
FIREWALL

FIREWALL

È un **oggetto che protegge dei computer connessi ad una rete (quindi un sistema) da attacchi intenzionali**

mirati a:

1) compromettere il funzionamento del sistema,

2) alterare i dati ivi memorizzati,

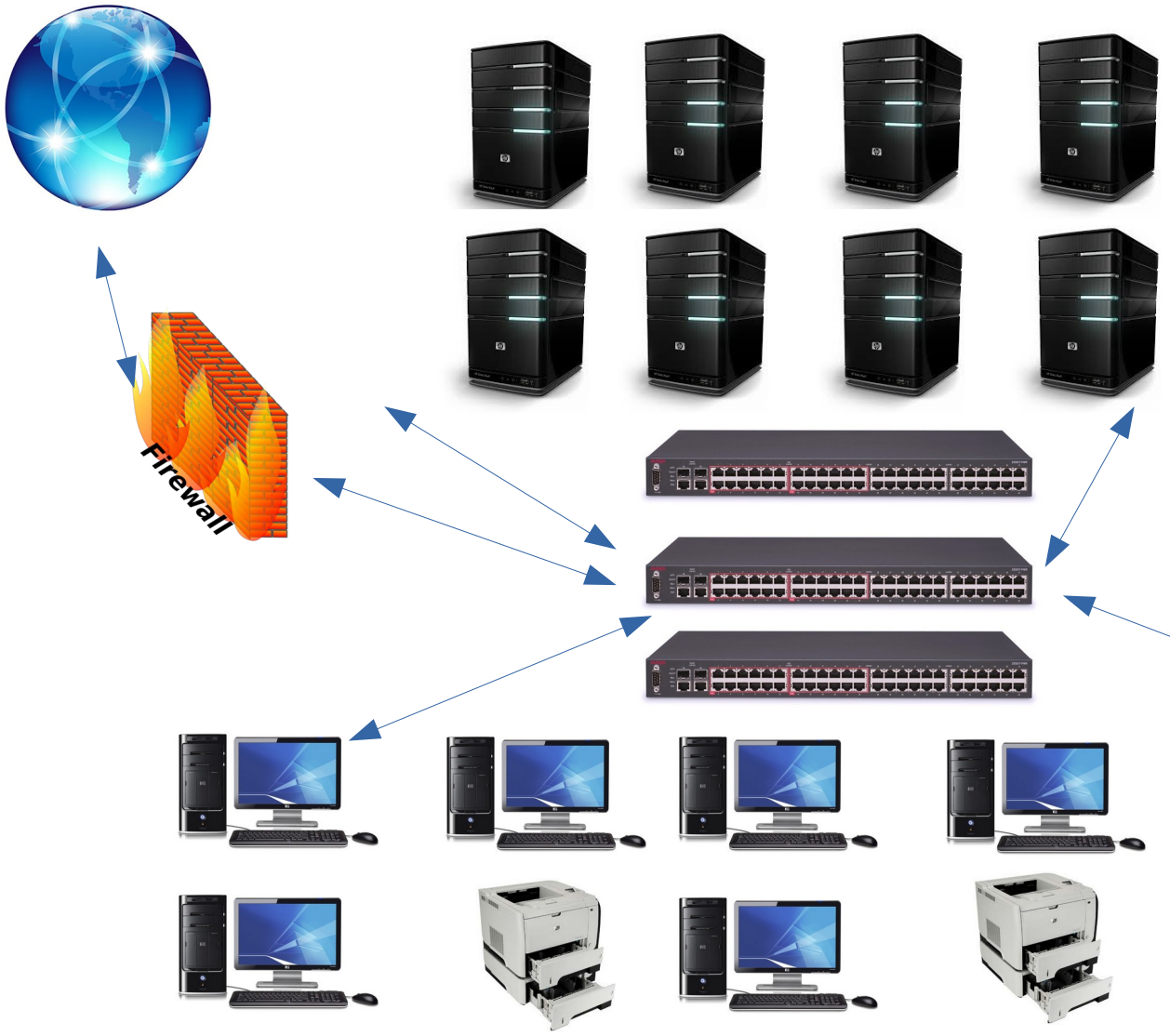
3) accedere a risorse private,

4) effettuare disservizi di tipo **DoS** (**Denial of Service**).

FIREWALL

- I firewall si possono distinguere usando diverse **classificazioni**. Una prima suddivisione può essere fra firewall **hardware** e firewall **software**.
- **Hardware**: si presenta sottoforma di scatolotto, più o meno delle dimensioni di un server rack 1U o di un piccolo pc tower o di uno switch. Può montare due o più schede di rete. Il vostro router adsl/fibra di casa è anche un firewall hardware. Ha un SO cucito su misura per la gestione di un firewall.
- **Software**: è un programma che gestisce l'hardware di un pc/server sottostante come se fosse un firewall poggiando su un SO pre-esistente. I principali firewall software sono:
  - Windows Firewall.
  - **Iptables (linux).**
  - PfSense (FreeBSD).
  - PF (openBSD).
- **Attenzione**: nell'ambito dei firewall, spesso, la dicitura "schede di rete" è un modo per identificare reti diverse.

Esempio di firewall... ricordate?



