Operazione Rif. PA 2023-19410/RER approvata con DGR 1317/2023 del 31/07/2023 finanziata con risorse del Programma Fondo sociale europeo Plus 2021-2027 della Regione Emilia –Romagna.

Progetto n. 1 - Edizione n. 1

TECNICO PER LA PROGETTAZIONE E LO SVILUPPO DI APPLICAZIONI INFORMATICHE

MODULO: N. 5 Titolo: SICUREZZA DEI SISTEMI INFORMATICI E DISPIEGO DELLE APPLICAZIONI
DURATA: 21 ORE DOCENTE: MARCO PRANDINI

MISURE PREVENTIVE CONTRO GLI ATTACCHI

Firewall

- 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

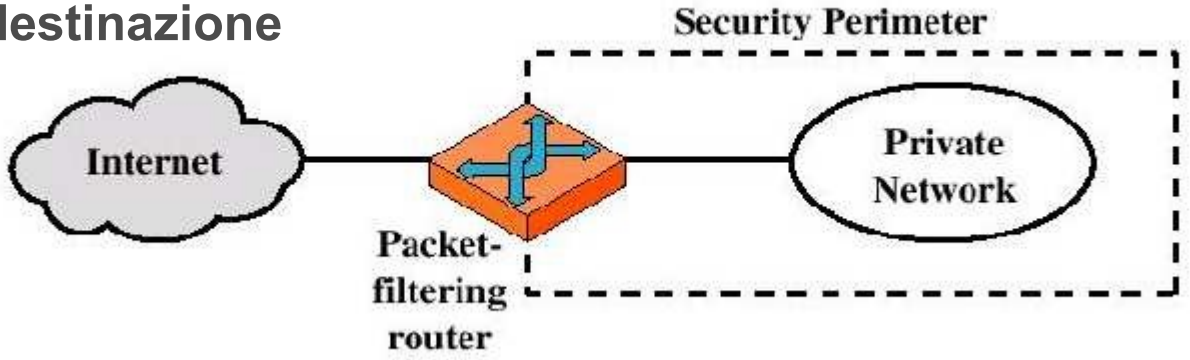
- *Traffico*
  - Esaminare indirizzi, porte, e altri indicatori del tipo di servizio che si vuol rendere accessibile
- *Direzione*
  - Discriminare a parità di servizio le richieste entranti verso la rete interna da quelle originate da essa
    - N.B.: il traffico è sempre composto da uno scambio bidirezionale 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

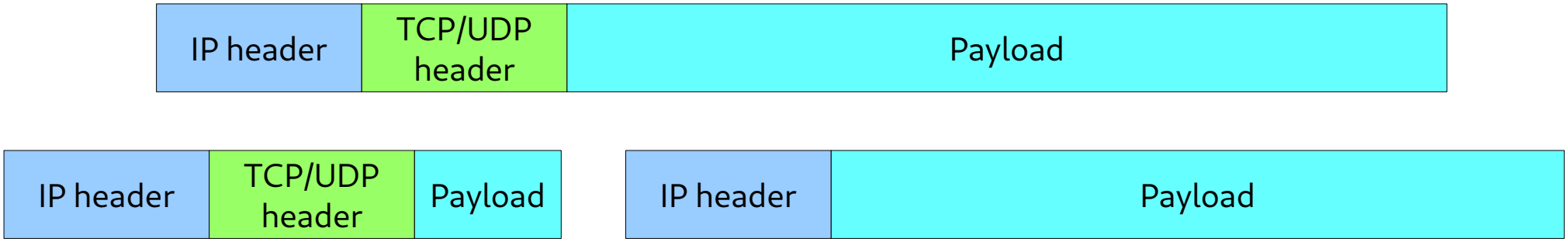
- Applica in serie un elenco di regole del tipo "se *condizione* allora *azione*"
  - Normalmente la prima trovata in cui il pacchetto soddisfa la condizione determina il destino del pacchetto e interrompe la scansione dell'elenco
  - Le azioni di base sono scartare o inoltrare il pacchetto
  - Altre 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

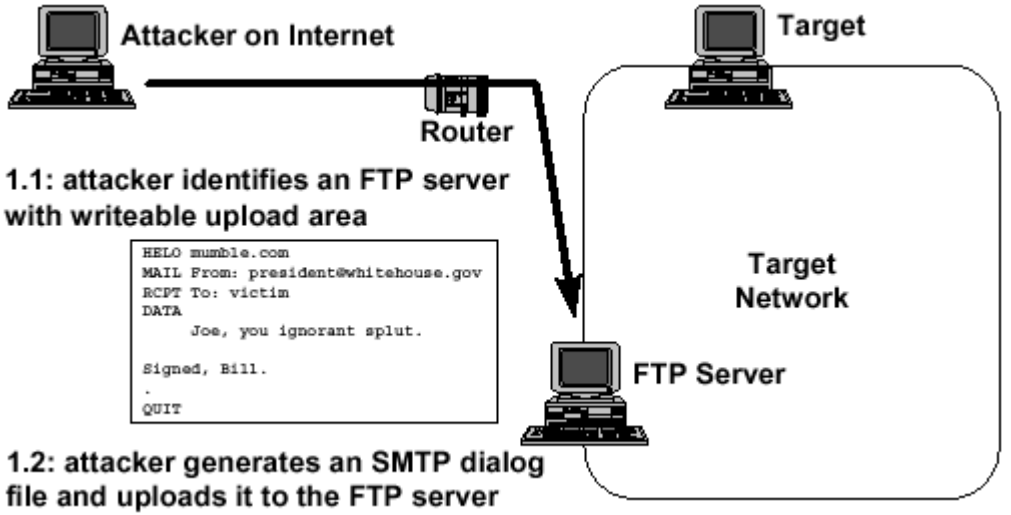
      Il trasferimento avviene sul Data Channel

