



Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in **Ingegneria Informatica**

NETWORK SECURITY

Sviluppo di un firewall in ambiente Linux con IPTables

Prof. Romano Simon Pietro

Anno Accademico 2020/2021

Studenti:

Coppola Vincenzo Matr. M63/1000

Della Torca Salvatore Matr. M63/1011

Indice

1	Firewall	1
2	IPTables	5
2.1	Guida all'uso di IPTables	6
3	Implementazione del firewall	10
3.1	Firewall di base	10
3.2	Regole specifiche per i servizi	13
3.2.1	SSH	13
3.2.2	Web Server	14
3.2.3	MySQL	15
3.2.4	Mail	15
4	Verifica del funzionamento del firewall	18

Capitolo 1

Firewall

Un **firewall** è un componente hardware o software di difesa perimetrale di una rete che può anche svolgere funzioni di collegamento tra due o più segmenti di rete, fornendo quindi protezione della rete stessa. Tutto il traffico in ingresso o in uscita deve passare attraverso il firewall, e il passaggio è consentito solamente se il traffico è autorizzato.

Le *security policies* permettono di specificare i tipi di traffico autorizzati a bypassare il firewall, e la loro realizzazione parte da un'accurata valutazione dei rischi per la sicurezza informatica, che permette di specificare:

- intervalli di indirizzi, utili ad esempio per specificare quali sono gli indirizzi IP che possono accedere al sistema;
- protocolli e applicazioni;
- user identity, per permettere l'accesso solamente ad utenti autenticati e autorizzati;
- tipi di contenuto.

Esistono due criteri per l'applicazione delle policy:

- **default-deny**: viene permesso solo ciò che viene dichiarato esplicitamente e il resto viene vietato;
- **default-allow**: viene vietato solo ciò che viene dichiarato esplicitamente e il resto viene permesso.

Tipicamente si usa la politica *default-deny* in quanto si garantiscono una maggiore sicurezza e precisione nella definizione delle regole. L'analisi dei pacchetti, in base ai criteri di sicurezza, si traduce in una delle seguenti azioni:

- **allow**: il firewall lascia passare il pacchetto;

- **deny**: il firewall blocca il pacchetto;
- **drop**: il firewall blocca il pacchetto e lo scarta senza inviare alcuna segnalazione al mittente.

Per quanto un firewall possa essere vantaggioso, presenta anche dei limiti legati al fatto che la sua efficacia è strettamente dipendente dalle regole con cui è stato configurato, e può presentare delle vulnerabilità come l'**HTTP tunneling**, che consente di bypassare le restrizioni Internet usando richieste HTTP, oppure **dll injection** che sovrascrive lo *shell code* all'interno di librerie di sistema utilizzate in programmi considerati sicuri e che per questo non vengono controllati.

È possibile distinguere diverse tipologie di firewall:

Packet Filtering Firewall

Sono responsabili di verificare se ogni pacchetto IP in ingresso o in uscita dalla rete matcha con una delle regole specificate, che tipicamente sono basate su valori dell'header IP o TCP (source address, destination address, campi del protocollo IP). In base al match con le regole, il pacchetto viene inoltrato o scartato. I firewall di questo tipo hanno il vantaggio di essere molto semplici da realizzare e sono tipicamente trasparenti all'utente finale, ma non possono prevenire attacchi che sfruttano vulnerabilità specifiche delle applicazioni, sono vulnerabili ad attacchi su bugs del protocollo TCP/IP e sono limitati alla funzionalità di logging.

