# Seguridad perimetral: sistemas de detección de ataques

## Seguridad en Redes de Ordenadores

Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación (GSyC)

Abril de 2018

# Contenidos

# Contenidos

# Nomenclatura

- **Vulnerabilidad**: fallo de seguridad que permite a un intruso conseguir acceder a recursos para los cuáles no tiene permiso de acceso.
- **Exploit**: aplicaciones cuyo objetivo es aprovecharse de una determinada vulnerabilidad para acceder a los recursos para los que no se tiene permiso de acceso.

# Introducción

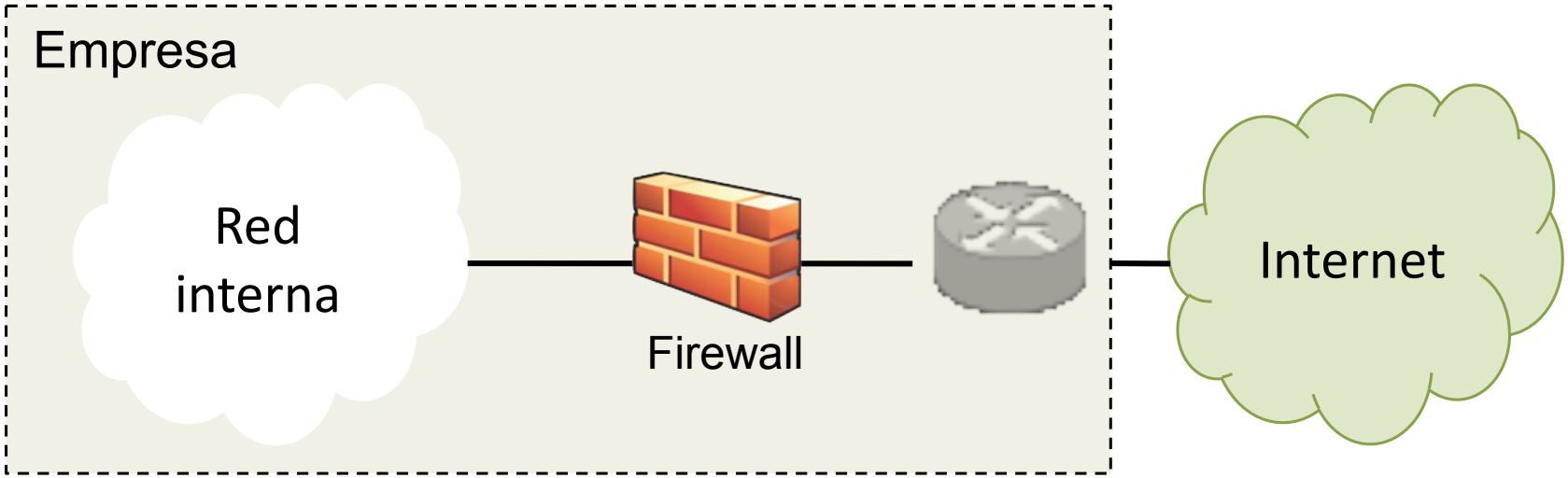
- La seguridad perimetral consiste en los elementos de red que proveen de seguridad al perímetro de una red interna frente a posibles ataques desde el exterior.
  - Cortafuegos (firewalls)
  - Sistemas de detección de intrusos y prevención de intrusos
  - Honeypots

# Contenidos

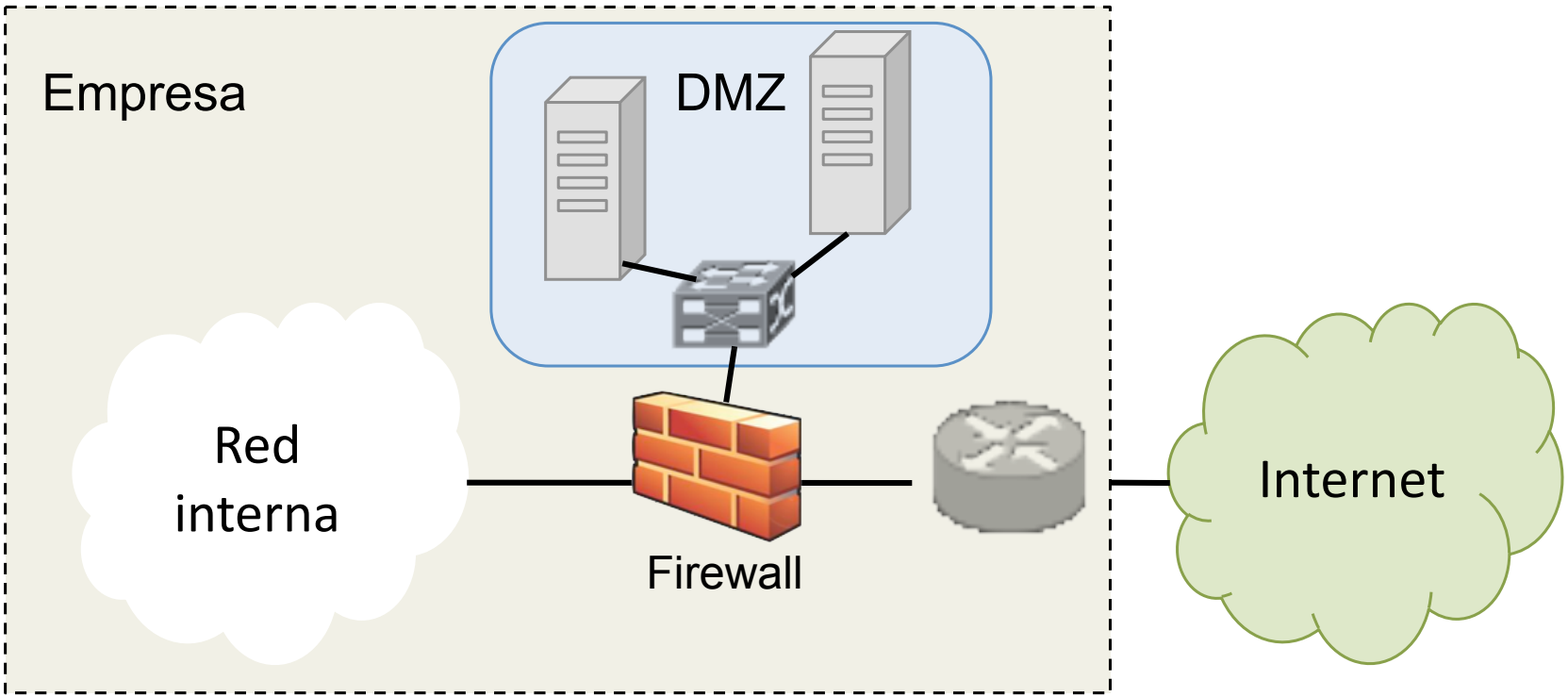
# La red frontera

- La red frontera es la parte de la red que comunica la red interna de una empresa con otras redes externas.
- La seguridad en la red frontera es clave para proteger los equipos y servicios de la empresa de ataques externos. Para ello, las empresas instalan *firewalls* que permiten filtrar el tráfico y detectar posibles ataques maliciosos desde el exterior. Adicionalmente los *firewalls* permiten restringir el tráfico que sale de los equipos internos de la empresa.