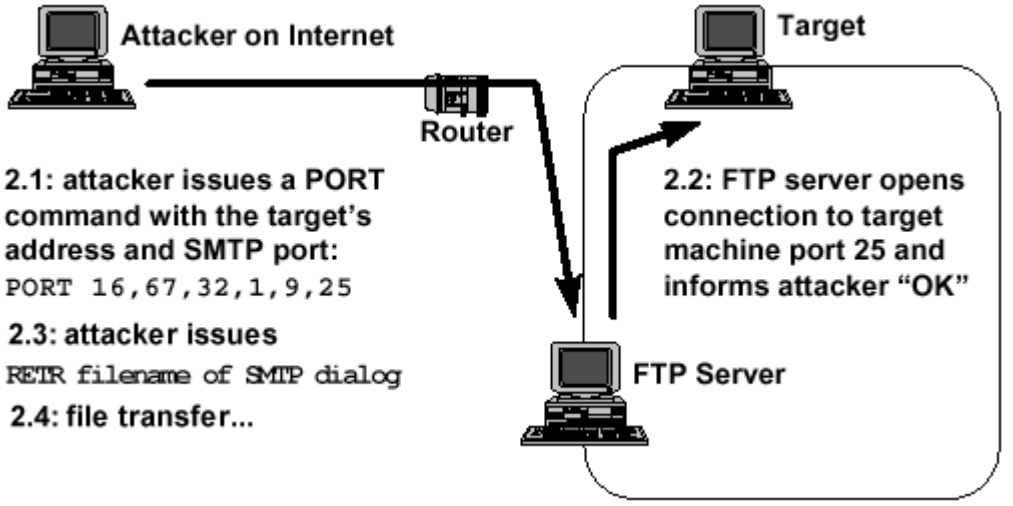
      Il Client sceglie una porta alta sulla quale si mette in ascolto e la comunica al server con il comando "PORT" es: PORT 1234
    - TCP open (S,20) → (C,1234) *Data Channel*