Statefull Inspection Firewall

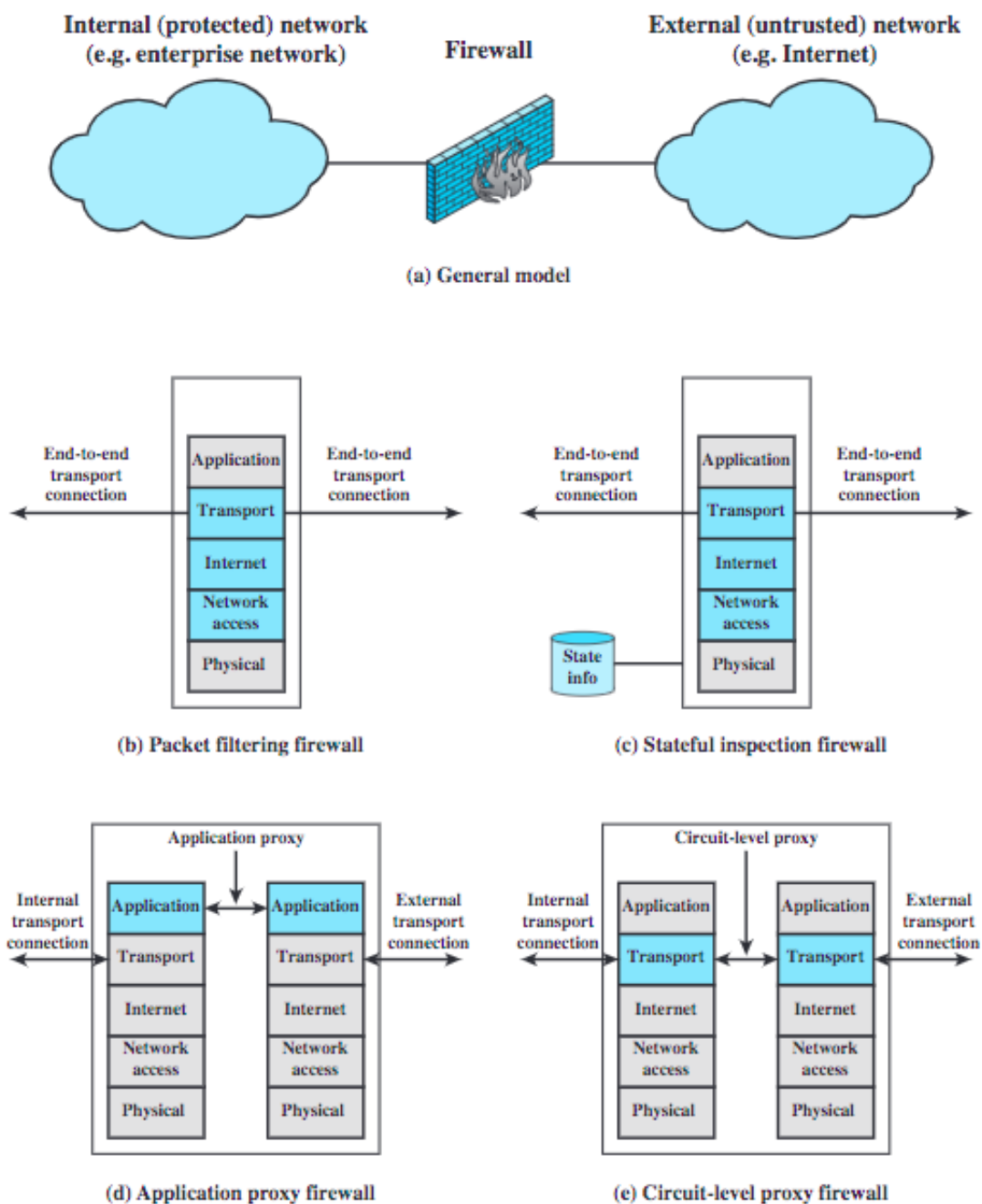
È un firewall che rafforza le regole per il traffico TCP (livello trasporto) creando una directory delle connessioni TCP in uscita che contiene una voce per ogni connessione stabilita; quindi il passaggio è consentito solamente ai pacchetti che corrispondono al profilo di una delle voci.

Application Level Gateway

Sono comunemente chiamati **proxy** e sono firewall che agiscono da filtro a livello applicazione: l'utente contatta il gateway utilizzando un'applicazione TCP/IP, viene autenticato, e il gateway contatta l'applicazione sull'host remoto e rilancia i segmenti TCP tra il server e l'utente.

Circuit Level Gateway

Si tratta di un firewall che si occupa di settare due connessioni TCP, una con un utente TCP su un host interno e una su un host esterno, e ritrasmette i segmenti TCP da una connessione all'altra senza esaminare i contenuti. L'obiettivo di questo firewall è determinare quali connessioni sono consentite, ed è utilizzato tipicamente quando gli utenti interni sono fidati.



È possibile fare anche un'altra distinzione tra **host-based firewall** e **network-based firewall**:

- i primi, anche detti *personal firewall* o *firewall software*, sono software che controllano il traffico in uscita da un singolo computer e bloccano le applicazioni installate sulla macchina a cui non è concessa la connessione con l'esterno;
- i secondi, detti anche *firewall hardware*, sono componenti hardware stand-alone che vengono inseriti sulla frontiera di una rete in modo da filtrare tutto il traffico che questa scambia con l'esterno, ed è per questo che spesso vengono anche definiti come **firewall perimetrali**.

Capitolo 2

IPTables

Settare un buon firewall è quindi un passo fondamentale per rendere un sistema sicuro. La maggior parte delle distribuzioni Linux sono dotate di diversi strumenti che possono essere utilizzati per la creazione dei firewall, tra cui **IPTables**: si tratta di un front-end per gli *hooks* del framework **netfilter** a livello di kernel, che possono manipolare lo stack di rete di Linux, e opera facendo corrispondere ogni pacchetto che attraversa l'interfaccia di rete con un insieme di regole che definiscono le caratteristiche un pacchetto dovrebbe avere per corrispondere alla regola e l'operazione da eseguire.

Le regole sono organizzate in gruppi, chiamati **catene (chains)**, e vengono controllate in maniera sequenziale fin quando il pacchetto corrisponde ad una delle regole; a quel punto le regole successive non vengono controllate.

L'azione che viene eseguita quando un pacchetto matcha con una regola prende il nome di **target**, e può essere accettare o eliminare il pacchetto, o anche spostare il pacchetto in un'altra *chain*.

Un utente può creare *catene* in base alle proprie necessità, ma esistono tre catene che sono definite di default:

- **INPUT**: gestisce tutti i pacchetti indirizzati al server;
- **OUTPUT**: contiene le regole per il traffico creato dal server;
- **FORWARD**: è usata per gestire il traffico destinato ad altri server che non sono creati sul proprio server. È quindi un modo per configurare il proprio server per instradare le richieste ad altre macchine.

Ogni catena può contenere zero o più regole ed ha una **politica** predefinita che determina cosa succede quando un pacchetto non matcha con nessuna delle regole della catena.

IPTables è anche in grado di tracciare le connessioni attraverso l'uso di un modulo che può essere caricato tramite le regole, ed è quindi possibile creare regole che definiscono cosa

succede ad un pacchetto in base alla sua relazione con i pacchetti precedenti (*connection tracking*).

Il firewall *netfilter* incluso nel kernel Linux mantiene separati il traffico IPv4 e IPv6, e allo stesso modo anche gli strumenti usati per manipolare le tabelle che contengono le regole del firewall sono distinti. *IPTables*, di default, permette di manipolare la tabella contenente le regole che gestiscono il traffico IPv4; per manipolare la tabella delle regole IPv6 è necessario utilizzare il comando **ip6tables**.

