ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di
Sicurezza Informatica

Firewall

Marco Prandini

Dipartimento di Informatica – Scienza e Ingegneria

Credits

- Materiale in parte tratto da presentazioni di
  - Henric Johnson
    Blekinge Institute of Technology, Svezia
    http://www.its.bth.se/staff/hjo/
  - Angelo Neri
    CINECA, Italia
  - Fabio Bucciarelli
    ex-DEIS, Università di Bologna, Italia

Firewall = difesa perimetrale

- Dall'inglese "muro tagliafuoco"
  - Un dispositivo per *limitare* la propagazione di un fenomeno indesiderato
- Immagine migliore: una cinta muraria con una porta
  - Divide il "dentro" dal "fuori"
    - Quel che avviene "dentro" non è visibile né controllabile
  - Si passa solo dalla porta
    - Politiche centralizzate di controllo dell'accesso
    - Funzionalità sofisticate implementate in un punto unico → non è necessario implementarle in tutti i sistemi
  - La porta serve per entrare, ma anche per uscire
    - *INGRESS* filtering, più intuitivo per impedire l'accesso a malintenzionati
    - *EGRESS* filtering, altrettanto importante, per impedire l'esfiltrazione di dati riservati e per evitare che i propri sistemi siano usati come base per attaccarne altri

Principi di base

- Firewall = *architettura*
  - Uno o più componenti
  - Hardware o software
- Punto di passaggio obbligato
  - Efficace solo se non ci sono altre strade per accedere alla rete da proteggere
- Default deny
  - Passa solo quel che è esplicitamente autorizzato
- Robustezza
  - Dev'essere immune agli attacchi → sistema dedicato, in cui sia possibile rinunciare a flessibilità e praticità in favore della riduzione delle vulnerabilità

Tecniche di controllo

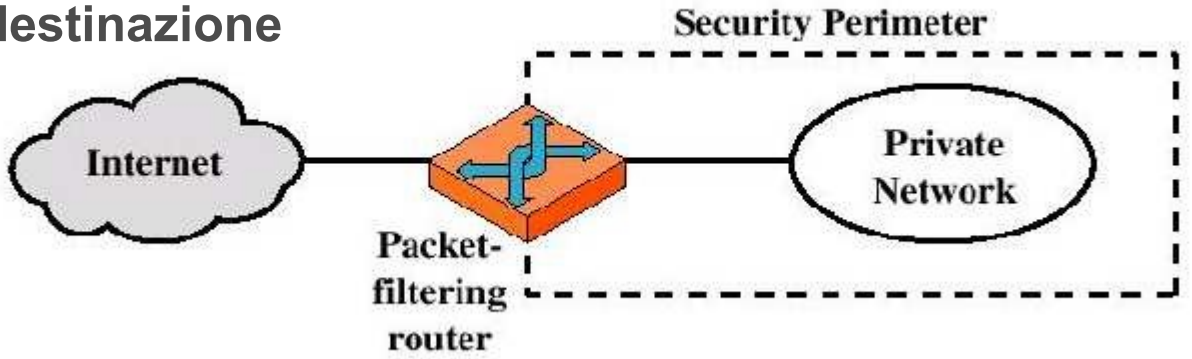
- *Traffico*
  - Esaminare indirizzi, porte, e altri indicatori del tipo di servizio che si vuol rendere accessibile
- *Direzione*
  - Discriminare a parità di servizio le richieste entranti verso la rete interna da quelle originate da essa
    - N.B.: il traffico è sempre composto da uno scambio bidirezionale di pacchetti, la direzione *logica* di una connessione è definita da chi prende l'iniziativa
- *Utenti*
  - Differenziare l'accesso ai servizi sulla base di chi lo richiede
    - N.B.: nel protocollo TCP/IP non c'è traccia dell'utente responsabile della generazione di un pacchetto!
- *Comportamento*
  - Valutare come sono usati i servizi ammessi, per identificare anomalie rispetto a parametri di "normalità"

Tipi di firewall

- Tre tipi fondamentali
  - Packet filter
  - Application-level gateway
  - Circuit-level gateway
- Due collocazioni particolari
  - Bastion host
  - Personal firewall

Tipi di firewall: packet filter (PF)

- Esamina unicamente l'header del pacchetto, es.:
  - Link layer:
    - Interfaccia fisica di ingresso o uscita
    - MAC address sorgente / destinazione
  - IP layer:
    - Indirizzi sorgente / destinazione
    - Protocollo trasportato (ICMP, TCP, UDP, AH, ESP, …)
    - Opzioni IP (ECN, TOS, …)
  - Transport layer:
    - TCP flags (SYN, ACK, FIN, RST, …)
    - Porte sorgente / destinazione



Tipi di firewall: PF

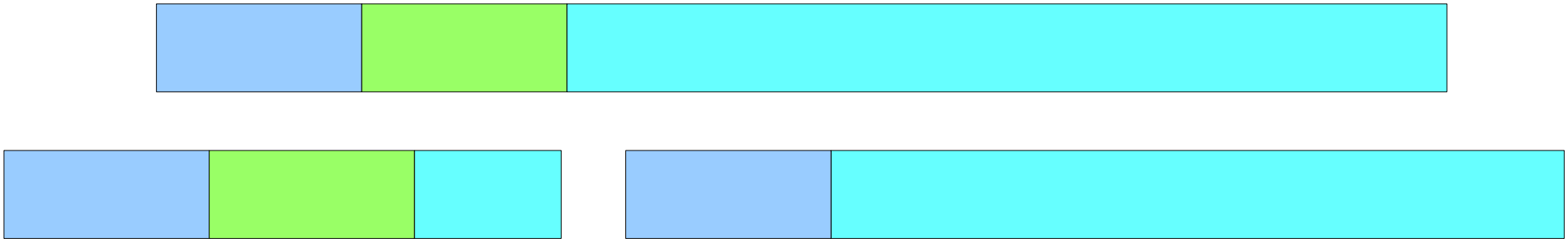
- Applica in serie un elenco di regole del tipo "se *condizione* allora *azione*"
  - Normalmente la prima trovata in cui il pacchetto soddisfa la condizione determina il destino del pacchetto e interrompe la scansione dell'elenco
  - Le azioni di base sono scartare o inoltrare il pacchetto
  - Altre comunemente implementate:
    - Loggare i dettagli del pacchetto
    - Modificare in qualche modo il pacchetto
  - Se nessuna regola viene attivata, si applica una politica di default (scartare o inoltrare il pacchetto)
- Normalmente le regole sono raccolte in più liste separate, corrispondenti a punti di controllo diversi
  - es. per i pacchetti in ingresso al firewall e quelli in uscita

Tipi di firewall: PF

- Vantaggi
  - Semplice e veloce
    - Implementato tipicamente in tutti i router
  - Trasparente agli utenti
    - Se il firewall coincide col default gateway di una subnet, per farlo attraversare non si deve riconfigurare nessun sistema
    - Nell'implementazione locale a un sistema, può intercettare il traffico locale e reindirizzarlo a componenti user-space arbitrari
- Svantaggi
  - Regole di basso livello
    - Comportamenti sofisticati richiedono set di regole molto complessi
  - Mancanza di supporto alla gestione utenti
    - Negli header non compaiono elementi identificativi
- La configurazione è importante
  - RFC2827, RFC3704, RFC8704 (best current practices)