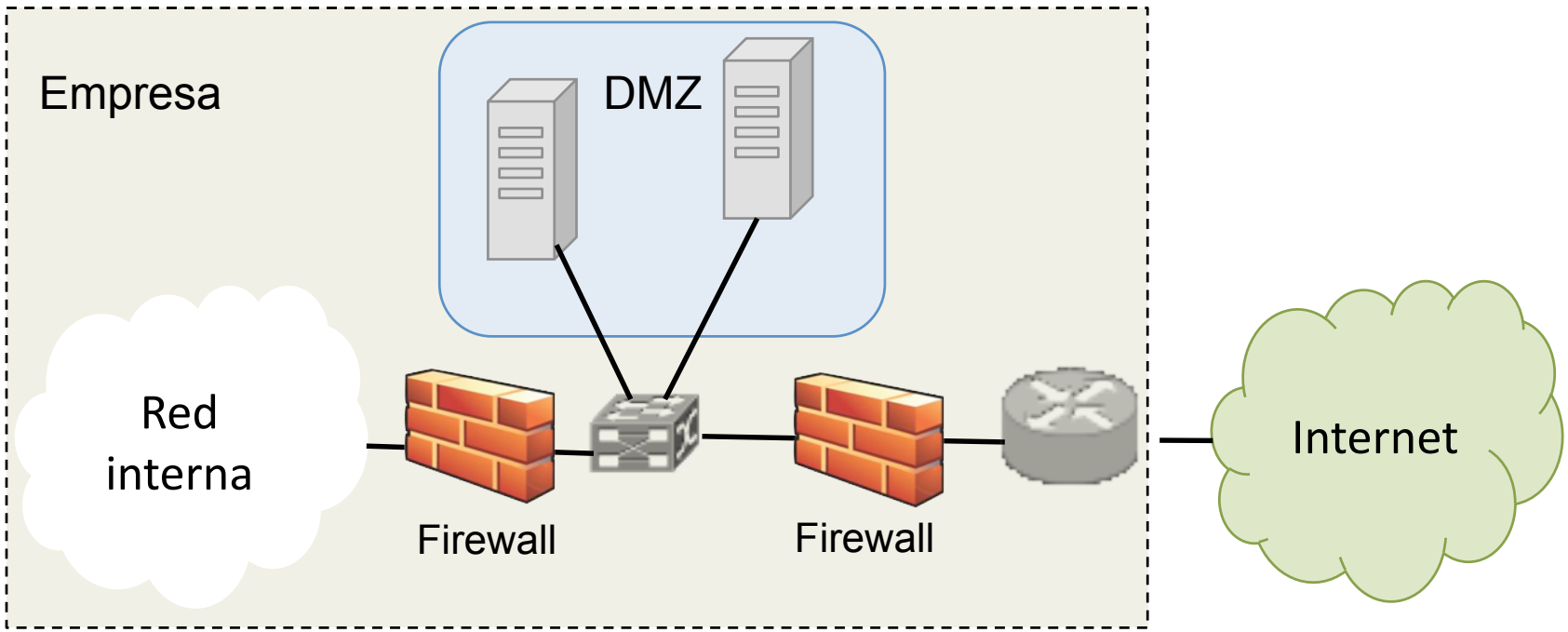
# Un único firewall



# Un único firewall con zona DMZ



# Dos firewalls



# Contenidos

# Netfilter - iptables

- Netfilter[1] es un framework de Linux que permite interceptar y modificar paquetes IP.
- iptables es una herramienta de Netfilter que permite al administrador la definición de conjuntos de reglas aplicables a los paquetes IP que entran y/o salen de una máquina para realizar las siguientes operaciones:
  - Filtrado de paquetes *(packet filtering)*.
  - Seguimiento de conexiones *(connection tracking)*.
  - Traducción de direcciones IP y puertos (NAT, *Network Address Translation*).
- Hay 3 conceptos básicos en iptables:
  - **reglas**
  - **cadenas**
  - **tablas**

[1] http://www.netfilter.org

# Reglas

- Una regla de `iptables` especifica una **condición** y una **acción**:
  - **condición**: características que debe cumplir un paquete para que la regla le sea aplicable. Ejemplos de condiciones:

    `-p tcp --dport 80`: el protocolo es TCP y el puerto destino es 80

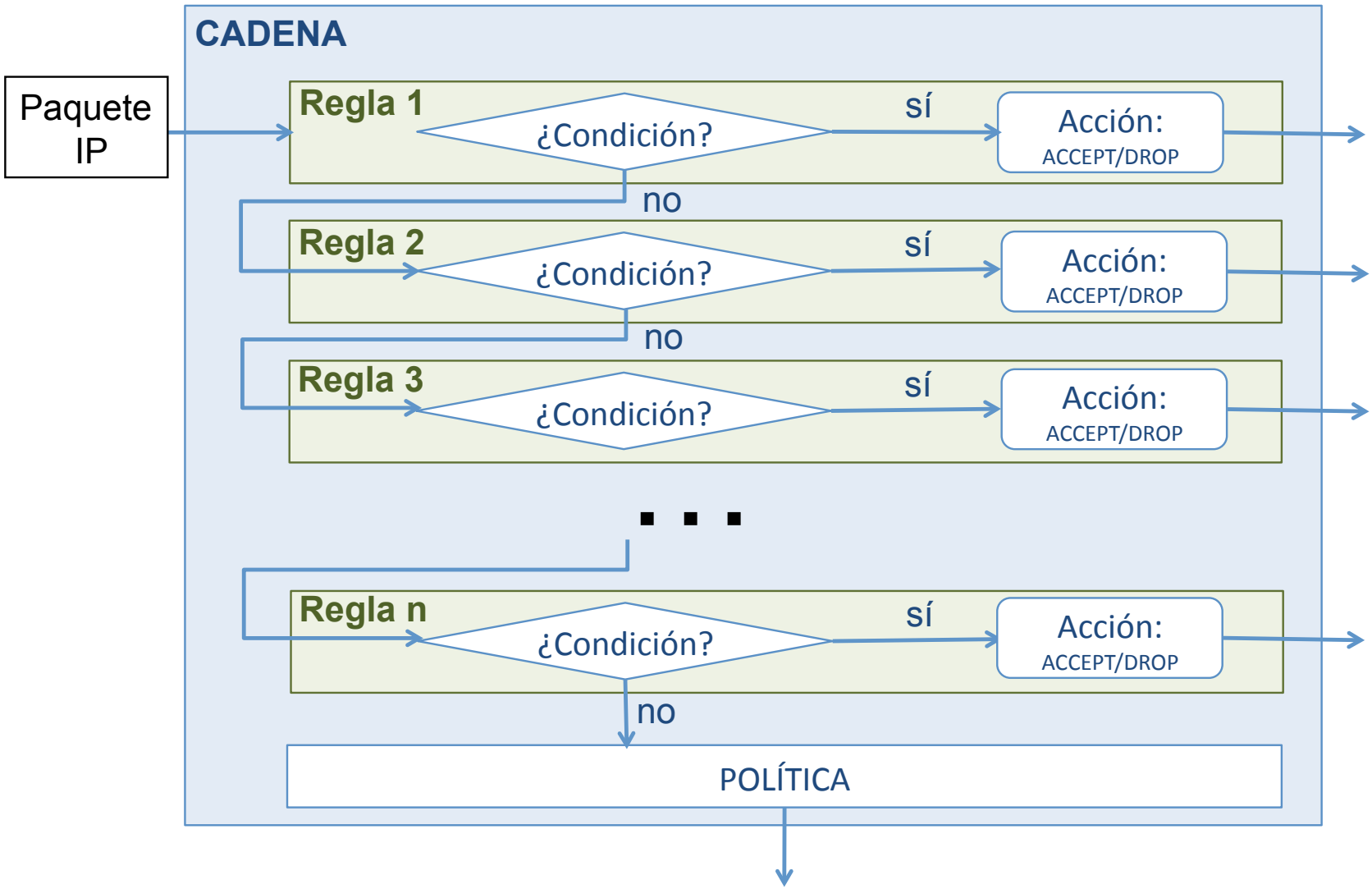
    `-s 13.1.2.0/24`: la dirección de origen es de la subred 13.1.2.0/24.
  - **acción**: indica lo que se hace con el paquete si cumple la condición de la regla. Ejemplos de acciones:

    `ACCEPT`: el paquete se acepta
    `DROP`: el paquete se descarta
    `SNAT --to-source 13.1.2.1`: se cambia la IP origen del paquete
- Las reglas se agrupan en listas de reglas, llamadas cadenas.
- Las cadenas se agrupan en tablas.