L'ordine in cui le regole vengono inserite all'interno di una catena è fondamentale in quanto le regole vengono verificate in maniera sequenziale dalla prima all'ultima. Per questo motivo, le regole all'inizio di una catena dovrebbero avere un livello più alto di specificità rispetto alle regole più in basso.

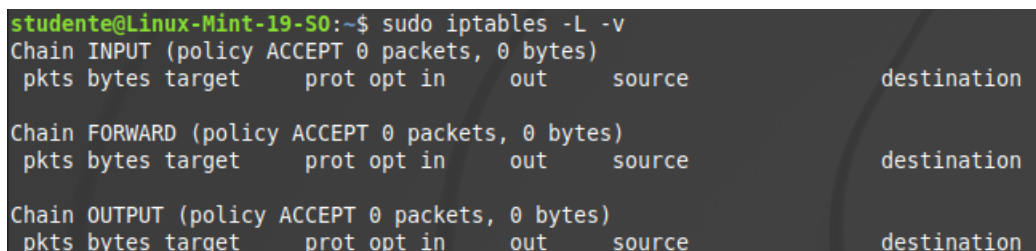
2.1 Guida all'uso di IPTables

Per poter utilizzare IPTables è necessaria la sua installazione mediante il comando

```
sudo apt-get install iptables
```

Per visualizzare le regole che di default sono presenti all'interno delle catene bisogna utilizzare il comando

```
sudo iptables -L -v
```



```
studente@Linux-Mint-19-S0:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source         destination
```

Le regole che vengono inserite devono essere rese persistenti, e per fare ciò bisogna usare il package **iptables-persistent**, che può essere installato col comando

```
sudo apt-get install iptables-persistent
```

Quando poi il firewall viene aggiornato con nuove regole, per rendere queste ultime persistenti è necessario eseguire il comando

```
sudo netfilter-persistent save
```


L'ordine in cui le regole vengono inserite è fondamentale per la protezione e, a tal proposito, quando si vuole inserire una nuova regola in una catena, con il flag **-A** la si inserisce in coda, con il flag **-I** invece si specifica l'esatta posizione in cui la si vuole inserire.

Per rimuovere una regola si possono invece utilizzare due tecniche differenti:

- cancellazione della regola in base alle specifiche, attraverso il flag **-D**;
- cancellazione della regola in base alla catena in cui si trova o al numero, attraverso l'opzione **-line-numbers**.

Come detto in precedenza, la maggiore sicurezza è garantita quando si utilizza la politica **default-deny**, secondo la quale tutto il traffico viene vietato ad eccezione di quello permesso attraverso le regole definite nella catena; come si può osservare però dalla schermata precedente, la politica di default è impostata ad **ACCEPT**, e per modificarla è necessario eseguire i comandi

```
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
```

In alternativa, si potrebbe implementare una nuova regola di drop da inserire alla fine della lista delle regole, e la distinzione tra i due metodi sta in ciò che accade quando le regole vengono cancellate: nel primo caso, in cui si imposta la politica di default su *drop*, se le regole vengono rimosse allora i servizi diventano non accessibili da remoto ma continua ad esserci protezione, mentre nel caso in cui si usa la regola di *drop*, se si cancellano le regole i servizi saranno accessibili ma non protetti. Una volta impostata la politica default-deny, è possibile procedere all'inserimento delle regole all'interno delle catene.

