Corso IFTS TST 2005-2007

Tecnico Superiore delle Telecomunicazioni

Firewall

Moreno Baricevic

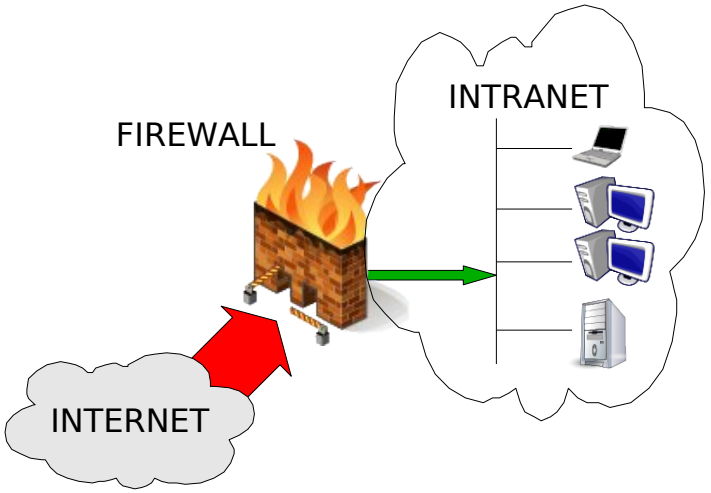
agenzia formativa
Friuli Venezia Giulia

IFTS TST
Trieste
15-16 Gennaio 2007

Panoramica sui firewall

Firewall: cos'è e a cosa serve

- NON è un *muro di fuoco*, è un *muro tagliafuoco* che ha il compito di isolare e compartimentare una struttura.



- Il firewall è l'insieme delle difese perimetrali, hardware e software, costituito da uno o più dispositivi.
- Il firewall è un elemento (ormai) fondamentale del networking il cui scopo principale è quello di proteggere una rete, frapponendosi, ad esempio, tra una rete privata *trusted* da proteggere (Intranet) e una pubblica *untrusted* (Internet).
- Un firewall controlla e limita il flusso dati verso (e da) una rete *protetta.*

Firewall: perché usarlo

- Aumentare la sicurezza della rete
  - alcuni servizi sono intrinsecamente insicuri e/o impossibili da mettere in sicurezza su ogni singolo host
  - un firewall può limitare o prevenire intrusioni e certi tipi di attacchi
- Controllo accessi/servizi
  - un firewall può aiutare a rafforzare la policy di sicurezza della rete consentendo selettivamente l'accesso a certi servizi (da tutti o certi host)
- Monitoring/Logging
  - un firewall deve esaminare tutto il traffico in ingresso o in uscita, può perciò aiutare a tracciare l'attività di rete (almeno quella che attraversa il firewall)

Firewall: limiti

- non protegge da:
  - virus, trojan, phishing, ingegneria sociale, ...
  - nuovi (sconosciuti) attacchi
  - connessioni che non lo attraversano (modem)
  - policy inefficienti (o inesistenti)
  - attacchi interni (75%-80%)
  - attacchi fisici
- può essere vulnerabile a sua volta
- non può fungere da unico punto di difesa

Classificazione/Tipi

Per funzionalità:

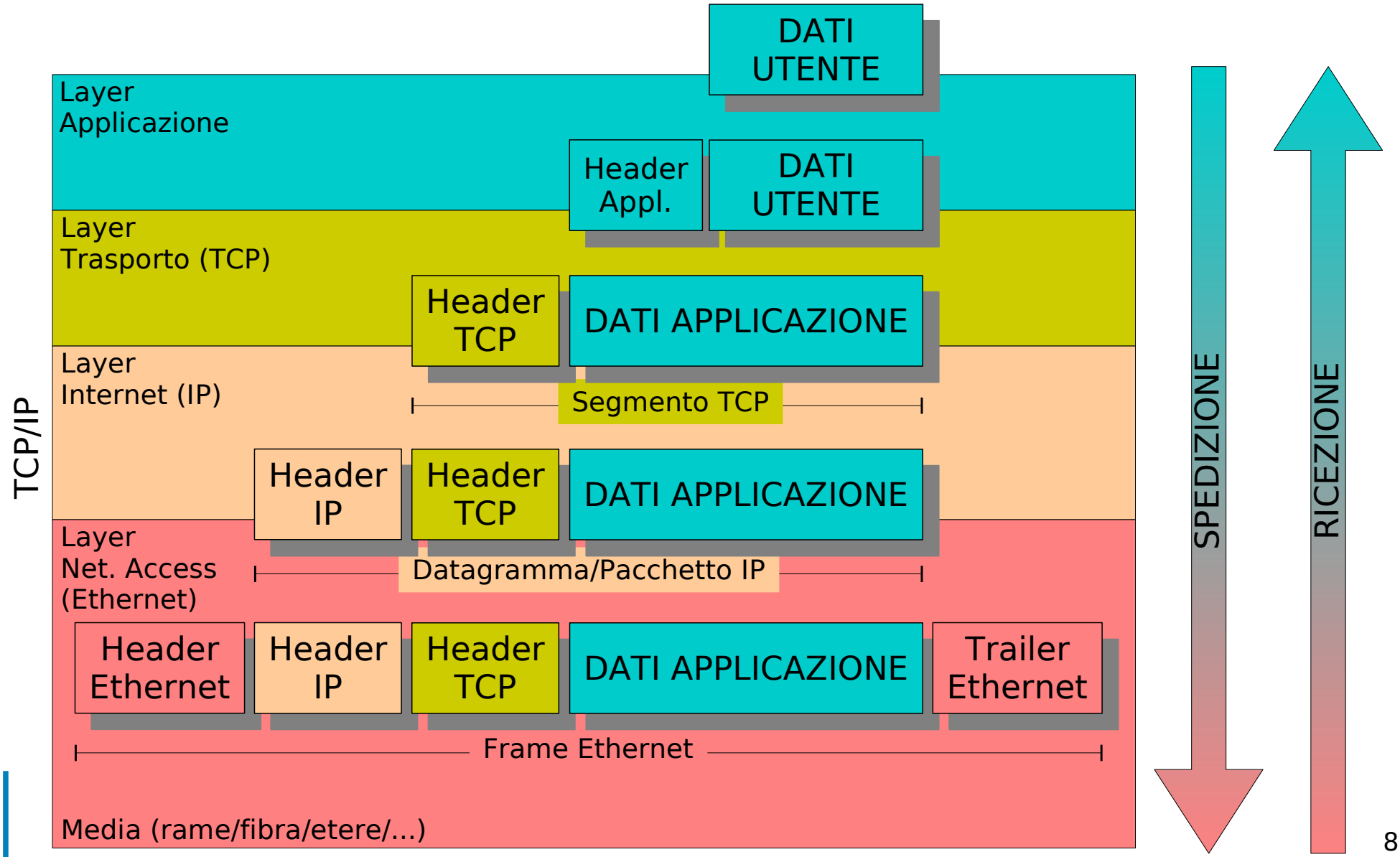
- **Packet Filtering**
  - ispeziona pacchetti nei layer OSI 3 e 4, ma non lo stato delle connessioni (modalità *stateless)*
  - semplice, veloce, richiede poche risorse; poco accurato
- **Stateful Multilayer Firewall**
  - in grado di ispezionare in modalità *stateful* il traffico che interessa i layer OSI 3, 4 (e, limitatamente, 2 e 7)
  - più efficiente e accurato del semplice packet filtering, ma richiede più risorse
- **Application Level Firewall**
  - agisce da (*transparent*) *proxy* filtrando layer applicazione
  - attivo (*application gateway*) o passivo (*protocol inspection firewall*)
  - “lento”, aggiunge overhead; filtraggio e logging molto accurati

Classificazione/Tipi 2/2

Per disposizione:

- **Appliance-Based**
  - piattaforma hardware dedicata
  - soluzione generalmente costosa
- **Host-Based**
  - applicazione per NOS
  - soluzione economica, può essere un semplice PC con OS e software adeguati (linux+netfilter)
- **Integrato**
  - dispositivo esistente con funzionalità firewall aggiuntive
  - costo del “pacchetto software” aggiuntivo

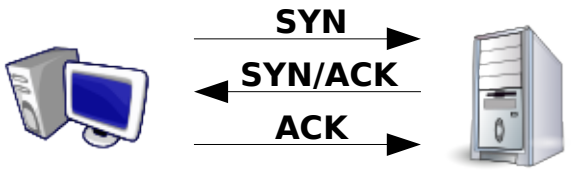
Cos'è un pacchetto ?



