanno riferimento a Docker, non sono rilevanti ai fini dell'esercitazione)

→

Su quale catena dovrete agire per proteggere la vostra macchina da connessioni esterne?

→

Ora formate delle coppie (diciamo Alice e Bob, dove Alice sarà la macchina da proteggere e Bob ne verificherà la configurazione) e procedete come segue:

- Alice attiva i server Apache e SSH;
- Bob verifica di poter raggiungere la macchina di Alice via `ping` e i servizi attivati, ad esempio:

  ```
  ping indirizzo_IP_Alice
  ```

  ```
  nmap -sT -Pn -n -p 80,22 -v indirizzo_IP_Alice
  ```

Scrivete il comando `iptables` per modificare la politica di autorizzazione di Alice affinché rifiuti qualsiasi tipo di traffico in ingresso (Suggerimento: dovete modificare la politica di default della catena INPUT da ACCEPT a DROP):

→

Bob verifica di non poter più inviare ping e connettersi via SSH e HTTP.

Ora scrivete il comando `iptables` per aggiungere una regola alla politica di Alice e abilitare tutto il traffico ICMP:

→

Allo stesso modo, scrivete il comando per aprire la porta 80/TCP:

→

Eseguite le modifiche e ispezionate la nuova configurazione di IPtables. Verificate che Bob sia nuovamente in grado di inviare ping e connettersi via HTTP alla macchina di Alice.

Ora procedete in questo modo:

1. Alice sceglie alcune porte e le rende accessibili tramite iptables (ad esempio la 22 e la 81);
2. Bob effettua una scansione con `nmap` e cerca di individuare quali porte Alice ha aperto:

   `nmap` *indirizzo_IP_Ali* `-p 10-100`

Come `nmap` riesce a distinguere tra porte filtrate o chiuse? Verificate analizzando il traffico scambiato tra le due macchine (ad esempio con `wireshark`).

→

Infine, avviate il server Apache sulla macchina di Bob e provate a connettervi dalla macchina di Alice.

Perchè la connessione non va a buon fine? Verificate analizzando il traffico tra le due macchine.

→

Come dovrebbe essere modificata la politica di autorizzazione di Alice utilizzando unicamente funzionalità di uno "stateless" packet filter, affichè possa connettersi al server HTTP di Bob senza che Bob possa individuare i servizi esposti da Alice?

→

E se avessimo a disposizione un packet filter "stateful"?

```
→
```

Ora cancellate tutte le regole create e ripristinate su Alice una politica di tipo “ACCEPT ALL” prima di passare al prossimo esercizio.

```
→
```

2.2 Stateless packet filter

Formate dei gruppi di tre macchine (diciamo Alice, Bob e Frank, dove Frank fornirà funzioni di “firewalling” tra Alice, il server, e Bob, il client).

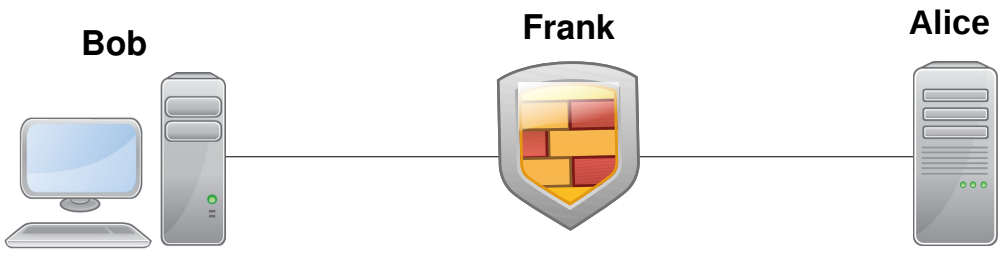

Figura 2: Configurazione di riferimento

Per “simulare” in laboratorio la configurazione di Figura 2, procedete come segue (potrebbe essere necessario eseguire i comandi con `sudo`):

1. Alice configura una regola di routing per instradare attraverso Frank i pacchetti per Bob:

   tramite comando `ip`

   ```
   ip route add IP_Bob via IP_Frank dev interfaccia
   ```

   oppute con i `net-tools`

   ```
   route add -host IP_Bob gw IP_Frank
   ```

2. analogamente, Bob configura una regola per instradare attraverso Frank i pacchetti per Alice:

   tramite comando `ip`

   ```
   ip route add IP_Alice via IP_Frank dev interfaccia
   ```

   oppure

   ```
   route add -host IP_Ali gw IP_Fra
   ```

3. Frank disabilita l’invio di ICMP redirect ed abilita il forwarding di pacchetti:

   ```
   echo 0 | sudo tee /proc/sys/net/ipv4/conf/all/send_redirects
   echo 0 | sudo tee /proc/sys/net/ipv4/conf/eth0/send_redirects
   echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
   ```