Lavoriamo sulla catena INPUT, che ci permette di filtrare il traffico in entrata. Come primo esempio, realizziamo una regola che accetta esplicitamente la connessione SSH e andiamo ad analizzarla nel dettaglio:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED
-j ACCEPT
```

- **-A INPUT**: questa porzione del comando specifica che la nuova regola deve essere aggiunta alla fine della catena INPUT;
- **-m conntrack**: *conntrack* è un modulo di IPTables che fornisce l'accesso a comandi che possono essere usati per prendere decisioni basate sulla relazione del pacchetto con le connessioni precedenti;

- **-ctstate ESTABLISHED,RELATED:** *ctstate* è uno dei comandi del modulo *conntrack* e ci permette di abbinare i pacchetti in base a come sono collegati ai pacchetti passati; i due valori *ESTABLISHED* e *RELATED* servono rispettivamente per consentire il passaggio di pacchetti che sono parte di una connessione esistente e per consentire il passaggio di pacchetti che sono associati ad una connessione stabilita;
- **-j ACCEPT:** l'ultima porzione serve per specificare il target della regola; in questo caso si dice ad IPTables di accettare i pacchetti che corrispondono ai criteri precedenti.

Una volta compreso come sono strutturate le regole, per poter realizzare un firewall che sia utile bisogna prendere delle decisioni.

Gestione del rifiuto di un pacchetto

In primis, bisogna capire come gestire il rifiuto di un pacchetto: di default IPTables non invia alcuna risposta al client che cerca di connettersi, per cui il client non riceverà alcuna conferma della ricezione. Questo implica che: nel caso di connessioni TCP, la connessione andrà in stallo fino al raggiungimento del timeout; nel caso di connessioni UDP, essendo questo un protocollo *connection-less*, la mancanza di risposta per i client è ancora più ambigua perchè potrebbe portare a pensare che il pacchetto sia stato accettato.

L'alternativa a ciò può essere quella di rifiutare esplicitamente i pacchetti che non sono consentiti, mediante l'uso di **ICMP** che permette l'invio di messaggi di stato e di errore tra gli host. Per fare ciò è necessario specificare come target della regola **REJECT**, che permette di restituire al mittente un pacchetto ICMP per informarlo che il suo traffico è stato ricevuto ma non accettato.

Tuttavia, per gli utenti malintenzionati, questo significa che possono completare lo scanning e il mapping delle porte aperte, chiuse e filtrate in un tempo più breve.

Accettare o rifiutare i pacchetti ICMP

Un'altra domanda che ci si pone spesso quando si realizza un firewall è se accettare o meno i pacchetti ICMP, che sono organizzati per **tipo** e **codice**: il *tipo* specifica il significato generale del messaggio, il codice fornisce ulteriori informazioni sul tipo.

Esempio: ICMP Type 3 Code 3 significa che la porta di destinazione non era disponibile. Alcuni tipi, come *type 1*, *type 2*, *type 4 code 0*, *type 7*, *type 15* e superiori, possono essere sempre bloccati in quanto sono deprecati oppure riservati per uso futuro.

Altri *tipi* invece possono essere bloccati a seconda della configurazione di rete; per esempio, *ICMP type 5* permette di segnalare al client la presenza di un percorso migliore per l'instradamento del pacchetto, e questo può essere utile quando la rete è fidata e si vogliono individuare inefficienze nelle tabelle di routing. Se la rete però non è fidata, un attaccante potrebbe inviare dei pacchetti *ICMP type 5* per manipolare le tabelle di routing sugli host.

Ci sono invece *tipi* che di solito sono sicuri e che pertanto possono essere abilitati, come *type 8 (echo request)* e *type 13 (timestamp request)*, anche se queste potrebbero essere usate in alcune tecniche di *fingerprinting*.

Limitare l'uso delle risorse

Per alcuni servizi si potrebbe voler permettere l'accesso a condizione che il client non ne faccia abuso, e per questo si può limitare l'uso delle risorse in due diversi modi:

- limitazione delle connessioni, che può essere implementata con l'estensione *connlimit* che permette di implementare limitazioni per indirizzo, per rete o su base globale;
- limitazione della velocità, che permette di costruire regole che governano la frequenza con cui il traffico sarà accettato, mediante l'utilizzo di estensioni come *limit* e *recent*: la prima applica la regola fino al raggiungimento del limite, dopo di che altri pacchetti verranno eliminati, mentre la seconda aggiunge dinamicamente gli indirizzi IP dei client ad una lista ed ha la capacità di specificare un *hit count* e un *time range* che può essere reimpostato se si ha del traffico aggiuntivo, forzando il client a fermare tutto il traffico se è stato limitato.

Gestione monolitica o chain-based

Nello sviluppo dei firewall è possibile adottare due diverse tecniche:

- cambiare la politica predefinita per le catene e aggiungere ad esse nuove regole, e questo è ciò che viene fatto quando si vogliono realizzare firewall abbastanza semplici
- quando invece si vogliono realizzare firewall più complessi, è utile creare catene aggiuntive alle catene di default.

Le nuove catene create dall'utente sono intrinsecamente legate alla loro catena chiamante e non hanno una politica di default, nel senso che se un pacchetto non matcha con nessuna regola presente nella nuova catena allora passerà alla catena chiamante e continuerà la valutazione.

Prese quindi le giuste decisioni, il passo successivo è l'implementazione del firewall.

Capitolo 3

Implementazione del firewall

3.1 Firewall di base

L'implementazione del firewall viene fatta direttamente da linea di comando mediante l'utilizzo dei comandi di IPTables. Come visto in precedenza, dopo l'installazione di IPTables verranno create le tre catene di default (INPUT, FORWARD, OUTPUT) e verrà impostata la politica di default su ACCEPT per tutte le catene.

Creazione delle chains specifiche per i protocolli

A questo punto si può procedere con la creazione vera e propria, e per avere una migliore leggibilità e gestione delle regole, è possibile creare le catene specifiche per i vari protocolli. Andremo a creare una catena per UDP, una per TCP e una per ICMP:

```
sudo iptables -N UDP
sudo iptables -N TCP
sudo iptables -N ICMP
```

```
studente@Linux-Mint-19-S0:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
Chain ICMP (0 references)
 pkts bytes target    prot opt in     out     source                   destination
Chain TCP (0 references)
 pkts bytes target    prot opt in     out     source                   destination
Chain UDP (0 references)
 pkts bytes target    prot opt in     out     source                   destination
```

Possiamo ora aggiungere l'eccezione per il traffico SSH, che usa TCP, e quindi possiamo aggiungere nella chain TCP una regola per accettare il traffico TCP destinato alla porta 22:

```
sudo iptables -A TCP -p tcp -dport 22 -j ACCEPT
```

- il flag **-p** ci permette di specificare il protocollo per il quale vogliamo creare la regola;
- il flag **-dport** serve invece per specificare il porto di destinazione.

Creazione delle regole **ACCEPT** e **DENY**

Nella catena di INPUT è necessario aggiungere le regole di scopo generale che servono per impostare la linea di base per il firewall, accettando il traffico a basso rischio e rifiutando il traffico che non è valido. Quindi per prima cosa creeremo una regola per accettare il traffico che fa parte di una connessione stabilita o che è collegato ad una connessione stabilita, così come spiegato in precedenza:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED  
-j ACCEPT
```

TCP è un protocollo *connection-based*, per cui si stabilisce una connessione prima di inviare i pacchetti; per UDP e altri protocolli di tipo *connection-less* invece i pacchetti vengono accettati solamente se la connessione che è stata avviata è associata ad una connessione già esistente.