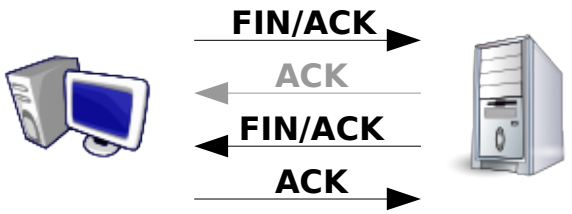
Stateless o stateful ?

CONNESSIONE TCP VERSO UNA PORTA APERTA (listening)

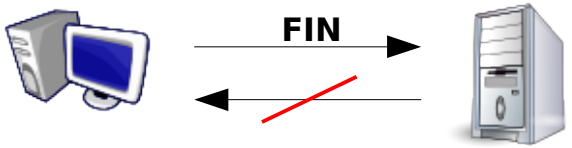
CLIENT SERVER



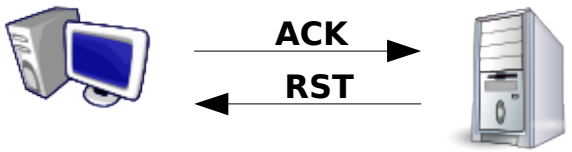
3-Way handshake
- SYN/half-open scan
- connect()



chiusura normale
(se ESTABLISHED)



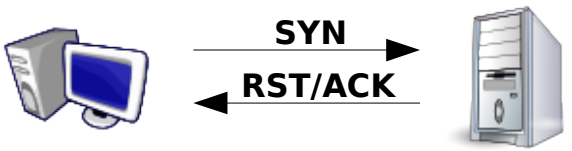
FIN scan e
Xmas Tree
(FIN/PSH/URG)

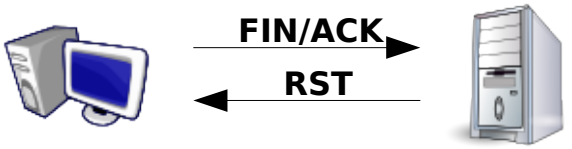


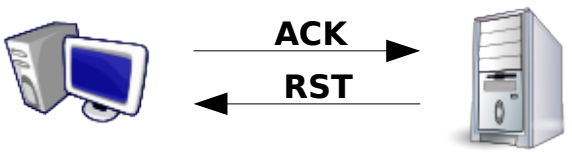
ACK scan
(stateless vs. stateful)

CONNESSIONE TCP VERSO UNA PORTA CHIUSA (non-listener)

CLIENT SERVER







RFC 793

Stateless o stateful ?

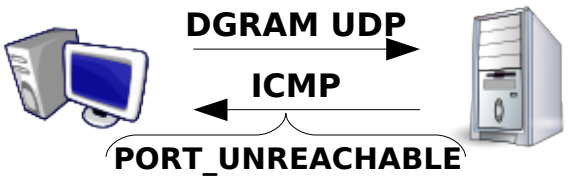
CONNESSIONE UDP VERSO UNA PORTA APERTA (listening)

CLIENT SERVER

A seconda dell'applicazione, potrebbe esserci risposta o meno

CONNESSIONE UDP VERSO UNA PORTA CHIUSA (non-listener)

CLIENT SERVER



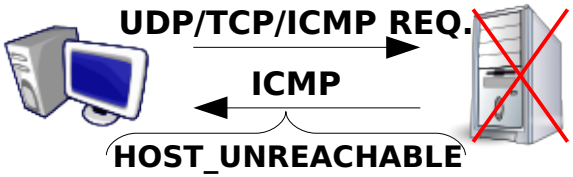
ICMP RICHIESTA/RISPOSTA

CLIENT SERVER

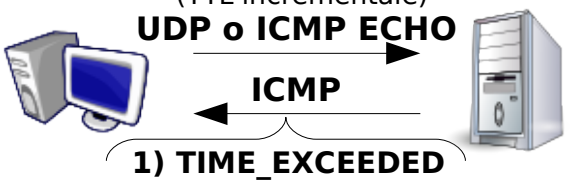
PING

ICMP NOTIFICA

CLIENT SERVER



TRACEROUTE (*)
(TTL incrementale)



2) PORT_UNREACHABLE / ECHO REPLY

RFC 768
RFC 792

(*) TCPTRACEOUTE fa lo stesso usando TCP SYN, SYN/ECN/CWR o ACK e attende SYN/ACK o RST, RST/ACK

Stateful !

- Per ogni nuova connessione viene creata una tabella che registra, per ogni pacchetto, lo stato e l'evoluzione della connessione. In particolare:
  - indirizzo sorgente, indirizzo destinazione
  - porta sorgente, porta destinazione (TCP e UDP)
  - TCP: flag di connessione (SYN, ACK, FIN), *sequencing*
  - UDP: “sessione” approssimata (direzione flusso e *idle timeout*)
  - ICMP: request/reply, ID, sequence number
- I pacchetti successivi vengono accolti solo dopo essere stati confrontati con i dati della tabella e aver verificato che siano associati ad una connessione attiva.
- In questo modo, si ottiene un'analisi più selettiva ed è possibile rendere una sessione immune a dirottamenti.

Cosa bloccare

- **IP address space** (layer OSI 3):
  - indirizzi riservati
    - reti private (rfc 1918: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)
    - indirizzi *loopback* (rfc 3330: 127.0.0.0/8)
    - *Zeroconf* (rfc 3927: 169.254.0.0/16)
    - *broadcast* e *multicast* (rfc 3171: 224.0.0.0/4)
    - indirizzi non validi e reti allocate per testing (vedi “IPv4 address space”, slide “Risorse”)
  - *spoofing* (nostro indirizzo IP dall'esterno, verifica coppie MAC/IP dei “vicini”, ...)
  - se possibile, far passare solo IP “amici” e bloccare il resto
- **Protocolli IP** dei layer OSI 3/4 che non siano TCP, UDP o ICMP.
- **TCP**: (necessario il controllo dell'header TCP)
  - pacchetti malformati, non validi o non associati a connessioni attive
  - ricognizioni/scansioni (nmap, hping, ...): ACK, NULL, FIN, Xmas, ...
  - (D)DoS: *SYN flood*, timeout connessioni
- **ICMP**: (utile per la diagnostica ma può essere usato per condurre attacchi)
  - richieste dei tipi *echo*, *timestamp*, *address mask*, ... (accettiamo risposte a nostre richieste e notifiche del tipo *destination unreachable, time exceeded, ...)*
  - (D)DoS: *Ping flood*, *smurf*, *Ping of death* (obsoleto)
- **UDP**: (difficile da controllare vista la sua natura *connectionless/unreliable)*
  - controllo approssimato su stato NEW (con timeout)
  - consentiamo lo stretto necessario (DNS, NTP) da server fidati