# Cadenas (I)

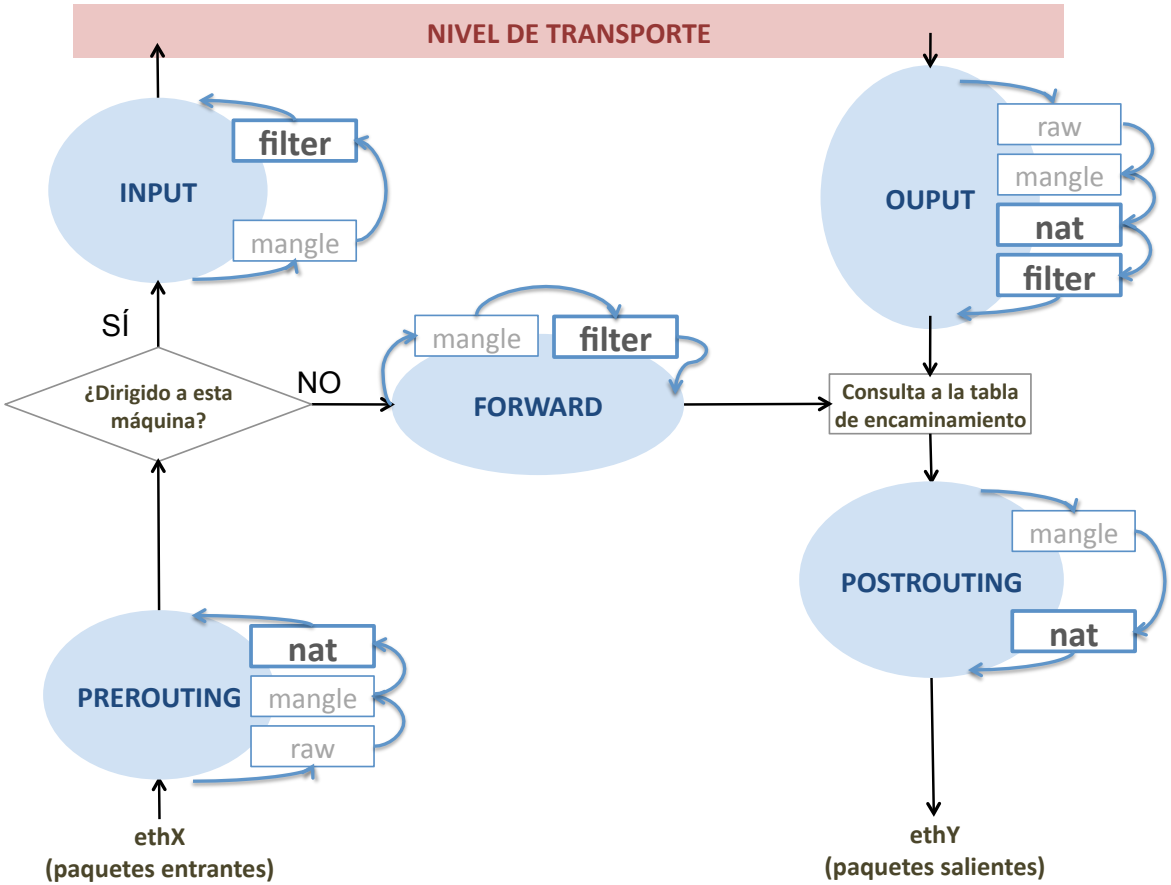
- Una cadena es una **lista ordenada de reglas**.
- Para cada paquete se va comprobando si se le aplica cada regla de la cadena (es decir, si cumple la **condición**):
  - Si una regla NO se aplica a un paquete, se pasa a la siguiente regla de la cadena.
  - Si una regla se aplica a un paquete, se ejecuta la **acción** definida en dicha regla. Dependiendo del tipo de acción:
    - El paquete abandona la comprobación del resto de las reglas y pasa a la siguiente cadena (acciones ej: ACCEPT, DROP)
    - El paquete continúa con la siguiente regla de la cadena (acción ej: LOG)
- Una cadena puede tener definida una política, que es la **acción por defecto** para la cadena. La política predefinida para todas las cadenas predefinidas es `ACCEPT` (es decir, aceptar el paquete).
- Cuando para un paquete NO se aplica NINGUNA de las reglas de la cadena, se ejecuta para él la política de la cadena.

# Cadenas (II)


CADENA
Paquete IP
Regla 1
¿Condición?
sí
Acción:
ACCEPT/DROP
no
Regla 2
¿Condición?
sí
Acción:
ACCEPT/DROP
no
Regla 3
¿Condición?
sí
Acción:
ACCEPT/DROP
Regla n
¿Condición?
sí
Acción:
ACCEPT/DROP
no
POLÍTICA

# Cadenas (III): Tipos de cadenas

- Existen diferentes tipos de cadenas:
  - Predefinidas: `PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT`, `POSTROUTING`
  - Definidas por el usuario. Dichas cadenas no tienen **política** predefinida.
- Cuando un paquete llega a una máquina se le aplican las reglas de las cadenas predeterminadas en distintos momentos según el esquema de la figura



# Cadenas (IV): Cadenas predefinidas

- Cadena `PREROUTING`:
  - Reglas que se aplican a los paquetes que llegan a la máquina. Esta cadena se ejecuta antes de comprobar si el paquete es para la propia máquina o hay que reenviarlo.
- Cadena `INPUT`:
  - Reglas que se aplican a los paquetes destinados a la propia máquina. Esta cadena se ejecuta justo antes de entregarlos a la aplicación local.
- Cadena `FORWARD`:
  - Reglas que se aplican a los paquetes que han llegado a la máquina pero van destinados a otra y hay que reenviarlos. Esta cadena se ejecuta antes de consultar la tabla de encaminamiento.
- Cadena `OUTPUT`:
  - Reglas que se aplican a los paquetes creados por la propia máquina. Esta cadena se ejecuta justo después de que la aplicación le pase los datos a enviar al *kernel* del sistema operativo y antes de consultar la tabla de encaminamiento.
- Cadena `POSTROUTING`:
  - Reglas que se aplican a los paquetes que salen de la máquina, tanto los creados por ella como los que se reenvían. Esta cadena se ejecuta después de consultar la tabla de encaminamiento.