Più di 30 server tra fisici e virtuali

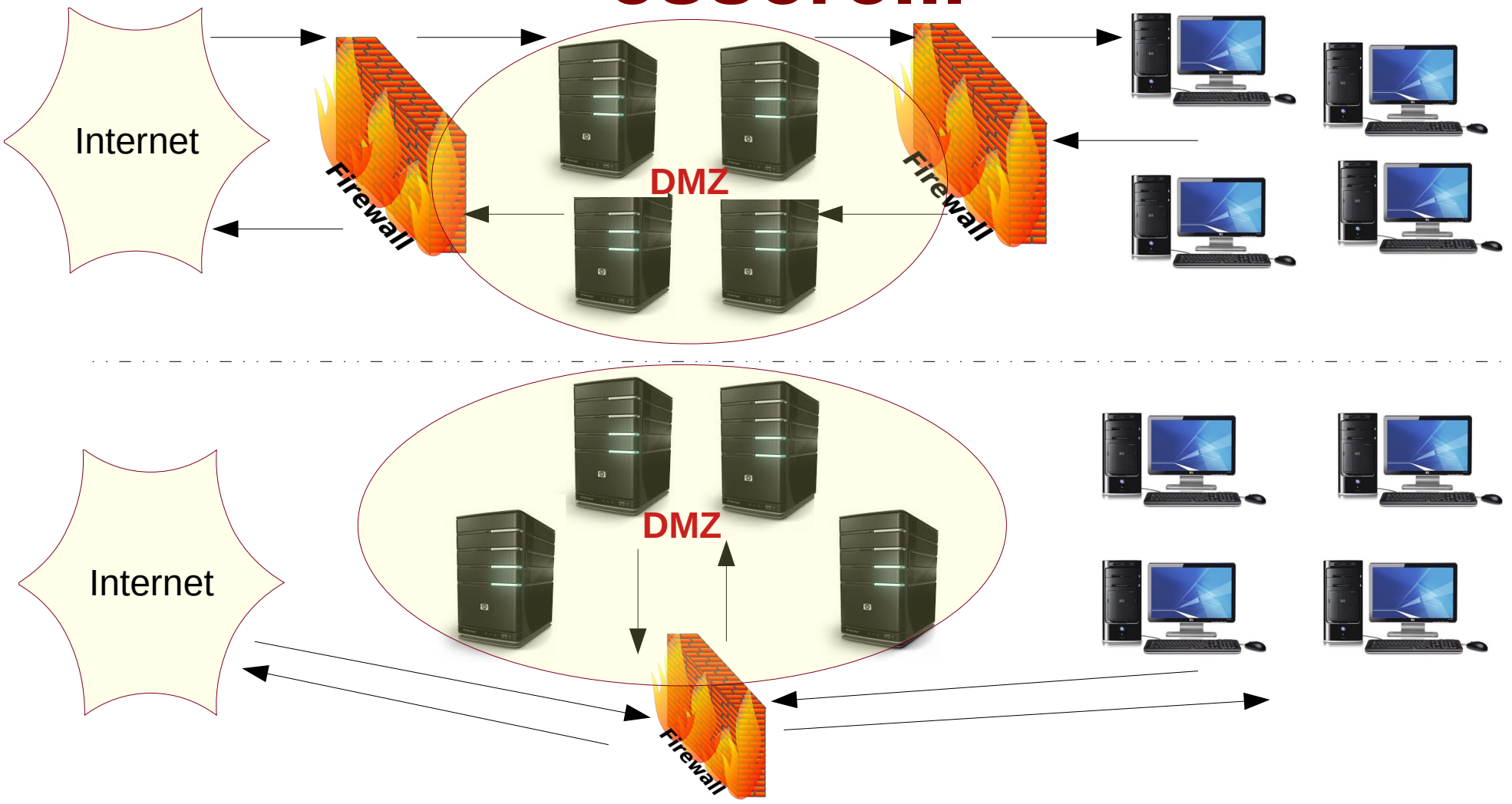
-Servers Segrerie
-Servers Studenti
bau broot roar proxy wwwstud
-Server www, mail, ssh, ftp, svn, printer
- SCSCF

Segreteria 20 pc
4 prt (solo alfa)

Laboratori 74 pc
2 prt (solo zeta)

Docenti 50 pc
3 prt (solo zeta)

Firewall - Come dovrebbe essere...



DMZ – **zona demilitarizzata**, ospita **i sever proteggendoli da attacchi esterni e interni e proteggendo la rete interna da essi**. In questo caso vi possono essere due firewall (**OUT-DMZ** e **DMZ-IN**) oppure una unica macchina con almeno tre schede di rete che filtra il traffico tra una scheda e l'altra (**scheda out – scheda dmz** e **scheda dmz – scheda in**).

FIREWALL

- Un **firewall** protegge i computer della rete agendo come **un sistema di controllo che verifica il traffico che lo attraversa.**
- È quindi un **ponte(bridge)** tra due o più reti.

- **Accetta o scarta il traffico basandosi su delle policy di sicurezza (regole) definite dall'uomo**.
  - In particolare:
    - Verifica i pacchetti in transito (**IP filtering**).
    - Può effettuare un mascheramento degli indirizzi interni (**NAT**).
    - Blocca pacchetti pericolosi e/o non ammessi dalle **policy di sicurezza (regole)**.

FIREWALL: cosa permette

1) **Controllo dei servizi**: permette di decidere a quali tipi di servizi si può accedere sia out/in che in/out.

2) **Controllo della direzione**: permette di decidere se le richieste di alcuni servizi posso essere avviate e inoltrate.

3) **Controllo utente**: regola l'accesso ad un servizio in base all'utente che ha effettuato la richiesta.

4) **Controllo del comportamento**: permette di controllare come sono utilizzati certi servizi (ad esempio il filtraggio dello spam).

- **NB**: I punti **3,4** si riferiscono a firewall che agiscono a **livello 5** nello stack TCP/IP. **1,2** a **livello 3,4**.

FIREWALL POLICY

- Un altro modo per classificare i **firewall** è la distinzione data dalle **politiche di protezione (policy) di default**.

- Vi sono due politiche principali (**duali**) applicabili quando si configura un firewall:

  1) **Tutto ciò che non è espressamente permesso è vietato.**

  OPPURE

  2) **Tutto cio che non è espressamente vietato è permesso.**

1) Tutto ciò che non è espressamente permesso è vietato.