Osservazioni

- Il firewall è solo la prima linea di difesa.
- Un sistema è tanto sicuro quanto ...
  - ... lo sforzo per renderlo tale.
  - ... l'anello più debole della catena.
- Ogni porta aperta è una possibile vulnerabilità.
- Il packet filter non esamina il *payload* del pacchetto.
- Il packet filtering su una macchina individuale non può sostituire un “vero” firewall.
- Configurare un packet filter da remoto può non essere una buona idea... ;-)

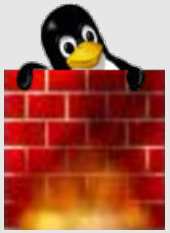
Modelli di sicurezza: approcci base

- Difesa perimetrale
  - assume che la protezione offerta dai dispositivi di frontiera sia sufficiente a bloccare un intruso, e quindi, che i sistemi interni siano al sicuro
  - non protegge da attacchi interni e, nel caso fallisca, non esiste più alcuna protezione
- Difesa in profondità
  - è il modello di difesa più robusto, ogni sistema interno deve implementare misure di sicurezza tali da renderlo indipendente dalle difese perimetrali
  - offre protezione anche da attacchi interni e rende possibile rilevare le attività dell'intruso
- *Security Through Obscurity*
  - protezioni “nascoste” (servizi su porte non standard, IP fidati, *port knocking,* ...), offrono una blanda sicurezza (generalmente temporanea) in grado di risolvere il “problema dell'orso” (anche noto come “problema della gazzella”). L'obiettivo è quello di offrire un bersaglio “più difficile” di altri, tanto da scoraggiare un attaccante poco esperto o non particolarmente motivato (*script-kiddies*), che potrebbe preferire un obiettivo più facile.
  - un bersaglio difficile o dal comportamento “bizzarro” potrebbe attrarre attaccanti “curiosi”

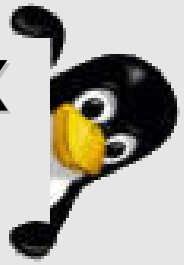
Considerazioni

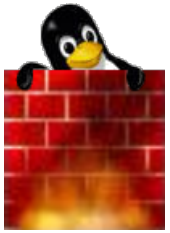
Per una policy di sicurezza accorta, andrebbero assicurate:

- SICUREZZA HARDWARE
  - accesso fisico e ridondanza
- SICUREZZA SOFTWARE
  - protezione, configurazione, aggiornamento, backup
- EDUCAZIONE UTENTI
  - Post-it con password appiccicati allo schermo o sessione lasciata aperta durante un mese di ferie possono vanificare qualunque sforzo per proteggere una macchina o un servizio... ;-)
  - prevenire/limitare ingegneria sociale



Fortificare un server LINUX con Netfilter/IPTables



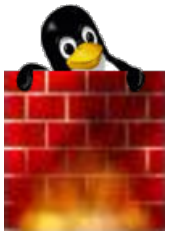


Premessa

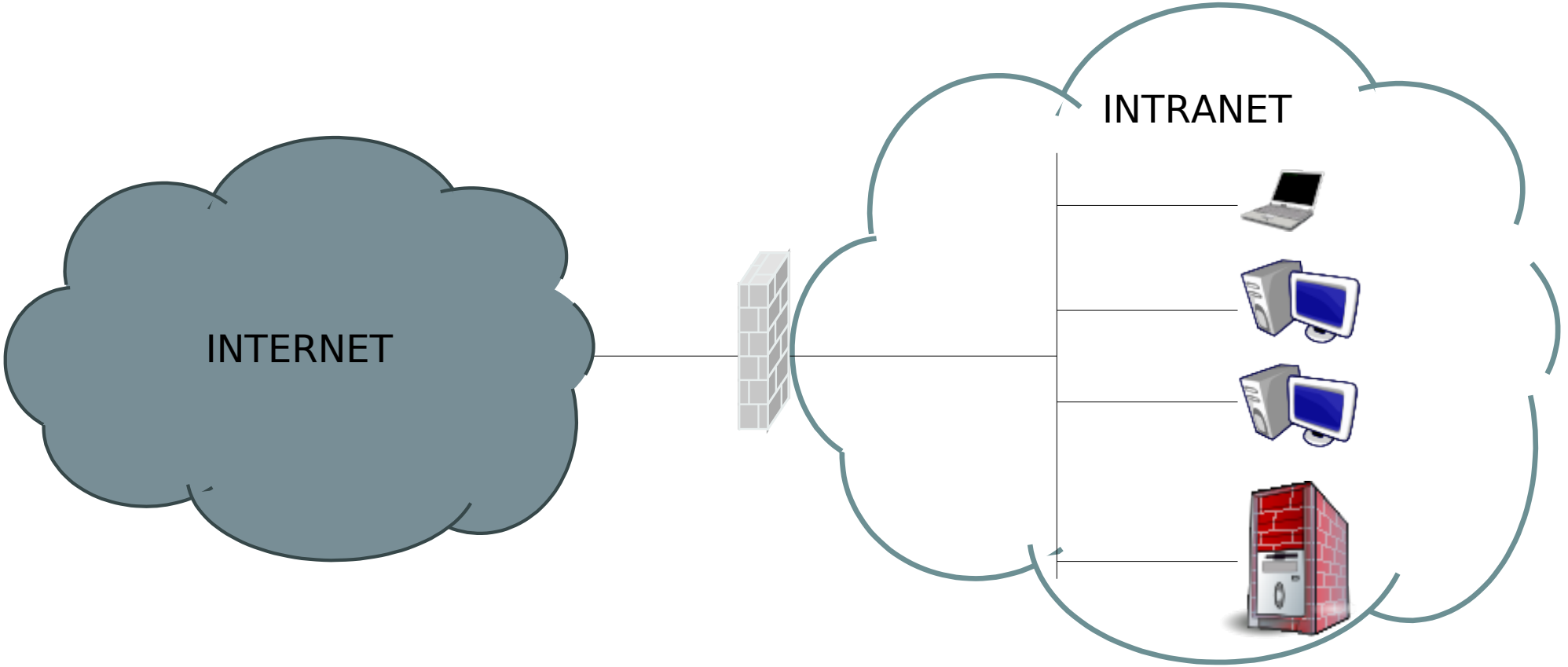
IL FIREWALL È SOLO LA PRIMA LINEA DI DIFESA,
AVERLO NON SIGNIFICA ESSERE AL SICURO

Un controllo fine dovrebbe prevedere:

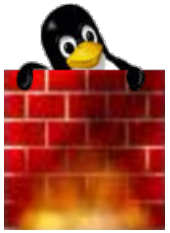
- controllo accesso fisico alla macchina e ai dispositivi di rete e cablaggi (layer OSI 1)
- *Arpwatch*, *EBTables*: controllo layer OSI 2 (MAC/IP gateway)
- *IPTables*: controllo layer OSI 2, 3, 4 (e, limitatamente, 7)
- *tcpwrapper*/[*x*]*inetd*: controllo layer OSI 3, 4 (per alcuni servizi)
- configurazione sicurezza servizi (layer OSI superiori, autenticazione)
- application proxy (proxy *SOCKS*, *Squid*) e filtri antivirus
- [HN]IDS: basati su log monitoring o prodotti come *AIDE*, *Tripwire*, *Snort*
- aggiornamento software / patch sicurezza
- **... paranoia**



Fortificare un server

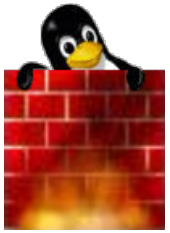


Non vogliamo che il router/gateway/firewall sia un "*single point of failure*", nella nostra implementazione assumeremo quindi che non esista e che il traffico diretto al server non sia filtrato.