Tipi di firewall: PF

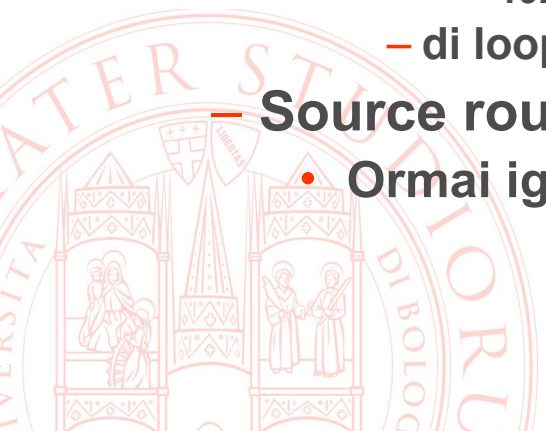
- Vulnerabilità e contromisure (parziali)
  - Frammentazione



- Frammenti successivi al primo non possono attivare condizioni che menzionano parametri dell'header di trasporto → evasione
- Molti altri attacchi basati su vulnerabilità dei riassemblatori
- Soluzione drastica: scartare i pacchetti frammentati
- Soluzione costosa: riassemblare sul firewall (non implementabile su packet filter puro)

Tipi di firewall: PF

- Vulnerabilità e contromisure (parziali)
  - Spoofing (falsificazione degli indirizzi del mittente)
    - Controllo di coerenza tra subnet e interfacce/configurazione
      - Multicast (224.0.0.0/4) se non utilizzato
      - Provenienti da “fuori” con IP sorgente della rete “dentro” e v.v.
        - Impossibile su router infrastrutturali
    - Controllo su indirizzi sorgente “alieni”
      - illegali (es. 0.0.0.0/8)
      - di broadcast (p.e. 255.255.255.255/32)
      - riservati; almeno quelli della rfc1918:
        - 10.0.0.0/8
        - 172.16.0.0/12
        - 192.168.0.0/16
      - di loopback: 127.0.0.0/8
  - Source routing (instradamento determinato dal mittente)
    - Ormai ignorato da tutti i router



Tipi di firewall: PF

- Limitazioni
  - Se non si introduce un livello di vera e propria analisi del protocollo applicativo, il filtraggio *stateful* non può gestire protocolli che negoziano dinamicamente le connessioni
  - Es. FTP:
    - TCP open (C,>1023) → (S,21) *Control Channel*

      Sul control channel si scambiano i comandi: es GET filename

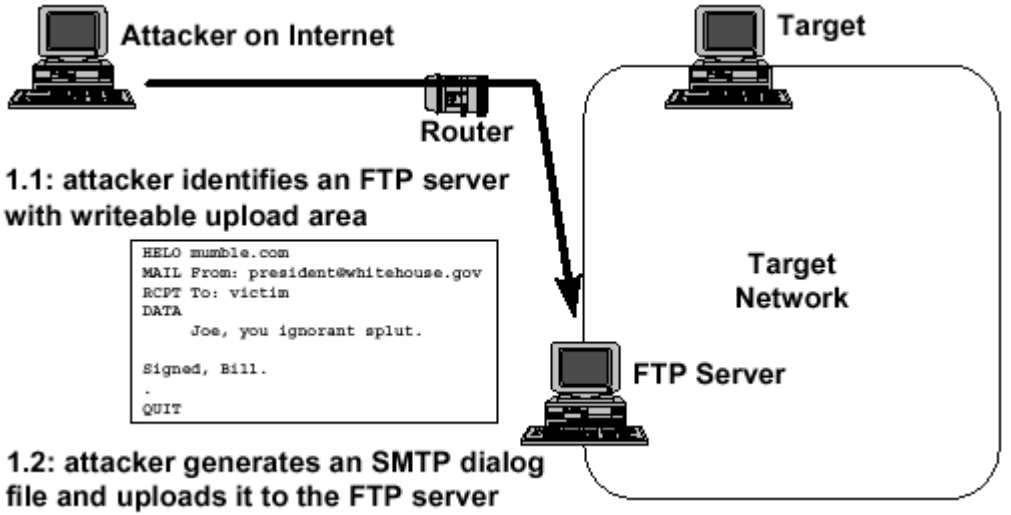
      Il trasferimento avviene sul Data Channel

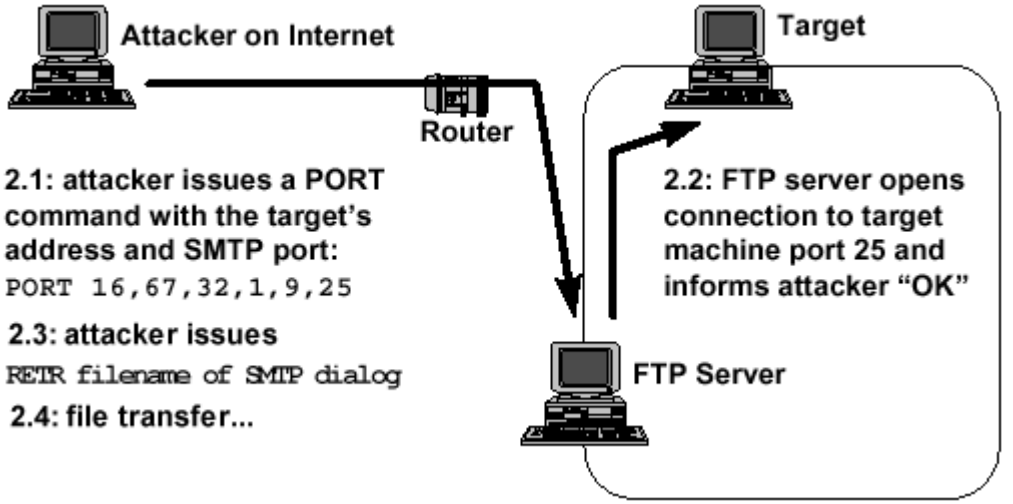
      Il Client sceglie una porta alta sulla quale si mette in ascolto e la comunica al server con il comando "PORT" es: PORT 1234
    - TCP open (S,20) → (C,1234) *Data Channel*

      Su questo canale il file viene effettivamente trasferito
    - La porta di destinazione del Data Channel non è nota a priori
      - Non esiste una regola del PF per ammetterla
    - … e viaggia nel *payload* del pacchetto
      - Il PF non la può vedere, non è nell'header
  - Altri casi molto comuni: streaming protocols per multimedia

Tipi di firewall: PF

- Limitazioni
  - Protezione assente contro attacchi data-driven (nel payload)
  - Es. FTP bouncing



