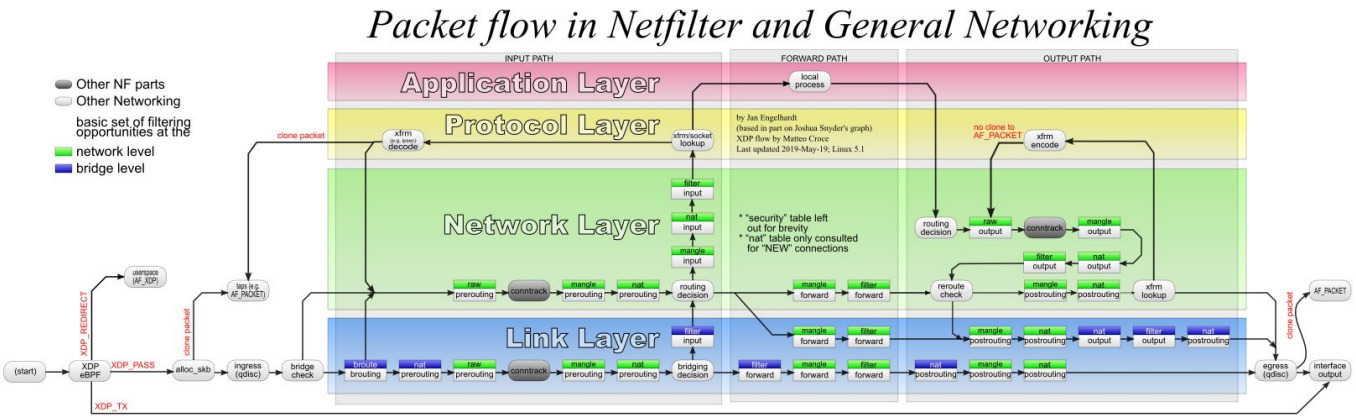
Linux Firewall

Ing. Filippo Poltronieri
filippo.poltronieri@unife.it

Netfilter

- Netfilter è un subsystem di Linux che fornisce gli strumenti per il controllo e il filtraggio del traffico di rete: iptables/nftables, traffic control (tc) e Network Emulation (netem)
- **tc** e **netem** sono strumenti utili per realizzare il controllo di flussi di traffico e realizzare una emulazione di una rete con determinate caratteristiche: latenza aumentata, banda ridotta, etc…
- **iptables** consente di definire un firewall per configurare il filtering di pacchetti IPv4 e IPv6.
- Iptables viene utilizzato in molti router / access-point di uso domestico che utilizzano Linux (BusyBox)
- Le regole inserite dall'interfaccia web vengono poi tradotte in regole iptables

Netfilter

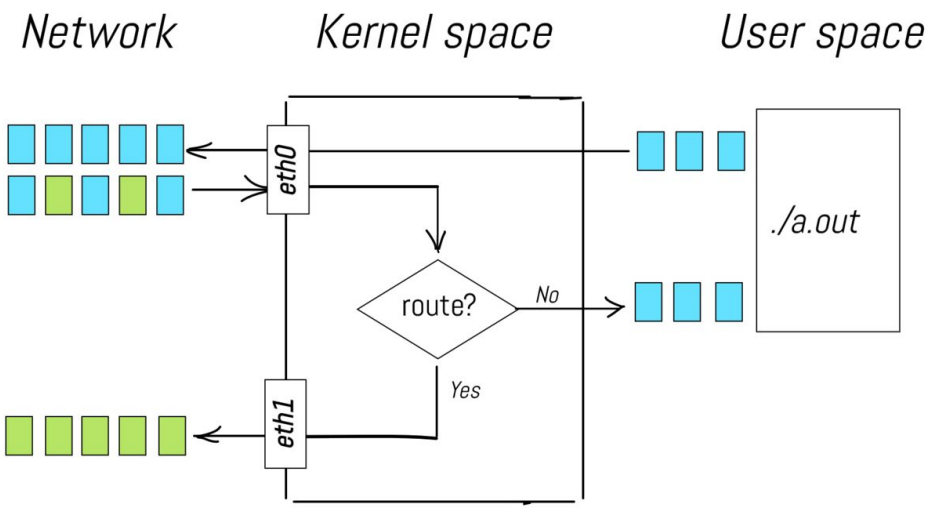


Iptables

Posso configurare un firewall utilizzando diversi requisiti. Per esempio posso aver bisogno di:

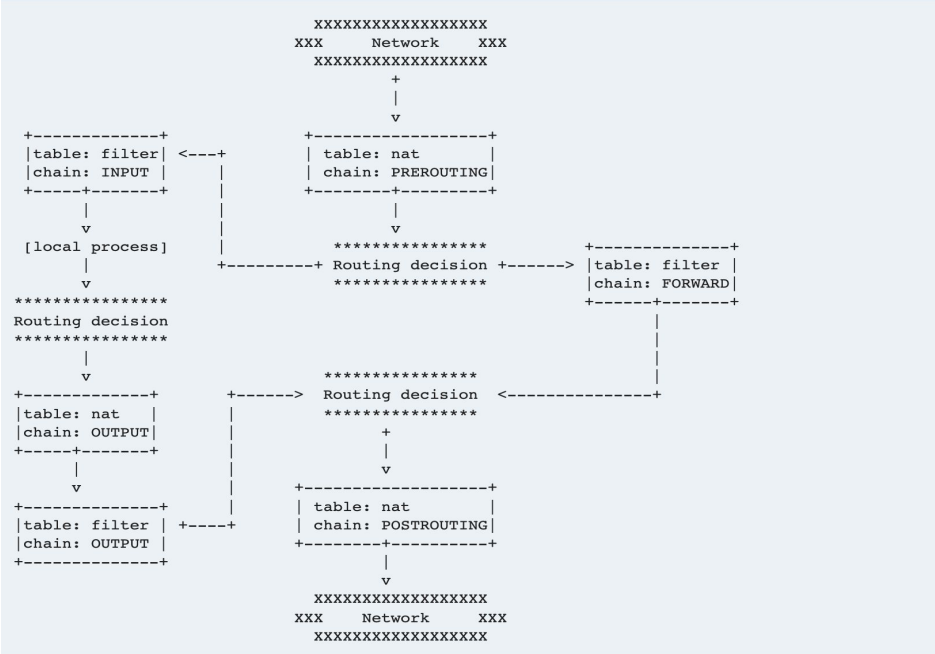
- Impostare un comportamento di default → **blocco**/**accetto** tutto il traffico in ingresso nella rete
- Possibilità di consentire del traffico in ingresso da una lista / range di IP sorgente verso una lista / range di IP destinazione
- Possibilità di filtrare traffico in base a porta sorgente e destinazione. Esempio, aprire la porta 80 per HTTP e 443 per HTTPS
- Porte application-specific per servizi di rete (logging), applicazioni etc…
- Traffico SSH in entrata per il controllo remoto di macchine UNIX e/o apparecchiature di rete