      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



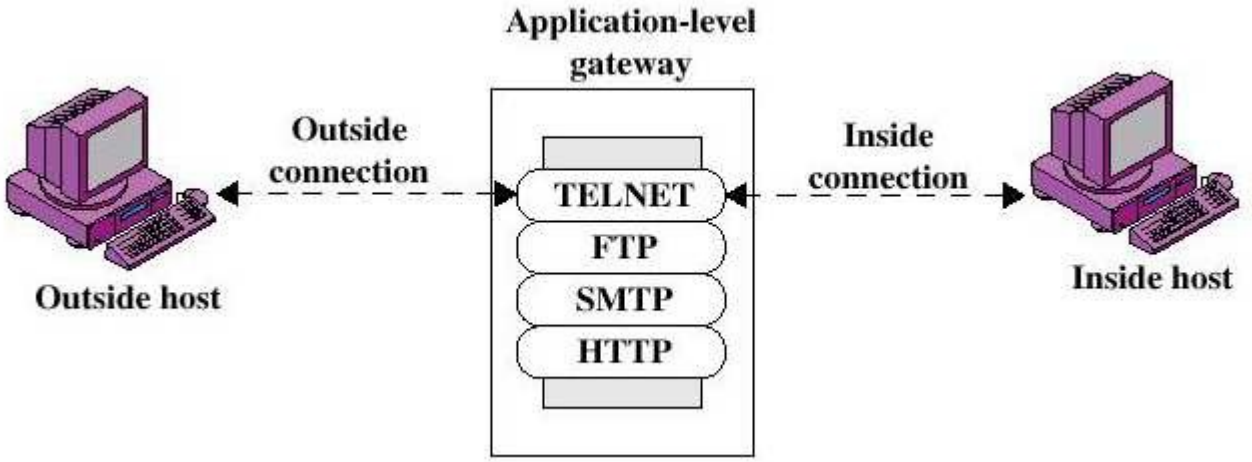


Tipi di firewall: PF

- 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

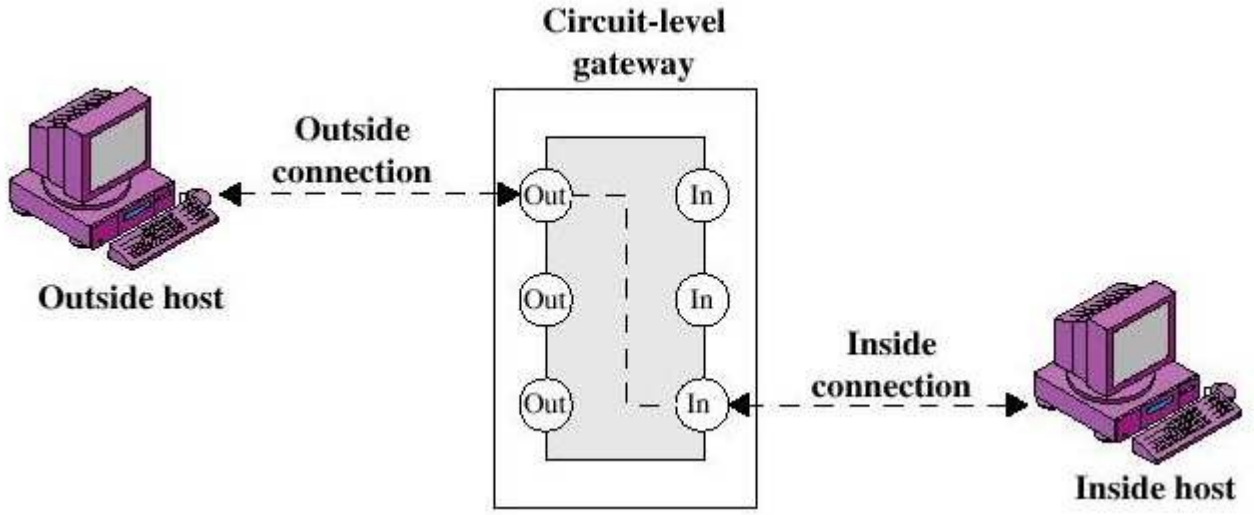


Tipi di firewall: ALG

- 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

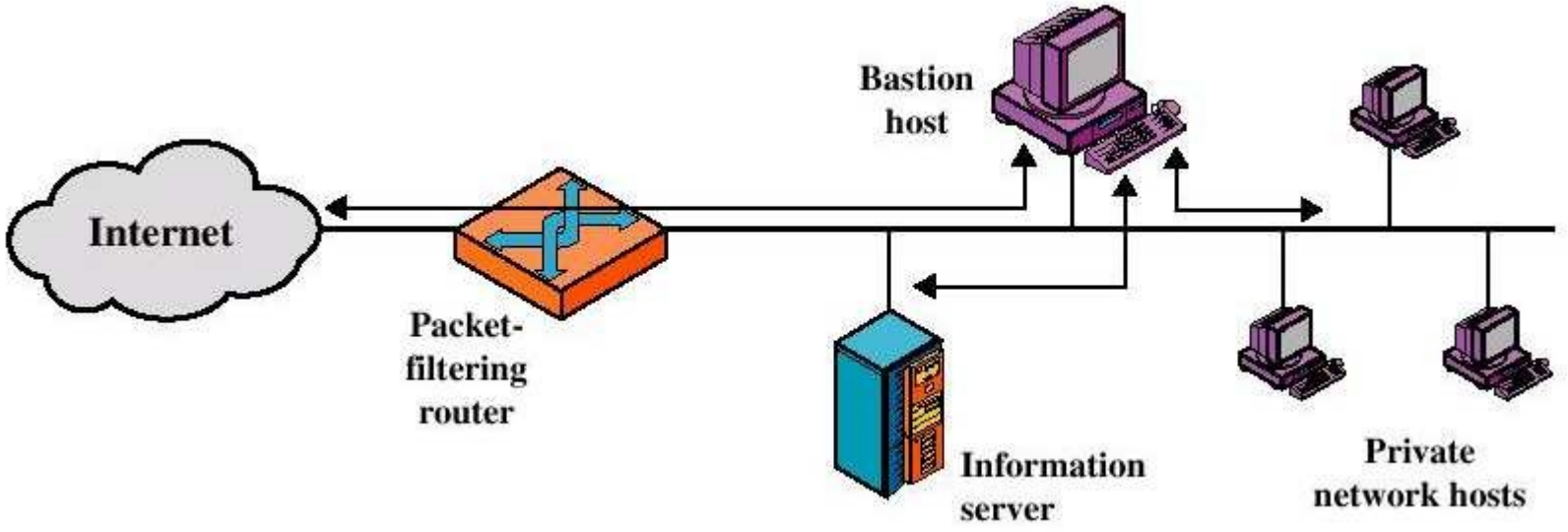
- La situazione più semplice è quella

  (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)