- **Il firewall blocca tutto il traffico e ogni servizio abilitato deve essere configurato caso per caso**.
  - **Maggior sicurezza:Chiaramente qualsiasi nuova minaccia, dovuta a nuovi servizi e/o protocolli viene bloccata sul nascere.**
  - Oneroso (ma neanche tanto) da gestire, ogni volta che bisogna autorizzare un servizio è necessario individuare quali porte aprire.
  - Limita il numero di servizi disponibili all'utente.
  - È necessaria elasticità verso le novità.

2) Tutto cio che non è espressamente vietato è permesso.

- **Il firewall inoltra tutto il traffico e ciascun servizio dannoso deve essere chiuso caso per caso.**
  - **Minor sicurezza:** ogni nuovo protocollo è permesso senza controllo.
  - Più facile da gestire…(insomma...)
  - **Aumenta la difficoltà nel garantire la sicurezza all'aumentare della dimensione della rete.**
  - È come avere la porta di casa aperta a tutti, tranne ad alcuni personaggi che l'hanno già svaligiata….

FIREWALL - Limiti

- **Valido quanto le regole che interpreta:** un **firewall** è un mero applicatore di regole ed in quanto tale è valido quanto le regole che applica!
- **Controlla solo il traffico che lo attraversa**:
  - Nel caso di intrusioni nella rete interna non può nulla.
  - Se il traffico dall'esterno arriva da un percorso non controllato dal firewall (tipo un altra connessione ad internet) il firewall ovviamente non può nulla.
- **È una macchina, quindi**:
  - Può essere **violata**.
  - Può diventare **obsoleta**.
  - Deve svolgere solo la funzione di firewall → deve essere la macchina più protetta della rete.

FIREWALL - Tipologie

- Un’altra classificazione dei **firewall** è data dalle **tipologie**.
- La **tipologia** di un **firewall** **non coincide** con la **politica** con cui è configurato.
- Si distinguono principalmente due tipologie di **firewall** a **seconda del livello dello stack tcp/ip in cui operano**:
  - **Livello rete o Packet filtering:** viene installato a monte della rete protetta ed ha il compito di bloccare o inoltrare i pacchetti IP (leggendo gli `IP Header`) secondo regole definite a priori (ad esempio **iptables**).
  - **Livello applicativo o Circuit/Application Gateway:** analizza e filtra il traffico a livello trasporto/applicazione (leggendo TCP/UDP header) sfruttando la conoscenza del servizio. Non deve essere necessariamente installato a monte della rete protetta.

ESEMPIO DI AG: PROXY SERVER

- **Proxy server**: sono applicazioni software con il compito di mediare il traffico tra rete esterna e rete interna e consentire l'accesso a un servizio specifico (`http`, `https`, `telnet`, `ftp`, ecc).
- I **servizi proxy** *possono* essere concentrati sull'host che funge da **firewall** e hanno le seguenti caratteristiche:
  - sono indipendenti tra di loro.
  - ciascun servizio implementa solo un sottoinsieme delle funzionalità (http o ftp o mail ecc).
  - un **servizio proxy** non accede al disco ad eccezione della lettura del suo file di configurazione.
  - ciascun **servizio proxy** viene eseguito come utente non privilegiato in una directory privata.
- I laboratori didattici del DAIS accedono ad internet tramite un **proxy server http/https**. La *configurazione è da qualche anno trasparente all'utente* grazie ad un insieme di regole realizzate ad hoc sul **firewall**. (**trasparent proxy**).
- Al DAIS usiamo `squid` → `www.squid-cache.org`

FIREWALL – PACKET FILTER

- **Un packet filter scarta o inoltra un pacchetto IP, da e verso la rete interna, sulla base di un insieme di regole di filtraggio.**
- Le regole di filtraggio si basano sul valore dei campi contenuti **nell'intestazione (IP)** e di **trasporto (TCP/UDP)**, tra cui:
  - l'indirizzo del sorgente e del destinatario.
  - il protocollo di trasporto.
  - il numero di porta del sorgente e del destinatario.
  - i flag SYN, ACK nell'header TCP.
- I **firewall packet filter**, o di livello 3, possono essere **Stateful** o **Stateless** (altra classificazione...).

Stateless

Stateless firewall: osserva il traffico sulla rete e blocca i pacchetti sulla base degli indirizzi di sorgente e di destinazione o su altri valori (ad esempio le porte).

- Quindi non tengono conto del modello di traffico o del flusso di dati.
- *Uno **stateless firewall** utilizza un semplice sistema di regole che non tengono in considerazione il fatto che un pacchetto ricevuto possa essere falso*: ad esempio un pacchetto non http che arriva sulla porta 80 viene fatto passare se la porta 80 è aperta…..

Stateful

Stateful firewall: osservano i flussi di traffico dalla sorgente alla destinazione.

- Sono a conoscenza dei percorsi di comunicazione e possono implementare diverse funzioni di **IP Security (IPSec**) come tunnel crittati e la crittografia.
- In termini tecnici, *ciò significa che i **firewall stateful** possono sapere in che stato è una connessione **TCP***: aperta, aperta inviata, sincronizzata, sincronizzazione stabilita, può dire se la MTU è cambiata, se i pacchetti sono frammentati ecc...

STATEFUL PACKET FILTER

- **Considera il traffico come uno scambio bidirezionale di pacchetti IP che costituisce una sessione di conversazione** (**conversation session**).
- *Permette di generare dinamicamente le regole per il prossimo pacchetto nella sessione di conversazione*.
  - In uscita/ingresso, se un pacchetto soddisfa il criterio di selezione della regola dinamica, esso viene lasciato passare e viene generata la regola per il prossimo pacchetto. Altrimenti ad esso sono applicate le regole statiche.
- Permette di concentrarsi sul lasciar passare o bloccare una nuova sessione: i successivi pacchetti della sessione subiranno la stessa sorte.
- Un esempio è **`IPTABLES`**.