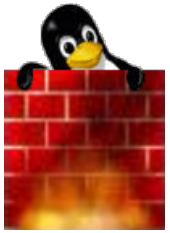
Netfilter: cos'è

- È un packet filter a livello kernel per linux.
- È un *framework* composto dal “filtro” residente in *kernel-space* e dai tool di configurazione in *user-space*.
- Accetta o rifiuta pacchetti basandosi su un set di regole.
- Esamina pacchetti confrontando sorgente, destinazione, porte e altri parametri a seconda del protocollo in esame.
- Fornisce ispezione *stateful* dei pacchetti.
- Supporta NAT/PAT, redirezione delle porte e alterazione dei pacchetti.
- Non nativamente, supporta anche filtering del layer 7.



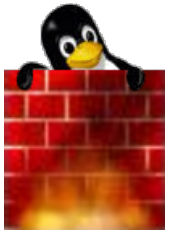
Netfilter: perché usarlo

- può bloccare rapidamente un attacco
- *TCPwrapper* protegge solo alcuni servizi
- tool per port monitoring/blocking si attivano quando la connessione è già avvenuta
- può filtrare il traffico in uscita
- può filtrare il protocollo ICMP
- ...perché non usarlo?



Netfilter: *user-space* tools

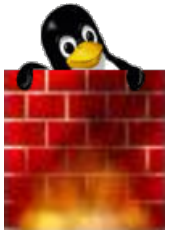
- linux 2.0 - **ipfwadm**
  - semplice (“insert”, “add”), struttura lineare: ogni regola deve essere processata finché non si ottiene un match o si raggiunge il default
  - non estensibile
  - stateless
- linux 2.2 - **ipchains**
  - più feature (QOS, “replace”, negazioni), struttura ad albero: una regola può saltare da una catena a un'altra
  - non molto più estensibile
  - ancora stateless
- linux 2.4/2.6 - **iptables**
  - molte più feature (ipv4, ipv6, *connection tracking*, NAT, PORTFW, log, ...)
  - flessibile ed estensibile (moduli)
  - stateful filtering



IPTables: Tabelle

iptables [**-t *TABELLA***] -A *CATENA* [-m *MODULO*] *REGOLA...* -j *AZIONE*

- ***filter***
  - tabella di default se “-t TABLE” non specificato
  - filtra il traffico in ingresso/uscita/transito
  - catene: INPUT, OUTPUT, FORWARD
- ***nat***
  - *Network Address Translation*
  - catene: PREROUTING, OUTPUT, POSTROUTING
- ***mangle***
  - alterazione dei pacchetti (TOS, MARK, ...)
  - catene: PREROUTING, INPUT, OUTPUT, FORWARD, POSTROUTING
- ***raw***
  - raramente utilizzata, viene invocata prima delle altre
  - configurazione, eccezioni al *connection tracking* (NOTRACK)
  - catene: PREROUTING, OUTPUT

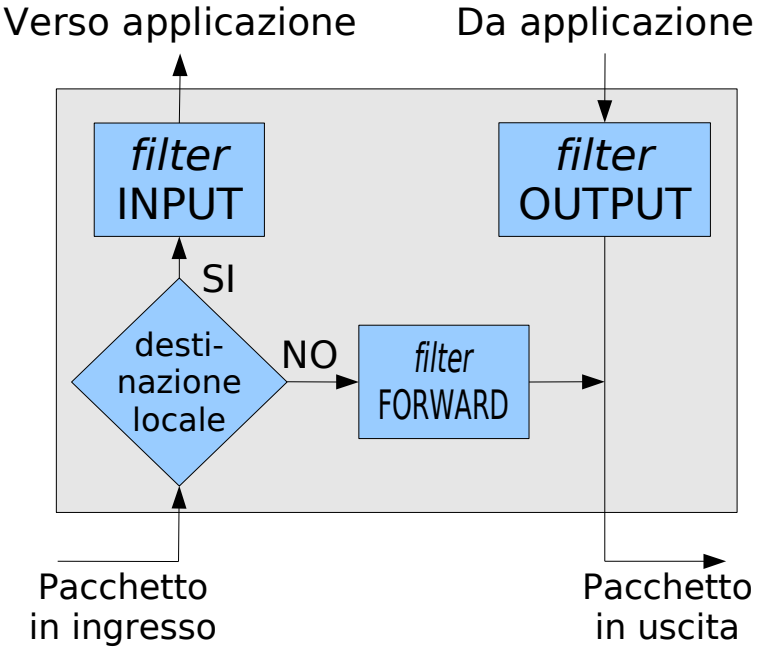


IPTables: Catene

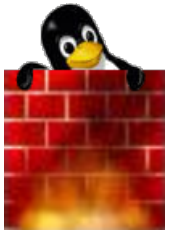
Una catena è una *ACL* costituita da una lista sequenziale di regole.

Le catene predefinite per la tabella ***filter*** sono 3:

- **INPUT**
  - processa pacchetti destinati all'host
- **OUTPUT**
  - processa pacchetti originati dall'host
- **FORWARD**
  - processa traffico in transito, instradato attraverso l'host



Altre catene possono essere create e aggiunte a piacimento.

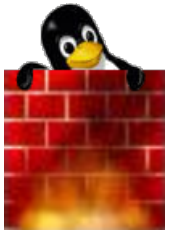


IPTables: Catene (sintassi)

Ogni catena built-in ha una policy di default che indica l'azione da compiere per i pacchetti che non matchano alcuna regola. Le policy possono essere: ACCEPT, REJECT, DROP.

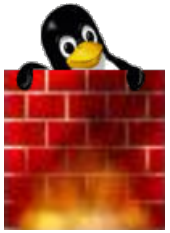
Le possibili operazioni sulle catene sono:

- **cambiare la policy di una catena**: "`iptables -P` *`CATENA POLICY`*" (`iptables -P INPUT DROP`)
- **creare una nuova catena**: "`iptables -N` *`NUOVA_CATENA`*"
- **cancellare una catena** (non built-in): "`iptables -X` *`NUOVA_CATENA`*"
- **rimuovere tutte le regole** da una catena: "`iptables -F` *`CATENA`*"
- **azzerare i contatori** associati a una catena: "`iptables -Z` *`CATENA`*"
- **rinominare** una catena: "`iptables -E` *`VECCHIO_NOME NUOVO_NOME`*"
- **lista delle regole** associate ad una catena: "`iptables -L` *`CATENA`*"



IPTables: Regole

- ogni regola definisce le caratteristiche di un pacchetto da controllare e l'azione da compiere nel caso ci sia un match. Alcuni dei parametri disponibili sono:
  - interfaccia di rete su cui transita il pacchetto
  - IP sorgente, IP destinazione
  - porte sorgente e destinazione (per TCP e UDP)
  - flag di connessione (TCP)
  - tipo e codice ICMP
  - stato della connessione
- se l'azione associata alla regola è "terminante", il match termina l'analisi del pacchetto, altrimenti l'analisi prosegue con la regola successiva. L'azione può anche essere il *jump* ad un'altra catena.