4. Frank verifica quale sia la default policy della chain FORWARD e la imposta su ACCEPT con l’apposito comando.

5. Alice e Bob attivano i server Apache e SSH

Prima di procedere, verificate (ad esempio con un ping) che tutti gli host siano raggiungibili tra loro. Verificate inoltre che il traffico scambiato da Alice e Bob passi effettivamente attraverso Frank (ad esempio con `wireshark`).

NOTA

In alcuni casi non si riesce proprio a convincere gli host coinvolti a non mandare pacchetti ICMP redirect. In teoria Frank, il default gateway, dice "Alice, se vuoi andare da Bob, vacci direttamente, non serve che passi attraverso me" (e dice lo stesso a Bob). Ma anche se disabilitamo i "send redirect", in alcuni casi, Alice riceve il pacchetto ICMP Host Redirect. Ve ne accorgete dal ping, il primo output visualizza

```
Redirect Host(New nexthop:  192.168.1.8)
```

Se questo dovesse verificarsi, per ottenere i risultati che ci aspettiamo da questa esercitazione, Alice e Bob devono eseguire i seguenti comandi:

- cancellare la cache locale di route

  ```
  ip route flush cache
  ```

- aggiungere una direttiva nella catena di INPUT di iptables che vieti la ricezione di pacchetti ICMP redirect

  ```
  iptables -A INPUT -p icmp --icmp-type redirect -j DROP
  ```

2.2.1 Traffico in uscita

In questo passo, Frank applicherà una politica di autorizzazione per permettere ad Alice di navigare su qualsiasi sito web esterno: Alice agirà da client sulla rete protetta e Bob da server web esterno.

Configurate la seguente politica di autorizzazione sul packet filter di Frank:

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp -s IP_Alice --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp -d IP_Alice --sport 80 -j ACCEPT
```

e rispondete alle seguenti domande:

Su quale catena si è agito?

→

Alice può collegarsi al server web di Bob?

→

A cosa serve l'ultima regola?

→

E al server web di un'altra macchina, sempre passando attraverso Frank?

→

Bob può collegarsi al server web di Alice? Suggerimento: un attaccante può scegliere qualunque porta sorgente.

→

Alice e Bob possono accedere ai rispettivi server SSH?

→

NOTA

Potete usare `nmap` per verificare la configurazione. Ad esempio, eseguite sulla macchina di Bob:

```
nmap -sS -Pn -n -p 22 IP_Ali --source-port 80
```

Scrivete (ed eseguite) il comando `iptables` per modificare la politica di Frank affinché siano effettivamente abilitate solo le connessioni web provenienti da Alice e dirette verso qualsiasi indirizzo IP (i.e. nessuno può connettersi al server di Alice in nessun modo ma Alice può ricevere pacchetti HTTP).

NOTA

Le regole sono elaborate dal kernel nell'ordine in cui appaiono visualizzate da:

```
iptables -L -v -n
```

→

Verificate la nuova configurazione con `nmap`.

2.2.2 Servizio pubblico

In questo passo, Frank applicherà una politica di autorizzazione per permettere di accedere remotamente alla macchina di Alice via SSH: Alice agirà da server SSH sulla rete protetta e Bob da client remoto autorizzato all'accesso SSH.

Scrivete i comandi `iptables` per modificare la politica di autorizzazione di Frank affinché accetti anche connessioni SSH verso Alice provenienti da qualsiasi indirizzo IP ma impedisca ad Alice di instaurare connessioni SSH:

→

Eseguite le modifiche e verificate con `nmap` che Bob possa accedere ad Alice via SSH, ma non viceversa.

2.2.3 Traffico ICMP

Ora modificate la politica di Frank per abilitare selettivamente il traffico ICMP.

Scrivete i comandi per abilitate unicamente il traffico ping da e verso Alice:

→

Perché è pericoloso abilitare il traffico ICMP senza restrizioni?
Suggerimento: eseguite `iptables -p icmp -h` per una lista di tipi di messaggio ICMP.

→

Supponete che Frank debba filtrare il traffico diretto verso la sottorete IP `1.2.3.0/24` invece della singola macchina di Alice: cosa accadrebbe con una configurazione simile alla precedente se Frank ricevesse un pacchetto `ICMP echo request` con indirizzo destinazione `1.2.3.255`?

→

2.2.4 Protezione da IP spoofing

Con la configurazione corrente, Frank può rilevare/impedire che Alice e Bob facciano spoofing dei rispettivi indirizzi IP?

→

E nella configurazione di Figura 3?

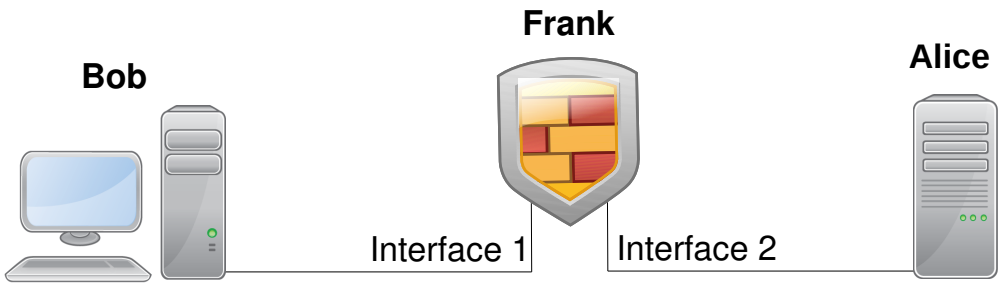

Figura 3: Configurazione con due interfacce di rete

→

2.3 Stateful packet filter

Continuate con gli stessi gruppi formati per l'esercizio 2.2.

Fate riferimento al passo 2.2.1 dell'esercizio precedente e verificate cosa succede eseguendo il seguente comando sulla macchina di Bob:

```
nmap -sA -Pn -n -p 22,25 --source-port 80 IP_Ali
```

Come si può ovviare al problema avendo a disposizione un packet filter?

Identificate la posizione delle regole relative al traffico web configurate in 2.2.1, e modificate la configurazione di Frank eseguendo i seguenti comandi `iptables`:

```
iptables -D FORWARD posizione_regola_web
iptables -A FORWARD -p tcp -s IP_Ali --dport 80 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Qual è il significato delle nuove regole? Quali vantaggi offre questa seconda soluzione? Suggerimento: analizzate l'introduzione della sezione "MATCH EXTENSIONS" e l'uso della direttiva "`--state`" nel manuale di iptables.

```
→
```

Verificate con il comando `nmap` usato in precedenza.

2.3.1 Limitazione banda

Fate riferimento al passo 2.2.3 dell'esercizio precedente.

Spiegate cosa cambierebbe se si sostituisse la regola

```
iptables -A FORWARD -p icmp -d IP_Ali --icmp-type echo-request -j ACCEPT
```

con la regola

```
iptables -A FORWARD -p icmp -d IP_Ali --icmp-type echo-request -m limit --limit
20/minute --limit-burst 1 -j ACCEPT
```

Suggerimento: analizzate l'uso della direttiva "`--limit`" nel manuale `iptables`.

NOTA

E' importante ripartire dalle regole del passo 2.2.3 o quantomeno rimuovere la regola

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

definita nella sezione 2.3 perché mentre lo stato del primo pacchetto `echo-request` è `NEW`, lo stato di quelli successivi è `RELATED`.

```
→
```

3 Approfondimenti ed ulteriori esercizi (opzionali)

Aggirare le regole di un firewall

In questo esercizio rivedremo i concetti fin qui incontrati da una prospettiva differente: ci concentreremo su come la politica di autorizzazione di un firewall può essere facilmente aggirata se un utente interno (Alice) e uno esterno (Bob) colludono.
In particolare, provate a rispondere alle seguenti domande:

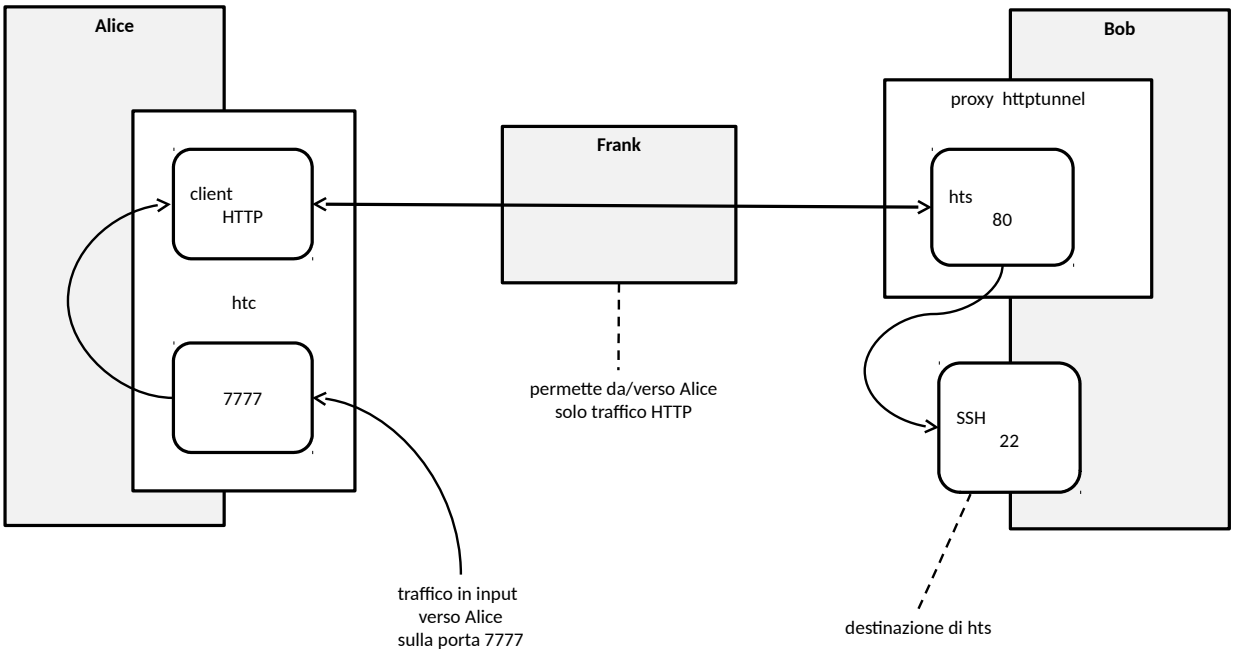

Figura 4: Lo scenario da implementare con HTTPtunnel

- supponete che Frank abbia abilitato il traffico web da Alice verso l'esterno via funzionalità di packet filtering come negli esercizi 2.2 e 2.3: come Alice e Bob possono violare la politica di Frank aprendo una connessione SSH al posto di una web?

  →

- perchè non è semplice filtrare un programma come Skype?

  →

3.1 Creazione di un tunnel HTTP

Proviamo a creare dei tunnel che aggirino le politiche di Frank. Il contesto è quello degli esercizi precedenti:

- Alice è l'utente interno alla rete
- Frank rappresenta il border firewall ed usa un packet filter per implementare una politica di autorizzazione: consentire agli utenti interni di accedere al web (e ricevere le risposte). Come visto negli esercizi precedenti Frank userà le seguenti regole (ricordate di fare il flush delle precedenti configurazioni ed accertatevi che la politica di default sia DROP):

  ```
  iptables -A FORWARD -p tcp -s IP_Ali --dport 80 -j ACCEPT
  ```

  ```
  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
  ```

- Bob è l'utente esterno che collude con Alice. Bob fa partire il server SSH

  ```
  systemctl start ssh
  ```

In questo esercizio utilizzeremo il tool HTTPtunnel per creare un tunnel HTTP che permetterà ad Alice di raggiungere il server presente sulla macchina di Bob nonostante la presenza di un firewall.
Trovate maggiori informazioni su questo strumento al seguente link: https://github.com/larsbrinkhoff/httptunnel
HTTPTunnel è costituito da due componenti, un server (`hts`) e un client (`htc`). HTTPTunnel Server (`hts`) si mette in ascolto in attesa di connessioni HTTPTunnel in arrivo (alla porta 8888 di default) e redirige tutto il traffico ricevuto verso un'altra destinazione, specificata tramite le opzioni `-F` o `-d`.
La sintassi del comando `hts` e le principali opzioni sono presentati di seguito:

```
hts [-F host:port] [port]
```

in cui:

- `-F host:port` specifica l'indirizzo IP e la porta su cui verrà rediretto il traffico ricevuto;
- `port` specifica la porta su cui rimarrà in ascolto `hts`

HTTPTunnel Client (`htc`) apre una connessione con un server HTTPtunnel, si mette in ascolto su una porta ed inoltra tutto il traffico ricevuto al server. La sintassi del comando `htc` e le principali opzioni sono presentati di seguito:

```
htc [-F port] [server:server_port]
```

in cui:

- `-F port` la porta su cui rimarrà in ascolto `htc`;
- `server:server_port` specifica indirizzo e porta del server `hts` con cui aprirà la connessione e verso cui inoltrerà il traffico(il tunnel vero e proprio) .