Iptables



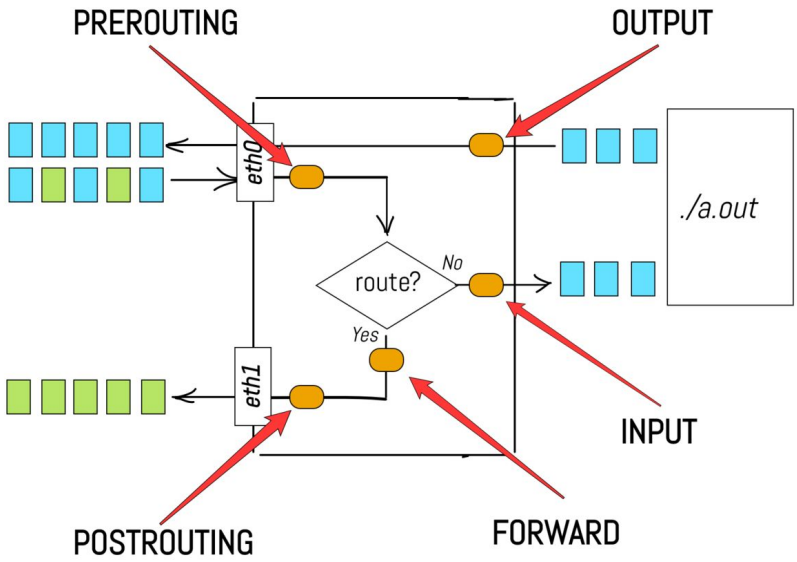
Courtesy of https://iximiuz.com/en/posts/laymans-iptables-101/
A very interesting post to read!

Iptables



Iptables

Courtesy of https://iximiuz.com/en/posts/laymans-iptables-101/



Iptables

- iptables applica catene (chains) ordinate di regole (rules) a pacchetti di rete
- L'insieme di chains definisce delle tables che vengono utilizzate per gestire diversi tipi di traffico.
- Esiste una tabella di default chiamata **filter** che viene utilizzata per filtrare il traffico di rete. **filter** contiene tre chains di default: FORWARD, INPUT e OUTPUT.
- Ogni pacchetto ricevuto dal kernel viene passato a una di queste tre chains.
- Le regole nella chain di FORWARD vengono applicate a *tutti i pacchetti che arrivano a una interfaccia di rete e devono essere inoltrati a un'altra interfaccia*.

Iptables

- Le regole nelle chain di INPUT e OUTPUT vengono invece applicate al traffico con destinazione o sorgente l'host locale.
- Con le tre chains di default è possibile realizzare un firewall completo tra due interfacce di rete.
- E' possibile definire configurazioni custom per scenari più complessi e specifici.

Iptables

- Oltre alla tabella di filter, **iptables** include anche le tabelle di *nat* e *mangle*.
- La tabella di *nat* include chains di rules che controllano la Network Address Translation (NAT).
- Attraverso NAT è possibile controllare le chain di PREROUTING e POSTROUTING che permettono di modificare, rispettivamente, le informazioni relative alla sorgente e destinazione di un pacchetto di rete.
- La table *mangle* invece contiene chains che permettono di modificare il contenuto di pacchetti di rete al di fuori del contesto di NAT e packet filtering. *Ad esempio effettuare il reset del parametro TTL di un pacchetto IP (non viene tipicamente utilizzata)*.

Iptables - rule targets

- Ogni rule facente parte di una chain definisce una clausola “target” che specifica cosa fare in caso di matching.
- Quando un pacchetto fa match con una rule di solito non vengono controllate ulteriori rule.
- Diversi target sono definiti all’interno di iptables ed è possibile specificare un’altra chain come target di una rule.
- Nella tabella di filter sono definiti diversi target: ACCEPT, DROP, REJECT, LOG, ULOG, REDIRECT, RETURN, MIRROR, e QUEUE.
- Banalmente quando un pacchetto fa match con una regola di ACCEPT, questo può procedere verso la sua destinazione.

Iptables - Default target

- DROP e REJECT specifica invece di “droppare” il pacchetto; A differenza di DROP, *REJECT ritorna al source del pacchetto un messaggio di errore ICMP*.
- LOG e ULOG consentono di loggare i pacchetti che fanno match: ULOG fornisce un logging esteso rispetto a LOG.
- REDIRECT effettua la redirezione dei pacchetti a un proxy invece che instradarli verso la loro destinazione (traffico web verso Squid).
- RETURN serve a specificare il termine di una chain user-defined.
- MIRROR effettua uno swap tra IP sorgente e destinazione prima di inviare il pacchetto
- QUEUE invia i pacchetti a un programma locale utilizzando un apposito modulo del Kernel.