IPTables: Regole (sintassi)

- **aggiungere una regola** in fondo a una catena: “`iptables -A` *`CATENA REGOLA...`*”

  `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`

- **inserire una regola** in un punto specifico della catena: “`iptables -I` *`CATENA POSIZIONE REGOLA`*”

  `iptables -I INPUT 1 -i lo -j ACCEPT`

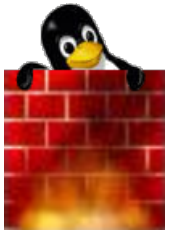
- **cancellare una regola**: “`iptables -D` *`CATENA POSIZIONE`*” o “`iptables -D` *`CATENA REGOLA`*”

  `iptables -D INPUT 2`

  `iptables -D INPUT -p tcp --dport 22 -j ACCEPT`

- **rimpiazzare una regola**: “`iptables -R` *`CATENA POSIZIONE NUOVA_REGOLA...`*”

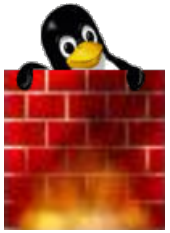
  `iptables -R INPUT 2 -p tcp --dport 22 --syn -j ACCEPT`



IPTables: Azioni

iptables [-t *TABELLA*] -A *CATENA* [-m *MODULO*] *REGOLA...* **-j *AZIONE***

- **ACCEPT** - il pacchetto viene accettato
- **REJECT** - il pacchetto viene rifiutato informando il sender
- **DROP** - il pacchetto viene ignorato
- **RETURN** - il pacchetto ritorna alla catena originale
- L'azione può essere una qualunque altra catena o una funzione particolare come “LOG/ULOG/TARPIT/...” (*filter*), “MARK/...” (*mangle*), “DNAT/SNAT/REDIRECT/...” (*nat*), “NOTRACK/...” (*raw*).
- Come nel caso di LOG, un'azione può essere “non terminante”, l'analisi continua con la regola successiva.

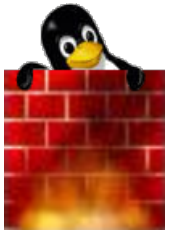


IPTables: Parametri (sintassi) 1/2

iptables [-t *TABELLA*] -A *CATENA* [-m *MODULO*] ***REGOLA...*** -j *AZIONE*

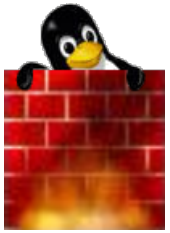
- "**-p [!] *PROTOCOLLO***" indica il protocollo (`-p tcp`, `-p 6`).
- "**-s [!] *IP_SORGENTE*[/*NETMASK*]**" specifica l'indirizzo (o il blocco di indirizzi) della sorgente (`-s 10.0.0.1`, `-s 10.0.0.0/255.0.0.0`, `-s 10.0.0.0/8`).
- "**-d [!] *IP_DESTINAZIONE*[/*NETMASK*]**" specifica l'indirizzo (o il blocco di indirizzi) della destinazione (`-d 10.0.0.1`, `...`)
- "**-i [!] *INTERFACCIA***" specifica l'interfaccia di rete dalla quale entra il pacchetto (`-i lo`, `-i eth0`, `-i eth1`, `...`)
- "**-o [!] *INTERFACCIA***" specifica l'interfaccia di rete dalla quale esce il pacchetto (`-o lo`, `...`)
- "**-m *MODULO***": carica il modulo di estensione (`-m tcp`, `-m state`, `-m recent`, `...`)

Nota: il '**!**', opzionale, indica la negazione "tutti tranne ...".



IPTables: Parametri (sintassi) 2/2

- Per ogni protocollo esistono parametri particolari, per TCP e UDP, ad esempio, possiamo usare:
  - "**--sport [!] *PORTA***" per indicare la porta sorgente
  - "**--dport [!] *PORTA***" per indicare la porta di destinazione
- Per il protocollo TCP, caricando il modulo opportuno (`-m tcp`), è possibile controllare i flag di connessione:
  - "**[!] --syn**" indica che il flag SYN e settato mentre FIN, RST e ACK non lo sono (richiesta nuova connessione)
  - "**--tcp-flags [!] *MASCHERA FLAG*[,...]**" consente di controllare i singoli bit: SYN, ACK, RST, FIN, URG, PSH. Il comando "`--tcp-flags FIN,SYN,RST,ACK SYN`" equivale a "`--syn`".
- Per il protocollo ICMP (`-m icmp`), è possibile controllare il tipo e il codice del messaggio (`--icmp-type echo-reply`, `--icmp-type time-exceeded`, `--icmp-type destination-unreachable`, ...).
- Altri moduli abilitano altre opzioni per match più accurati su parametri aggiuntivi.

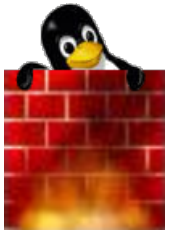


IPTables: Moduli

iptables [-t *TABELLA*] -A *CATENA* [**-m *MODULO***] *REGOLA...* -j *AZIONE*

Tra i vari moduli disponibili, alcuni esempi degni di nota sono:

- **tcp**, **udp**, **icmp**, **mac**: abilitano opzioni aggiuntive per relativi protocolli (`-m mac --mac-source 00:11:22:33:44:55 -j ACCEPT`)
- **state**: abilita controllo sul connection tracking
- **limit**: limita il numero di pacchetti per unità di tempo che matchano la regola (`-p icmp --icmp-type echo-request -m limit --limit 3/s -J ACCEPT`)
- **recent**: consente di creare una lista dinamica di IP per tracciare, ed eventualmente limitare, in maniera intelligente il numero di connessioni da un host
- **multiport**: consente di definire più porte (`--dports 80,443`) o sorgente e destinazione con un unica opzione (`--ports 123`)
- **owner**: permette di controllare, solo nella catena OUTPUT, l'owner del pacchetto (`-m tcp --syn --owner --uid-owner webuser -j DROP`)



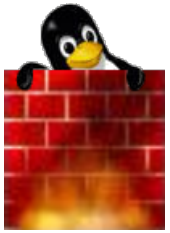
IPTables: *connection tracking*

iptables *[...]* **-m state --state *STATO***

- **NEW**: il pacchetto inizia una nuova connessione
- **ESTABLISHED**: il pacchetto è associato a una connessione esistente
- **RELATED**: il pacchetto inizia una nuova connessione ma è in relazione a una connessione già esistente (*ftp-data* quando una sessione *ftp-control* è attiva, notifiche ICMP)
- **INVALID**: il pacchetto non appartiene ad una connessione esistente

Esempio:

```
iptables -A INPUT -p tcp -m tcp ! --syn -m state --state NEW -j DROP
iptables -A INPUT -m state --state INVALID -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```



/proc filesystem

- disattivazione IP forwarding

  echo "**0**" > /proc/sys/net/ipv4/**ip_forward**

- protezione da SYN flood (DoS, DDoS)

  echo "**1**" > /proc/sys/net/ipv4/**tcp_syncookies**

- logging e protezione da spoofing, redirect, broadcast (DDoS, smurf)

  echo "**1**" > /proc/sys/net/ipv4/**icmp_ignore_bogus_error_responses**

  echo "**1**" > /proc/sys/net/ipv4/**icmp_echo_ignore_all**

  echo "**1**" > /proc/sys/net/ipv4/**icmp_echo_ignore_broadcasts**

  echo "**100**" > /proc/sys/net/ipv4/**icmp_ratelimit**

  echo "**0**" > /proc/sys/net/ipv4/conf/all/**accept_redirects**

  echo "**0**" > /proc/sys/net/ipv4/conf/all/**accept_source_route**

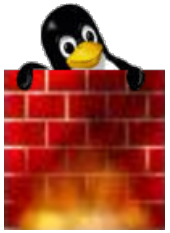
  echo "**0**" > /proc/sys/net/ipv4/conf/all/**secure_redirects**

  echo "**1**" > /proc/sys/net/ipv4/conf/all/**rp_filter**

  echo "**1**" > /proc/sys/net/ipv4/conf/all/**log_martians**

Per ulteriori informazioni:

/usr/src/linux/Documentation/networking/ip-sysctl.txt
http://cr.yp.to/syncookies.html
http://securityfocus.com/infocus/1711



Configurazione

- command line (slide precedenti)