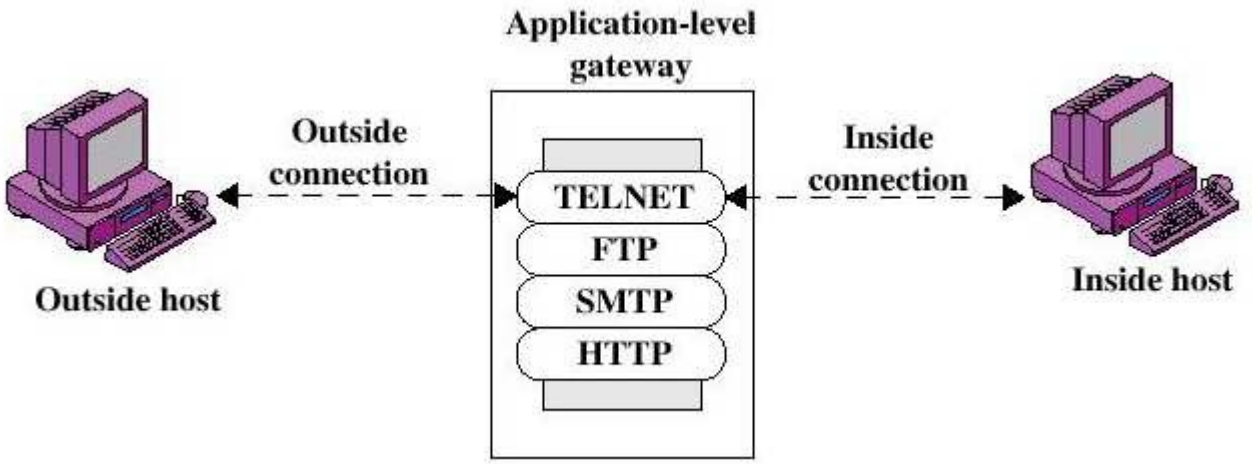


Tipi di firewall: PF

- Formalmente un PF è *stateless*
  - Non ha memoria del traffico passato
  - Decide su ogni pacchetto solo sulla base delle regole
- Evoluzione: PF *stateful*
  - Ha memoria di qualche aspetto del traffico che vede passare
  - Può decidere su di un pacchetto riconoscendolo parte di un flusso di traffico già instaurato
    - Implementazione specifica del tipo di PF
    - Utile soprattutto per protocolli senza connessione
- Evoluzione: Multilayer protocol inspection firewall
  - Tiene traccia dell'intera storia della connessione per verificare la coerenza del protocollo
  - In alcuni casi anche oltre il livello di trasporto

Tipi di firewall: Application-Level Gateway

- Anche chiamato *proxy server*
  - In questo ruolo può svolgere anche altre funzioni, es. caching
- Un ALG è un "man in the middle buono" che agisce da server nei confronti del client, e propaga la richiesta agendo da client nei confronti del server effettivo

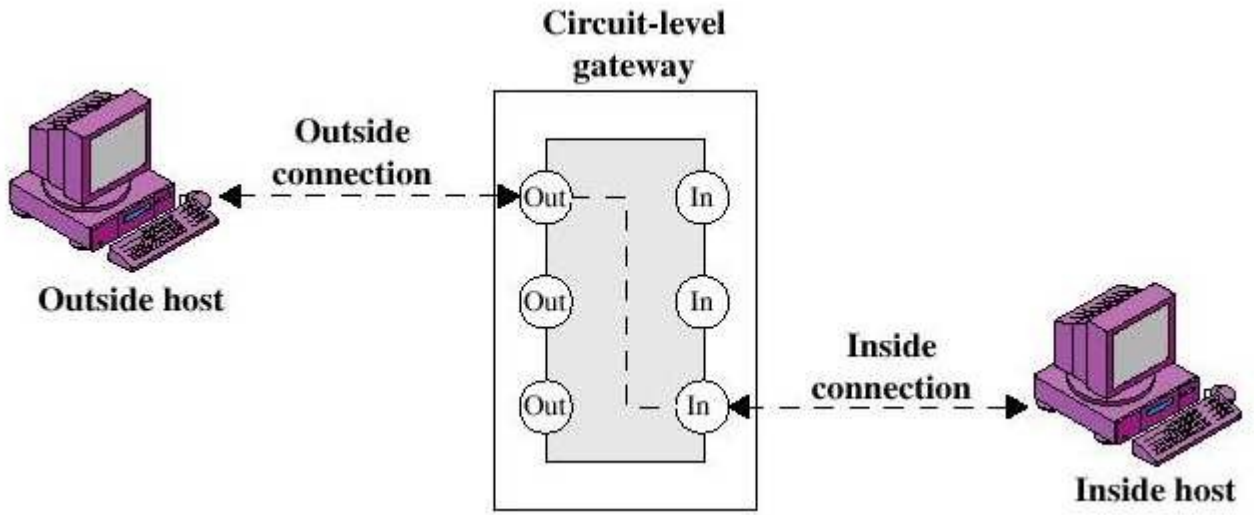


Tipi di firewall: ALG

- Vantaggi
  - Comprende il protocollo applicativo, quindi permette filtraggi avanzati come
    - Permettere/negare specifici comandi
    - Esaminare la correttezza degli scambi protocollari
    - Attivare dinamicamente regole sulla base della negoziazione C/S
  - Sono integrabili con processi esterni per l'esame approfondito del payload, es:
    - Antispam/antivirus per la posta
    - Antimalware/antiphishing per il web
  - Permette di tenere log molto dettagliati delle connessioni
    - Privacy permettendo!
- Svantaggi
  - Molto più pesante di un PF
  - Specifico di un singolo protocollo applicativo
  - Non sempre trasparente, può richiedere configurazione del client

Tipi di firewall: Circuit-level gateway (CLG)

- Spezzano la connessione a livello di trasporto
  - Diventano endpoint del traffico, non intermediari
  - Inoltrano i payload senza esaminarli



Tipi di firewall: CLG

- Utilizzo tipico
  - Determinare quali connessioni sono ammissibili dall'interno verso l'esterno
- Vantaggi
  - Può essere configurato trasparentemente agli utenti per autorizzare le connessioni da determinati host considerati fidati
  - Può agire da intermediario generico, senza bisogno di predefinire quali protocolli applicativi gestire
  - Può essere usato in combinazione con le applicazioni per differenziare le politiche sulla base degli utenti
- Svantaggi
  - Le regole di filtraggio sono limitate a indirizzi, porte, utenti
    - Si può combianare con un PF per gestire più dettagli di basso livello, con un ALG per gestire più dettagli applicativi
  - Richiede la modifica dello stack dei client
    - O la consapevole configurazione delle applicazioni

Collocazioni dei firewall

- Bastion Host (BH)
  - Un sistema dedicato a far girare un software firewall, tipicamente per realizzare un ALG o un CLG
  - Può servire anche per un PF, ma tipicamente questo è integrato nei router che servono la rete
- Personal Firewall
  - Costituiscono un'eccezione al principio del controllo alla frontiera, essendo installati sulle singole macchine da proteggere
  - Vantaggi
    - Correlazione fra applicazione sorgente/destinazione e pacchetto → altissima precisione nel controllo di cosa è lecito vs. anomalo
  - Svantaggi
    - Perdita della centralizzazione della configurazione (o necessità di utilizzare sistemi di deploy piuttosto invasivi)
    - Spesso configurati "learning by doing" → molti alert → ignorati