Useremo i comandi `hts` e `htc` per stabilire i due endpoint del tunnel, rispettivamente server ed il client:
Scrivere il comando che Bob deve lanciare per far partire tramite il comando `hts` un demone HTTP in ascolto sulla porta 80 che redirige il traffico ricevuto verso il server SSH:

```
→
```

Scrivere il comando che Alice deve lanciare per avviare tramite `htc` un processo che intercetta il traffico diretto alla porta locale 7777 e lo rediriga sulla porta 80 di Bob (Figura 4):

```
→
```

Scrivere il comando che Alice deve usare per connettersi al server SSH di Bob tramite il tunnel:

```
→
```

Verificate il contenuto dei pacchetti HTTP scambiati tra Alice e Bob. Sebbene tutto il traffico sia teoricamente instradabile in questo modo, riconoscete un modo semplice per riconoscere il traffico generato da `httptunnel`?

```
→
```

3.2 Creazione di un tunnel ICMP (ping)

Un'altra possibilità per aggirare le politiche di un firewall è usare il tool Ptunnel che permette di creare un tunnel TCP affidabile incapsulato all'interno di pacchetti di tipo ICMP echo request ed echo reply. Visto che spesso i firewall lasciano passare il traffico ICMP di tipo echo (anche se meno frequentemente di HTTP), Ptunnel può tornare utile al nostro scopo.
Anche in questo caso il contesto è quello degli esercizi precedenti:

- Alice è l'utente interno alla rete
- Frank rappresenta il border firewall. Frank usa un packet filter per implementare una politica di autorizzazione: consentire agli utenti interni di inviare e ricevere il ping. Come visto negli esercizi precedenti Frank userà le seguenti regole (ricordate di fare il flush delle precedenti configurazioni ed accertatevi che la politica di default sia DROP):

  ```
  iptables -A FORWARD -p icmp -s IP_Ali --icmp-type echo-request -j
  ACCEPT
  ```

  ```
  iptables -A FORWARD -p icmp -d IP_Ali --icmp-type echo-reply -j ACCEPT
  ```

  ```
  iptables -A FORWARD -p icmp -d IP_Ali --icmp-type echo-request -j
  ACCEPT
  ```

  ```
  iptables -A FORWARD -p icmp -s IP_Ali --icmp-type echo-reply -j ACCEPT
  ```

- Bob è l'utente esterno che collude con Alice. Bob fa partire il server SSH

  ```
  systemctl start ssh
  ```

Nella configurazione di Ptunnel ci sono tre entità da considerare, un server destinazione, un proxy ed un client:

- il server destinazione è il server (di solito esterno) che si vuole raggiungere;
- il client è un processo che rimane in ascolto su una specifica porta TCP ed incapsula il payload ed alcuni dei campi dell'header TCP dei pacchetti ricevuti in pacchetti ICMP echo request, comprese le informazioni sul server destinazione. Inoltre il client riceve in risposta dal proxy i pacchetti ICMP echo reply e ricompone il pacchetto TCP inviato dal server destinazione;
- il proxy è un processo che riceve i pacchetti di tipo ICMP echo request generati dal client, estrae le informazioni relative al server destinazione, ricrea il pacchetto TCP e lo inoltra al server destinazione. Inoltre, il proxy riceve le risposte dal server destinazione e le incapsula in pacchetti di tipo ICMP echo reply diretti al client.

Ulteriori informazioni sono disponibili al sito https://www.mit.edu/afs.new/sipb/user/golem/tmp/ptunnel-0.61.orig/web/
La sintassi di `ptunnel` è la seguente:

```
ptunnel [-p proxy_address] [-lp listening_port] [-da dest_addr] [-dp
dest_port]
```

in cui:

- `-p proxy_address`, serve a specificare l'indirizzo IP del proxy (client);
- `-lp listening_port`, serve a specificare la porta in ascolto per i pacchetti da inoltrare al server (client);
- `-da dest_addr`, serve a specificare l'indirizzo del server destinazione (client);
- `-dp dest_port`, serve a specificare la porta del server destinazione da raggiungere (client).