Altro traffico che vorremmo permettere è quello proveniente dall'interfaccia di loopback locale, ossia il traffico che viene generato dal server e che è destinato al server stesso e che viene usato dai servizi sull'host per comunicare tra loro.

```
sudo iptables -A INPUT -i lo -j ACCEPT  
sudo iptables -A OUTPUT -o lo -j ACCEPT
```

Infine, si vogliono negare tutti i pacchetti non validi, come quelli che si riferiscono a connessioni non esistenti o quelli destinati ad indirizzi, porte o interfacce che non esistono.

```
sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
```

Creazione delle regole di salto tra le catene

Finora ci siamo limitati a creare regole generali nella catena di input e una regola nella catena TCP; tuttavia, per far sì che il traffico possa raggiungere la catena da noi inserita è necessario reindirizzarlo attraverso un'apposita regola, che opererà sul *tipo* di protocollo per identificare la giusta catena.

Inoltre, poichè stiamo trattando i pacchetti TCP, bisogna aggiungere il requisito che il pacchetto in arrivo sia un pacchetto di **SYN**, che è l'unico valido per iniziare una connessione TCP.

```
sudo iptables -A INPUT -p udp -m conntrack --cstate NEW -j UDP
sudo iptables -A INPUT -p tcp --syn -m conntrack --cstate NEW -j TCP
sudo iptables -A INPUT -p icmp -m conntrack --cstate NEW -j ICMP
```

Rifiuto del traffico rimanente

Se un pacchetto che è stato passato ad una catena specifica del protocollo non corrisponde a nessuna delle regole al suo interno, il controllo verrà passato alla catena INPUT e, per tutto ciò che è stato detto in precedenza, il pacchetto dovrà essere negato. Per fare ciò si usa il target **REJECT**, che invia un messaggio di risposta al client che dipende dal protocollo utilizzato dal client:

- il tentativo di raggiungere una porta UDP chiusa provocherà un messaggio ICMP *port unreachable*

```
sudo iptables -A INPUT -p udp -j REJECT
--reject-with-icmp-port-unreachable
```

- il tentativo di stabilire una connessione TCP su una porta chiusa provocherà un messaggio di TCP RST

```
sudo iptables -A INPUT -p tcp -j REJECT --reject-with-tcp-reset
```

- per tutti gli altri pacchetti si può restituire un messaggio ICMP *protocol unreachable* per indicare che il server non risponde a pacchetti di quel tipo

```
sudo iptables -A INPUT -j REJECT
--reject-with-icmp-proto-unreachable
```

Queste tre regole dovrebbero essere in grado di gestire tutto il traffico rimanente nella catena di input, ma come già detto è importante settare la politica di default a DROP

sudo iptables -P INPUT DROP

A questo punto si ha a disposizione una configurazione iniziale del firewall, alla quale si possono aggiungere le singole regole per permettere o negare l'accesso ai servizi che si vogliono rendere disponibili.

3.2 Regole specifiche per i servizi

Permettere alla rete interna di accedere a quella esterna

Supponendo che *eth0* sia la rete esterna e *eth1* sia la rete interna, il comando seguente permetterà alla rete interna di accedere a quella esterna.

```
sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Bloccare un indirizzo IP

È possibile, laddove non si voglia permettere l'accesso ad un determinato indirizzo IP bloccare le connessioni di rete che vengono originate da quell'IP.

```
sudo iptables -A INPUT -s "IP address" DROP
```

Se invece di bloccare la connessione la si vuole rifiutare, bisogna sostituire *DROP* con **REJECT**; facendo in questo modo il sistema risponderà alla richiesta di connessione con un messaggio di errore "*connessione rifiutata*".

Bloccare le connessioni a un'interfaccia di rete

Combinando i due comandi precedenti è possibile bloccare le connessioni da uno specifico IP verso una specifica interfaccia di rete.

```
sudo iptables -A INPUT -i eth0 -s "IP address" DROP
```

3.2.1 SSH

Supponiamo di avere un server senza una console locale, per cui si vogliono permettere le connessioni SSH in entrata, sul porto 22, in modo da potersi connettere in remoto. Per permettere tutte le connessioni SSH in ingresso bisogna utilizzare il comando

```
sudo iptables -A INPUT -p tcp -dport 22 -m conntrack -cstate NEW,  
ESTABLISHED -j ACCEPT
```

Nel caso in cui la policy di default della catena OUTPUT non è ACCEPT, bisogna eseguire anche un altro comando che permette il traffico in uscita delle connessioni SSH stabilite

```
sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --cstate  
ESTABLISHED -j ACCEPT
```

Per permettere connessioni SSH in entrata da uno specifico IP address o da una subnet, bisogna utilizzare il comando

```
sudo iptables -A INPUT -p tcp -s "IP/subnet" --dport 22 -m conntrack  
--cstate NEW, ESTABLISHED -j ACCEPT
```

In maniera analoga al caso precedente, se la policy di default della catena OUTPUT non è ACCEPT bisogna eseguire anche il comando di output

```
sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --cstate  
ESTABLISHED -j ACCEPT
```

3.2.2 Web Server

I Web Server tipicamente sono in ascolto sul porto 80 per le connessioni **HTTP** e sul porto 443 per le connessioni **HTTPS**.

Per permettere tutte le connessioni HTTP in ingresso si usa il comando

```
sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --cstate NEW,  
ESTABLISHED -j ACCEPT
```

Analogamente al caso di SSH, se la policy di default di OUTPUT non è ACCEPT, bisogna inserire anche la regola che permette il traffico in uscita delle connessioni HTTP