Topologie di filtraggio

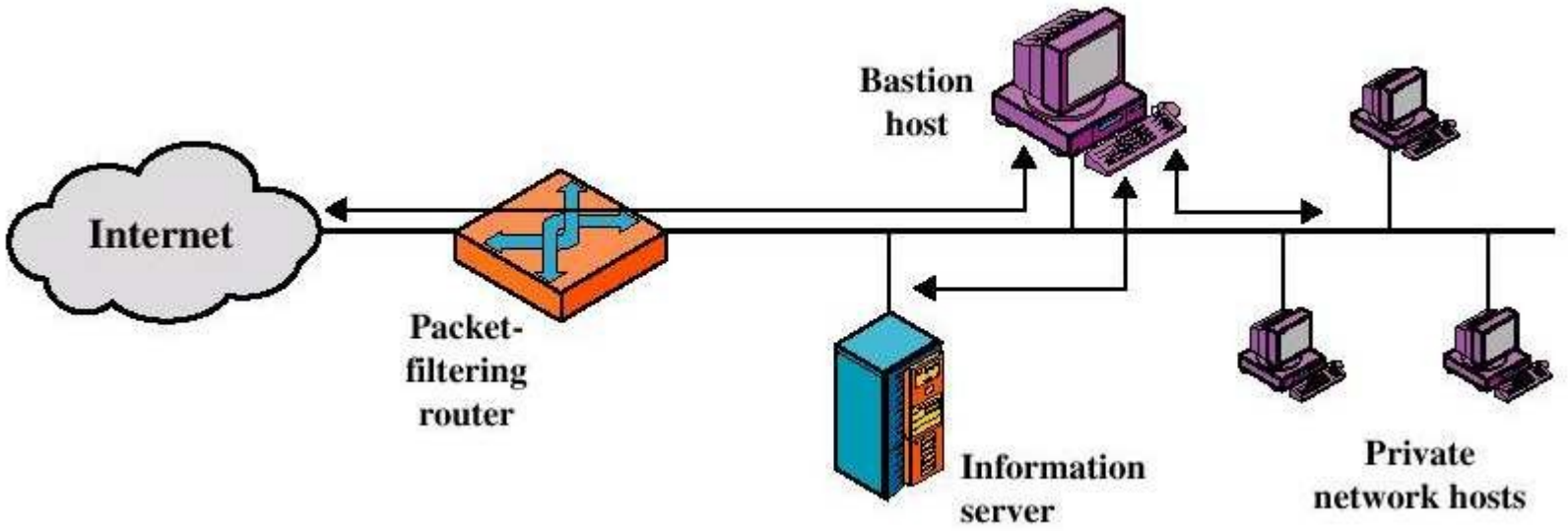
- La situazione più semplice è quella

  (rete esterna) --- (firewall) --- (rete interna)

- Non è adatta a reti in cui siano presenti contemporaneamente
  - Client
    - generano traffico uscente
    - devono essere totalmente schermati dagli attacchi esterni
  - Server
    - devono ricevere selettivamente traffico dall'esterno
    - possono essere più facilmente compromessi e non devono poter essere usati per attaccare i client
- Utilizzo di molteplici dispositivi per generare reti con zone differenziate

Topologie – screened single-homed BH

- Un PF garantisce che solo un BH possa comunicare con l'esterno
- Il BH implementa un ALG (eventualmente con autenticazione)

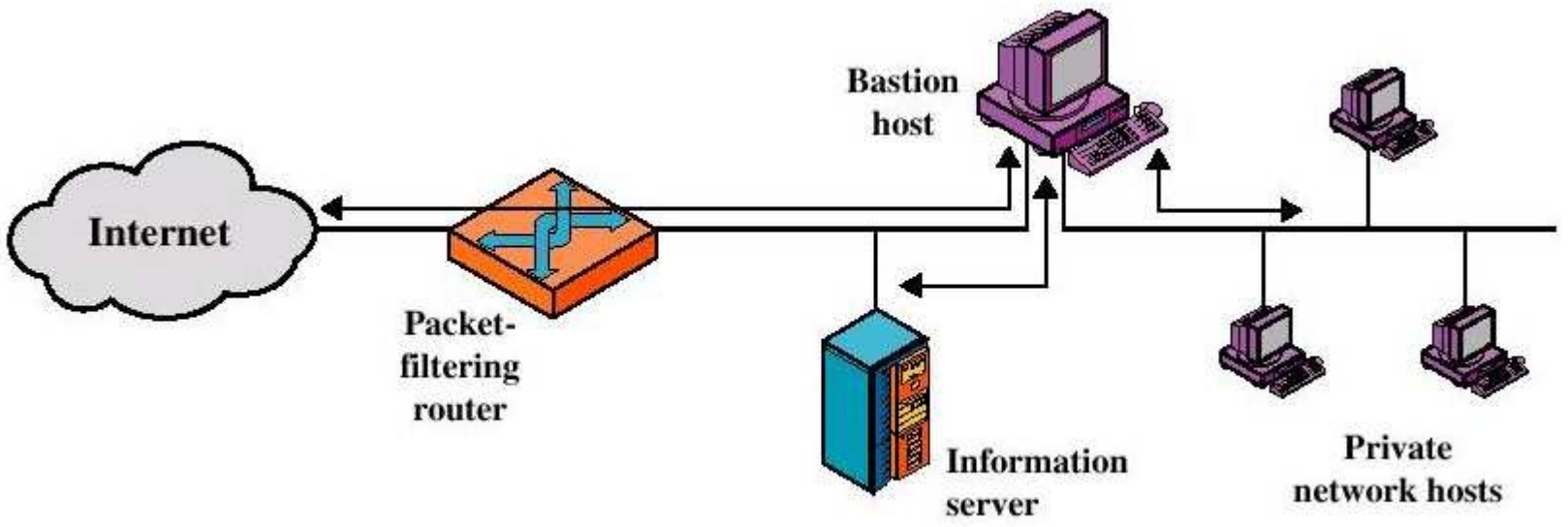


Topologie – screened single-homed BH

- Doppio filtraggio
  - a livello header (PF)
  - e applicativo (BH)
- Per prendere il controllo completo della rete interna, due sistemi da compromettere
  - Ma per un accesso significativo è sufficiente compromettere il PF (per contro, questo è tipicamente un sistema embedded o che comunque offre una superficie di attacco ridottissima)
- Semplice fornire accesso diretto a server totalmente pubblici