```
iptables -A ...
```

- script

```
#!/bin/bash
MIO_IP=192.0.2.1
RETE_LOCALE=192.0.2.0/24
iptables -A ... -s $RETE_LOCALE -d $MIO_IP ...
```

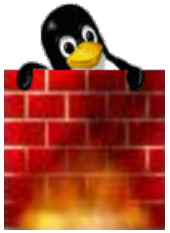
- file di configurazione: il contenuto del file è costituito dalle cmdline senza “iptables” davanti (prossime slide)

```
iptables-restore < plain-text-file
iptables-save > plain-text-file
```

- */proc* filesystem
  - sysctl

```
sysctl net.ipv4.ip_forward
sysctl -w net.ipv4.ip_forward=1
sysctl -p /etc/sysctl.conf
```

  - echo “*VALORE*” > /proc/sys/net/ipv4/ip_forward
  - cat /proc/sys/net/ipv4/ip_forward

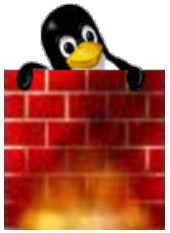


Cosa filtrare

- **IP address space**: ignoriamo indirizzi non validi, indirizzi riservati, broadcast, multicast. `-A INPUT -j indirizzi`
- **Protocolli IP**: accettiamo TCP, UDP e ICMP, ignoriamo il resto. `-A INPUT -j protocolli`
- **TCP**: accettiamo SYN per servizi offerti pubblicamente e accettiamo sessioni già stabilite, ignoriamo il resto.
- **ICMP**: accettiamo *destination unreachable*, *time exceeded* e risposte. Accettiamo un numero limitato di *echo request*. Ignoriamo tutto il resto.
- **UDP**: accettiamo solo DNS e NTP da server fidati, ignoriamo il resto.

`-A INPUT -j protezione`
`-A INPUT -j servizi`

Nota: se il *connection tracking* è abilitato (-m state/conntrack), non serve filtrare i frammenti (-f), vengono già riassemblati.



Cosa filtrare: comandi

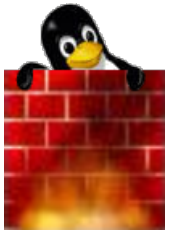

```
-A INPUT -i lo -j ACCEPT
-A INPUT -j indirizzi
-A INPUT -j protocolli
-A INPUT -j protezione
-A INPUT -j servizi
-A INPUT -j DROP
```

```
-A indirizzi -j bad_things
-A indirizzi -j riservati
-A indirizzi -m pkttype --pkt-type broadcast -j DROP
-A indirizzi -m pkttype --pkt-type multicast -j DROP
-A indirizzi -d ! MIO_IP -j DROP

  -A bad_things -i ! lo -s MIO_IP -j DROP
  -A bad_things -i ! lo -s 127.0.0.0/8 -j DROP

  -A riservati -s 10.0.0.0/8     -j DROP
  -A riservati -s 172.16.0.0/12  -j DROP
  -A riservati -s 192.168.0.0/16 -j DROP
  -A riservati -s ...

-A protocolli -p tcp -j RETURN
-A protocolli -p udp -j RETURN
-A protocolli -p icmp -j RETURN
-A protocolli -j DROP
```



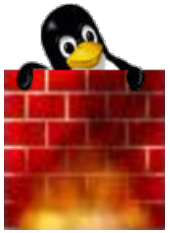
Cosa filtrare: comandi


```
-A INPUT -i lo -j ACCEPT
-A INPUT -j indirizzi
-A INPUT -j protocolli
-A INPUT -j protezione
-A INPUT -j servizi
-A INPUT -j DROP
```

```
-A protezione -p icmp -j ICMP
-A protezione -p tcp -m tcp --tcp-flags ALL NONE -j DROP
-A protezione -p tcp -m tcp ! --syn -m state --state NEW -j DROP
-A protezione -p tcp -m tcp -m state --state INVALID -j DROP
-A protezione -p tcp -m tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
-A protezione -p tcp -m tcp --syn -j syn_flood
```

```
-A syn_flood -m limit --limit 3/s --limit-burst 10 -m recent --set --name SF --rsource -j RETURN
-A syn_flood -m limit --limit 5/m --limit-burst 60 -j LOG --log-prefix "iptables: synflood: "
-A syn_flood -m limit --limit 5/m --limit-burst 20 -j ULOG
-A syn_flood -j DROP
```

```
-A ICMP -p icmp -m pkttype --pkt-type broadcast -j DROP
-A ICMP -p icmp -m icmp --icmp-type echo-request -m length --length 128:65535 -j DROP
-A ICMP -p icmp -m icmp --icmp-type echo-request -s RETE_LOCALE -j ACCEPT
-A ICMP -p icmp -m icmp --icmp-type echo-request -m state --state NEW -m limit --limit 3/s -j ACCEPT
-A ICMP -p icmp -m icmp --icmp-type destination-unreachable -j ACCEPT
-A ICMP -p icmp -m icmp --icmp-type time-exceeded -j ACCEPT
-A ICMP -p icmp -m icmp --icmp-type timestamp-reply -j ACCEPT
-A ICMP -p icmp -m icmp --icmp-type echo-reply -j ACCEPT
-A ICMP -p icmp -j DROP
```



Cosa filtrare: comandi

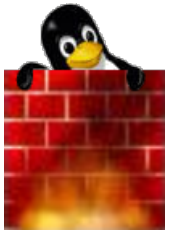

```
-A INPUT -i lo -j ACCEPT
-A INPUT -j indirizzi
-A INPUT -j protocolli
-A INPUT -j protezione
-A INPUT -j servizi
-A INPUT -j DROP
```

```
-A servizi -p tcp -m tcp --syn --dport 22 -j amici
-A servizi -p tcp -m tcp --syn -m multiport --dports 80,443 -j ACCEPT

-A servizi -p tcp -m tcp --dport 113 -j REJECT       \
   --reject-with icmp-port-unreachable
-A servizi -p udp -m udp --dport 113 -j REJECT       \
   --reject-with icmp-port-unreachable

-A servizi -p udp -m udp -s DNS_PRIMARIO --sport 53 -d MIO_IP -j ACCEPT
-A servizi -p udp -m udp -s DNS_SECONDARIO --sport 53 -d MIO_IP -j ACCEPT
-A servizi -p udp -m udp -s RETE_LOCALE --sport 123 --dport 123 -j ACCEPT
-A servizi -p udp -m udp -j DROP
```

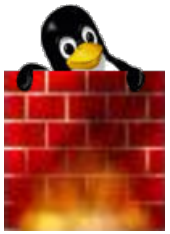
```
  -A amici -m mac --mac-source 00:11:22:33:44:55 -s IP_AMICO -j ACCEPT
  -A amici -j DROP
```



Filtrare l'OUTPUT ?

- A livello firewall non possiamo controllare né proibire il *binding* delle porte, non possiamo infatti decidere quale utente può aprire e utilizzare quale porta (*grsec*, ad esempio, potrebbe farlo), possiamo però bloccare i pacchetti di un certo utente che "tentano" di uscire da una specifica porta (o da più porte o persino da qualunque porta).
- Effettuando questo genere di controllo possiamo evitare che una porta non filtrata dal firewall, perché appartenente ad un servizio offerto, possa essere (ri)utilizzata da un utente quando il servizio non sta girando (o dopo che sia stato abbattuto).
- Evitiamo che un *demone*, legittimamente in ascolto e che dovrebbe solo rispondere a delle richieste, possa aprire nuove connessioni (*code injection* che tenta il download di un *exploit*).

Nota: il controllo sullo *user ID* non funziona nel caso di programmi con *setuid*, come ping e traceroute.



Filtrare l'OUTPUT: comandi

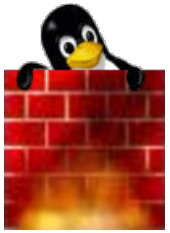
```
-A indirizzi_out -j riservati_out
-A indirizzi_out -m pkttype --pkt-type broadcast -j REJECT
-A indirizzi_out -m pkttype --pkt-type multicast -j REJECT
```