Iptables - Abilitare IP forwarding

Prima di poter utilizzare iptables come un firewall, bisogna abilitare la variabile (flag) di IP forwarding a livello del kernel. Questo procedimento potrebbe variare in base alla distribuzione Linux utilizzata.

Per abilitare la funzionalità di IP forwarding in modo permanente su distro Debian-based si può modificare il file `/etc/sysctl.conf`, decommentando o inserendo l'opzione relativa:

```
net.ipv4.ip_forward=1
```

Puoi possiamo utilizzare il comando sysctl per ricaricare le configurazioni

```
$ sudo sysctl -p /etc/sysctl.conf
```


Iptables - Abilitare IP forwarding

Sempre utilizzando il comando sysctl è possibile abilitare temporaneamente Ip forwarding:

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

o disabilitarlo:

```
$ sudo sysctl -w net.ipv4.ip_forward=0
```

Invece possiamo verificare la sua attivazione con:

```
$ cat /proc/sys/net/ipv4/ip_forward
```

(se attivo l'output sarà 1, 0 viceversa)

Iptables - Comandi

Nella configurazione di un firewall ad-hoc il primo passo è la pulizia delle tabelle.

```
iptables -F chain-name
```

l'opzione -F elimina tutte le regole definite all'interno di chain.

```
iptables -P chain-name target
```

dove il parametro -P va a impostare la policy di default per la chain. Per la chain target viene di solito utilizzato `DROP` come default policy (blocco tutto, poi scelgo cosa fare passare).

Iptables - Comandi

```
iptables -A chain-name -i interface -j target
```

dove l'opzione -A va ad aggiungere un comando in append alla chain. Se non viene specificata una table con l'opzione **-t**, il comando viene applicato alla chain nella filter table. Il parametro **-i** applica la rule all'interfaccia specificata, mentre **-j** serve per specificare il target.

```
iptables -L -v
```

permette di visualizzare quante volte una rule ha fatto match con un pacchetto.

Iptables - Comandi

Il comando iptables accetta molte opzioni. Eccone alcune:

-p proto	Permette di specificare un protocollo: tcp, udp e icmp
-s source-ip	Match con hostname o indirizzo IP sorgente
-d dest-ip	Match con hostname o indirizzo IP destinazione
--sport #port	Match con porta sorgente
--dport #port	Match con porta destinazione
-t	Specifica la tabella a cui applicare il comando
!	nega una clausula

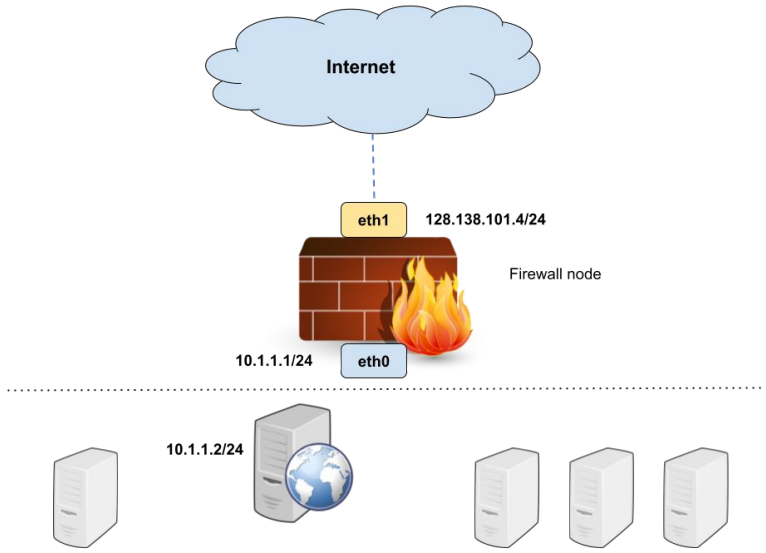
Iptables - Esempio

Supponiamo di avere un nodo firewall con due interfacce di rete: *eth1* è collegata alla rete Internet e l'interfaccia *eth0* è collegata a una rete interna. L'indirizzo dell'interfaccia eth1 è 128.138.101.4, quelllo dell'interfaccia eht0 è 10.1.1.1/24.