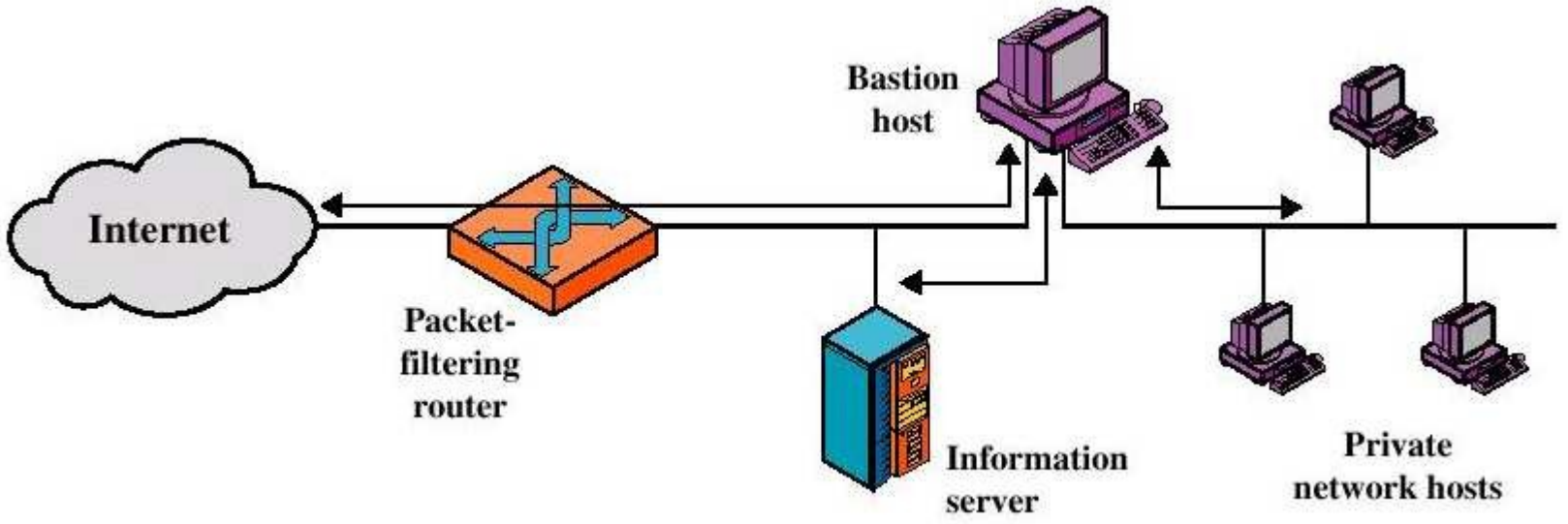


Topologie – screened single-homed BH

- 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

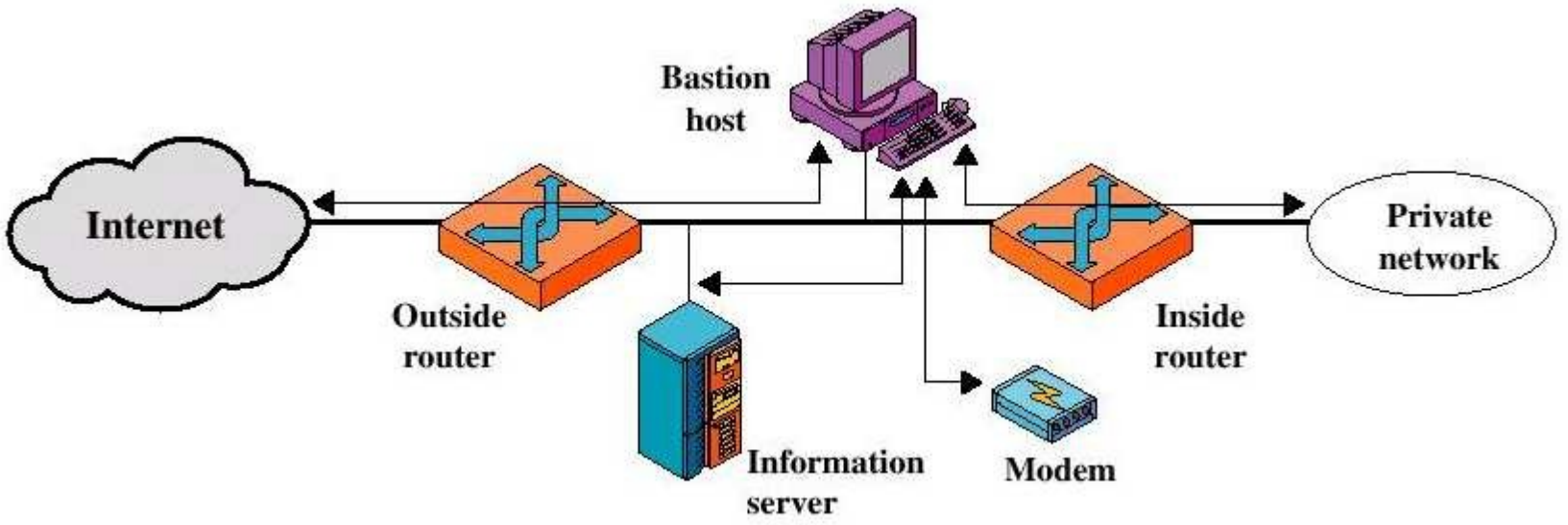
Topologie – screened dual-homed BH

- 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



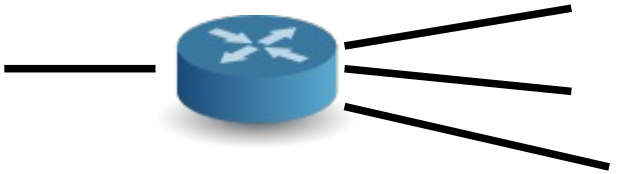
Topologie – screened subnet

- 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

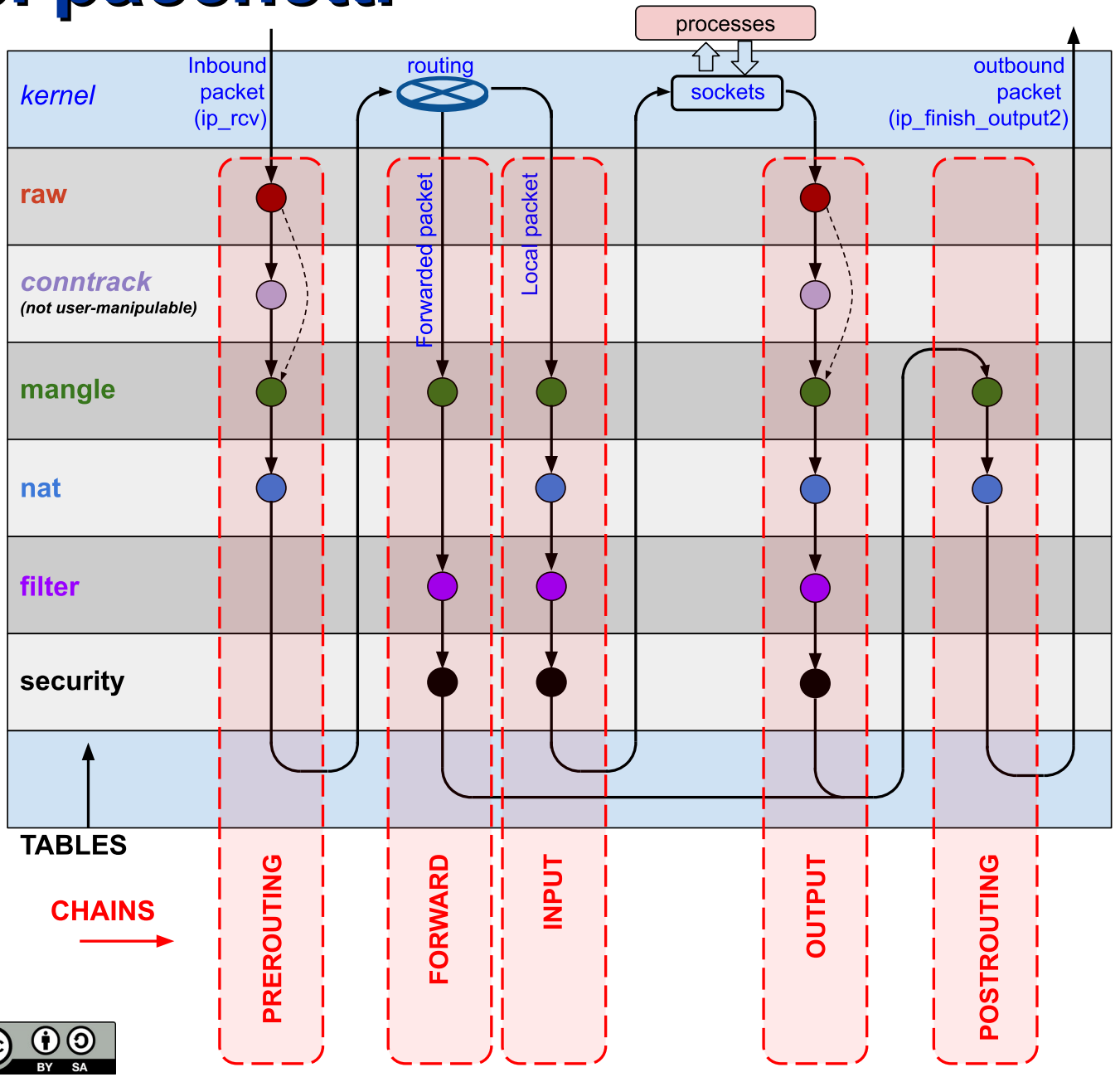
- corrispondono esattamente agli hook di netfilter
  - PREROUTING: attivata dall'hook NF_IP_PRE_ROUTING
    - regole da applicare appena il pacchetto entra
  - INPUT: attivata dall'hook NF_IP_LOCAL_IN
    - regole da applicare prima di consegnare il pacchetto a un processo
  - FORWARD: attivata dall'hook 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