IPTABLES

IPTABLES

- **è un sistema di firewalling, implementato a livello kernel, che permette, tramite la definizione di regole, di avere una protezione per il filtraggio del traffico**.
- Si basa su un *meccanismo di regole che determinano il lasciapassare o meno di pacchetti*.
- Le regole devono essere inserite in un contesto ben preciso che prende il nome di **catena** (**chain** da cui il vecchio nome **ipchain**).
- Permette di realizzare firewall di rete ma anche firewall locali all'interno di server che forniscono servizi (come ad esempio un server http).

IPTABLES: Struttura e comandi

- La struttura di **iptables** poggia sul concetto di **catena**.

- Esistono delle **catene di default**: **INPUT**, **OUTPUT**, **FORWARD**, **PREROUTING** e **POSTROUTING**:
  - **INPUT**: lavora sui pacchetti in entrata nel sistema.
  - **OUTPUT**: lavora sui pacchetti in uscita dal sistema.
  - **FORWARD**: lavora sui pacchetti che sono diretti ad un altro host della rete che però devono passare dal nostro sistema: in pratica il sistema agisce come un router.
  - **PREROUTING**: lavora sui pacchetti in entrata ai quali vengono già applicate delle regole ben definite prima di essere instradate nel sistema.
  - **POSTROUTING**: lavora sui pacchetti in uscita dal sistema ma solamente dopo che è stato deciso il loro instradamento.

IPTABLES: Struttura e comandi

- Ogni catena fa parte di una **tabella**:
  - **Filter**: **INPUT**, **OUTPUT**, **FORWARD**.
  - **Nat**: **OUTPUT**, **PREROUTING**, **POSTROUTING**.
- **Filter**: regole per il filtraggio dei pacchetti.
- **Nat**: regole per l'instradamento dei pacchetti.
- Esiste una terza tabella**,** **Mangle** che viene usata quando le opzioni dei pacchetti devono essere modificate. La vediamo più avanti.
- Si possono creare anche nuove tabelle utili per la gestione di firewall più complessi.

IPTABLES: Struttura e comandi

- L'utilizzo di un firewall basato su **`iptables`** passa attraverso la chiamata ad un omonimo comando (sempre da root): `iptables`.
- Per visualizzare l'elenco completo delle regole usiamo:

```
# iptables -L
```

- Per visualizzare le regole della tabella **filter** usiamo il seguente comando:

```
# iptables -t filter -L
```

- Per visualizzare le regole della tabella **nat** (**PREROUTING** e **POSTROUTING**):

```
# iptables -t nat -L
```

- Se ci interessa solo la catena **INPUT**:

```
# iptables -L INPUT
```

IPTABLES: Struttura e comandi

- Prima di scrivere le regole di un nuovo firewall è sempre bene eliminare le regole vecchie…

```
# iptables -F ──► cancella le regole
# iptables -X ──► cancella le catene definite
                  dall'utente
# iptables -Z ──► azzera i contatori
# iptables -t nat -F
```

- È poi necessario stabilire la politica con cui vogliamo partire. Ad esempio se decidiamo di accettare tutto per la catena di **INPUT**:

```
# iptables -P INPUT ACCEPT;
```

  - Con questo comando accettate tutti i pacchetti di input verso il vostro host.
  - Cambiando `ACCEPT` con `DROP` o `REJECT` rifiutate tutti i pacchetti in ingresso… occhio a non chiudervi fuori!!!!!

IPTABLES – Esempio, header di uno script.

- **`iptables`** è un **comando** che permette di definire delle regole attraverso **chiamate successive** al comando stesso: **un firewall con `iptables` è uno script di shell.**

```
#!/bin/bash
echo 1 > /proc/sys/net/ipv4/ip_forward # abilita ip_forward
IPTABLES=$(which iptables)
# definizione di altre variabili utili
 . . . . . .
# istruzioni del firewall, iniziamo svuotando le catene.
$IPTABLES -F
$IPTABLES -X
$IPTABLES -Z
$IPTABLES -t nat -F
```

IPTABLES – Esempio

- In questo esempio blocchiamo tutto in ingresso e decidiamo successivamente cosa far passare (ad esempio `ssh`, porta `22`).

```
$IPTABLES -P INPUT DROP # -P definisce la politica della
                        # catena

$IPTABLES -A INPUT -i lo -j ACCEPT # è fondamentale abilitare
                                   # sempre l'accesso al server
                                   # stesso.

$IPTABLES -A INPUT -p tcp --dport 22 -j ACCEPT
```

- Dove: **`-A (Append)`** inserisce la regola alla fine della catena, **`-p`** indica il protocollo (tcp/udp), **`-j`** indica cosa fare del pacchetto, **`--dport`** indica la porta di destinazione. **`-P`** definisce la politica per la catena.

- La regola può essere ulteriormente raffinata specificando la scheda di rete da cui arriva la richesta (**`-i`**) o/e la rete e l'host sorgente(**`-s`**)/destinazione(**`-d`**), ecc:

```
$IPTABLES -A INPUT -p tcp -s 192.168.0.2 -d 10.0.0.10 -i eth0 --dport
22 -j ACCEPT
```

- Questo è un firewall **stateless**: nel momento in cui avviamo questo script, tutte le connessioni pre-esistenti vengono troncate comprese quelle permesse (compresa `ssh` che abbiamo autorizzato sulla porta `22`).

IPTABLES – Esempio: stateful

- Per evitare la caduta delle connessioni è essenziale il **concetto di stato** che tiene traccia e mantiene le connessioni già avviate (firewall **stateful**):

```
$IPTABLES -P INPUT DROP #definisce la politica della
   #catena

$IPTABLES -A INPUT -i lo -j ACCEPT

$IPTABLES -I INPUT 1 -m state --state ESTABLISHED,RELATED -j
ACCEPT
# La clausola -I permette di inserie una riga in una
# determinata posizione.

$IPTABLES -A INPUT -p tcp --dport 22 -m state --state NEW -j
ACCEPT
# è necessario garantire la persistenza delle connessioni ssh.
```

- Le **connessioni**, gestite dal modulo **CONNTRACK** di **iptables**, classificano il pacchetto in arrivo in uno di questi **stati**: **NEW**, **ESTABLISHED**, **RELATED**, **INVALID**, a seconda delle caratteristiche dello stesso.
- Il modulo **CONNTRACK**, è abitualmente già inserito nel kernel.