Sulla rete interna è presente un web server con indirizzo IP 10.1.1.2/24 che dobbiamo rendere visibile sulla rete Internet.

Vogliamo configurare il firewall utilizzando il comando iptables.

Iptables - Esempio



Iptables - Esempio

La prima operazione consiste nel fare il flush (pulizia) delle regole presenti in tutte le chain. Poi andiamo impostare il comportamento di default droppando tutti i pacchetti in INPUT e FORWARDING (la strategia più sicura):

```
iptables -F

iptables -P INPUT DROP

iptables -P FORWARD DROP
```

Iptables - Esempio

I passi successivi consistono nello specificare il traffico ammesso.

Accettare tutte le connessioni che hanno origine all'interno della rete sicura:

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
```

Abilitare sulla catena di forward solamente le connessioni ai servizi di rete sul nodo con ip 10.1.1.2 (SSH, HTTP, HTTPS):

```
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j ACCEPT
```


Iptables - Esempio

Per quanto riguarda il nodo firewall (10.1.1.1) possiamo abilitare solamente il traffico SSH (utile per controllare il firewall stesso).

```
iptables -A INPUT -i eth0 -d 10.1.1.1 -p tcp --dport 22 -j ACCEPT
```

Inoltre abilitiamo il traffico sull'interfaccia di loopback e il ping dalla rete interna (per verificare il funzionamento del nodo):

```
iptables -A INPUT -i lo -d 127.0.0.1 -p ANY -j ACCEPT
iptables -A INPUT -i eth0 -d 10.1.1.1 -p icmp --icmp-type 8 -j ACCEPT
```

--icmp-type 8 corrisponde a un messaggio di ECHO_REQUEST

Iptables - Esempio

Oltre al ping dalla rete interna dobbiamo abilitare anche alcuni pacchetti ICMP (diretti al web server) provenienti dalla rete Internet verso il firewall:

```
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 5 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
```


Iptables - Esempio

… e al web server sulla rete interna:

```
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 0 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 3 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 5 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 11 -j ACCEPT
```

Iptables - Esempio

Il passo successivo è quello di aggiungere regole alla chain di PREROUTING per realizzare un filtro antispoofing. Dato che tutti i pacchetti transitano per la chain di PREROUTING, una volta droppati non devono saranno presenti nelle chain di INPUT e FORWARD.

```
iptables -t nat -A PREROUTING -i eth1 -s 10.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 192.168.0.0/16 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 127.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i eth1 -s 224.0.0.0/4 -j DROP
```


Iptables - Esempio

Infine, possiamo specificare nelle chain di INPUT e FORWARD una regola che ci permette di effettuare il log di tutti i pacchetti che provengono dalla rete Internet:

```
iptables -A INPUT -i eth1 -j LOG
iptables -A FORWARD -i eth1 -j LOG
```

Questi pacchetti saranno comunque bloccati in quanto abbiamo specificato il comando -P come prima regola delle chain di INPUT e FORWARD.

Iptables - Esempio

Una alternativa consiste nell'utilizzare regole stateful per il filtraggio dei pacchetti. Invece che abilitare servizi specifici in entrata, possiamo utilizzare il firewall per abilitare risposte in entrata alle richieste dei client.

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
```

Abilita tutto il traffico in uscita

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

consente solo il traffico in ingresso che è collegato alle connessioni iniziate da host presenti all'interno della rete.