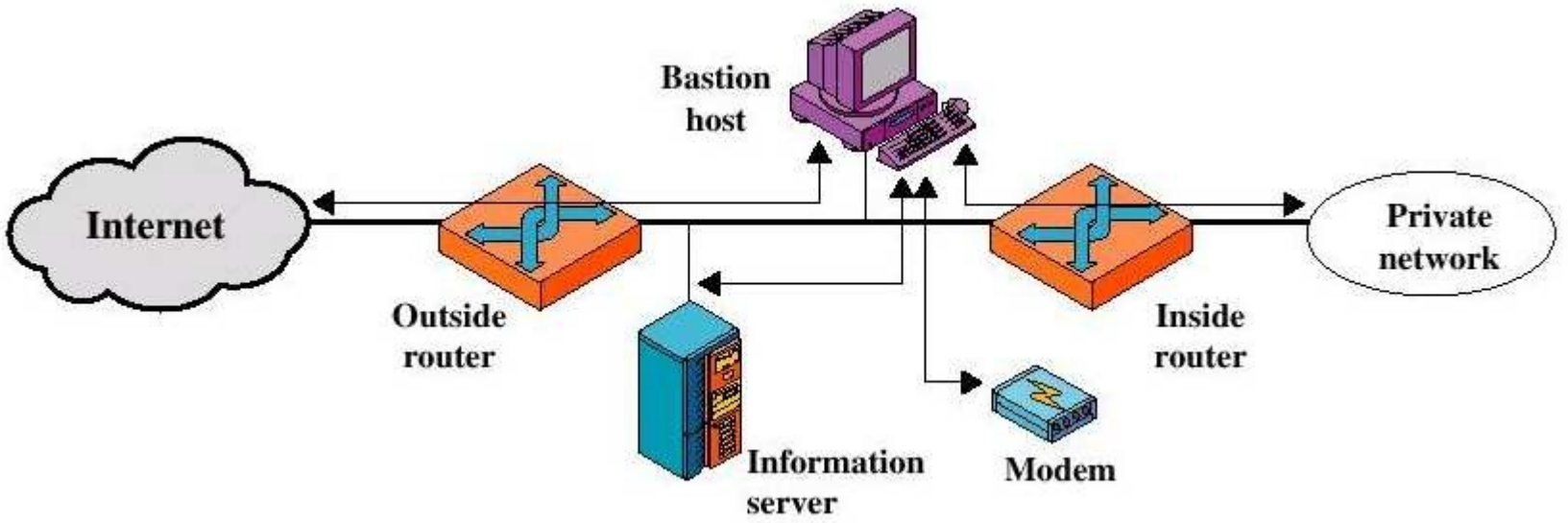
Topologie – screened dual-homed BH

- Come prima, ma il BH separa fisicamente due segmenti di rete
  - La compromissione del PF non dà accesso alla rete interna
  - Si crea una zona intermedia detta “demilitarizzata” (DMZ)
    - I server sono collocati qui
- Svantaggio: tutto il traffico dai client *deve* fluire attraverso il BH, anche quello del tutto innocuo



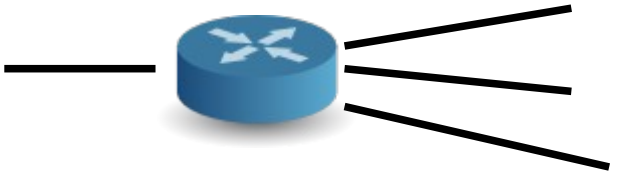
Topologie – screened subnet

- L'uso di due PF router
  - Rafforza la separazione tra esterno e interno
  - Nasconde completamente all'esterno l'esistenza della subnet privata, ostacolando l'enumerazione da parte degli attaccanti
  - Nasconde l'esistenza di Internet alla rete privata, ma consente ai router di inoltrare il traffico "banale" senza passare dal BH



Topologie – variazioni sul tema

- Sacrificando il doppio livello di protezione, se si dispone di un PF molto affidabile o di poco budget
  - Si possono unificare le funzioni di R1 e R2 della topologia screened subnet
  - con >3 interfacce si possono realizzare diverse DMZ



- Al contrario, se si deve gestire con elevata sicurezza una topologia di rete caratterizzata da molte zone con esigenze di protezione via via più elevate, si possono concatenare in serie DMZ con vari PF

IPTables

- **iptables** è il packet filter integrato nel kernel Linux
- recentemente è stato affiancato da **nftables**, che in prospettiva lo sostituirà
  - https://www.netfilter.org/projects/nftables/
  - https://paulgorman.org/technical/linux-nftables.txt.html
  - https://wiki.nftables.org/wiki-nftables/index.php/Main_Page
  - ma iptables è ancora la soluzione più diffusa
- si appoggia sul framework **netfilter**
  - definisce degli hook nello stack di rete del kernel
  - ogni pacchetto che attraversa lo stack di rete innesca gli hook
  - si possono registrare programmi agli hook in modo da far eseguire controlli e manipolazioni sui pacchetti

netfilter hooks

- Cinque hook in punti strategici dello stack di rete
  - NF_IP_PRE_ROUTING
    - attivato da un pacchetto appena entra nello stack di rete. Questo hook viene elaborato prima di prendere qualsiasi decisione di instradamento riguardo a dove inviare il pacchetto.
  - NF_IP_LOCAL_IN:
    - attivato dopo che un pacchetto in arrivo è stato instradato se il pacchetto è destinato al sistema locale.
  - NF_IP_FORWARD:
    - attivato dopo che un pacchetto in arrivo è stato instradato se il pacchetto deve essere inoltrato a un altro host.
  - NF_IP_LOCAL_OUT:
    - attivato da qualsiasi pacchetto in uscita creato localmente non appena raggiunge lo stack di rete.
  - NF_IP_POST_ROUTING:
    - attivato da qualsiasi pacchetto in uscita o inoltrato dopo che l'instradamento ha avuto luogo e appena prima di essere messo in rete.
- Più programmi possono registrarsi allo stesso hook, dichiarando un ordine di priorità
  - invocati in ordine
  - ognuno restituisce una decisione sul destino del pacchetto

iptables connesso a netfilter

- iptables gestisce il traffico registrandosi agli hook
- concetti fondamentali:
- tabelle
  - organizzano i controlli a seconda del **tipo di decisione** da prendere sul pacchetto
- catene
  - organizzano i controlli a seconda dell'hook a cui sono agganciate, quindi del **momento in cui decidere** cosa fare del pacchetto durante il suo ciclo di vita nel sistema
- regole
  - sono gli elementi costitutivi delle catene
  - espressioni del tipo "SE il pacchetto rispetta queste condizioni, ALLORA esegui questa azione"