IPTABLES – Stati

- **New**: quando il pacchetto in arrivo non fa parte di nessun flusso o **socket** pre-esistente e il flag `SYN` di tcp è `ON`.
- **Established**: quando il pacchetto in arrivo fa parte di un flusso o di un **socket** pre-esistente tracciato dal modulo **CONNTRACK** e ha attivo almeno un TCP flags (`SYN`, `ACK`, `RST`). Dopo aver stabilito la connessione TCP il flag `SYN` del pacchetto deve essere impostato a `OFF` affinchè lo stato sia **established**.
- **Related**: quando il pacchetto in arrivo non fa parte di nessun flusso o socket, ma è atteso poichè esiste un un **socket** che lo prevede. Ad esempio:
  - Pacchetti di dati sulla porta `20` quando vi è una connessione `FTP` sulla `21`.
  - Pacchetti di dati via UDP quando vi è una connessione `SIP` sulla porta TCP `5060`.
- **Invalid**: Se nessuno dei precedenti stati è applicabile al pacchetto in arrivo allora lo stato del pacchetto è **INVALID**. Ciò capita quando vi sono vari tipi di tentativi di accessi fraudolenti, o semplicemente quando arrivano pacchetti inaspettati dalle regole di **CONNTRACK**. Di solito nei log di sistema si trova traccia del passaggio di tali pacchetti, che va sottolineato, possono far parte di traffico non malevolo.

IPTABLES – Ordine delle regole

- **ATTENTI: l'ordine in cui inserite le regole di iptables è importante!!!**
- **`iptables`** inserisce le regole in sequenza dando loro una numerazione progressiva (da 1 a N). Ad esempio, la prima regola inserita, ha il numero 1.
- Abbiamo visto che con `-A` inseriamo la regola in coda alle altre, ciò non sempre è corretto, soprattutto se aggiungiamo regole al volo da terminale…
  - Perchè quando arriva un pacchetto, **`iptables`** inizia a scandagliare le regole una ad una, e **nel momento in cui trova la regola che rispetta i requisiti del pacchetto in esame la esegue senza preoccuparsi di ciò che c'è dopo**.
- Possiamo aggirare questo probema con l'opzione **`-I`** che ci permette di inserire la regola dove vogliamo… se siamo bravi a fare i conti….

```
$IPTABLES -I INPUT 2 -p tcp….
```

IPTABLES - ESEMPIO

```
#!/bin/bash
echo 1 > /proc/sys/net/ipv4/ip_forward
IPTABLES = $(which iptables)
#. . .  regole di pulizia delle catene . . .
$IPTABLES -P INPUT DROP
$IPTABLES -A INPUT -i lo -j ACCEPT
# Accettiamo prima http:
$IPTABLES -A INPUT -p tcp --dport 80 -j ACCEPT
# ……
#     Altre regole
# ……
# Fine dello script
```

Num. Regola
1
2
3
. . .
1000000000

- Dopo aver lanciato lo script, decidiamo di cambiare la regola relativa ad http da terminale: Neghiamo accesso http:

```
# iptables -A INPUT -p tcp --dport 80 -j DROP
```

Funziona?

IPTABLES - ESEMPIO

- Dovete prima cancellare la regola precedente tramite l'opzione **-D** di `iptables`:

```
# iptables -D INPUT 3
# iptables -A INPUT -p tcp --dport 80 -j DROP
```

- Oppure sovrascrivere la precedente:

```
#  iptables -I INPUT 3 -p tcp --dport 80 -j DROP
```

- **O meglio ancora… correggere lo script e rilanciarlo.**..

IPTABLES - ESEMPIO

- Il modo corretto di procedere è correggere lo script su cui si sta lavorando e rilanciarlo!:

```
$IPTABLES -I INPUT 1 -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPTABLES -P INPUT DROP

$IPTABLES -A INPUT -i lo -j ACCEPT

$IPTABLES -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT

$IPTABLES -A INPUT -p tcp --dport 80 -j DROP

$IPTABLES -A INPUT -p tcp --dport 443 -j ACCEPT

$IPTABLES -A INPUT -p tcp --dport 8080 -s 157.138.22.0/24 -j ACCEPT

$IPTABLES -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT # manteniamo persistenti le connessioni in uscita
```

- Se non vogliamo la 80 aperta… la chiudiamo e basta!
- È sempre necessario definire anche una regola per la catena di **OUTPUT** per la gestione di un firewall **STATEFUL**, altrimenti **iptables** non riesce a tenere traccia delle connessioni.

IPTABLES – Logica di attraversamento

- Il kernel gestisce ogni pacchetto secondo logiche e sequenze ben precise, che determinano il modo con cui vanno processate, e configurate, le regole di **`iptables`**.
- Possiamo distinguere tre scenari a seconda del percorso dei pacchetti:
  1) Pacchetti destinati al sistema locale.
  2) Pacchetti in uscita dal sistema locale.
  3) Pacchetti che attraversano il firewall.

- Fonte: `http://openskill.info/topic.php?ID=125`

Pacchetti destinati al sistema locale

1) Un pacchetto entra da una interfaccia (es: `eth0/ens33`).
2) Attraversa la tabella/catena **mangle/PREROUTING** (set TOS ecc).
3) Attraversa la tabella/catena **nat/PREROUTING** (DNAT).
4) Viene fatta la decisione di routing sulla base dell'IP destinazione attuale (in questo caso locale).
5) Attraversa la tabella/catena **mangle/INPUT.**
6) Attraversa la tabella/catena **filter/INPUT.**
7) Viene processato da una applicazione locale.