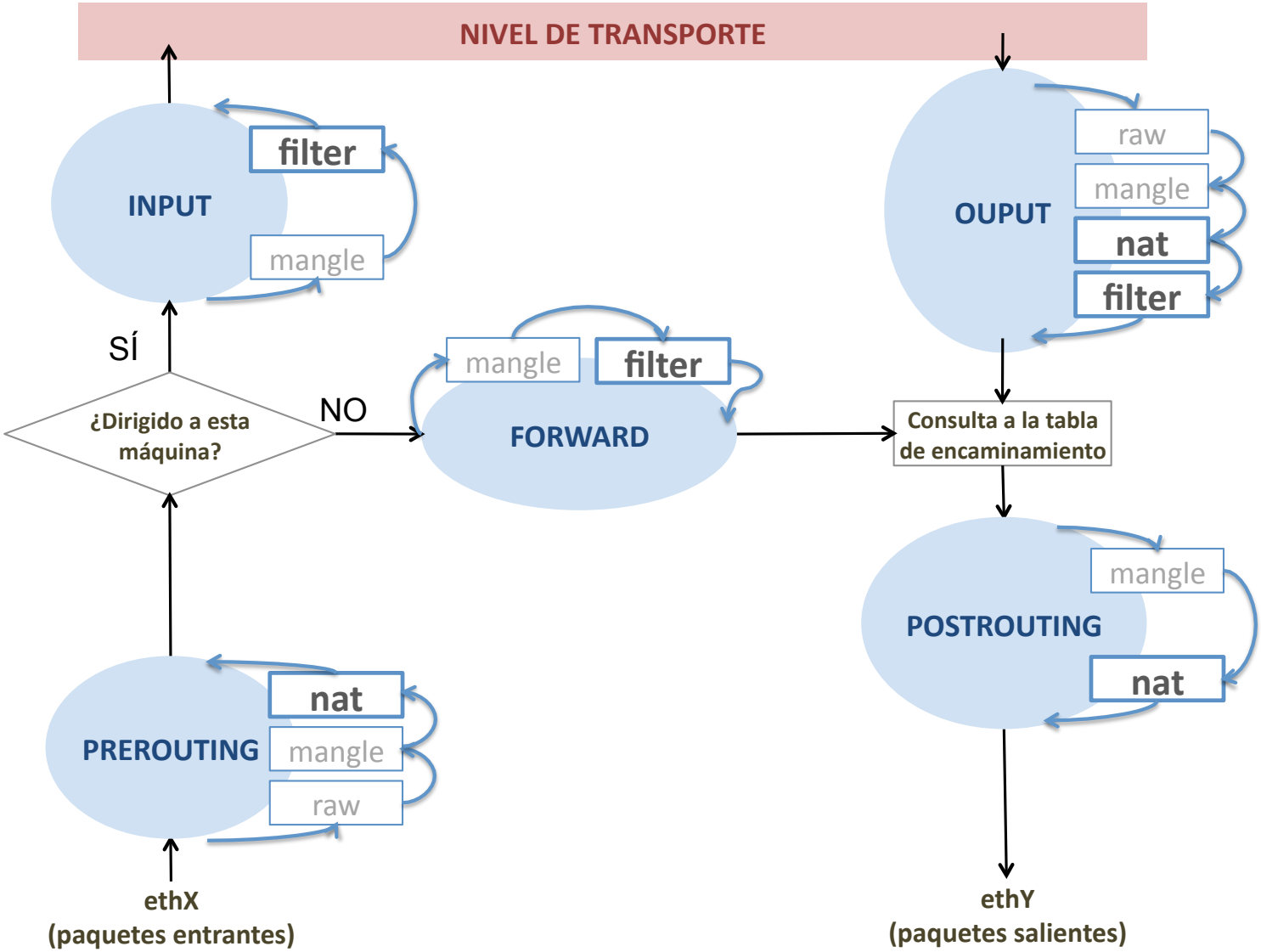
# Tablas (I)

- Una tabla de iptables contiene un conjunto de cadenas, tanto predefinidas como de usuario.
- Una tabla concreta engloba las reglas (agrupadas en cadenas) relacionadas con un tipo de procesamiento de los paquetes.
- Netfilter define las siguientes tablas:
  - filter: engloba las reglas de filtrado de paquetes, es decir, de las que deciden que un paquete continúe su camino o sea descartado.
  - nat: engloba las reglas de modificación de direcciones IP y puertos de los paquetes
  - mangle: engloba las reglas de modificación de algunos campos de las cabeceras del paquete. Ejemplo: ToS
  - raw: engloba las reglas que permiten marcar excepciones al seguimiento que hace el *kernel* de las comunicaciones de la máquina.

# Tablas (II): Cadenas predefinidas de cada tabla

- La tabla `filter` incluye las cadenas:
  - `FORWARD`
  - `INPUT`
  - `OUTPUT`
- La tabla `nat` incluye las cadenas:
  - `PREROUTING`
  - `OUTPUT`
  - `POSTROUTING`
- La tabla `mangle` incluye las cadenas:
  - `PREROUTING`
  - `FORWARD`
  - `INPUT`
  - `OUTPUT`
  - `POSTROUTING`
- La tabla `raw` incluye las cadenas:
  - `PREROUTING`
  - `OUTPUT`

# Movimiento de los paquetes por tablas y cadenas



# Seguimiento de "conexiones" (I)

- Las "conexiones" (en sentido amplio) de las comunicaciones TCP, UDP, ICMP que atraviesan una máquina se pueden monitorizar a través del módulo `conntrack` de `iptables`.
- El **estado** en el que puede estar una determinada "conexión" es:
  - **ESTABLISHED**: el paquete está asociado a una "conexión" donde se han transmitido paquetes en ambos sentidos. Por ejemplo: conexión TCP.
  - **NEW**: el paquete está asociado a una nueva conexión o a una conexión en la que no se han transmitido paquetes en ambos sentidos
  - **RELATED**: el paquete está relacionado con una conexión existente pero no pertenece a ella (ej: datos FTP, ICMP error).
  - **INVALID**: el paquete no puede ser identificado o no está asociado a ningún estado.
- Los cálculos de seguimiento se realizan en la cadena PREROUTING (para los paquetes que recibe y reenvía el router) o en la cadena OUTPUT (para los paquetes que crea el router).

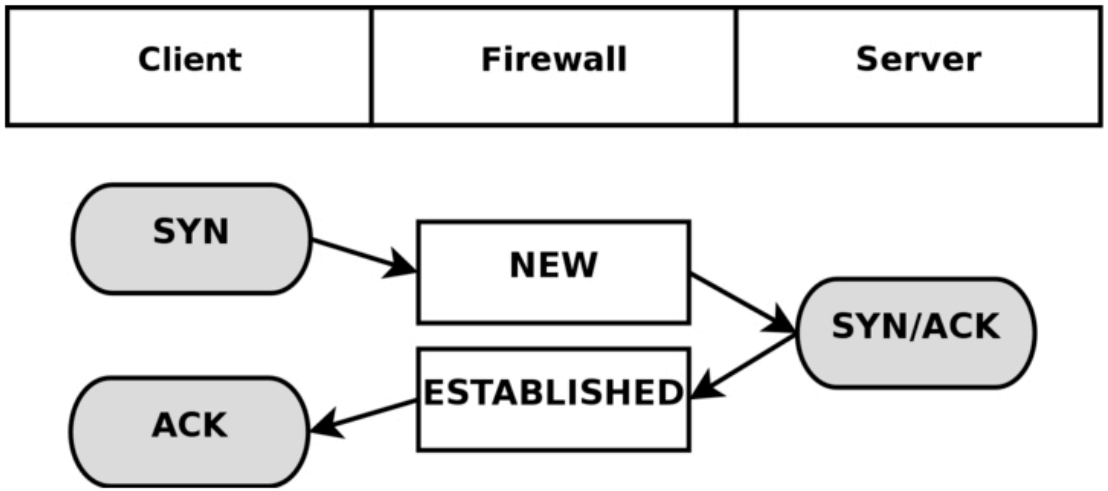
# Seguimiento de "conexiones" (II)

- Para visualizar dichas conexiones hay que mostrar el contenido del fichero /proc/net/ip_conntrack.

```
r1:~# cat /proc/net/ip_conntrack
tcp      6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775 dport=22
            [UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22    dport=32775 use=2
```

- Para cada conexión se mostrará la siguiente información:
  - El primer y segundo campo muestran el protocolo utilizado (el nombre y el código).
  - El tercer campo muestra el tiempo que le queda a dicha conexión para que el sistema de seguimiento borre su entrada. Con cada paquete recibido se actualiza este campo con el valor configurado por defecto, que va disminuyendo hasta que llega a cero.
  - El cuarto y noveno campos muestran información de la "conexión": SYN_SENT, SYN_RECV, ESTABLISHED, UNREPLIED, etc. Esta información incluye el estado de la conexión según la máquina de estados de TCP, que da información más detallada que simplemente **NEW, ESTABLISHED, RELATED, INVALID** (por ejemplo, SYN_SENT implica que la conexión para `conntrack` está en estado `NEW`).
  - El resto de los campos muestran información de los paquetes en ambos sentidos.