```
sudo iptables -A OUTPUT -p tcp --sport 80 -m conntrack --cstate  
ESTABLISHED -j ACCEPT
```

Se invece di connessioni HTTP si hanno connessioni HTTPS, bisogna utilizzare le regole viste per HTTP modificando però il porto, sostituendo 80 con 443.

Nel caso in cui invece si vuole permettere sia il traffico HTTP che il traffico HTTPS, esiste il modulo **multiport** che permette di creare un'unica regola per entrambi i porti

```
sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack  
--cstate NEW, ESTABLISHED -j ACCEPT
```

```
sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack  
--cstate ESTABLISHED -j ACCEPT
```


3.2.3 MySQL

Se un database MySQL deve essere usato da un client su un server remoto, bisogna garantire che il traffico sia consentito. MySQL ascolta le connessioni dei client sul porto 3306, quindi per permettere le connessioni MySQL da specifici indirizzi IP o subnet è necessario il comando

```
sudo iptables -A INPUT -p tcp -s "IP/Subet" -dport 3306 -m conntrack  
-ctstate NEW, ESTABLISHED -j ACCEPT  
sudo iptables -A OUTPUT -p tcp -sport 3306 -m conntrack -ctstate  
ESTABLISHED -j ACCEPT
```

Come al solito, il secondo comando è necessario solo se la policy di OUTPUT non è ACCEPT.

È possibile permettere le connessioni MySQL anche ad una specifica interfaccia di rete, specificando nel comando, prima del protocollo, il flag **-i eth0**.

3.2.4 Mail

I mail server ascoltano su porte diverse a seconda dei protocolli che vengono utilizzati per la consegna della posta.

Quando si parla di mail bisogna conoscere tre protocolli:

- **SMTP (Simple Mail Transfer Protocol)**: è il protocollo che viene utilizzato per inviare il messaggio al mail server, che poi lo inoltrerà al client destinatario. Utilizza a livello trasporto il protocollo **TCP**: il client apre una sessione TCP verso il server, sul porto 25. Combinando SMTP con **TLS** si genera **SMTPS**, grazie al quale è possibile rendere sicura una connessione.
- **POP (Post Office Protocol)**: è un protocollo di livello applicativo di tipo client-server che ha il compito di permettere, mediante un processo di autenticazione, l'accesso da parte di un client ad un account di posta elettronica presente su un server. Il porto di default sul quale è in ascolto è il 110 e per leggere i messaggi di posta è necessario scaricarli prima sul proprio PC.
- **IMAP (Internet Message Access Protocol)**: è il protocollo di comunicazione per la ricezione delle mail da parte del client. Opera di default sul porto 143, mentre se si utilizza una connessione sicura tramite SSL, il porto è il 993. Questo protocollo permette ad un client di accedere, leggere e cancellare le mail da un server, e a differenza di POP, non è necessario scaricare i messaggi per poterli leggere. Inoltre,

a differenza di POP, permette anche procedure complesse di sincronizzazione che consentono l'accesso alla posta sia online che offline, consentono a più utenti di utilizzare la stessa casella di posta, permette l'accesso a molteplici caselle di posta sul server e fornisce supporto all'accesso a singole parti MIME di un messaggio. Dati i molteplici vantaggi di IMAP, il protocollo POP è raramente utilizzato.

A questo punto si possono aggiungere le regole necessarie a seconda del caso specifica. Se il server non dovrebbe inviare posta in uscita, conviene bloccare la posta SMTP sul porto 25.

```
sudo iptables -A OUTPUT -p tcp --dport 25 -j REJECT
```

Se invece si vuole consentire al server di rispondere alle connessioni SMTP sul porto 25 sono necessari due comandi

```
sudo iptables -A INPUT -p tcp --dport 25 -m conntrack --ctstate NEW,  
ESTABLISHED -j ACCEPT  
sudo iptables -A OUTPUT -p tcp --sport 25 -m conntrack --ctstate  
ESTABLISHED -j ACCEPT
```

Per permettere al server di rispondere alle connessioni IMAP sul porto 143, è necessario eseguire due comandi

```
sudo iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,  
ESTABLISHED -j ACCEPT  
sudo iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate  
ESTABLISHED -j ACCEPT
```

Nel caso in cui si trattasse di connessioni IMAPS basta sostituire al porto 143 il porto 993.

Quando invece di IMAP si utilizza POP3, per permettere al server di rispondere alle connessioni POP3 sul porto 110, bisogna eseguire due comandi

```
sudo iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,  
ESTABLISHED -j ACCEPT  
sudo iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate  
ESTABLISHED -j ACCEPT
```

Al termine dell'inserimento delle regole, digitando il comando

sudo iptables -L

è possibile visualizzare tutte le regole che sono state inserite.