catene

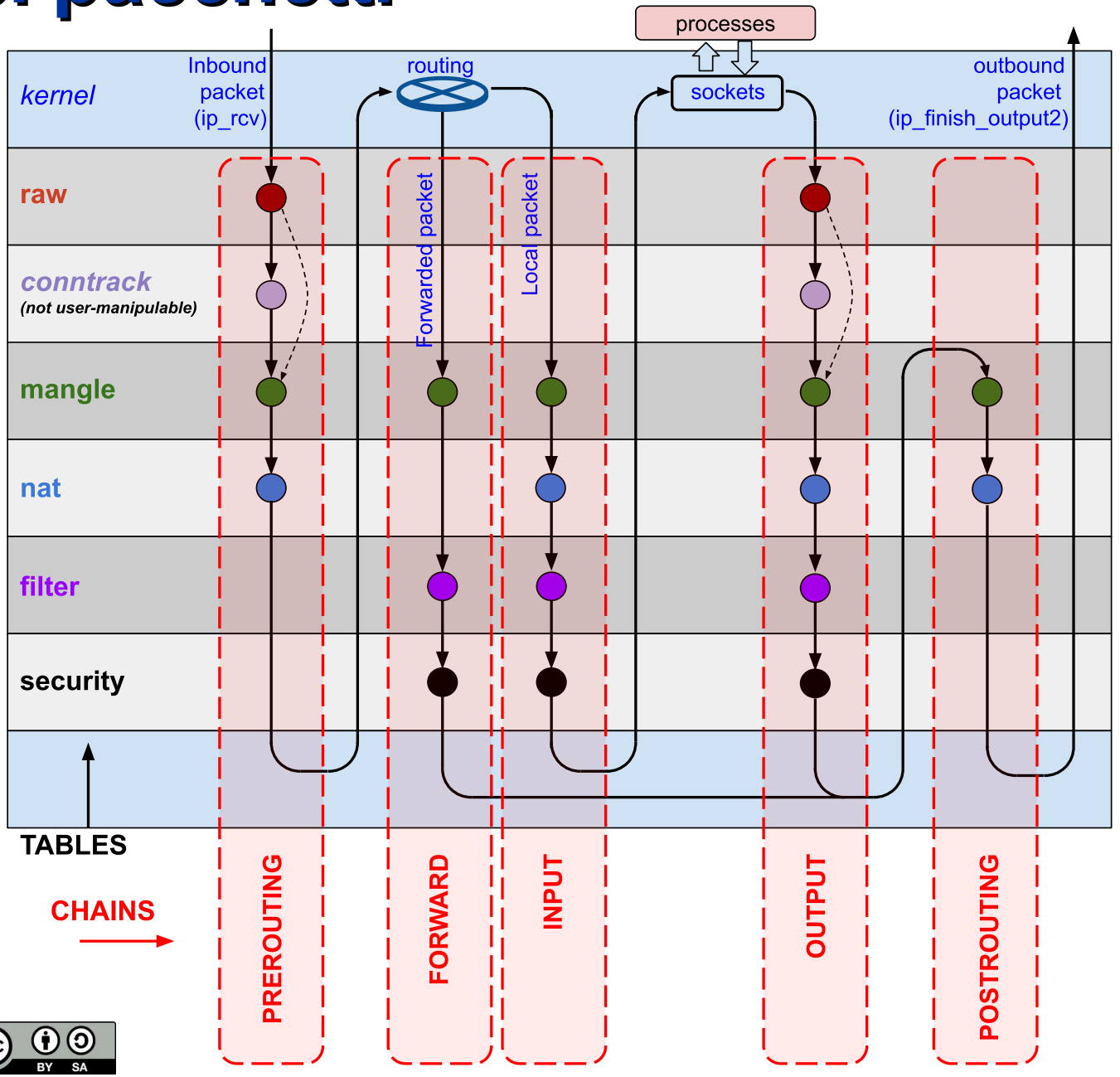
- corrispondono esattamente agli hook di netfilter
  - PREROUTING: attivata dall'hook NF_IP_PRE_ROUTING
    - regole da applicare appena il pacchetto entra
  - INPUT: attivata dall'hook NF_IP_LOCAL_IN
    - regole da applicare prima di consegnare il pacchetto a un processo
  - FORWARD: attivata dall'hook NF_IP_FORWARD
    - regole da applicare prima di inoltrare un pacchetto a un altro host
  - OUTPUT: attivata dall'hook NF_IP_LOCAL_OUT
    - regole da applicare a un pacchetto appena generato da un processo
  - POSTROUTING: attivata dall'hook NF_IP_POST_ROUTING
    - regole da applicare a un pacchetto appena prima che lasci il sistema
- non tutte le tabelle registrano tutte le catene possibili

tabelle

- **raw** — iptables è stateful, quindi tratta i pacchetti come parte di una connessione; raw fornisce un meccanismo per contrassegnare i pacchetti al fine di disattivare il tracciamento della connessione saltando conntrack
- **conntrack** — implementa automaticamente (cioè con una logica non configurabile dall'utente) il riconoscimento delle connessioni e l'attribuzione dei pacchetti alle stesse
- **filter** — la tabella principale, utilizzata per decidere se lasciare che un pacchetto continui verso la destinazione prevista o bloccarlo.
- **nat** — utilizzata per implementare le regole di traduzione degli indirizzi di rete, modificando gli indirizzi di origine o di destinazione del pacchetto
- **mangle** — utilizzata per modificare l'intestazione IP del pacchetto (es. cambiare il valore TTL o qualsiasi altro campo), e può marcare la rappresentazione kernel di un pacchetto per renderlo riconoscibile da altre tabelle e da altri strumenti
- **security** — utilizzata per impostare i contrassegni di contesto di sicurezza SELinux interni sui pacchetti

Il percorso dei pacchetti

- Ogni pacchetto che entra nello stack di rete viene sottoposto all'esame di varie catene nell'ordine mostrato da questo schema
- Il percorso ha un inizio comune per i pacchetti di origine esterna (tutte le catene PREROUTING delle tabelle che la supportano)
- Il percorso si dirama a seconda che il pacchetto sia destinato a un processo locale (catene INPUT) o a un host remoto (catene FORWARD)
- I pacchetti di origine interna sono processati dalle catene OUTPUT
- I pacchetti di qualunque origine destinati a lasciare il sistema sono infine processati dalle catene POSTROUTING



Marco Prandini marco.prandini@unibo.it Based on http://linux-ip.net/nf/nfk-traversal.png by Martin. A. Brown, martin@linux-ip.net

regole

- Ognuna delle catene illustrate è composta da una **sequenza** di regole
- Ogni regola può stabilire un elenco di condizioni (**match**), e un'azione (**target**) da eseguire su ogni pacchetto che rispetti tutte le condizioni
- I target possono essere classificati in due categorie
  - **terminating target**: concludono l'esame delle regole della catena e ritornano il controllo a netfilter (che attuerà un'operazione dipendente dallo specifico target incontrato, es. scartare il pacchetto, contrassegnarlo, …)
  - **non-terminating target**: eseguono un'azione sul pacchetto, che non lascia però la catena e viene sottoposto all'analisi delle regole successive
  - **l'ordine delle regole è quindi fondamentale!**
- Se un pacchetto non soddisfa le condizioni di alcuna regola, o solo di regole con non-terminating target, la catena deve comunque restituire a netfilter un risultato (**default policy**)

terminating target di base

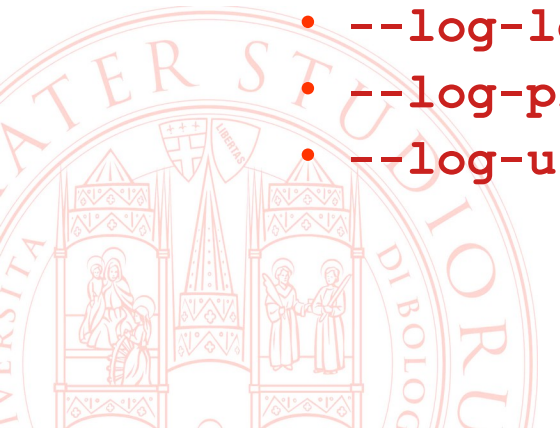
- **ACCEPT** termina la scansione della catena corrente indicando a netfilter di proseguire l'analisi con le catene successive
- **DROP** termina la scansione della catena corrente indicando a netfilter di scartare il pacchetto
  - è comune utilizzare questo target unicamente nelle catene della tabella filter, le altre tabelle sono utilizzate per modifiche o marcature specifiche ma non è opportuno utilizzarle per decidere il "destino" del pacchetto
- **RETURN** termina la scansione della catena corrente passando a netfilter come risultato la default policy della catena