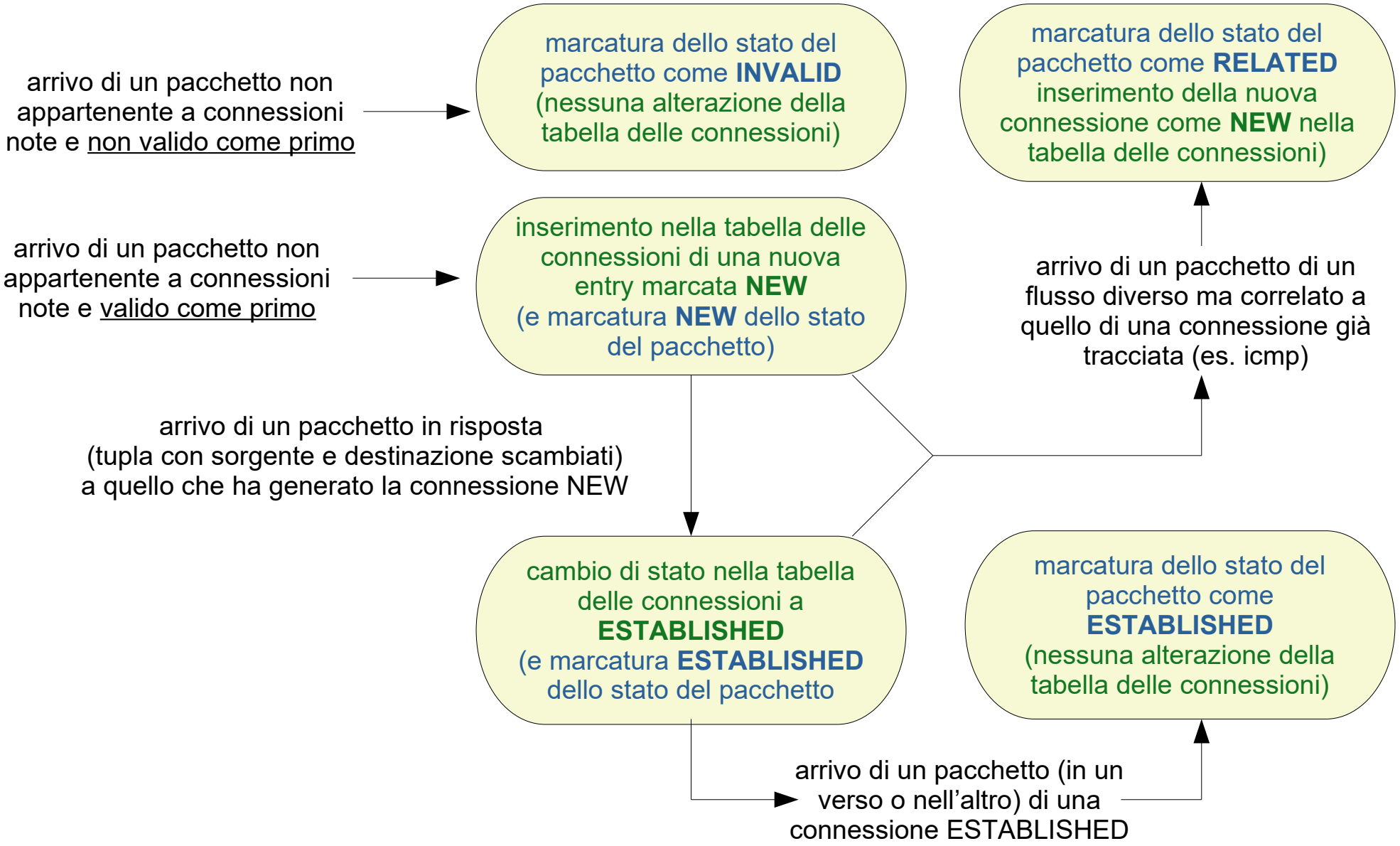
```
iptables -vnxL > contatori
iptables -Z
```