# Conexiones TCP (I)




- El firewall recibe SYN, estado **NEW**:

```
tcp          6 117 SYN_SENT src=192.168.1.5  dst=192.168.1.35 sport=1031 dport=23
                 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5  sport=23   dport=1031 use=1
```

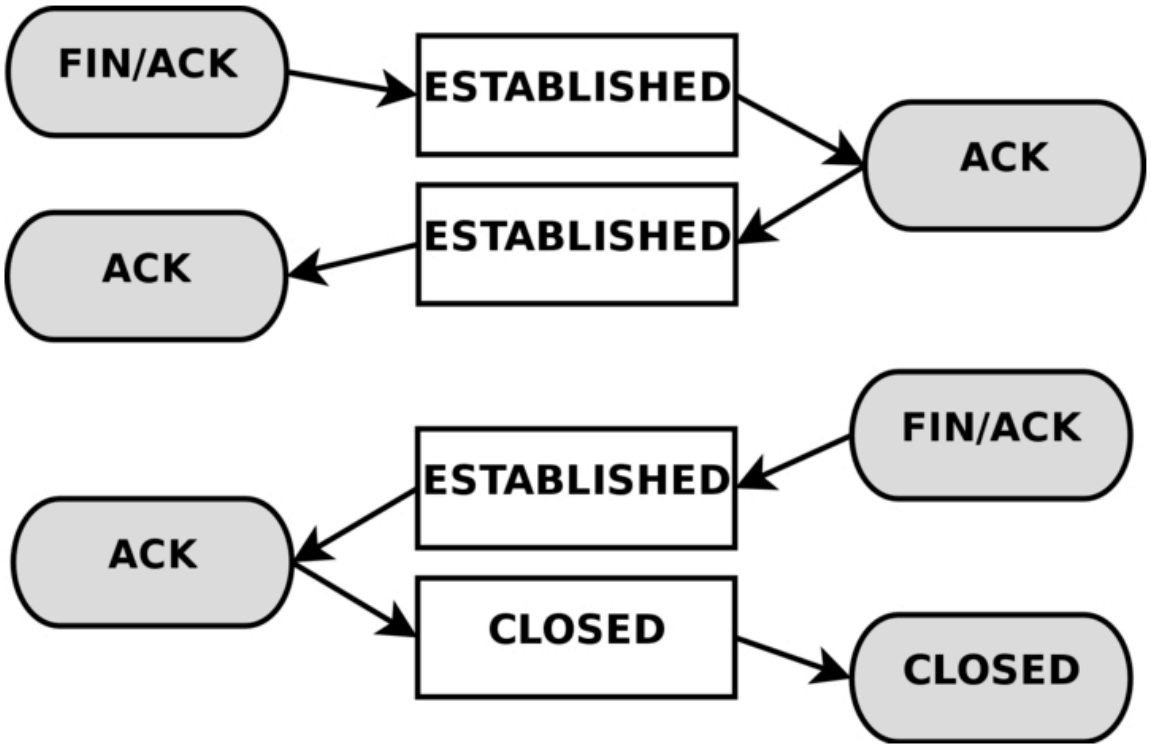
- El firewall recibe SYN/ACK, estado **ESTABLISHED**:

```
tcp          6 57 SYN_RECV src=192.168.1.5  dst=192.168.1.35 sport=1031 dport=23
                           src=192.168.1.35 dst=192.168.1.5  sport=23   dport=1031 use=1
```

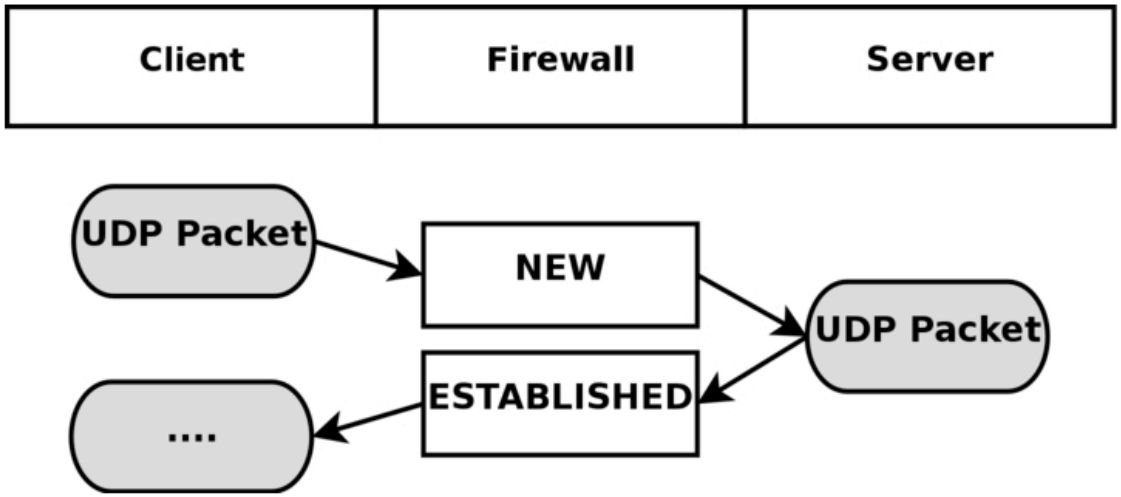
- El firewall recibe ACK y datos TCP, estado **ESTABLISHED**:

```
tcp          6 431999 ESTABLISHED src=192.168.1.5  dst=192.168.1.35 sport=1031 dport=23
                                  src=192.168.1.35 dst=192.168.1.5  sport=23   dport=1031
                                  [ASSURED] use=1
```

# Conexiones TCP (II)



# "Conexiones" UDP




- El firewall recibe un primer paquete UDP, estado **NEW**:

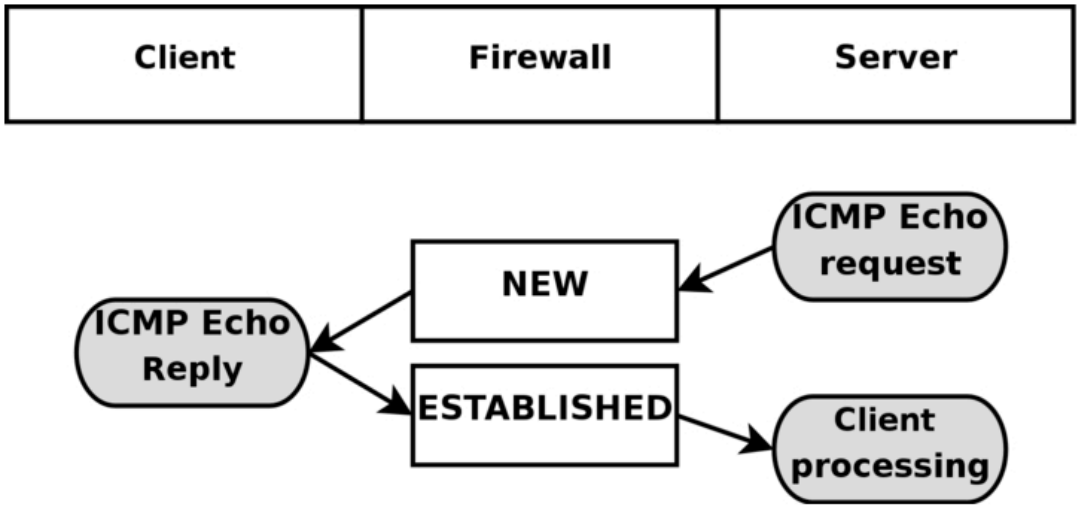
```
udp          17 20    src=192.168.1.2 dst=192.168.1.5 sport=137   dport=1025
   [UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025  dport=137 use=1
```