Useremo il comando `ptunnel` per creare sia il client che il proxy. Nel nostro caso semplificato, Bob farà partire sia il proxy che il server destinazione. La situazione è presentata in Figura 5.
Bob fa partire il server SSH ed il proxy con il seguente comando:

```
ptunnel &
```

Scrivere il comando che Alice dovrà eseguire per far partire un client `ptunnel` che si metta in ascolto sulla porta 8000, che permetta di comunicare con SSH di Bob tramite il proxy, sempre sulla macchina di Bob.

→

Scrivere il comando che Alice dovrà eseguire per collegarsi col server SSH di Bob.

→

Supponendo che si aggiunga una nuova macchina dove Carol abbia fatto partire il server Apache, cosa deve fare Alice per raggiungerlo? (Suggerimento: continuate ad usare Bob come proxy).
E se volesse raggiungere più di una destinazione alla volta?

→

Provate ad intercettare il traffico. Notate differenze tra i pacchetti ICMP generati da `ptunnel` e quelli normali? Sarebbero distinguibili con un IDS?

→

TUTORIAL: Circuit-level gateway

Ipotizzando gli stessi gruppi formati per l'esercizio 2.2, Frank rimuove le regole relative al traffico web: il traffico web sarà nel seguito riabilitato facendo uso di un circuit-level gateway.
Sfrutteremo la funzionalità "dynamic port forwarding" del client e server SSH per sperimentare le funzionalità di un circuit-level gateway.
Prima di cominciare però, Frank aggiunge l'utente `ali` e fa partire il server SSH.

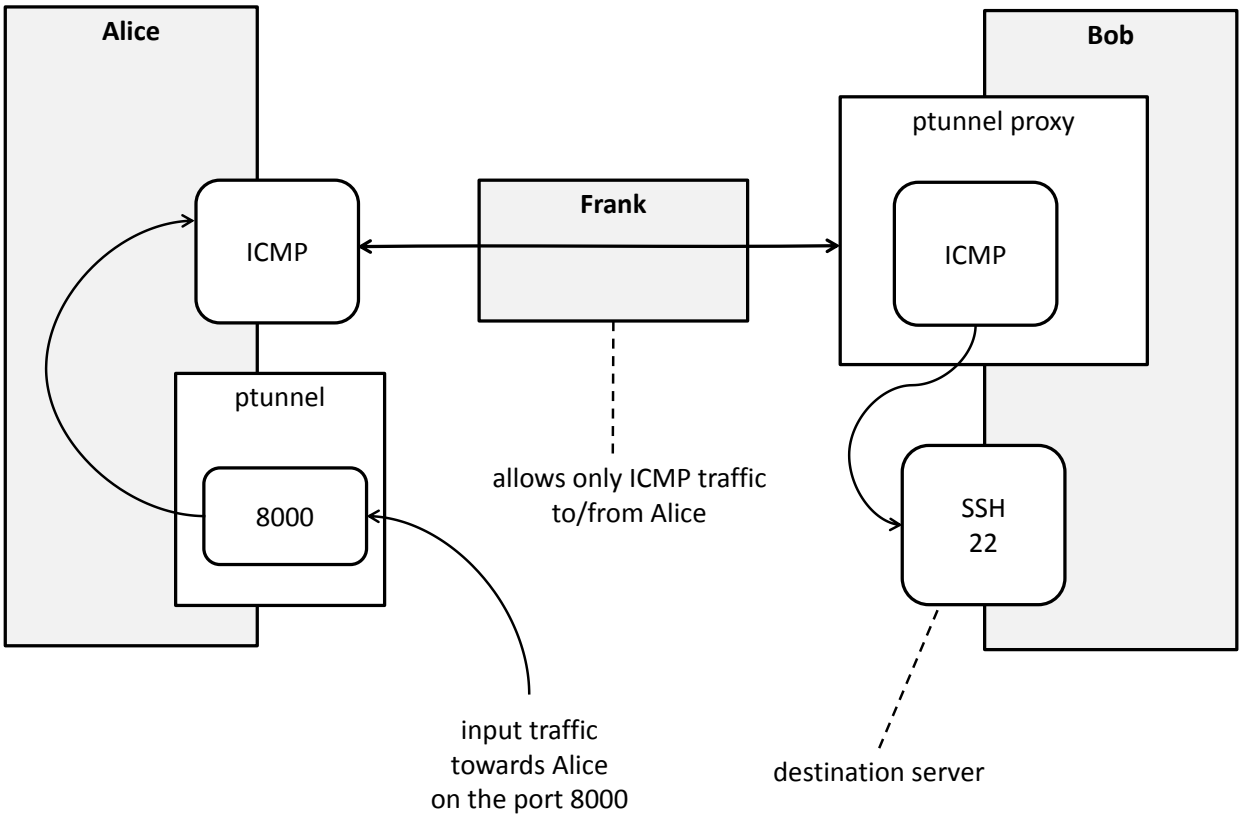

Figura 5: Lo scenario da implementare con Ptunnel.