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
```

Offensive security

- Porsi nel ruolo degli attaccanti
  - verificare l'esistenza di vulnerabilità
  - stimare con precisione l'impatto degli attacchi
  - testare l'efficacia delle contromisure
- Reconnaissance = primo anello della kill chain

  https://attack.mitre.org/tactics/TA0043/
- In questa parte del corso
  - enumerazione
  - scansione
  - brute forcing
  - vulnerability assessment

  limitatamente all'esposizione dei sistemi

Offensive security!

- Usare le stesse tecniche degli attaccanti è delicato
- MAI farlo su risorse non proprie senza permesso
- "permesso" è un termine da definire in modo ampio
  - semplici grattacapi da operazioni sospette
  - conseguenze legali
  - effetti imprevisti anche in buona fede
  - effetti sulle reti attraversate per raggiungere l'obiettivo lecito
- Scopo, efficacia ed efficienza dei test
  - velocità o precisione?
  - ricerca esaustiva di vulnerabilità o verifica della sensibilità dei sistemi di rilevazione?

Testing

- Fondamentale per
  - verificare se sono sfuggite vulnerabilità
  - verificare se il sistema è esposto a rischi nuovi rispetto al momento della progettazione
- Problema concettuale: copertura
  - Non su può dimostrare l'assenza di problemi
  - Solo tentare di sollecitare il sistema nel modo più completo possibile per trovare eventuali problemi esistenti
- Tre livelli di approfondimento
  - Vulnerability Assessment
  - Penetration Testing
  - Red Team Operations

VA

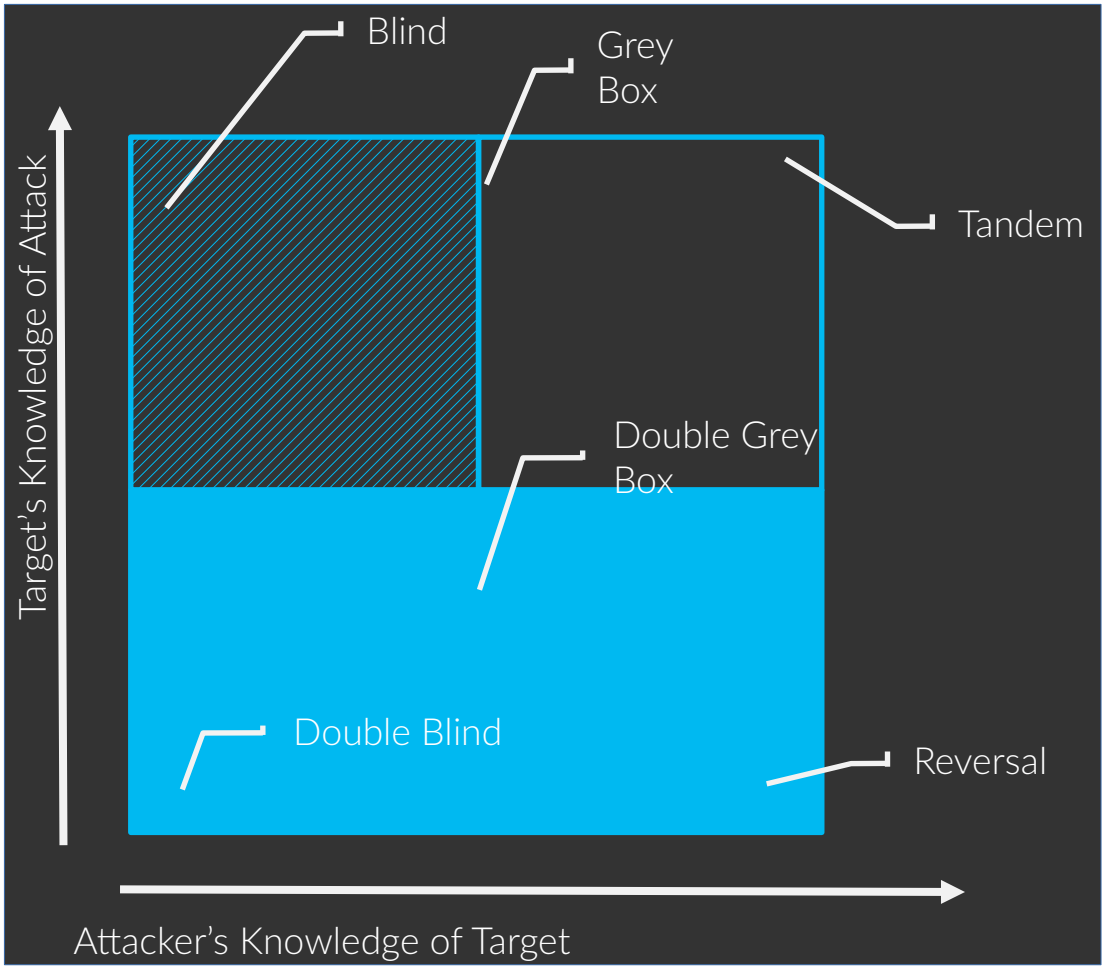
- La comunità pubblica le vulnerabilità scoperte, secondo un principio di *responsible disclosure*
- Esistono database human e machine-readable, es.
  - Common Vulnerabilities and Exposures http://cve.mitre.org/
  - National Vulnerability Database http://nvd.nist.gov/
  - Open Sourced Vulnerability Database http://osvdb.org/
  - SecurityFocus http://www.securityfocus.com/vulnerabilities
  - US-CERT http://www.kb.cert.org/vuls/
- Esistono software per cercarle sui sistemi
  - es. OpenVAS – dettagli in seguito
- Esistono database di exploit pronti per sfruttarle
  - https://www.cvedetails.com/
  - https://www.exploit-db.com/
  - https://packetstormsecurity.com/

VA → PT

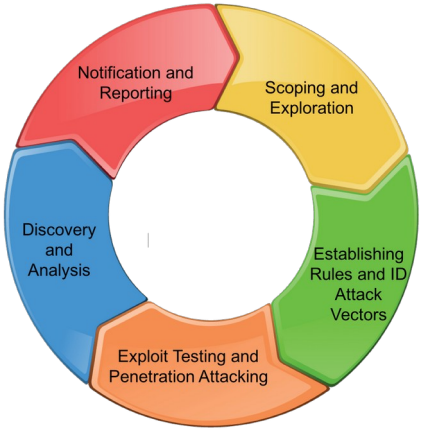
- VA trova solo vulnerabilità note
- Non procede oltre
  - Sfruttando una vulnerabilità si potrebbe accedere a una vista più interna e approfondita del sistema, svelandone altre
- Non considera la specificità del sistema
  - Anche falsi positivi, es. servizi che dichiarano una versione vulnerabile ma sono stati corretti
- PT: il tester (umano) avanza fin dove può, sfruttando le vulnerabilità per mezzo di exploit
  - Più realistico
  - Report più dettagliato
  - RISCHIOSO

PT - punti di partenza

- **Valutazione del target**
  - vengono stabilite le regole di ingaggio
  - mappatura, prioritizzazione, tracciamento dei confini
- **Postura e visibilità**
  - gli attacchi ciechi possono sembrare più realistici, ma fanno solo perdere tempo al tester esperto che è meglio spendere sui dettagli veramente nascosti
- **Protezione del bersaglio**
  - dove possibile, viene creata una replica per evitare di danneggiare il bersaglio, ma...
  - alcuni sistemi sono semplicemente troppo complessi
  - alcuni sistemi sono troppo critici per rischiare di perdere qualche dettaglio nella replica che potrebbe alterare il test



PT - metodologie



- Seguire una metodologia consente di
  - assicurarsi che il test sia coerente e ripetibile
  - eseguire una misurazione accurata della sicurezza (nessun pregiudizio o ipotesi o prove aneddotiche)
- Esistono alcune metodologie generalmente accettate:
  - Open Source Security Testing Methodology Manual (OSSTMM)
    - consente a qualsiasi tester di sicurezza di fornire idee per eseguire test di sicurezza più accurati, attuabili ed efficienti.
    - consente la libera diffusione delle informazioni e della proprietà intellettuale
- Open Web Application Security Project (OWASP)
  - specifico per applicazioni web
- Payment Card Industry Data Security Standard (PCI DSS)
  - settore finanziario; la sezione 11.3 riguarda il pentesting
- Technical Guide to Information Security Testing and Assessment (NIST800-115)
  - uno standard ufficiale del governo degli Stati Uniti
- Information Systems Security Assessment Framework (ISSAF)
  - completo ma non sviluppato attivamente

Preparazione

- Reconnaissance
  - raccolta di informazioni utili
  - estensione del perimetro di test
  - preparazione degli strumenti
- Enumeration
  - delimitazione del perimetro di test
  - verifica puntuale delle risorse e delle loro proprietà

OSINT

- Open Source INTelligence
  - L'uso di qualsiasi fonte pubblicamente disponibile per ricavare informazioni su di uno specifico obiettivo
  - Un campo di applicazione più ampio rispetto alla cybersecurity!
- OSINT su altri (vedremo più avanti)
  - componente della threat intelligence
  - componente dell'incident response
- OSINT su se stessi
  - cosa possono scoprire gli avversari?
  - come possono essere usate queste informazioni?
- È legale?
  - sostanzialmente sì
  - attenzione alle aree grigie

    https://mediasonar.com/2020/03/11/10-tips-for-doing-osint-legally/

OSINT – strumenti e fonti online

- https://osintframework.com/
- Ad esempio, per misurare l'esposizione dell'infrastruttura
  - collocazione fisica
    - geolocation
    - rilevazione di indirizzi da documenti e pagine web
  - collocazione in rete
    - domini DNS associati all'obiettivo
    - range di indirizzi IP
    - provider di connettività e autonomous systems
    - certificati X.509
  - accesso ai servizi
    - porte raggiungibili
    - fingerprinting dei sistemi → anche notoriamente vulnerabili
    - username validi → anche con relative password

Raccolta di informazioni – DNS

- I record DNS possono svelare
  - gli IP registrati dall'obiettivo
  - l'esistenza e la collocazione di specifici server applicativi
  - l'esistenza di sottoreti non direttamente raggiungibili
  - alias per sistemi collocati al di fuori del perimetro dell'obiettivo
    - risorse in cloud
    - sistemi legati da relazioni di fiducia es. domini di una foresta Active Directory
- Questo consente un notevole risparmio di tempo rispetto alla forza bruta
- Aggravanti
  - plateali: abilitazione di domain transfer
  - sottili: permanenza di record rimossi nelle cache
- Strumenti
  - lookup di base: `host`, `dig`, `nslookup`
  - strumenti di ricerca che includono guessing e forza bruta: `dnsenum`, `dnsmap`, `dnsrecon`, `fierce`, ...

Raccolta di informazioni – IP blocks

- La conoscenza dei dettagli organizzativi o di pochi indirizzi IP validi può permettere di espandere la conoscenza
  - agli interi blocchi allocati all'obiettivo
  - ad altri blocchi non evidentemente collegati
- Esempio
  - da www.unibo.it riuscite a risalire a tutte le reti degli enti di ricerca e delle università italiane?
  - hint: strumenti di ricerca RIPE

Enumerazione – host

- Una volta individuati i blocchi di indirizzi da analizzare si procede con l'individuare gli host effettivamente attivi (live host)
- Banale `ping`
  - 1 indirizzo per volta
  - bloccato da router e firewall?
  - ignorato da host?
    - scansione mirata ai servizi (in seguito)
- Scansioni massive
  - `masscan`
- Su rete locale più strumenti
  - sniffing passivo (`wireshark`, `tshark`, `tcpdump`, ...)
  - `arping`

Enumerazione – servizi

- Determinati gli host raggiungibili, si cercano le porte aperte
  - le due fasi possono collassare in una, se si sospetta che gli host interessanti ignorino i ping → test di vitalità fatto direttamente sondando le porte
- Il tool più diffuso: `nmap`
  - scansione contemporanea di range di indirizzi e porte
    - set predefinito di porte "più popolari"

      https://nmap.org/book/port-scanning.html
  - diverse tipologie di scansione
  - fingerprinting del sistema operativo e delle versioni dei servizi
- Alcuni vantaggi di `unicornscan` su `nmap`
  - fingerprinting più affidabile
  - relativamente più veloce
  - può salvare le risposte per analisi con altri strumenti

Tentativi di accesso ai servizi

- Analisi dei protocolli applicativi più comuni
  - SMB, SMTP, SNMP
  - può portare ad accesso a dati o a raccolta di ulteriori informazioni per le fasi successive
- Brute forcing applicativo (fuzzing)
  - invio di payload randomizzati per tentare di sollecitare risposte impreviste (Es. `bed`, `doona`, vari tool per SIP, …)
- Framework per lo sviluppo e l'esecuzione di exploit

Permessi e ACL

- Ricerche tipiche di file che possono causare problemi
- Bit SUID/SGID