- El firewall recibe un paquete UDP de respuesta, estado **ESTABLISHED**:

```
udp          17 170 src=192.168.1.2 dst=192.168.1.5 sport=137  dport=1025
                    src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 [ASSURED] use=1
```

# "Conexiones" ICMP: Mensajes ICMP de diagnóstico

- Para paquetes ICMP que tienen respuesta: ICMP diagnóstico.



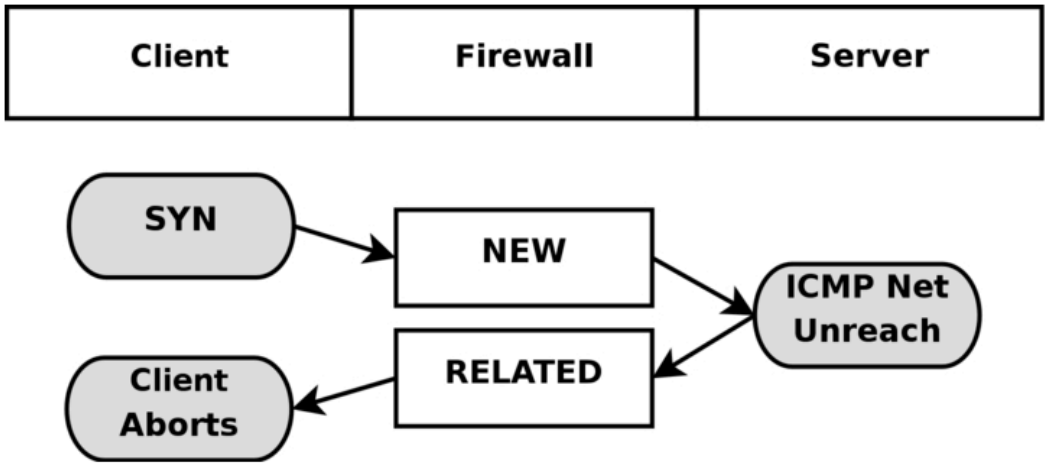

- El firewall recibe primer paquete ICMP Echo Request, estado **NEW**:

```
icmp          1 25   src=192.168.1.6  dst=192.168.1.10 type=8 code=0 id=33029
          [UNREPLIED] src=192.168.1.10 dst=192.168.1.6  type=0 code=0 id=33029 use=1
```

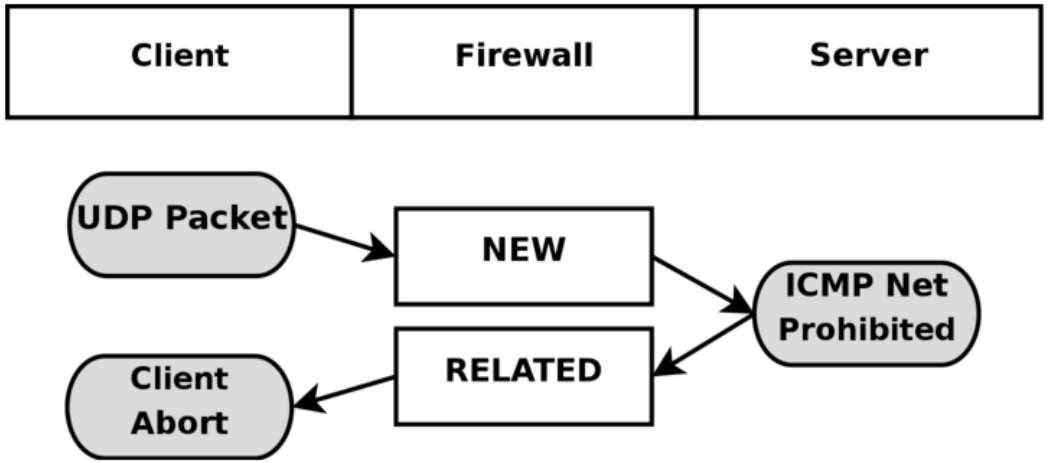
- Cuando el firewall recibe ICMP Echo Reply, el estado pasa a **ESTABLISHED** y se elimina del sistema de seguimiento. Es una "conexión" terminada.

# "Conexiones" ICMP: Mensajes ICMP error

- Mensajes ICMP error: no son respuesta a otros mensajes ICMP.
  - En conexión TCP:




  - En "conexión" UDP:



# Contenidos

# Intrusion Detection/Prevention System (IDS/IPS)

- Un **sistema de detección de intrusos** (IDS) aplica un conjunto de técnicas para detectar ataques maliciosos a un entorno. Por tanto, un IDS identifica tráfico potencialmente malicioso y elabora un informe para analizarlo posteriormente.
- El objetivo de los sistemas de detección de intrusos (IDSs) es detectar intentos de ataques para:
  - conseguir acceso a una determinada red o máquina
  - degradar el rendimiento de una red o máquina
  - robar información
- Utilizan la captura de paquetes en modo promiscuo, recibiendo todos los paquetes que circulan a través de una determinada red.
- Los **sistemas de prevención de intrusos** (IPSs) o también llamados IDs activos, además de detectar el ataque son capaces de evitarlo, reaccionan ante él, descartando los paquetes del atacante malicioso. Para ello el sistema IPS debe encontrarse conectado "en línea" para poder retirar de la red aquellos paquetes que cumplan un determinado patrón.

# Tipos de IDSs

- Según el ámbito al que aplican:
  - **N-IDS (Network IDS)**: detección de intrusiones de red. Por ejemplo: snort. Normalmente se encuentran justo detrás de un firewall, o en diversos puntos de la red interna de una organización.
  - **H-IDS (Host IDS)**: detección de intrusiones de máquina. Se encuentan instalado en la máquina para detectar accesos maliciosos a los recursos de dicha máquina, por ejemplo, en el propio Firewall.
- Según el método de detección:
  - Firmas
  - Patrones de comportamiento

# Contenidos

# Snort

- Herramienta de código abierto para la monitorización de red y la detección de intrusos. El comportamiento sospecho de un intruso se detecta por comparación con patrones de ataques.
- Los patrones de ataque se van actualizando a medida que se van descubriendo nuevos.
- A veces se combina con honeypot para estudiar ataques reales en un entorno controlado.
- Funcionamiento:
  - Decodificador de paquete
  - Preprocesadores: preparan los datos antes de dárselos al motor de búsqueda, pueden detectar anomalías en los paquetes.
  - Motor de detección: aplica las reglas que contienen patrones de comportamientos sospechos sobre los datos recibidos y comprueba si se satisfacen una o varias reglas, en cuyo caso se aplica la acción asociada a la regla.
  - Logs: almacena el mensaje de alerta asociado a la regla, y el propio paquete en formato pcap para que después se pueda analizar.
  - Plugin de salida: permiten generar notificaciones con las alertas detectadas, por ejemplo, enviar las alertas por correo, SNMP, etc.