Iptables - NAT

Caricare i moduli kernel per il connection track per abilitare il NAT (oltre a ip forwarding)

Per fare il routing di pacchetti utilizzando NAT dobbiamo utilizzare la seguente regola iptables:

iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 128.138.101.4

eth1 *è l'interfaccia connessa alla rete Internet e non appare nel comando*, ma il suo indirizzo IP è indicato come argomento di --to.

eth0 *è l'interfaccia connessa alla rete interna*. Agli host della rete internet sembra che tutti i pacchetti originati da host interni alla rete abbiano l'indirizzo dell'interfaccia eth1. L'host che implementa il NAT quando riceve pacchetti, controlla la loro effettiva destinazione e li riscrive utilizzando l'indirizzo IP del nodo sulla rete interna.

Iptables - Salvare una configurazione

iptables-save permette di fare il dump delle tabelle iptables sia su stdout che su un file.

```
iptables-save -f iptables_rules
```

salva le regole in un file chiamato iptables_rules.

E' possibile specificare diverse opzioni per il comando: salvare solamente table specifiche, etc… Si consulti il man associato (**man iptables-save**).

Si veda anche il man dei comandi **iptables-apply** e **iptables-restore** per caricare una configurazione iptables.

Iptables - Cheat Sheet

Un cheat sheet è un insieme di regole che può aiutarci nella definizione dei nostri firewall.

Una raccolta molto interessante di regole:

https://andreafortuna.org/2019/05/08/iptables-a-simple-cheatsheet/

Iptables

- Iptables è attualmente un framework legacy, sostituito da **nftables** dalla versione mainline 3.13 del kernel Linux.
- Esistono tool che permettono di tradurre delle regole iptables in un formato compatibile con nft.
- Questi tool offrono una sintassi molto simile a quella che veniva utilizzata dal comando iptables.
- Tuttavia la compatibilità non è sempre garantita. Si vede l'interessante post: https://www.redhat.com/en/blog/using-iptables-nft-hybrid-linux-firewall

nftables

Nftables è il sostituto di iptables dalla versione del kernel 3.13. Lo scopo di nftables è quello di sostituire con un unico comando sia la versione IPv4 che IPv6 di iptables oltre a arptables e brtables.

Come iptables, anche nftables fa parte del subsystem Netfilter.

Nftables ha l'obiettivo di fornire una sintassi semplificata (molto simile a quella utilizzata nel comando tcpdump) e più facile gestione delle regole.

A differenza di iptables, in nftables le regole vengono specificate attraverso una client utility che va poi a tradurre regole ottimizzate per il kernel.

nftables

Il primo passo per l'utilizzo di nftables è quello di verificare se il suo modulo è presente nel kernel utilizzando il comando modinfo:

```
modinfo nf_tables
```

Con il comando lsmod è possibile controllare i moduli kernel nftables attivati:

```
lsmod | grep nf_tables
```

tra questi dovremmo trovare nf_tables_ipv4 e nf_tables_ipv6.

Il passo successivo è quello di disabilitare iptables se attivo (con un flush delle tabelle):

```
iptables -F
```

```
ip6tables -F
```


nftables - Sintassi

viene utilizzato per i commenti

; per specificare diversi comandi

**** (backslash) per dividere un'istruzione su diverse linee (per favorire la leggibilità di una istruzione / comando)

nftables supporta il concetto di variabili per evitare ripetizioni e quindi errori di inserimento:

```
define ext_if = eht0; define int_if = eth1
```

```
define all_if = { $ext_if, $int_if }
```

nftables

Una rule è composta di più parti: una (o più) expression, un operator e una action.

Una expression è valutata da sinistra a destra. Se la prima expression fa match vengono valutate anche le altre expression (se presenti).

```
nft add rule Firewall Incoming ip daddr 192.168.0.1-192.168.0.19 drop
```

Set è una collezione di elementi (indirizzi ip o numeri di porte). La lunghezza massima di un nome di set è di 15 caratteri.

