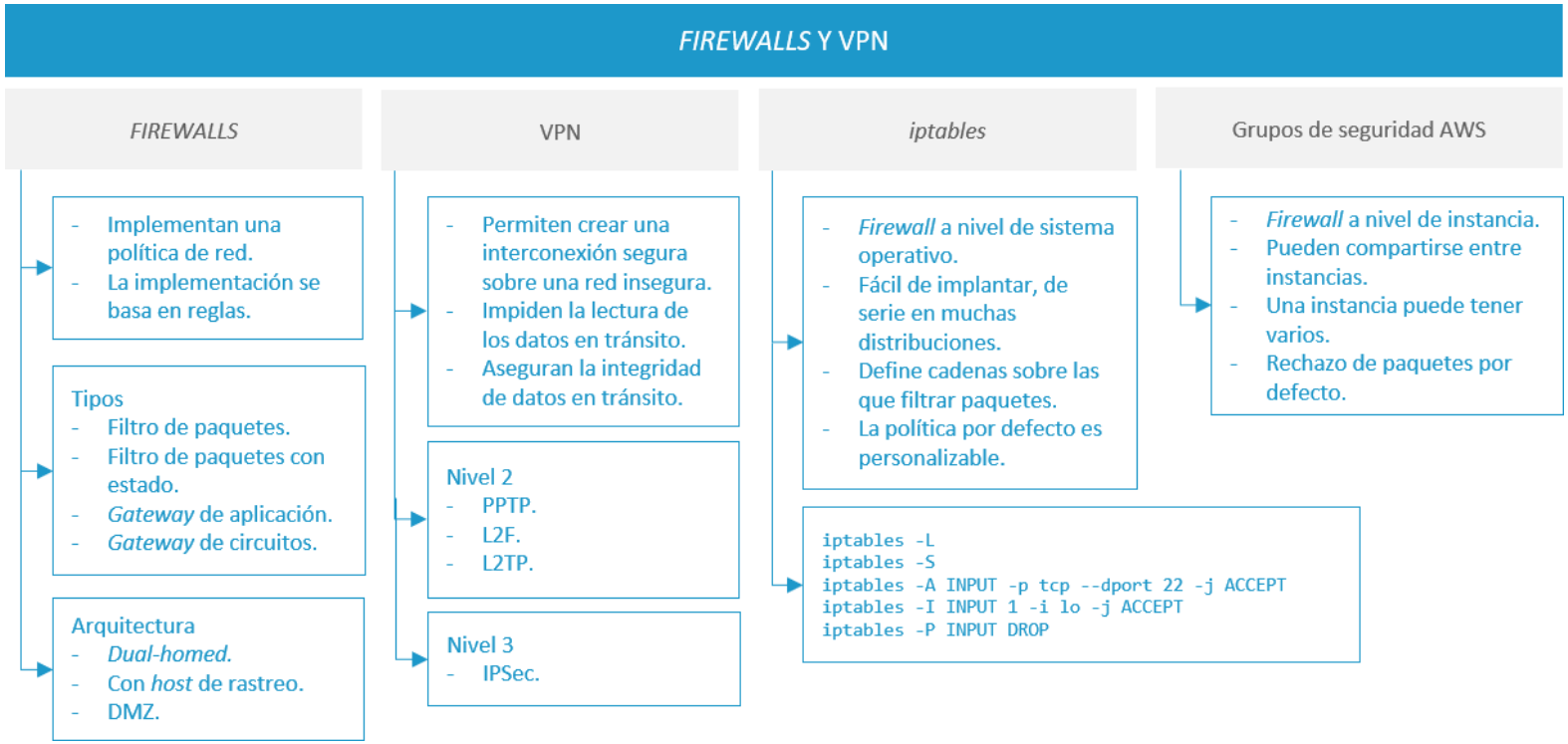
SecDevOps y Administración de Redes para Cloud

Firewalls y VPN

Índice

Esquema



Ideas clave

5.1. Introducción y objetivos

Este tema tratará el problema de la **seguridad en el acceso remoto de redes.** En particular, analizará cómo se utilizan los *firewalls,* o cortafuegos, para **proteger** los recursos corporativos de **intrusos externos** y cómo permitir a las sucursales y a los usuarios remotos acceder a la **red interna de forma segura,** a través de redes públicas no seguras.