```
  -A riservati_out -d 0.0.0.0/8 -j REJECT
  -A riservati_out -d 10.0.0.0/8 -j REJECT
  -A riservati_out -d 127.0.0.0/8 -j REJECT
  -A riservati_out -d 172.16.0.0/12 -j REJECT
  -A riservati_out -d 192.168.0.0/16 -j REJECT
  -A riservati_out -d 224.0.0.0/4 -j REJECT
  -A riservati_out -d 240.0.0.0/4 -j REJECT
  -A riservati_out -d ...
```

```
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -s ! MIO_IP -j REJECT
-A OUTPUT -j indirizzi_out
-A OUTPUT -j protocolli
-A OUTPUT -j utenti
-A OUTPUT -j ACCEPT
```

```
-A protocolli -p tcp -j RETURN
-A protocolli -p udp -j RETURN
-A protocolli -p icmp -j RETURN
-A protocolli -j DROP
```

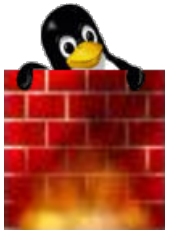
```
-A utenti -p udp -m udp -d DNS_PRIMARIO --dport 53 -j ACCEPT
-A utenti -p udp -m udp -d DNS_SECONDARIO --dport 53 -j ACCEPT
-A utenti -p udp -m udp -m owner --uid-owner root --dport 123 -j ACCEPT
-A utenti -p udp -m udp -m owner --uid-owner ntpuser --sport 123 --dport 123  \
   -j ACCEPT
-A utenti -p udp -j REJECT
-A utenti -p tcp -m state --state INVALID -j REJECT
-A utenti -p tcp -m owner --uid-owner webuser -m state --state NEW -j REJECT
-A utenti -p tcp -m tcp -m multiport --sports 80,443 -m owner  \
   ! --uid-owner webuser -j REJECT
```



Logging

Decidere cosa loggare è complesso, bisogna evitare che la macchina attaccata possa crollare sotto il peso dei log (dischi pieni e tempo CPU occupato a scrivere i log). Sarebbe bello poter loggare ogni singolo pacchetto, ma di solito è ragionevole loggare solo i pacchetti a cui si è negato l'accesso (usando opportunamente "*-m limit*" per evitare un auto-DoS...).

- Le azioni che consentono il logging sono:
  - **LOG** (*kernel-space logging*): consente di definire il *log-level* (*debug*, *warn*, *alert*, ...), *log-prefix* (commento), logging delle opzioni TCP e IP, del *sequencing* e dello *userid* (solo in OUTPUT). È possibile accedere ai log con *dmesg* o *syslogd*.
  - **ULOG** (*user-space logging*): è possibile accedere ai log con *ulogd*. Utilizzando l'estensione *pcap*, è possibile loggare l'intero pacchetto per una successiva analisi (tcpdump, ethereal). Come curiosità, usando *p0f* è possibile azzardare un *passive OS fingerprinting* sui pacchetti catturati.



Logging: comandi

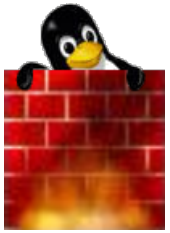
Utilizzando le catene qui riportate, basta rimpiazzare opportunamente nelle regole precedenti i vari "*-j DROP/REJECT*" con "*-j logndrop/lognreject*".

INPUT:

```
-A logndrop -m unclean -j LOG --log-level debug                \
   --log-prefix "iptables: unclean packet: "                   \
   --log-tcp-sequence --log-tcp-options --log-ip-options
-A logndrop -m limit --limit 5/m --limit-burst 60 -j LOG       \
   --log-level debug --log-prefix "iptables: "
-A logndrop -j UDROP
-A UDROP -m limit --limit 5/m --limit-burst 30 -j ULOG
-A UDROP -j DROP
```

OUTPUT:

```
-A lognreject -m unclean -j LOG --log-level debug              \
   --log-prefix "iptables: unclean: " --log-tcp-sequence       \
   --log-tcp-options --log-ip-options --log-uid
-A lognreject -m limit --limit 5/m --limit-burst 60 -j LOG     \
   --log-level debug --log-prefix "iptables: outdrop: "        \
   --log-uid
-A lognreject -j REJECT --reject-with icmp-admin-prohibited
```



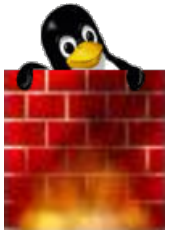
Verifica: tools

VERIFICA ATTIVA: port-scanner, utility di diagnostica, applicazioni. L'obiettivo è simulare un attacco (port-scan, ping-sweep, ...) o semplicemente stimolare le regole del firewall.

- nmap, hping, netcat, nessus
- ping, traceroute, tcptraceroute
- telnet, client ssh, browser, ...

VERIFICA PASSIVA: sniffing, diagnostica, logging. Lo scopo è controllare il traffico con l'obiettivo di verificare la risposta del firewall.

- tcpdump, wireshark/ethereal, iptraf, ettercap, dsniff
- iptstate, ss
- lsof, netstat, fuser
- syslogd, ulogd



Verifica: TCP/UDP

Alcuni esempi per verificare le regole riguardanti i protocolli TCP e UDP:

- usando **nmap**

```
nmap [-sS|-sA|-sN|-sF|-sX|-sT] -p 21,22,23,25,80,443 -P0 INDIRIZZO
nmap -sU -sS -p T:21-23,25,80,443,U:53,123 INDIRIZZO -g 80
```

- usando **hping**

```
hping -c 1 [-S,-A,-F,-R,-U,-P,-X,-Y] -p 80 -V INDIRIZZO
hping --udp -c 1 -p 123 -V INDIRIZZO -s 123 [--spoof INDIRIZZO_NTPSERVER]
hping -A --scan 22,80,443 -V INDIRIZZO -s 80
```

- usando **nc (netcat)**

```
nc -z -vv INDIRIZZO 21 22 23 25 80 443
nc -z -vv -u INDIRIZZO 53 123
nc -v -u INDIRIZZO 53
```

  per aprire una porta in ascolto:

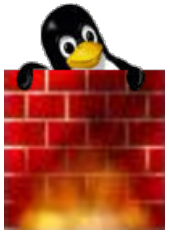
```
echo ciao | nc -v -l -u -p 53 INDIRIZZO_ASCOLTO
nc -v -l -p 80 INDIRIZZO_ASCOLTO
```

- usando **telnet**