Un set può essere anche anonimo (non avere un nome associato):

```
dport { 22, 23, 80, 443 }
```


nftables

Visualizzare la lista della tabelle inet (sia IPv4 che IPv6):

```
nft list tables inet
```

nftables

```
nft add set inet blacklist blacklist4-perm { type ipv4_addr \; }

nft add element inet blacklist blacklist4-perm { 192.168.1.21, 192.168.1.22
}

nft add rule inet blacklist input ip saddr @blacklist4-perm drop
```

Alcuni esempi di regole, *viene lasciata allo studente la possibilità di approfondire il tool nftables.*

Il manuale è un'ottimo punto di partenza per uno studio approfondito del comando e del suo funzionamento.

ufw

Un'alternativa interessante per la configurazione di un Firewall è rappresentata da **UFW** (Uncomplicated Firewall).

ufw è disponibile in Linux e viene utilizzato di default in Ubuntu per controllare l'interfaccia netfilter (viene utilizzato **iptables** come backend).

UFW può essere configurato sia tramite il comando ufw, ma anche una versione con interfaccia grafica del comando (Gufw).

Se non è presente sul proprio sistema, ufw può essere installato utilizzando il package manager della distribuzione di interesse (apt, zypper, pacman, …).

ufw

Una volta installato, il primo passo è verificarne lo stato:

```
# ufw status
```

Si può abilitare/disabilitare utilizzando i relativi comandi di enable/disable. Se abilitato, il firewall viene creato del sistema (come per i servizi configurati con systemctl).

```
# ufw enable
```

Possiamo valutare ancora una volta lo stato per verificare quale sono le regole attive sulla macchina con un'interfaccia del tipo:

```
To                         Action                 From
```


ufw

Le regole del firewall si configurano poi con una DSL semplice e intuitiva, che può essere consultata nel manuale del comando (man ufw).

Le principali azioni sono quelle di **allow**, **deny**, **reject**, che consentono di cambiare le policy per il traffico incoming, outgoing e routed. Si può specificare anche una policy di default che va poi modificata.

Alcuni esempi:

```
# ufw allow in on eth0 from 192.168.0.0/16

# ufw allow out on eth1 to 10.0.0.0/8

# ufw route allow in on eth0 out on eth1 to 10.0.0.0/8 from
192.168.0.0/16
```

ufw

Per esempio, nel caso di un web server posso configurare il firewall per consentire connessioni http e https in entrata (**in**):

```
# ufw allow in http
```

```
# ufw allow in https
```

O anche nella sua forma più generica in e out:

```
# ufw allow https
```

Disabilitare il traffico telnet in uscita:

```
# ufw reject out telnet
```

Se non viene specificata una direction (in/out), ufw applica la regola al traffico in ingresso.

ufw

Oppure, regole più specifiche in cui devo indicare IP e porta sorgente e IP e porta destinazione. Per esempio, voglio specificare una particolare tupla sorgente (IP/porta/protocollo) e una tupla destinazione:

```
# ufw allow proto udp from 1.2.3.5 port 5469 to 1.2.3.4 port 5469
```

Il manuale del comando è ricco di esempi e una fonte utilissima di materiale per comprendere lo strumento.

Si guardi anche la possibilità di stabilire ***regole specifiche per le singole applicazioni (app***)...

ufw

route viene utilizzato per specificare tutte quelle operazioni in cui il *traffico viene instradato attraverso il firewall*, ad esempio un dispositivo di rete con più interfacce.

```
# ufw route allow in on eth1 out on eth2
```

Consente al traffico proveniente da eth1 e destinato a eth2 di attraversare il firewall.