```

studente@Linux-Mint-19-S0:/$ sudo iptables -L -v --line-number
Chain INPUT (policy DROP 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source               destination
1    8929 4414K ACCEPT    all  --  any    any    anywhere             anywhere
2    830 54523 ACCEPT    all  --  lo     any    anywhere             anywhere
3     61 2440 DROP      all  --  any    any    anywhere             anywhere
4     2  146 UDP       udp  --  any    any    anywhere             anywhere
5     0   0 TCP       tcp  --  any    any    anywhere             anywhere
6     0   0 ICMP      icmp  --  any    any    anywhere             anywhere
7     2  146 REJECT    udp  --  any    any    anywhere             anywhere
8     0   0 REJECT    tcp  --  any    any    anywhere             anywhere
9     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
10    0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
11    0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
12    0   0 ACCEPT    tcp  --  any    any    203.0.113.0/24       anywhere
13    0   0 ACCEPT    tcp  --  any    any    192.0.2.0/24         anywhere
14    0   0 ACCEPT    tcp  --  any    any    198.51.100.0/24      anywhere
15    0   0 REJECT    all  --  any    any    anywhere             anywhere
                                     reject-with icmp-port-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source               destination
1     0   0 ACCEPT    all  --  eth1   eth0    anywhere             anywhere

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target    prot opt in     out     source               destination
1    408 37190 ACCEPT    all  --  any    lo     anywhere             anywhere
2    1975 279K ACCEPT    all  --  any    any    anywhere             anywhere
3     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
4     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
5     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
6     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
                                     tcp spt:ssh ctstate ESTABLISHED
                                     tcp spt:ssh ctstate NEW,ESTABLISHED
                                     tcp spt:http ctstate ESTABLISHED
                                     tcp spt:https ctstate ESTABLISHED
                                     tcp spt:mysql ctstate ESTABLISHED

Chain ICMP (1 references)
num  pkts bytes target    prot opt in     out     source               destination

Chain TCP (1 references)
num  pkts bytes target    prot opt in     out     source               destination
1     0   0 ACCEPT    tcp  --  any    any    anywhere             anywhere
                                     tcp dpt:ssh

Chain UDP (1 references)
num  pkts bytes target    prot opt in     out     source               destination

```

Le regole che costituiscono il firewall possono essere anche visualizzate e modificate direttamente dal *file delle regole*, che si trova nella cartella `/etc/iptables` col nome di **rules.v4**.

```

# Generated by iptables-save v1.6.1 on Tue Sep  7 16:47:49 2021
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:ICMP - [0:0]
:TCP - [0:0]
:UDP - [0:0]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate INVALID -j DROP
-A INPUT -p udp -m conntrack --ctstate NEW -j UDP
-A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -m conntrack --ctstate NEW -j TCP
-A INPUT -p icmp -m conntrack --ctstate NEW -j ICMP
-A INPUT -p udp -j REJECT --reject-with icmp-port-unreachable
-A INPUT -p tcp -j REJECT --reject-with tcp-reset
-A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 203.0.113.0/24 -p tcp -m tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 192.0.2.0/24 -p tcp -m tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 198.51.100.0/24 -p tcp -m tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-proto-unreachable
-A FORWARD -i eth1 -o eth0 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT
-A TCP -p tcp -m tcp --dport 22 -j ACCEPT
COMMIT
# Completed on Tue Sep  7 16:47:49 2021

```

Capitolo 4

Verifica del funzionamento del firewall

Una volta che si è sviluppato un firewall sufficientemente sicuro, il passo successivo è quello di testare le regole del firewall.

Esistono diversi strumenti che possono essere utilizzati a tal proposito:

- **Packet Analyzers:** possono essere usati per guardare tutto il traffico di rete che passa su un'interfaccia. Strumenti che permettono di fare ciò sono **tcpdump** e **Wireshark**.
- **Port Scanners:** possono essere utilizzati per creare e inviare vari tipi di pacchetti a host remoti, al fine di scoprire il tipo di traffico che il server accetta. Spesso questo strumento viene usato dagli attaccanti per scoprire servizi vulnerabili da sfruttare per attaccare il sistema. Uno strumento per fare ciò è **nmap**.

Per scansionare il sistema target, **nmap** mette a disposizione diversi metodi; per iniziare, un buon metodo, è il **TCP SYN scan**, che consiste nell'inviare il segmento di SYN senza però inviare l'ACK quando il server risponde, in modo da non chiudere il *3-way-handshaking*. Così facendo, non verrà mai negoziata una connessione TCP completa e l'attaccante è difficilmente tracciabile.

Prima di iniziare la scansione, bisogna impostare **tcpdump** per catturare il traffico generato, così da analizzare i pacchetti inviati e ricevuti; dopo aver creato una directory in cui salvare le informazioni, possiamo avviare l'acquisizione col comando

```
sudo tcpdump host "ip_target" -w "file_name"
```

Per avviare tcpdump in background bisogna digitare il comando **bg**; a questo punto possiamo eseguire la scansione, e per farlo sono necessari alcuni flag:

- **-sS:** avvia la SYN scan;

- **-Pn**: segnala ad nmap di saltare la fase di scoperta dell'host;
- **-p-**: di default nmap proverà solo le 1000 porte più comuni, mentre con questo flag controllerà ogni porta disponibile;
- **-T4**: accelera il test;
- **-vv**: aumenta la verbosità dell'output;
- **-reason**: segnala ad nmap di fornire la ragione per cui lo stato di una porta è stato riportato in un certo modo;
- **-oN**: scrive i risultati in un file che possiamo usare in fasi successive di analisi.