Pacchetti in uscita dal sistema locale

1) Il pacchetto viene processato/generato da una applicazione locale.
2) Il kernel decide come instradare il pacchetto, sulla base dell'IP destinazione(può essere remoto o locale).
3) Attraversa la tabella/catena **mangle/OUTPUT.**
4) Attraversa la tabella/catena **nat/OUTPUT.**
5) Attraversa la tabella/catena **filter/OUTPUT.**
6) Attraversa la tabella/catena **mangle/ POSTROUTING.**
7) Attraversa la tabella/catena **nat/POSTROUTING** (SNAT)**.**
8) Esce dall'interfaccia per raggiungere la sua destinazione (es: `eth1/ens34`).

Pacchetti che attraversano il firewall

1) Un pacchetto entra da una interfaccia (es: `eth0/ens33`).
2) Attraversa la tabella/catena **mangle/PREROUTING.**
3) Attraversa la tabella/catena **nat/PREROUTING.**
4) Viene instradato verso l'IP di destinazione (in questo caso remoto).
5) Attraversa la tabella/catena **mangle/FORWARD.**
6) Attraversa la tabella/catena **filter/FORWARD.**
7) Attraversa la tabella/catena **mangle/POSTROUTING.**
8) Attraversa la tabella/catena **nat/POSTROUTING.**
9) Esce dall'interfaccia per raggiungere la sua destinazione (es: `eth1/ens34`).

FIREWALL DI RETE CLASSICO

- Un **firewall di rete classico permette di filtrare il traffico esterno prima che raggiunga dei server in una DMZ**.
- Ha almeno **2** interfacce di rete (una esterna, esposta ad Internet, una sulla **DMZ**, che costituisce il default gateway dei server pubblici).
- Può agire in 2 modi principali:
  - **Routing** fra rete esterna e DMZ con **IP** (della rete interna) **pubblici**.
  - **Natting** fra rete esterna e DMZ con **IP** (della rete interna) **privati** ed eventualmente nattati (riconvertiti in un indirizzo pubblico) dal firewall.
- In entrambi i casi **l'IP forwarding va attivato**. Si tratta di aggiungere al vostro script di iptables la riga:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

FIREWALL DI RETE CLASSICO

- Definite le regole di **INPUT** a protezione del firewall, la parte interessante riguarda la catena di **FORWARD** (ciò che passa attraverso il firewall…):

```
#politica
$IPTABLES -P FORWARD DROP
#WEB
$IPTABLES -A FORWARD -d 157.138.20.11 -i eth0 -p tcp
 --dport 80 -j ACCEPT
#WEB
$IPTABLES -A FORWARD -d 157.138.20.11 -i eth0 -p tcp

--dport 443 -j ACCEPT
#SSH
$IPTABLES -A FORWARD -d 157.138.20.17 -i eth0 -p tcp
 --dport 22 -j ACCEPT
#DNS
$IPTABLES -A FORWARD -d 157.138.20.12 -i eth0 -p udp
 --dport 53 -j ACCEPT
#Stateful
$IPTABLES -A FORWARD -m state --state
 RELATED,ESTABLISHED -j ACCEPT
```

FIREWALL DI RETE CLASSICO CON NAT (ROUTER ADSL)

```
$IPTABLES -A FORWARD -d 10.0.0.2 -i eth0 -p tcp --dport 80 -j
ACCEPT

$IPTABLES -A FORWARD -d 10.0.0.2 -i eth0 -p tcp --dport 443 -j
ACCEPT

# Accettiamo tutto dalla rete 10.0.0.0/8 su eth1
$IPTABLES -A FORWARD -p tcp -s 10.0.0.0/255.0.0.0 -i eth1 -j
ACCEPT

$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -j
ACCEPT

$IPTABLES -t nat -A POSTROUTING -s 10.0.0.0/255.0.0.0 -j SNAT
```

Esercizi per l'esame - 1

- Scrivere una regola di **`iptables`** che blocchi in input l'accesso ftp alla rete `157.138.22.0`.

```
# iptables -A INPUT -p tcp --dport 21 \
  -s 157.138.22.0/24 -j DROP
```

- Data la politica `iptables -P INPUT -j DROP`, realizzare delle regole `iptables` che permettano l'accesso a `http` e `https` per un web server:

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Esercizi per l'esame - 2

- Supporre di essere amministratore di un firewall la cui politica di default sulla catena di **FORWARD** è: `iptables -P FORWARD DROP`:
  - Accettare per la macchina 192.168.1.11 le connessioni ssh:

```
# iptables -A FORWARD -d 192.168.1.11 -i eth0 -p tcp
   --dport 22 -j ACCEPT
```

  - Accettare per la macchina 192.168.1.141 le connessioni http, https:

```
# iptables -A FORWARD -d 192.168.1.141 -i eth0 -p tcp
  --dport 80 -j ACCEPT

# iptables -A FORWARD -d 192.168.1.141 -i eth0 -p tcp
  --dport 443 -j ACCEPT
```

  - Accettare per la macchina 192.168.1.110 le connessioni ftp solo dalla rete 192.168.100.0:

```
# iptables -A FORWARD -d 192.168.1.110 -i eth0 -s
192.168.100.0/255.255.255.0 -p tcp --dport 21 -j ACCEPT

# iptables -A FORWARD -d 192.168.1.110 -i eth0 -s
192.168.100.0/255.255.255.0 -p tcp --dport 20 -j ACCEPT
```

Tabella mangle

- È utilizzata raramente.
- *Utilizzata quando le opzioni dei pacchetti devono essere modificate al volo* in quanto permette la modifica di vari header IP o TCP di un pacchetto.
  In particolare è usata per alterare il **TOS** (**throughput of service**).
- Esempio di modifica **TOS** per ridurre la latenza di pacchetti DNS:

```
# iptables -t mangle -A OUTPUT -p udp --dport 53
  -j TOS --set-tos Minimize-Delay
```