Un altro esempio interessante è rappresentato da **limit**, che può essere utilizzato per "proteggersi" dagli attacchi di tipo brute-force. Per esempio, voglio limitare le richieste di connessione SSH sul mio host:

```
# ufw limit ssh/tcp
```


ufw

Si possono eliminare le regole utilizzando il sub-command di delete in cui vado a indicare i) la regola o ii) il numero della regola (rilevato dallo status):

```
# ufw status numbered
     To                         Action      From
     --                         ------      ----
[ 1] 80/tcp                     ALLOW IN    Anywhere
[ 2] 443                        ALLOW IN    Anywhere
[ 3] 22/tcp                     ALLOW IN    Anywhere


# ufw delete number #RULE
```

ufw

Materiale interessante:
https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands

Pros:

- risulta molto semplice e intuitivo definire delle regole standard, come abilitare e disabilitare traffico in una direzione;
- si interfaccia con il comando iptables:
- può essere un buon compresso per la configurazione del firewall di postazioni di lavoro;

Cons:

- configurazioni complicate possono essere difficili da implementare;
- andrebbe verificata la compatibilità con il comando nf-tables;

Esercizi 1/3

Nota: per questi esercizi, potrebbe tornarvi utile anche la funzionalità clone di VirtualBox che vi permette di creare un clone di una macchina virtuale.

Avviare la macchina virtuale, trovare il suo indirizzo IP e fare un ping dalla macchina host per verificare la connettività [bridge/nat] (posso utilizzare ping anche su Windows).

Avviare il servizio di iptables e configurare il firewall della macchina virtuale, per implementare le seguenti politiche.

- per default non accetti traffico in ingresso;
- accetti connessioni HTTP/HTTPS in entrata;
- accetti il traffico DNS in entrata;
- accetti connessioni TCP su un ipotetico servizio in ascolto sulla porta 32000;
- visualizzare le regole inserite utilizzando gli opportuni parametri del comando iptables;

Esercizi 2/3

Una volta configurato il firewall, provare a:

- fare un ping alla macchina virtuale dalla macchina host. Funziona?
- se non funziona che regola dovrei impostare ricevere e rispondere ai ping?
- utilizzare nc per creare un server sulla porta 32000, provare a connettersi dall'host e scambiare dei messaggi;
- avviare una sessione di cattura con Wireshark e analizzare il traffico in ingresso e in uscita;

- Terminato l'esercizio, disabilitare iptables e provare a impostare le stesse regole utilizzando però **UFW**

Esercizi 3/3

Impostare una regola NAT

- Salvare le regole impostate su un file.
- Rimuovere tutte le regole impostate sul firewall con un unico comando.
- Ricaricare le regole pre
- Utilizzare iptables e la tabella di nat per modificare tutte le richieste DNS a un nameserver esterno (ad esempio 8.8.8.8). Su quale chain devo agire?
- Verificare il funzionamento di questa regola utilizzando Wireshark. Che comando posso utilizzare per generare una richiesta DNS?

Esempio di Configurazione Access Point - 2

L'interfaccia wlan0 può essere configurata con un IP statico. Per farlo dobbiamo modificare il file /etc/dhcpcd.conf

```
interface wlan0
    static ip_address=192.168.4.1/24
    nohook wpa_supplicant
```

Dobbiamo poi attivare il servizio.

Esempio di Configurazione Access Point - 3

Dobbiamo anche configurare un server DHCP per assegnare indirizzi IP ai dispositivi che si conneteranno all'access point.

```
#/etc/dnsmaq.conf
interface=wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
```

E infine abilitare il servizio.

Esempio di Configurazione Access Point - 4

L'ultimo servizio da configurare è hostapd, che fornisce le funzionalità di access point.

```
#/etc/hostapd/hostapd.conf
country_code=IT
interface=wlan0
ssid=YOURSSID
channel=9
auth_algs=1
wpa=2
wpa_passphrase=YOURPWD
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
```


Esempio di Configurazione Access Point - 5

Dobbiamo poi ricordarci di configurare il NAT andando a impostare il masquerade per i pacchetti generati dai dispositivi connessi all'access point su wlan0 e diretti sull'interfaccia eth0 (verso l'esterno).

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```