El acceso a la red en remoto es una **necesidad** para la mayoría de las **corporaciones.** Para casi la totalidad de ellas, **Internet** proporciona una **forma económica** de conectar sucursales y trabajadores itinerantes a la **red corporativa.** Esta conexión, entre una red corporativa e Internet, sin embargo, expone la red interna a **riesgos** del mundo exterior. Por tanto, las empresas, entonces, deben tomar **medidas** para proteger la **información confidencial** de usuarios externos no autorizados.

Los **objetivos** que se pretenden conseguir en este tema son:

- Entender el concepto de *firewall* y de VPN.
- Mostrar ejemplos de implementación y configuración.

5.2. Introducción a los cortafuegos

Un **cortafuegos** (aunque el término cortafuegos es una traducción válida de *firewall,* el término en inglés está tan extendido que se usarán ambos a lo largo del tema) es un **sistema de seguridad** que controla el acceso a una **red protegida,** como una red corporativa, desde **otra red.**

Esta red puede ser una red pública, como Internet, u otra red interna. Por ejemplo, es habitual disponer de cortafuegos **regulando el tráfico** entre la red a la que se conectan los equipos de usuario y las redes con aplicaciones corporativas. Como resultado, cada **solicitud de acceso,** desde la red de origen a la red protegida, debe pasar a través del cortafuegos, eliminando la necesidad de protección individual en cada servidor y *host* de la red protegida.

Para **controlar el acceso** desde una red pública, el cortafuegos se encuentra en el punto en el que se interconectan las redes (Figura 1). Esta ubicación permite que un mismo dispositivo ejerza de *firewall* y proporcione **autenticación** y otros servicios de **seguridad** a **usuarios remotos.**

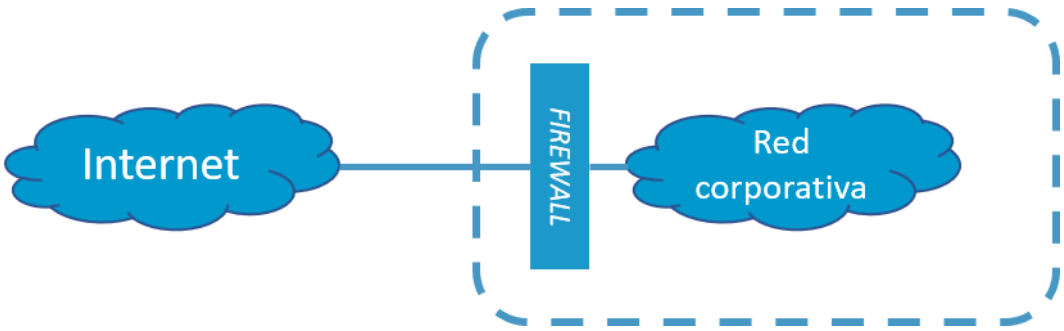


Figura 1. Diagrama de interconexión entre red pública, red privada y cortafuegos. Fuente: elaboración propia.

Para que un *firewall* sea efectivo, las organizaciones deben tener bien definida una **política de seguridad de red.** Esta política identifica los recursos que necesitan protección y las **amenazas** que existen contra ellos. A partir de estos datos, define cómo estos recursos pueden ser utilizados y por quién, y establece las **acciones** a llevar a cabo cuando se violan estas políticas.

Una política se aplica sobre los **dispositivos de red** como un conjunto de **reglas** que comprueban los paquetes que llegan a cada dispositivo. Estas reglas incluyen qué **tráfico IP** desea permitir la organización para que acceda a su red, qué **direcciones** de origen deben excluirse de la red y qué direcciones de destino pueden ser accedidas desde fuera de la red. En cuanto a las **acciones específicas,** estas incluyen aceptar o rechazar paquetes.

5.3. Tipos de *firewall*

Los *firewalls* se pueden clasificar en tres categorías básicas: **filtros de paquetes** *(packet filters),* **filtros de paquetes con estado** *(stateful packet filters)* y **servidores *proxy*** (que incluyen *gateways* de aplicación y *gateways* de circuito). Hay una cuarta categoría que es, esencialmente, un **híbrido** de las tres principales.

Filtros de paquetes

Un filtro de paquetes es un cortafuegos que **inspecciona** cada paquete de acuerdo con las **reglas de filtrado** que haya definido el usuario. Por ejemplo, una regla de filtrado podría requerir que todas las solicitudes de Telnet se eliminen. Teniendo en cuenta esta Información, el *firewall* bloqueará todos los paquetes con el **puerto TCP 23** como destino (el puerto predeterminado para Telnet).