terminating target specifici della tabella nat

- nelle catene **PREROUTING** o **OUTPUT**
  - il target **DNAT** indica a netfilter che deve essere modificato l'indirizzo di destinazione del pacchetto
  - l'opzione `--to-destination [ipaddr[-ipaddr]][:port[-port]]` permette di specificare la nuova destinazione
  - il target **REDIRECT** funziona come **DNAT** ma è necessario se si vuole specificamente ridirigere il pacchetto alla macchina locale
  - l'opzione --to-ports permette di cambiare la porta di destinazione
- nelle catene **POSTROUTING** o **INPUT**
  - il target **SNAT** indica a netfilter che deve essere modificato l'indirizzo di sorgente del pacchetto
  - l'opzione `--to-source [ipaddr[-ipaddr]][:port[-port]]` permette di specificare la nuova sorgente
  - il target **MASQUERADE** (valido solo in **POSTROUTING**) funziona come **SNAT** assegnando automaticamente al pacchetto l'indirizzo dell'interfaccia di uscita
- se vengono utilizzati intervalli, di default, i diversi indirizzi e porte vengono utilizzati a turno (round-robin) via via che arrivano pacchetti

non-terminating target

- **nessun target!**
  - ad ogni regola sono associati due contatori che vengono incrementati ogni volta che un pacchetto “fa match”
    - un contatore di pacchetti
    - un contatore di byte cumulativamente da essi trasportati
  - una regola senza target permette di conteggiare il traffico con certe caratteristiche senza interferire col transito dei pacchetti in netfilter
- **LOG**
  - il kernel logga i dettagli del pacchetto
  - opzioni utili:
    - `--log-level <priority>`
    - `--log-prefix <prefisso>`
    - `--log-uid`



match di base sull'header IPv4

- Caratteristiche "Layer 1"
  - `-i <input interface>`
  - `-o <output interface>`
- Caratteristiche "Layer 2"
  - solo caricando l'estensione con `-m mac`
    - `--mac-source <source mac address>`
- Caratteristiche "Layer 3"
  - `-s <source address>`
  - `-d <destination address>`
  - `-f` (fa match coi frammenti dal secondo in poi)
  - solo caricando l'estensione con `-m iprange`
    - `--src-range from-to`
    - `--dst-range from-to`

innumerevoli altre:
`man iptables-extensions`

match di base sull'header IPv4

- Caratteristiche "Layer 4"

  `-p <tcp|udp|udplite|icmp|icmpv6|esp|ah|sctp|mh>`
- se il protocollo supporta le porte, abilita l'interpretazione di

  `--dport port[:port]`

  `--sport port[:port]`
- se il protocollo è tcp, abilita l'interpretazione di

  `--tcp-flags mask comp`
  - i flag sono `SYN ACK FIN RST URG PSH` *`ALL NONE`*
  - **mask** = elenco flag "interessanti" (gli altri flag del pacchetto sono ignorati)
  - **comp** = elenco flag tra quelli interessanti che devono essere settati per fare match
- se il protocollo è icmp, abilita l'interpretazione di

  `--icmp-type <type>`
  - elenco tipi: `iptables -p icmp -h`

gestione delle catene

- sintassi base del comando

```
iptables [-t <tabella>] -CMD [catena] [match] [-j <target>]
```

se omesso si assume -t filter

- comandi (*CMD*) principali:

– `L`	list	elenca le regole della catena (se presente, o tutte)
– `C`	check	ritorna true se una regola coi match e il target specificati esiste nella catena
– `A`	append	aggiunge una regola in fondo alla catena
– `I [n]`	insert	inserisce una regola (in n-esima posizione, o in testa se manca n)
– `D [n]`	delete	rimuove una regola (in n-esima posizione, o quella che ha esattamente i match e il target specificati)
– `R n`	replace	sostituisce la regola in n-esima posizione
– `F`	flush	svuota la catena
– `P`	policy	imposta la policy di default della catena

esempi

- `iptables -A FORWARD -i ppp0 -d 87.15.12.0/24 -p tcp --dport 80 -j ACCEPT`
- `iptables -P INPUT DROP`
- `iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ppp0 -j SNAT --to-source 87.4.8.21`
- `iptables -t nat -A PREROUTING -i ppp0 -d 87.4.8.21 -p tcp --dport 2222 -j DNAT --to-destination 192.168.0.1:22`
- `iptables -D FORWARD 1`
- `iptables -I INPUT 13 -p icmp ! --icmp-type echo-reply -j DROP`
- `iptables -A OUTPUT -p tcp --tcp-flags SYN,ACK,FIN FIN`

gestione dei contatori

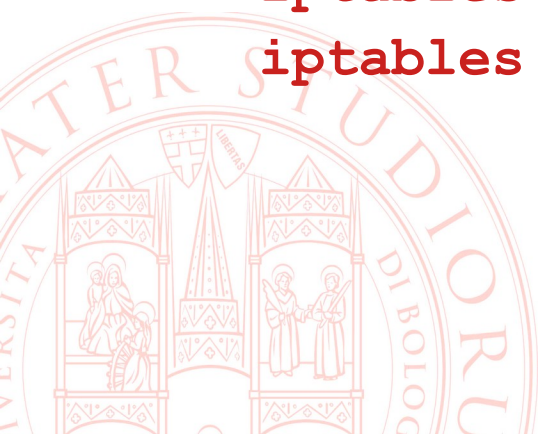
- I contatori vengono visualizzati col comando list (`-L`)
  - se unito all'opzione -v
  - in forma "human readable" (K, M, G)
    a meno che non si usi l'opzione `-x`
- I contatori possono essere azzerati col comando `-Z`
- Per una lettura reiterata precisa, è importante svolgere lettura e azzeramento in modo contestuale

```
iptables -vnxL -Z > contatori
```