- Esempio di impostazione del massimo **TOS** per protocolli di trasferimento file:

```
# iptables -t mangle -A FORWARD -p tcp -m
  multiport --dport 21,20,43,88,109,110,143,873
  -j TOS --set-tos Maximize-Throughput
```

NAT

- **NAT** ovvero **Network Address Translation** permette di manipolare un **pacchetto IP modificando l'indirizzo IP sorgente o di destinazione in esso contenuto**.
- Ovviamente il dispositivo che esegue il **NAT,** quando riceve il pacchetto di ritorno, esegue l'operazione inversa, sulla base di una **tabella di natting** che mantiene in memoria.
- Il **NAT** viene utilizzato nelle occasioni dove si ha la necessità di redirezionare il traffico su un unico IP, oppure forwardare il traffico con una certa porta di destinazione ad un'altro host oppure su un'altra porta.
- Fonte: `http://openskill.info/topic.php?ID=125`

NAT

- Due modalità:
  - **SNAT**: **Source NAT**, cioè *l'alterazione dell'IP sorgente del primo pacchetto che apre la connessione.* Avviene sempre dopo che il pacchetto ha subito il routing (post-routing).
    - Esempio ne è il **masquerading**, con cui tutti gli IP sorgenti di una rete locale vengono convertiti in un unico IP sorgente con cui sono inviati in rete.
  - **DNAT**: **Destination NAT**, cioè *l'alterazione dell'IP di destinazione del primo pacchetto che apre una connessione*.
    - Esempi sono: **port-forwarding e trasparent proxy.**

Source NAT

- Utilizzato quando è necessario che **le connessioni effettuate da uno o più computer siano modificate in modo da presentare verso l'esterno uno o più indirizzi IP diversi da quelli originali**. Chi riceve le connessioni le vede provenire da un indirizzo diverso da quello utilizzato da chi effettivamente le genera.
- **SNAT** si specifica usando '`-j SNAT`'. L'opzione '`--to-source`' permette di indicare un indirizzo o un intervallo di indirizzi IP e opzionalmente una o un intervallo di porte (solo UDP e TCP).
- Es: Cambia l'indirizzo sorgente in 157.138.22.6:

```
# iptables -t nat -A POSTROUTING -o ens33 -j SNAT
  --to 157.138.22.6
```

- Es: Cambia l'indirizzo sorgente in 157.138.22.6, 157.138.22.7 oppure 157.138.22.8:

```
# iptables -t nat -A POSTROUTING -o ens33 -j SNAT
  --to 157.138.22.6-157.138.22.8
```

- Es: Cambia l'indirizzo sorgente in 157.138.22.6, porte 8000-8100

```
# iptables -t nat -A POSTROUTING -p tcp -o ens33
-j SNAT --to 157.138.22.6:8000-8100
```

Masquerading

- **Dovrebbe essere utilizzato solo se gli indirizzi IP sono assegnati dinamicamente**, come ad esempio nel caso di connessione via modem (dial up). Nel caso di indirizzi IP statici invece si usi il già citato **SNAT**.
- Non è necessario indicare l'indirizzo sorgente in quanto sarà utilizzato l'indirizzo dell'interfaccia da cui il pacchetto uscirà.
- Se il collegamento dovesse interrompersi, la connessione sarà dimenticata.
- Es: Maschera qualsiasi cosa esca da `fb0`:

```
# iptables -t nat -A POSTROUTING -o fb0
-j MASQUERADE
```

Destination NAT

- Utilizzato quando è necessario alterare le connessioni in modo da ridirezionarle verso indirizzi IP diversi da quelli originali (ad esempio trasparent proxy).
- **DNAT** si specifica usando '`-j DNAT`', l'opzione '`--to-destination`' permette di specificare un indirizzo o un intervallo di indirizzi IP e opzionalmente una o un intervallo di porte (solo per i protocolli UDP e TCP).
- Es: Cambia l'indirizzo di destinazione in 192.168.1.10:

```
# iptables -t nat -A PREROUTING -i eth1 -j
DNAT --to 192.168.1.10
```

- Es: Cambia l'indirizzo di destinazione in 192.168.1.11,192.168.1.12 oppure 192.168.1.13:

```
# iptables -t nat -A PREROUTING -i eth1 -j
DNAT -to 192.168.1.11-192.168.1.13
```

Destination NAT

- Es: Cambia l'indirizzo di destinazione del traffico web in 192.168.1.11, porta 8080.

```
# iptables -t nat -A PREROUTING -p tcp
--dport 80 -i eth1 -j DNAT --to
192.168.1.11:8080
```

- *Per alterare la destinazione dei pacchetti generati localmente si può usare la catena OUTPUT, anche se tipicamente non si fa*.
- Es: redireziona i pacchetti locali diretti all'indirizzo 192.168.1.100 verso loopback.

```
# iptables -t nat -A OUTPUT
-d 192.168.1.100 -j DNAT --to 127.0.0.1
```

Destination NAT - Redirection

- Esiste un caso speciale di **DNAT** chiamato **redirection** (redirezione).
- è semplicemente una comodità in più in quanto esattamente uguale al **DNAT** ma agisce esclusivamente sull'indirizzo associato all'interfaccia di ingresso.
- Es: invia i pacchetti diretti alla porta 80 verso un proxy (***trasparente***) **squid**:

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp \
--dport 80 -j REDIRECT --to-port 3128
```

Mignis

- Tool sviluppato dal **gruppo di sicurezza** (**Cookies**) del DAIS.
- Semplifica l'uso di **IPTABLES**: introduce delle regole semantiche per la descrizione di uno script di firewalling successivamente tradotte in regole IPTABLES da un'interprete.
- In uso al DAIS.
- `https://secgroup.dais.unive.it/teaching/security-course/mignis/`

Mignis - Esempio

- A partire da una condizione in cui **tutto ciò che non è esplicitamente permesso è vietato**, aprire le porte per un mail server:
- IPTABLES