3.3 Configurazione di SSH come circuit-level gateway

Per poter usare un circuit-level gateway Alice crea le credenziali RSA utili all'autenticazione presso il gateway con il seguente comando:

```
ssh-keygen
```

Salvate la chiave generata nel file `.ssh/id_rsa` (default) e usate una passphrase nulla. Potrete notare che `ssh-keygen` ha in effetti creato due file: `id_rsa` contiene la chiave privata e `id_rsa.pub` contiene la corrispondente chiave pubblica RSA, che sarà inviata a Frank.
Alice deve quindi inviare la sua chiave pubblica sulla macchina di Frank per potersi autenticare tramite sfida asimmetrica con la chiave appena generata. Per far questo usa il comando:

```
ssh-copy-id -i .ssh/id_rsa.pub ali@IP_Fra
```

Questo comando apre una connessione SSH con Frank

- crea la cartella `/home/ali/.ssh`, se non esiste già,
- aggiunge la chiave in `id_rsa.pub` alle chiavi associate all'utente `ali` (nel `/home/ali/.ssh/authorized_keys`),
- registra tramite `ssh-add` la nuova chiave

A questo punto configuriamo il circuit-level gateway sulla macchina di Frank che:

- effettua i seguenti cambiamenti al file `/etc/ssh/sshd_config`:
  1. sposta la porta di ascolto cambiando

```
Port 22
```
→
```
Port 1080
```

2. disabilita l'autenticazione con username e password cambiando

```
#PasswordAuthentication yes
```
→
```
PasswordAuthentication no
```

3. disabilita il port forwarding, aggiungendo (al fondo) il seguente comando

```
AllowTcpForwarding no
```

- riavvia il server SSH con la nuova configurazione (`systemctl restart ssh`);

Verificate che Alice possa collegarsi alla porta tcp/1080 sulla macchina di Frank con il seguente comando:

```
nmap -sT -Pn -n -p 1080 IP_Fra
```

Ora Alice prova a instaurare un canale SSH con il gateway:

```
ssh -p 1080 ali@IP_Fra
```

Alice, dopo aver chiuso la connessione precedente, prova a connettersi al server web di Bob attraverso il circuit-level gateway:

- Alice apre una connessione SSH con il gateway attivando il "dynamic port forwarding":

  ```
  ssh -D 1080 -p 1080 -i .ssh/id_rsa ali@IP_Fra
  ```

  l'opzione `-D` serve per abilitare il "dynamic port forwarding", in quanto alloca un socket in ascolto in locale sulla porta specificata come parametro (in questo caso *1080*). Qualora una connessione sia aperta verso questa porta, la connessione sara' inoltrata su canale sicuro `ssh` (per ulteriori approfondimenti possibile analizzare l'opzione `-D` nel manuale `ssh` usando il comando `man ssh`)

- Alice configura il browser per contattare all'indirizzo `127.0.0.1` l'interfaccia SOCKS messa a disposizione dal client `ssh` per il "dynamic port forwarding". Con Firefox, bisogna andare in `Preferenze` → `Generale` → `Impostazioni di rete` → `Impostazioni...` → `Configurazione manuale dei proxy` → `Host SOCKS`;

- prova a collegarsi al server web di Bob.

Alice non è ancora in grado di collegarsi al server web di Bob, in quanto non ancora autorizzata ad aprire connessioni web. Possibile verificare cosa succede analizzando il traffico di rete con Wireshark, in cui in particolare si vede

- Ali→Fra: SSH Encrypted request packet

- Fra→Ali: SSH Encrypted response packet (errore) ?