invece di

```
iptables -vnxL > contatori
iptables -Z
```

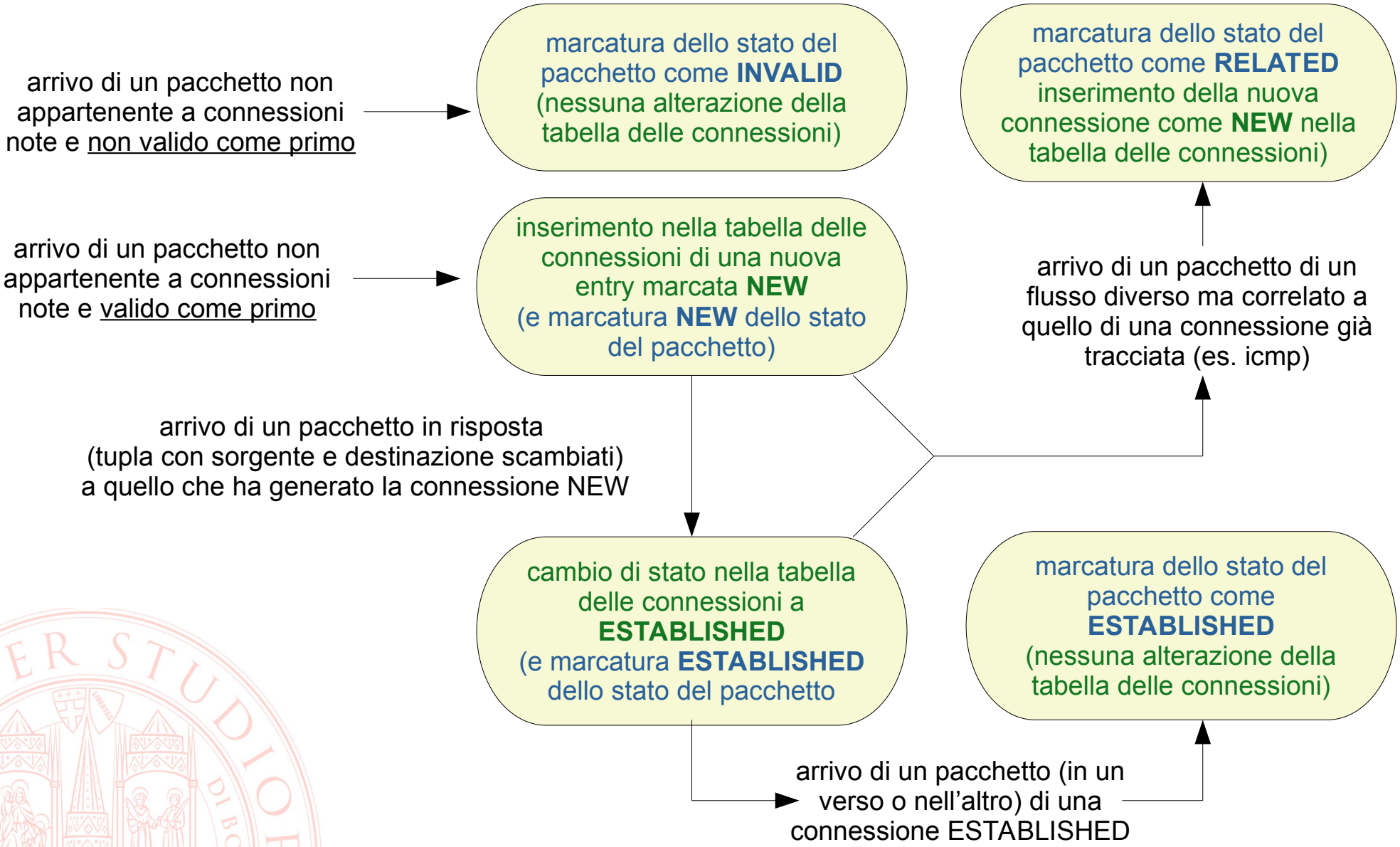
nel tempo che trascorre tra i due comandi, possono arrivare pacchetti
• non inclusi nella lettura del primo comando
• azzerati prima che il ciclo di lettura si ripeta



connection tracking

- una delle prime operazioni svolte da iptables sui pacchetti è gestirne lo stato rispetto a una connessione
  - operazione trasparente svolta da conntrack
  - si può imporre che un pacchetto salti le procedure di connection tracking marcandolo col target `-j CT --notrack` nella catena appropriata della tabella raw
- il connection tracker
  - opera con una propria logica, indipendente dal fatto che il protocollo del pacchetto sia connection-oriented o connection-less
    - definiamo connessione semplicemente la tupla <protocollo, ip sorgente, ip destinazione [, porta sorgente, porta destinazione]>
    - il comando `conntrack -L` mostra le entry della tabella delle connessioni
  - riconosce pacchetti che iniziano nuove connessioni
  - riconosce l'appartenenza di pacchetti a connessioni esistenti
  - applica automaticamente alcune operazioni a tutti i pacchetti di una connessione (vedi NAT)
  - permette di utilizzare lo stato della connessione come match

stati di tracciamento



stati di tracciamento per nat

- quando un pacchetto viene sottoposto ad address translation, alla sua connessione viene assegnato uno stato “virtuale” (aggiuntivo) SNAT o DNAT
- due effetti automatici:
  - i pacchetti della connessione nella stessa direzione vengono automaticamente modificati nello stesso modo
    - le regole della tabella **nat** operano quindi solo sul primo pacchetto della connessione, poi ci pensa **conntrack**
  - i pacchetti della connessione in direzione opposta (le risposte) vengono automaticamente ripristinati con l’indirizzo originale
    - se sono risposte a pacchetti sottoposti a SNAT, l’indirizzo di destinazione viene ripristinato all’indirizzo sorgente originale
    - se sono risposte a pacchetti sottoposti a DNAT, l’indirizzo della sorgente viene ripristinato all’indirizzo destinazione originale
    - queste operazioni avvengono in conntrack, prima di qualsiasi altra analisi da parte di iptables → tenerne conto nel filtraggio

stateful filtering

- dal punto di vista del filtraggio, il connection tracking è molto utile perché permette di utilizzare gli stati dei pacchetti per raffinare i match
- per accettare solo pacchetti validi come iniziatori di una connessione (nella direzione lecita da un client verso un server) e seguenti

```
-m state --state NEW,ESTABLISHED
```

- per accettare solo pacchetti validi come risposte a una connessione già iniziata (nella direzione lecita da un server verso un client) e seguenti

```
-m state --state ESTABLISHED
```

catene custom

- registrare nuovi hook è complesso, ma iptables offre un'alternativa semplice: creare catene custom
  - `iptables [-t TAB] -N <NOME>`
    - crea la catena NOME nella tabella TAB
    - la catena è ignorata fintanto che non ci si "salta dentro" da una catena builtin:
  - `iptables [-t TAB] -A <BC> …match… -j <NOME>`
    - regola inserita nella catena builtin BC della tabella TAB
    - se c'è match, il pacchetto salta alla catena NOME
    - NOME e BC devono essere definite nella stessa tabella
    - iptables inizia a scorrere le regole di NOME, che sono gestibili in modo identico a quello delle catene builtin, con un'eccezione:
  - `iptables [-t TAB] -A <NOME> -j RETURN`
    - termina lo scorrimento delle regole di NOME e ritorna alla catena BC da cui era stato fatto il salto, riprendendo l'esame delle regole da quella successiva
  - `iptables [-t TAB] -X <NOME>`
    - rimuove la catena NOME dalla tabella TAB
    - funziona solo se non ci sono salti verso NOME in altre catene

utilità delle catene custom – esempi

- tenere in ordine set di regole molto ampi creando gerarchie di classificazione
  - per finalità: filtraggio vs. monitoraggio
  - per località: sottoreti, protocolli, ...
  - per tempo di vita: regole persistenti vs. temporanee
  - set di regole predefinite “plug-in” scelte da librerie vs. personalizzazioni
- approccio usato da vari framework per la gestione di firewall, es. ufw, shorewall, ...

utilità delle catene custom – esempi

- rendere più semplice e robusta la gestione di set di regole dinamici
  - es. creando catene custom quando appare in rete una nuova entità da gestire
    - un solo salto dalle catene built-in → pulizia organizzativa
    - facilmente individuabili → semplicità di monitoraggio
- esempio: per ogni server web rilevato dinamicamente a valle del router-firewall

```
iptables -N SRV_$IPSERVER
iptables -I FORWARD -d $IPSERVER -j SRV_$IPSERVER
iptables -I FORWARD -s $IPSERVER -j SRV_$IPSERVER
```

inserimento in SRV_$IPSERVER di regole di filtraggio di base e regole dinamicamente aggiunte e tolte per affrontare situazioni come DoS, account bruteforcing, ecc.

  - allo spegnimento del server, indipendentemente da quante regole sono via via state aggiunte alla catena custom, basterà:

```
iptables -F SRV_$IPSERVER
iptables -D FORWARD -d $IPSERVER -j SRV_$IPSERVER
iptables -D FORWARD -s $IPSERVER -j SRV_$IPSERVER
iptables -X SRV_$IPSERVER
```