```
telnet INDIRIZZO PORTA
```

- usando (**tcp**)**traceroute**

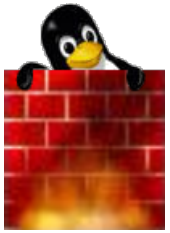
```
(UDP) traceroute INDIRIZZO
(TCP) tcptraceroute [-S|-A] INDIRIZZO [PORTA]
```



Verifica: ICMP

Alcuni esempi per verificare le regole riguardanti il protocollo ICMP:
(vedi /usr/include/linux/icmp.h per dettagli su tipi e codici ICMP)

- *echo request* (ping)
  ```
  ping INDIRIZZO
  traceroute -I INDIRIZZO
  ```
- ping flood
  ```
  ping -f INDIRIZZO
  hping --icmp --icmptype 8 -c 100 --faster INDIRIZZO
  ```
- *timestamp request*
  ```
  hping --icmp --icmptype 13 -c 2 INDIRIZZO
  ```
- *destination unreachable - host unreachable*
  ```
  hping --icmp --icmptype 3 --icmpcode 1 -c 1 INDIRIZZO
  ```
- *destination unreachable - administratively prohibited*
  ```
  hping --icmp --icmptype 3 --icmpcode 13 -c 1 INDIRIZZO
  ```
- ICMP in sovrappeso
  ```
  ping -c 1 -s 65507 INDIRIZZO
  hping --icmp --icmptype 8 -d 65000 -c 1 INDIRIZZO
  ```



Verifica: monitoraggio 3/3

Alcuni comandi per verificare lo stato delle connessioni e il traffico:

- counters e lista regole di iptables:
  `iptables -nvL [`*`CATENA`*`]`
- stato *connection tracking*:
  `iptstate [-1]`
- connessioni e risorse:
  ```
  lsof -i TCP[:PORTA] -n -P
  netstat -putan
  fuser -v -n tcp 22 25 80 443
  ss -4 -n -l
  ```
- sniffing del traffico:
  ```
  tcpdump -i INTERFACCIA -nn [-v] \
      [proto PROTOCOLLO and port PORTA and host INDIRIZZO ...]
  {tshark|tethereal} -i INTERFACCIA -np [-V] \
      [proto PROTOCOLLO and port PORTA and host INDIRIZZO ...]
  {wireshark|ethereal} &
  iptraf -i INTERFACCIA
  ```
- controllo log del kernel (iptables -j LOG):
  ```
  dmesg
  tail -f /var/log/iptables.log (configurando opportunamente syslogd)
  ```

That's All Folks!


```
( questions ; comments ) | mail -s uheilaaa baro@democritos.it

( complaints ; insults ) &>/dev/null
```

RISORSE E LINK UTILI

SOFTWARE:

- Linux Kernel http://www.kernel.org
- Netfilter http://www.netfilter.org

- nmap http://www.insecure.org/nmap/
- hping http://www.hping.org/
- netcat http://netcat.sourceforge.net/
- iptstate http://www.phildev.net/iptstate/
- ss http://linux-net.osdl.org/index.php/Iproute2
- lsof ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/
- netstat http://www.tazenda.demon.co.uk/phil/net-tools/
- tcpdump http://www.tcpdump.org
- wireshark http://www.wireshark.org
- ethereal http://www.ethereal.com (vedi wireshark)
- iptraf http://iptraf.seul.org/
- ettercap http://ettercap.sourceforge.net
- dsniff http://www.monkey.org/~dugsong/dsniff/
- tcptraceroute http://michael.toren.net/code/tcptraceroute/
- (telnet, traceroute, ping, ...)

DOCUMENTAZIONE:

- IPTables HOWTO http://www.netfilter.org/documentation/HOWTO/
- IPTables tutorial http://iptables-tutorial.frozentux.net/
- Having fun with IPTables
  http://www.ex-parrot.com/~pete/upside-down-ternet.html
- Denial of Service http://www.cert.org/tech_tips/denial_of_service.html
- IPv4 Address space
  - http://www.cymru.com/Documents/bogon-bn.html
  - http://www.iana.org/assignments/ipv4-address-space
  - http://www.oav.net/mirrors/cidr.html
  - http://en.wikipedia.org/wiki/IPv4
  - IANA http://www.iana.org
  - RIPE http://www.ripe.net
  - RFC 3330 http://www.rfc.net/rfc3330.html
- SANS: http://www.sans.org/reading_room/whitepapers/firewalls/
  http://www.sans.org/reading_room/

RFC: (http://www.rfc.net)

- RFC 791 – Internet Protocol (IPv4)
  http://www.rfc.net/rfc791.html
- RFC 793 – Transmission Control Protocol (TCP)
  http://www.rfc.net/rfc793.html
- RFC 768 – User Datagram Protocol (UDP)
  http://www.rfc.net/rfc768.html
- RFC 792 – Internet Control Message Protocol (ICMP)
  http://www.rfc.net/rfc792.html
- RFC 1180 – A TCP/IP Tutorial
  http://www.rfc.net/rfc1180.html
- RFC 1700 / IANA db – Assigned Numbers
  http://www.rfc.net/rfc1700.html
  http://www.iana.org/numbers.html
- RFC 3330 – Special-Use IPv4 Addresses
  http://www.rfc.net/rfc3330.html
- RFC 1918 – Address Allocation for Private Internets
  http://www.rfc.net/rfc1918.html
- RFC 2196 – Site Security Handbook
  http://www.rfc.net/rfc2196.html
- RFC 2827 – Network Ingress Filtering
  http://www.rfc.net/rfc2827.html
- RFC 2828 – Internet Security Glossary
  http://www.rfc.net/rfc2828.html
- RFC 1149 – Transmission of IP Datagrams on Avian Carriers
  http://www.rfc.net/rfc1149.html
- Unofficial CPIP WG
  http://www.blug.linux.no/rfc1149/
- RFC 2549 – IP over Avian Carriers with Quality of Service
  http://www.rfc.net/rfc2549.html
- Firewalling the CPIP
  http://www.tibonia.net/
  http://www.hotink.com/wacky/dastrdly/

Acronimi, servizi, varie ed eventuali...

DEMOCRITOS – Democritos Modeling Center for Research In aTOmistic Simulations

IP – Internet Protocol
TCP – Transmission Control Protocol
UDP – User Datagram Protocol
ICMP – Internet Control Message Protocol
ARP – Address Resolution Protocol
MAC – Media Access Control

OS – Operating System
NOS – Network Operating System
LINUX – LINUX is not UNIX

PING – Packet Internet Groper

IDS – Intrusion Detection System
HIDS – Host-based IDS
NIDS – Network-based IDS
AIDE – Advanced Intrusion Detection Environment

DoS – Denial Of Services
DDoS – Distributed Denial Of Services

SSH – Secure SHell – (TCP/22)
DNS – Domain Name System – (UDP/53)
NTP – Network Time Protocol – (UDP/123)
FTP – File Transfer Protocol – (TCP/21,20)
HTTP – HyperText Transfer Protocol – (TCP/80)
HTTPS – HyperText Transfer Protocol over TLS/SSL – (TCP/443)

TLS – Transport Layer Security
SSL – Secure Sockets Layer

RFC – Request For Comments

ACL – Access Control List

PDU – Protocol Data Unit

TCP flags:
- **URG**: Urgent Pointer field significant
- **ACK**: Acknowledgment field significant
- **PSH**: Push Function
- **RST**: Reset the connection
- **SYN**: Synchronize sequence numbers
- **FIN**: No more data from sender

RFC 3168 TCP flags:
- **ECN**: Explicit Congestion Notification
- (**ECE**: ECN Echo)
- **CWR**: Congestion Window Reduced

ISN – Initial Sequence Number

"The bear problem" is when you're hiking in the woods and start getting chased by a bear, you don't have to be able to outrun the bear. You just have to be able to outrun at least one of the guys you're with. The moral is if you make yourself a more difficult target, the attackers will move on to easier pickings.