# Configuración de Snort

- La configuración general de Snort se encuentra en `/etc/snort/snort.conf`. Este fichero tiene varias secciones de configuración, en particular, existe una sección para definir las reglas que hay que comprobar sobre los paquetes para decidir qué acción hay que tomar.
- Normalmente el fichero `snort.conf` no contiene directamente las reglas, sino que hace un `include` de otros ficheros que son los que contienen las reglas y que se encuentran en: `/etc/snort/rules`.
- Las reglas siguen este formato:

| cabecera de la regla | (opciones de la regla) |
|---|---|

  donde el texto antes del primer paréntesis es la cabecera de la regla y el texto contenido en el paréntesis son las opciones.

- La **cabecera de la regla** es la que especifica qué campos deben cumplir los paquetes para que se realice una acción sobre dicha regla. En la cabecera de la regla se encuentran los siguientes campos:

| acción | protocolo | direcciónIP1 | puerto1 | sentido de la comunicación | direcciónIP2 | puerto2 |
|---|---|---|---|---|---|---|

  Las acciones permitidas:
  - `alert`: genera alerta y registra el paquete
  - `log`: registra el paquete
  - `pass`: ignora el paquete
  - `activate`: genera alerta y activa otra regla dinámica
  - `dynamic`: regla inactiva hasta que la activa con activate y pasa a modo log
  - `drop`: bloquea el paquete y lo registra como log
  - `reject`: bloque el paquete y responde: RST para TCP, Port Unreachable para UDP
  - `sdrop`: bloquea el paquete pero no lo registra
- Las **opciones** se escriben separadas por ';', donde cada opción queda definida por el formato:

```
nombre_opción:valor_opción.
```

# Ejemplo de regla en Snort

```
alert tcp any any -> 192.168.1.0/24 111 \
     (content: "|00 01 86 a5|"; msg: "mountd access";)
```

- Cabecera de la regla: acción a aplicar `alert` (guarda el paquete en un fichero de captura y escribe un mensaje en el fichero log), el protocolo `tcp`, direcciónIP1=`any`, puerto1=`any`, sentido de la comunicación -> (desde la direcciónIP1 y puerto1 dirigido a la direcciónIP2, puerto2), dirección IP2=`192.168.1.0/24` y puerto2=`111`.
- Opciones: `msg` indica el mensaje que se va a guardar en el fichero de log (`"mountd access"`). La opción `content` busca un patrón en el contenido de un paquete.
- Otras opciones:
  - `reference:arachnids,28` ofrece información de sistemas externos de idenficación de ataques en los que se puede encontrar más información sobre esta alerta.
  - `classtype:attempted-recon` Categoría de la alerta según unos niveles predefinidos y prioridades que permiten organizar las reglas. consultar en el fichero `/etc/snort/classification.config`. Números bajos de prioridad significa, prioridad muy alta.
  - `sid:628` Identificación única para una regla snort. Permite realizar búsquedas de una determianda regla en el sistema de alertas.
  - `rev:1` Identificación de la revisión o versión de la regla.
  - Entre las opciones se encuentra su clasificación y prioridad, con el nombre de opción `classtype` se asocia un valor cuyo significado y el valor de prioridad de la alerta se pueden consultar en el fichero `/etc/snort/classification.config`. Números bajos de prioridad significa, prioridad muy alta.
  - La opción `reference` da información de sistemas en los que se puede encontrar más información sobre esta alerta. La opción `sid` es el identificador Snort de la regla, se usa para herramientas de búsqueda de reglas en la configuración. La opción `rev` hace referencia al número de versión de esa regla.

# Ejemplo de alerta generada con Snort

Dada la regla:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING"; icode:0; itype:8;
classtype:misc-activity; sid:384; rev:5;)
```

La alerta generada por un paquete que cumpla la condición de la regla podría ser:

```
04/16-10:39:59.140738  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity]
[Priority: 3] {ICMP} 10.100.1.2 -> 10.100.0.10
```

- `04/16-10:39:59.140738` es una marca de tiempo del momento en que se ha generado la alerta.
- `[1:384:5]`: el primer número es el identificador del generador (la parte de Snort que ha generado la alerta). El 1 significa que la alerta la ha generado el subsistema de reglas. Otros números están asociados al preprocesador, el decodificador, etc. El segundo número es el identificador de la regla Snort (sid=384). Y el tercer número es la revisión de la regla (rev=5).
- `ICMP PING`: es el campo `msg` de la regla.
- `[Classification: Misc activity]`: clasificación de la regla.
- `[Priority: 3]`: prioridad de la regla.
- `{ICMP} 10.100.1.2 -> 10.100.0.10`: datos del paquete que ha satisfecho la regla: protocolo, dirección IP, (opcionalmente el puerto), sentido de la comunicaciám, dirección IP (opcionalmente el puerto).

# Contenidos

# Honeypot

- Trampa puesta intencionadamente por una entidad para detectar intentos de acceso a recursos de forma maliciosa y conseguir información de cómo se realizan los ataques.
- Los atacantes creen que están accediendo a un sistema protegido y sin embargo, están accediendo a un señuelo.
- Deben ser máquinas vigiladas y sin acceso a los servicios reales internos de la red privada.

# Contenidos

# Pentesting (Penetration Testing)

- Es una práctica que consiste en atacar un entorno para descubrir vulnerabilidades con el objetivo de proteger dicho entorno de ataques maliciosos. Las pruebas consisten en el análisis de la red, aplicaciones e incluso el factor humano (ingeniería social) para conseguir información del entorno y realizar pruebas ofensivas contra el mismo.
- Es una práctica legal siempre y cuando lo realicemos con el consentimiento de la organización a la que pertenece el entorno. Sin el consentimiento, es una práctica que puede conllevar penas de prisión en algunos países (delito de hacking).
- Frecuentemente el sistema a atacar no está directamente accesible y es necesario usar otras máquinas intermediarias para poder acceder (*pivoting*).
- Kali Linux: distribución que incluye herramientas para realizar pentesting.

# Fases de Pentesting

- Las pruebas que comprenden una práctica de Pentesting se dividen en varias fases que culminan con un informe de la auditoría:
  - **Fase de reconocimiento**: búsqueda de información del objetivo que se desea atacar: direcciones IP, topología de la red, aplicaciones, etc. Es una de las fases más importantes y que requiere más tiempo. A partir de la información conseguida se desarrollan las siguientes fases. Esta fase también se le llama OSINT (Open-Soource Intelligence), se obtiene información de fuentes públicas.
  - **Fase de escaneo**: escaneo de puertos, SO, etc.
  - **Fase de enumeración**: obtención de datos de usuarios, equipos de la red, información sobre los servicios activos, etc.
  - **Fase de acceso o explotación**: acceso al sistema en el que se han encontrado las vulnerabilidades para saber qué información puede estar comprometida.
  - **Fase de mantenimiento de acceso o post-explotación**: estudiar como preservar el acceso para poder utilizarlo sucesivas veces.

# Conocimiento profundo de los protocolos de red

- Es fundamental entender el funcionamiento de la pila de protocolos TCP/IP que está usando un sistema para poder realizar una prueba de pentesting.
  - Funcionamiento del mecanismo ARP.
  - Funcionamiento del sistema de resolución de nombres, DNS.
  - Diagrama de estados de TCP: establecimiento de conexión, finalización de conexión.
  - Mensajes de error ICMP generados en determinadas circunstancias.

# Contenidos

# Contenidos

# Nmap

- Nmap (Network Mapper) es una herramienta de código abierto que se puede utilizar para exploración de una red y realizar una auditoría de seguridad un sistema.
- Multiplataforma, de código abierto.
- Programada por un hacker famoso, Fyodor (apodo, en honor a escritor ruso Fyodor Dostoyevsky, www.insecure.org).
- Explora una red enviando diferentes paquetes TCP/IP, **el tipo de respuestas o no respuestas permiten realizar suposiciones sobre la red**.
- Utilidad:
  - Descubrimiento de subredes
  - Pentesting de redes y equipos
  - Evaluación de configuración en cortafuegos.
  - Descubrimiento de SO, puertos abiertos, servicios instalados y sus versiones, etc.

# Escaneado: nmap

- La salida de nmap muestra las máquinas analizadas e información sobre las mismas, por ejemplo el SO que usan, el nombre de la máquina, dirección Ethernet y los servicios que tienen arrancados en determinados puertos TCP o UDP. La información que se obtiene sobre un puerto es la siguiente:
  - **Cerrado**: no existe ninguna aplicación esperando paquetes en dicho puerto.
  - **Abierto**: existe una aplicación esperando paquetes en dicho puerto.
  - **Filtrado**: un firewall está bloqueando el acceso a ese puerto y nmap no puede saber si está abierto o cerrado.
  - **No filtrado**: puerto que responde a los sondeos de nmap pero nmap no puede saber si está abierto o cerrado.

# Fases de un escaneo con nmap

- **Script pre-scanning**: NSE (Nmap Script Engine) utiliza scripts para obtener más información de los sistemas remotos. Por ejemplo: `dhcp-discover` que usa mensajes DHCPINFORM para obtener información de un servidor de DHCP esperando peticiones en el puerto udp 67.
- **Enumeración de objetivos**: a partir de la descripción de objetivos del usuario (nombre de máquinas, rango de subredes, etc), nmap consigue una lista de direcciones IP.
- **Descubrimiento de máquinas**: nmap envía diferentes tipos de mensajes para descubrir si una máquina está activa: solicitudes de ARP, mensajes TCP, ICMP, etc.
- **Resolución inversa de DNS**: nmap obtiene los nombres de las direcciones IP activas solicitando una resolución inversa al DNS (registros PTR)
- **Escaneo de puertos**: nmap envía mensajes a puertos TCP/UDP y se obtiene información de la respuesta o no respuesta.
- **Detección de versión**: en los puertos que estén abiertos se puede detectar la versión de la aplicación que se está ejecutando. Nmap manda diferentes tipos de mensajes y contrasta las respuestas con su base de datos para saber qué versión se está ejecutando.
- **Detección de SO**: los SOs implementan estándares de red de diferente forma. Nmap envía diferentes tipos de mensajes y contrasta las respuestas con su base de datos.
- **Traceroute**: nmap contiene una versión optimizada de traceroute para determinar rutas.
- **Script scanning**: NSE ejecuta una serie de scripts para cada máquina y puerto con el objetivo de determinar vulnerabilidades.
- **Salida**: nmap presenta la información recopilada.
- **Script post-scanning**: scripts para presentar estadísticas de los resultados obtenidos.

# Protocolos utilizados

- ICMP echo request/echo reply (ping, fping): para conocer si una máquina o conjunto de máquinas se encuentran activas.
- Para el escaneo de puertos hay que revisar todos los posibles valores desde 0-65535, aunque los servicios más comunes se encuentran en una serie de puertos reservados: FTP (20, 21), SSH (22), telnet(23), STMP (25), DNS(53), HTTP (80), NetBIOS (137-139), BGP(179), HTTPS (443), SMB (445), etc.
- Three-way handshake de TCP: una conexión TCP comienza con 3 mensajes para el establecimiento de la conexión: SYN, SYN+ACK, ACK. Si no hay una aplicación escuchando en dicho puerto se envía un segmento TCP con el flag RST.
- UDP: si no hay una aplicación escuchando en un determinado puerto se envía ICMP Port Unreachable.

# Descubrimiento de equipos

- Existen diversas técnicas para el descubrimiento de equipos utilizando `nmap`. El objetivo es mostrar si la máquina se encuentra activa o no.
  - **Sondeo de equipos en una red**: permite conocer qué equipos están arrancados en una red Ethernet, se utilizan sólo mensajes ARP (opción `-PR`) ya que usar ICMP puede llegar a ser lento:

    ```
    nmap -sP -PR -n <subred/máscara>
    ```

  - **Sondeo de equipos Ping Scan**: si la máquina está en la misma subred el sondeo se realizará de forma suave a través de una consulta de ARP, si la máquina está en otra subred es necesario usar otro tipo de paquetes que normalmente son detectados por IDS.

    ```
    nmap -sP -n <dirIP1>, <dirIP2>
    ```

    La opción `-n` si usa si no se desea que haya resolución inversa de DNS.

# Sondeo TCP SYN

- **TCP SYN Stealth scan**: nmap puede construir segmentos TCP con el flag de SYN activo con el objetivo de determinar si hay un servicio esperando recibir paquetes. No completa el establecimiento de la conexión porque si hay respuesta desde el servicio sondeado se envía RST:

  `nmap -sS -Pn -p <puerto o rangoPuertos> -n -v <dirIP>`

  Con la opción `-Pn` evitamos la fase de descubrimiento de equipos.
- Por ejemplo:

```
# nmap -sS -Pn -p 21-22 -n -v 10.100.1.1

Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-19 12:11 UTC
Initiating ARP Ping Scan at 12:11
Scanning 10.100.1.1 [1 port]
Completed ARP Ping Scan at 12:11, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:11
Scanning 10.100.1.1 [5 ports]
Discovered open port 22/tcp on 10.100.1.1
Completed SYN Stealth Scan at 12:12, 0.10s elapsed (2 total ports)
Nmap scan report for 10.100.1.1
Host is up (0.00084s latency).

PORT   STATE  SERVICE
21/tcp closed ftp
22/tcp open   ssh
23/tcp closed telnet
24/tcp closed priv-mail
25/tcp closed smtp
MAC Address: 00:14:5C:97:72:63 (Intronics)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.97 seconds
           Raw packets sent: 6 (248B) | Rcvd: 6 (232B)
```

# Sondeo UDP

- nmap puede construir paquetes UDP con el objetivo de determinar si hay un servicio esperando recibir paquetes. Como UDP es un servicio NO orientado a conexión, si el servidor está esperando paquetes en un determinado puerto, no se recibe respuesta. Si el servidor no está esperando paquetes en dicho puerto se enviará un mensaje ICMP Port Unreachable:

  `nmap -Pn -sU -p <puerto o rangoPuertos> -n -v <dirIP>`

  Con la opción -Pn evitamos la fase de descubrimiento de equipos.
- Por ejemplo:

```
# nmap -Pn -sU -p 7775-7780 -n -v 10.100.1.1

Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-19 13:25 UTC
Initiating ARP Ping Scan at 13:25
Scanning 10.100.1.1 [1 port]
Completed ARP Ping Scan at 13:25, 0.10s elapsed (1 total hosts)
Initiating UDP Scan at 13:25
Scanning 10.100.1.1 [6 ports]
Completed UDP Scan at 13:25, 1.19s elapsed (6 total ports)
Nmap scan report for 10.100.1.1
Host is up (0.0011s latency).

PORT     STATE  SERVICE
7775/udp closed unknown
7776/udp closed unknown
7777/udp open|filtered cbt
7778/udp closed interwise
7779/udp closed vstat
7780/udp closed unknown
MAC Address: 00:14:5C:97:72:63 (Intronics)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.02 seconds
           Raw packets sent: 8 (224B) | Rcvd: 5 (364B)
```