```
$MAILSERVER = 192.168.1.100

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 25 -j ACCEPT

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 143 -j ACCEPT

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 110 -j ACCEPT

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 465 -j ACCEPT

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 993 -j ACCEPT

iptables -A FORWARD -d $MAILSERVER-p tcp --dport 995 -j ACCEPT
```

- Mignis

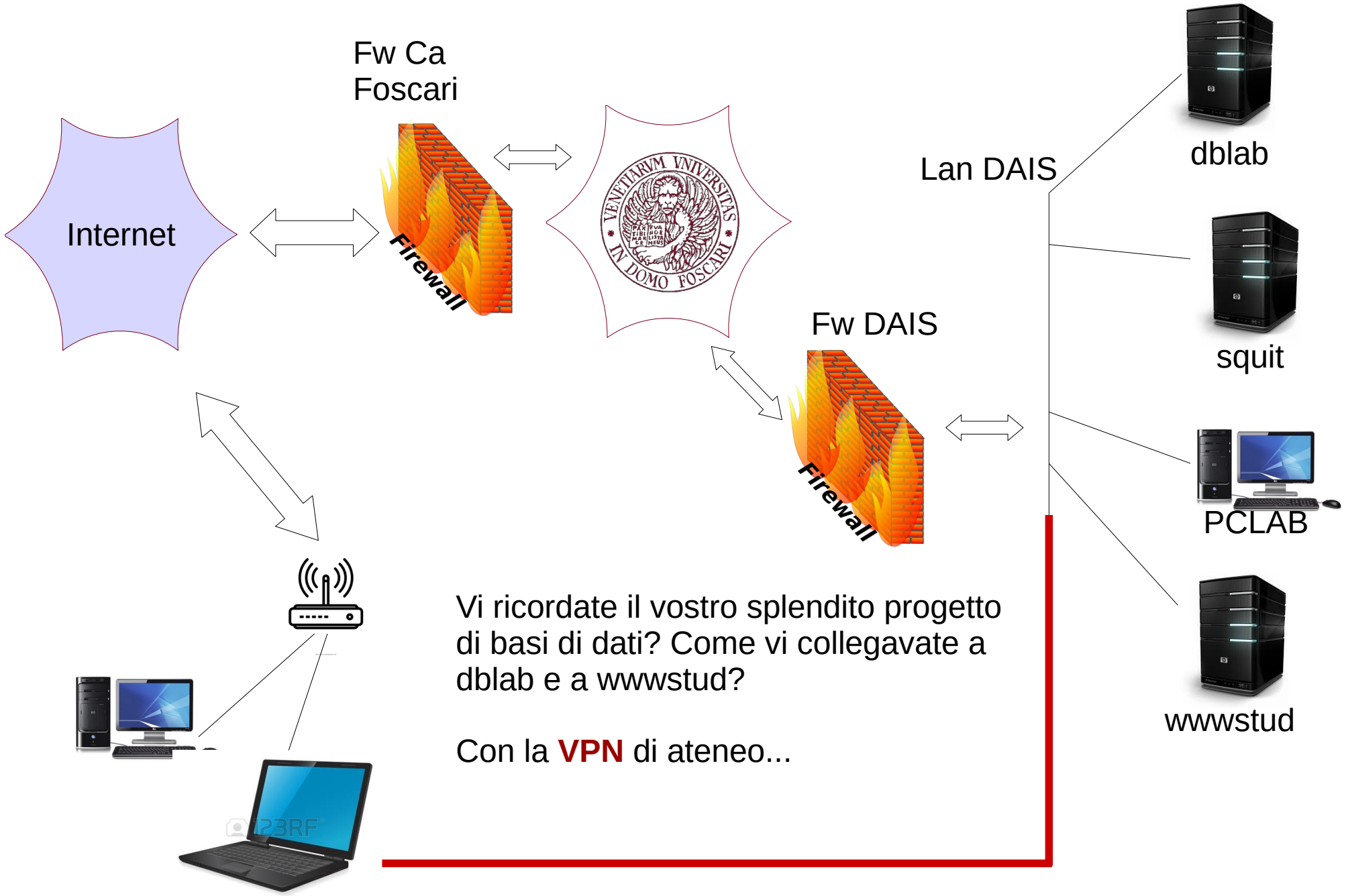
```
mailserver   192.168.1.100

* > mailserver:(25,110,143,465,993,995) tcp;    * indica tutta internet!
```

VIRTUAL PRIVATE NETWORK

- Letteralmente **Rete Privata Virtuale**.
- Pensate per permettere ai dipendenti delle aziende di accedere in **modo sicuro alle risorse all'interno dell'azienda al di fuori delle rete aziendale**.
- Una **VPN** crea un tunnel virtuale crittato tra un host esterno e un server sicuro di proprietà del fornitore del servizio VPN. Tutto il traffico passa in modo criptato dal computer al server, per poi uscire "normalmente" (quindi criptato per HTTPS e in chiaro per HTTP, FTP, e altro) sulla rete attraverso la rete aziendale.
- Cisco VPN, Open VPN, Windows VPN Server, Forti VPN, ecc...

VPN - Esempio


Fw Ca
Foscari
Internet
Firewall
VENETIARVM VNIVERSITAS
IN DOMO FOSCARI
Lan DAIS
dblab
squit
Fw DAIS
Firewall
PCLAB
wwwstud
Vi ricordate il vostro splendito progetto di basi di dati? Come vi collegavate a dblab e a wwwstud?
Con la VPN di ateneo...

Guide

- Script iptables di esempio:

https://www.cyberciti.biz/tips/linux-iptables-examples.html

- Guida veloce per realizzare un firewall con iptables:
  http://guide.debianizzati.org/index.php/Impostare_un_firewall_con_uno_script_iptables
- Documentazione su iptables:
  https://netfilter.org/documentation/

Avete un firewall!!!