```
sudo nmap -sS -Pn -p- -T4 -vv -reason -oN "file_name" "target_ip_addr"
```

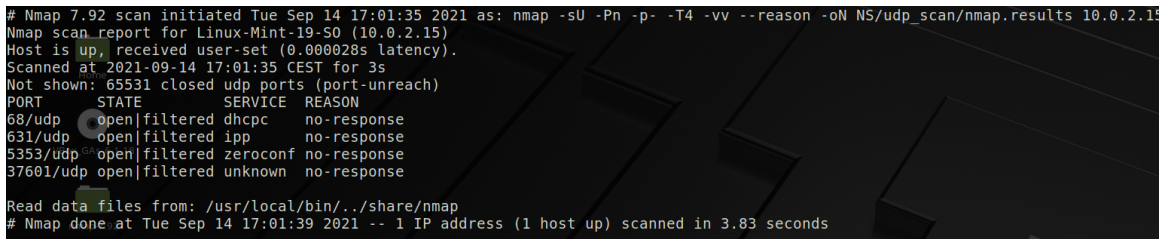
Quando termina la scansione, col comando **fg** si riporta in primo piano *tcpdump* e digitando **ctrl+c** lo si interrompe.

A questo punto si hanno a disposizione due file: quello generato da *tcpdump* e quello generato da nmap, chiamato *nmap.results*. Col comando **less nmap.results** si visualizza il file. Analogamente la scansione può essere fatta con UDP, che molto probabilmente impiegherà più tempo dato che UDP è un protocollo *unfair* e *connection-less*.

La scansione richiede l'uso di un flag aggiuntivo rispetto a quelli precedenti, ossia il flag **-sU**, che indica ad nmap di eseguire una scansione UDP.

```
sudo nmap -sU -Pn -p- -T4 -vv -reason -oN "nmap.results"
"target_ip_addr"
```

Anche qui, col comando **less** è possibile visualizzare il file.



```

# Nmap 7.92 scan initiated Tue Sep 14 17:01:35 2021 as: nmap -sU -Pn -p- -T4 -vv --reason -oN NS/udp_scan/nmap.results 10.0.2.15
Nmap scan report for Linux-Mint-19-S0 (10.0.2.15)
Host is up, received user-set (0.000028s latency).
Scanned at 2021-09-14 17:01:35 CEST for 3s
Not shown: 65531 closed udp ports (port-unreach)
PORT      STATE      SERVICE    REASON
68/udp    open|filtered dhcpc      no-response
631/udp   open|filtered ipp        no-response
5353/udp  open|filtered zeroconf   no-response
37601/udp open|filtered unknown    no-response

Read data files from: /usr/local/bin/./share/nmap
# Nmap done at Tue Sep 14 17:01:39 2021 -- 1 IP address (1 host up) scanned in 3.83 seconds
  
```

Possiamo vedere come nmap ha dovuto inviare molti pacchetti alle porte che sono state segnalate come aperte o filtrate, chiedendo di vedere il traffico UDP verso una delle porte segnalate:

```
sudo tcpdump -nn -Q out -r "file_name" 'udp and port x'
```

dove x sarà il numero della porta aperta o filtrata.

```

studente@Linux-Mint-19-S0:~/Scrivania$ sudo tcpdump -nn -Q out -r ~/scan_results/udp_scan/packets 'udp and port 53'
tcpdump: /home/studente/scan_results/udp_scan/packets: No such file or directory
studente@Linux-Mint-19-S0:~/Scrivania$ sudo tcpdump -nn -Q out -r NS/udp_scan/pacchetti 'udp and port 68'
reading from file NS/udp_scan/pacchetti, link-type EN10MB (Ethernet)
studente@Linux-Mint-19-S0:~/Scrivania$ sudo tcpdump -nn -Q out -r NS/udp_scan/pacchetti 'udp and port 631'
reading from file NS/udp_scan/pacchetti, link-type EN10MB (Ethernet)
studente@Linux-Mint-19-S0:~/Scrivania$ sudo tcpdump -nn -Q out -r NS/udp_scan/pacchetti 'udp and port 5353'
reading from file NS/udp_scan/pacchetti, link-type EN10MB (Ethernet)
17:04:05.974344 IP 10.0.2.15.5353 > 224.0.0.251.5353: 0 [2q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ipp._tcp.local. (45)
  
```

È possibile poi eseguire dei test per vedere se nmap ha identificato il sistema operativo in esecuzione oppure la versione dei vari servizi.

Per l'analisi delle versioni dei servizi si può usare il *fingerprinting*; la scansione nmap che dobbiamo usare è attivata dal flag `-sV`, e dato che abbiamo già fatto scansioni SYN e UDP, possiamo passare le porte esatte che vogliamo guardare con il flag `-p`. Per scoprire invece il sistema operativo dell'host ci si può basare sulle caratteristiche del suo software, più o meno allo stesso modo del service versioning. Il flag di cui abbiamo bisogno per eseguire il rilevamento del sistema operativo è `-O`.

```

# Nmap 7.92 scan initiated Tue Sep 14 17:18:12 2021 as: nmap -O -Pn -vv --reason -oN NS/versioni/os_version.nmap 10.0.2.15
Nmap scan report for Linux-Mint-19-S0 (10.0.2.15)
Host is up, received user-set (0.00018s latency).
Scanned at 2021-09-14 17:18:12 CEST for 1s
All 1000 scanned ports on Linux-Mint-19-S0 (10.0.2.15) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
Too many fingerprints match this host to give specific OS details
TCP/IP fingerprint:
SCAN(V=7.92E=4%D=9/14%OT=%CT=1%CU=41119%PV=Y%DS=0%DC=L%G=N%TM=6140BD35%P=x86_64-unknown-linux-gnu)
SEQ(CI=I%II=I)
SEQ(II=I)
T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 0 hops

Read data files from: /usr/local/bin/./share/nmap
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Tue Sep 14 17:18:13 2021 -- 1 IP address (1 host up) scanned in 2.04 seconds
NS/versioni/os_version.nmap (END)
  
```

L'identificazione del sistema operativo può aiutare un attaccante a determinare quali exploit possono essere utili sul sistema. Configurare il firewall per rispondere a meno richieste può aiutare ad ostacolare l'accuratezza di alcuni di questi metodi di rilevamento.