  ```
  find / -type f -perm /6000
  ```

- File scrivibili da tutti

  ```
  find / -perm /2
  ```

- File che non sono di proprietà di alcun utente valido

  ```
  find / -nouser
  ```

- Per le ACL è più complicato, si può partire da

  ```
  getfacl -R /
  ```

Capabilities

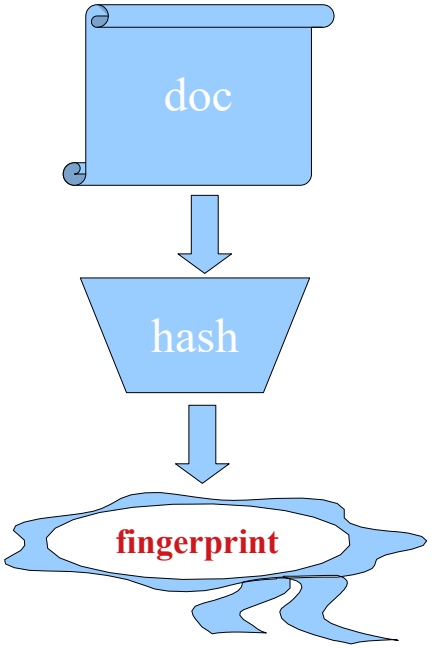
- Le capabilities sono ciò che distingue root dagli utenti standard
  - root le ha tutte (~40)
  - possono essere assegnate singolarmente a un processo per mezzo di attributi estesi associati al programma
- Vediamole per capire quali sono le più pericolose

  https://man7.org/linux/man-pages/man7/capabilities.7.html
- Ricerca di file con capabilities settate:

  ```
  getcap -r /
  ```

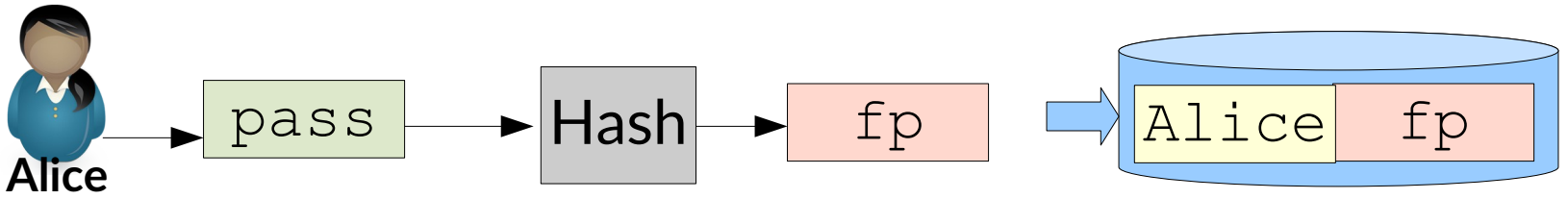
Utenti

- Le credenziali degli utenti sono memorizzate in file del sistema
  - protetti dai permessi
  - contenenti non le password in chiaro, ma le loro *impronte* generate da un algoritmo di *hash*
- Approfondiremo il tema quando parleremo di crittografia, per ora sintetizziamo e accettiamo "a scatola chiusa" che una funzione hash
  - Produce un'impronta di dimensione fissa a partire da un input arbitrario (quindi non è direttamente invertibile)
  - È costruita in modo che dedurre l'input originale dall'impronta sia pressochè impossibile
  - È costruita in mdo che produrre due documenti che abbiano la stessa impronta sia pressoché impossibile

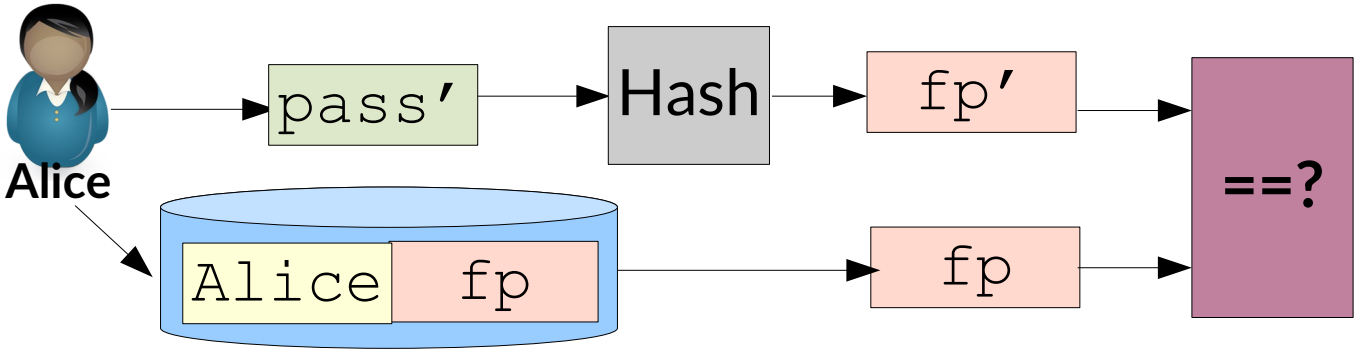


Password - semplificato

- Scelta della password



- Verifica della password



- Se le ipotesi sulla funzione hash sono corrette, non c'è modo più efficiente per dedurre una password che tentare di indovinarla

Attacco alle credenziali utente

- Il più classico degli assessment: robustezza delle password
- Nella posizione più vantaggiosa (root) si potrebbero impersonare tutti gli utenti
  - perché rubare password?
  - utilizzo frequente su altri sistemi!
- Password cracking a forza bruta
  - interattivo → lento, tendente all'impossibile
  - avendo gli hash → ricerca con rainbow tables
    http://project-rainbowcrack.com/
    - compromesso spazio-tempo da valutare
- Password cracking con dizionario
  - John the ripper https://www.openwall.com/john/
    - wordlist enormi disponibili online
  - costruzione di wordlist su misura per caratteristiche note dell'obiettivo
    https://github.com/Mebus/cupp
    https://github.com/digininja/CeWL

Normalmente contenuti in file non leggibili dall'utente standard, ma potrebbero essere esfiltrati se è presente una vulnerabilità che consente una privilege escalation

File e socket accessibili

- `lsof` lists open files
  - TCP and UDP sono solo namespace differenti

```
# lsof -i -n | egrep 'COMMAND|LISTEN|UDP'
COMMAND    PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd      2317 root    3u  IPv6   6579       TCP *:ssh (LISTEN)
xinetd    2328 root    5u  IPv4   6698       TCP *:auth (LISTEN)
sendmail  2360 root    3u  IPv4   6729       TCP 127.0.0.1:smtp (LISTEN)
```

  - provate `lsof | grep '(deleted)'`

Punti di acccesso esposti via rete

- `netstat` mostra lo stato delle socket Unix e di rete
  - di default, già connesse a un altro endpoint
    - opzione `-l` : listening
    - opzione `-a` : entrambe le categorie
  - altre opzioni utili
    - `-p` processo in ascolto
    - `-n` output numerico
    - `-t` tcp socket
    - `-u` udp socket
- `ss` è il rimpiazzo più recente, però non è SELinux-aware

netstat sample output

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:901             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:904             0.0.0.0:*               LISTEN      11325/rpc.mountd
tcp        0      0 0.0.0.0:3689            0.0.0.0:*               LISTEN      11438/mt-daapd
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      20600/mysqld
tcp        0      0 0.0.0.0:3690            0.0.0.0:*               LISTEN      11441/mt-daapd
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      3717/smbd
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      2953/portmap
tcp        0      0 0.0.0.0:6001            0.0.0.0:*               LISTEN      14660/Xrealvnc
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      3457/inetd
tcp        0      0 137.204.58.80:993       137.204.58.138:51929    ESTABLISHED 8190/imapd
```

Processi a orologeria

- Sono un modo per garantire persistenza
  - presenti tra i processi solo quando necessario
  - riavviati dopo terminazione o reboot
- Eseguiti periodicamente
  - crontab di ogni utente
  - crontab di sistema
- Accodati per l'esecuzione ritardata
  - spool del demone atd

Iniezione di software

- Non banale ma estremamente impattante
- I sistemi di package management prendono, di default, sempre l'ultima versione di ogni pacchetto
- Aggiungere repository è un rischio
  - spesso lo si fa in modo legittimo per installare un'applicazione magari semisconosciuta ma innocua
  - un pacchetto messo nel repository "minore" potrebbe sostituirne uno cruciale con lo stesso nome
    - repo meno sorvegliati
    - repo senza firma digitale → MITM
- Verificare se esiste l'opportunità di iniettare pacchetti
  - file di configurazione delle sorgenti
  - keyring per la verifica delle firme
  - utilizzo dei tool di package management della distribuzione

Strumenti di ricerca locali

- Una miriade di altre possibili vulnerabilità o semplici informazioni utili per test più efficaci
- Servono strumenti di scansione approfondita, es.
  https://github.com/rebootuser/LinEnum
  - Informazioni sul sistema
  - Informazioni sugli utenti
  - Esecuzione automatica di programmi
  - Servizi installati e in esecuzione
- Esempi di impostazioni insicure riportate
  - default umask
  - permessi e capabilities
  - dati sensibili nella history, env vars, …
  - credenziali di default
- Non necessariamente coprono tutto!

Strumenti di ricerca completi

- Esistono scanner di vulnerabilità completi
  - configurabili per eseguire la scansione di una combinazione arbitraria di host e porte
  - possono testare l'esistenza di vulnerabilità a livello di rete, sistema operativo e applicazione tramite plug-in caricabili
  - possono collegare ogni vulnerabilità alla documentazione pertinente (ad esempio CVE)
- Il più noto è Nessus, un prodotto commerciale di Tenable, attualmente alla versione 8
  - www.nessus.org
- OpenVAS è l'open-source equivalente, essendo un fork di nessus 2.2 avviato nell'agosto 2008 e attivamente sviluppato da allora
  - www.openvas.org

OpenVAS – caratteristiche principali

- architettura
  - scan engine (svolge i test)
  - manager (coordina i task di scansione)
  - interfaccia (pianifica i task per il manager, mostra i risultati)
  - amministrazione (gestisce utenti e database)
- si appoggia su di un vulnerability database
  - Network Vulnerability Tests
    - descrizione della vulnerabilità
    - piattaforme colpite
    - processo di verifica
  - aggiornato ogni giorno!