Quindi, dovete ancora raffinare la politica di autorizzazione del circuit-level gateway affinché permetta a Alice di aprire solo connessioni web verso Bob:

- Frank modifica `/etc/ssh/sshd_config` ed aggiunge, sempre al fondo, le seguenti linee:

  ```
  Match User ali
  AllowTcpForwarding yes
  PermitOpen IP_Bob:80
  ```

- Frank riavvia il server SSH
- Alice riavvia il client SSH

Alice prova nuovamente a collegarsi al server web e SSH di Bob ed ora può navigare! Analizzando il traffico con wireshark si nota:

- Ali→Fra: SSH Encrypted request packet
- Fra↔Bob: scambio (ad esempio con protocollo HTTP)
- Fra→Ali: SSH Encrypted response packet

Da notare che rispetto alla abilitazione del traffico web via funzionalità di packet filtering come negli esercizi 2.2 e 2.3 si ottengono diversi vantaggi rilevanti, in particolare autenticazione utente, che impedisce lo spoofing IP, e l'instaurazione di un canale protetto. Inoltre, questa configurazione, introducendo l'intermediario Frank non rientra nel modello client-server, permettendo quindi di non dipendere da una macchina meno controllabile come genericamente sono i client, e permettendo un maggiore isolamento e protezione dello stack TCP/IP della macchina di Alice.

NOTA

Prima di procedere con gli esercizi successivi, disabilitate il server SSH sulla macchina di Frank.

TUTORIAL: Application level gateway

3.4 Configurazione di mod_proxy come application-level gateway

In questo esercizio Alice dovrà usare un application level gateway invece del circuit-level gateway dell'esercizio 3.3 per accedere al server web di Bob. In particolare, useremo il server Apache e il modulo `mod_proxy` per sperimentare le funzionalità di un application level gateway web.
Come prima cosa Frank attiva e configura l'application level gateway:

1. disabilita il server Apache se attivo (`systemctl stop apache2`);
2. sostituisce il file `/etc/apache2/sites-enabled/000-default.conf` con il file `frank-apache2-config` fornito;
3. Ora sono necessari chiave e certificato del server, potete generare un certificato per il server nel modo seguente
   (a) crea una CA di prova come fatto nell'esercitazione precedente:
   ```
   /usr/lib/ssl/misc/CA.pl -newca
   ```
   (b) crea una richiesta di certificato per il server (avente il nome DNS della sua macchina come *Common Name*):
   ```
   openssl req -new -keyout server_pkey.pem -out server_creq.pem
   ```
   (c) emette il nuovo certificato:
   ```
   openssl ca -in server_creq.pem -out server_cert.pem
   ```
4. crea la cartella `/etc/apache2/ssl` e copia in essa i certificati (della CA e del server) e la chiave privata
5. carica i moduli di Apache necessari ad usare SSL/TLS

```
a2enmod ssl
```

6. carica i moduli di Apache necessari al funzionamento come proxy

```
a2enmod proxy
```

```
a2enmod proxy_http
```

7. riavvia Apache (`systemctl restart apache2`)

Bob configura il proprio server web:

- sostituisce il file `/etc/apache2/sites-enabled/000-default.conf` con il file `bob-apache2-config` fornito;
- riavvia il server Apache (`systemctl restart apache2`)

Ora Alice prova a connettersi al server web di Bob, utilizzando l'application level gateway di Frank:

- configura il browser testuale `lynx` per usare Frank come proxy HTTPS, impostando la variabile d'ambiente `http_proxy`:

```
export http_proxy='https://IP_Fra:443/'
```

- si collega al web server di Bob (potete ignorare il problema della CA non riconosciuta premendo 'y'):

```
lynx http://IP_Bob
```

Ora modificate la politica di autorizzazione dell'application gateway di Frank:

- editate il file di configurazione di Apache e decommentate la sezione "Proxy Filter"
- riavviate Apache

Connettendosi dalla macchina di Alice, ad esempio con

```
lynx http://IP_Bob/icons/
```

la regola blocca l'accesso alle risorse del tipo *.gif. Quindi, provando a scaricare una gif segnala che Alice non sia autorizzata.
Per concludere, possiamo proteggere i servizi offerti dall'application gateway mediante autenticazione del client:

- editate nuovamente il file di configurazione di Apache e decommentate la sezione "Proxy Auth"
- dovete inoltre creare un file con username e password per Apache:

```
htpasswd -c /etc/apache2/htpasswd ali
```

- riavviate Apache

Come fatto sopra, Alice si connette al server web di Bob con il browser `lynx`:
Ora, `lynx` chiederà la password di autenticazione a Alice (che dovrà farsela fornire da Frank)
Con questa soluzione, si ottengono gli stessi vantaggi di quella basata su circuit-level gateway, ed in aggiunta la possibilita di instaurare dei filtri applicativi, con tutti i vantaggi che ne conseguono (es. filtraggio di contenuti e vulnerabilita specifiche di determinate applicazioni, minore suscettibilita ad occultamento di dati all'interno di traffico legittimo).