Las reglas de filtrado pueden estar **basadas** en:

- Dirección IP de origen.
- Dirección IP de destino.
- Protocolo de capa 4 (TCP/UDP).
- Puerto de destino.

Por lo tanto, un filtro de paquetes toma decisiones basadas en la **capa de red** y la **capa de transporte.** Los filtros de paquetes son rápidos y pueden implementarse, fácilmente, en **rúters existentes.**

Por desgracia, son los **menos seguros** de todos los *firewalls.* Una desventaja que presentan es que no tienen ninguna **facilidad de registro** que se pueda utilizar para detectar cuando se ha producido una intrusión. Además, un *firewall* de filtrado de paquetes concede o deniega acceso a la red, de acuerdo con las **direcciones** de origen y de destino y los **puertos** de origen y de destino. Estos puertos pueden ser

falsificados y, como resultado, cualquiera puede acceder a los recursos de la red una vez que se haya dado acceso a un usuario autorizado.

Servidores *proxy*

Un servicio *proxy* es una aplicación que **redirige las solicitudes** de los usuarios hacia servicios basados en la **política de seguridad de una organización.** Así, un servidor *proxy* actúa como un **intermediario** de comunicaciones entre los clientes y servidores de aplicaciones. Dado que actúa como un punto de control donde se validan aplicaciones específicas, un servidor *proxy* puede convertirse en un **cuello de botella** si hay demasiado tráfico.

Los servidores *proxy* pueden funcionar tanto en la capa de aplicación como en la de transporte. Los primeros se denominan *gateway* de aplicación y los segundos *gateway* de circuito.

Gateway de aplicación

Un *gateway* de aplicaciones es un servidor *proxy* que proporciona el **control de acceso** a la capa de aplicación. Dado que opera en la capa de aplicación, es capaz de **examinar el tráfico** de la capa más alta en detalle y, por lo tanto, es considerado el tipo de *firewall* más seguro. Genera registros de todas las actividades y aplicaciones de la red, de acuerdo con las necesidades de **auditoría** de seguridad.

Los *gateways* de aplicación también pueden **ocultar información** hacia el exterior. Dado que todos los servicios en la red protegida pasan a través del gateway, este puede proporcionar la funcionalidad de **traducción de direcciones de red** (u ocultar direcciones IP) y **ocultar direcciones IP** en la red protegida desde Internet, reemplazando la dirección IP de cada paquete saliente (es decir, paquetes que van desde la red protegida a Internet) con su propia dirección IP.

La **traducción de direcciones de red** también permite que las direcciones IP no registradas sean libremente utilizadas en la red protegida, porque el *gateway* las mapea a su propia **dirección IP.**

Gateway de circuito

Un *gateway* de circuito es un servidor *proxy* que **valida las sesiones TCP y UDP** antes de permitir una conexión o un circuito. Está activamente involucrado en el **establecimiento de la conexión** y no permite que los paquetes se envíen hasta que hayan pasado con éxito las normas de control de acceso. No son tan seguros como los de aplicación, porque no analizan la capa superior. Además, una vez que se ha establecido una **sesión,** cualquier aplicación puede **ejecutarse** a través de esa conexión. Este comportamiento expone la red protegida a los ataques de intrusos.

Filtros de paquetes con estado

Los *gateway* de aplicación ofrecen la mejor seguridad, pero tienen, también, los **requisitos** de procesamiento más alto, lo que puede reducir el **rendimiento de la red.** Un filtro de paquetes con estado intenta proporcionar **seguridad** sin comprometer el **rendimiento.**

A diferencia de un *gateway* de aplicación, un filtro con estado comprueba los datos que pasan a través de la capa de red, pero **no los procesa.** El *firewall* mantiene la información de estado para cada sesión. Si los paquetes nuevos no pertenecen a una sesión válida, ni intentan crear una sesión que cumple con las políticas del *firewall,* se **rechazan.**

5.4. Arquitectura del *firewall*

La arquitectura del *firewall* se refiere a la forma en que los componentes del cortafuegos están dispuestos para proporcionar una **protección eficaz** contra usuarios no autorizados. Una red corporativa, usualmente, tiene **múltiples redes perimetrales** que se pueden clasificar en tres grupos:

Clasificación de redes perimetrales	La red perimetral más externa.
	Una o más redes internas del perímetro.
	La red perimetral más interna.

Tabla 1. Clasificación de redes perimetrales. Fuente: elaboración propia.

El perímetro exterior proporciona un **límite** entre los **recursos corporativos** (que necesitan ser protegidos) y los **recursos externos** (recursos que la corporación no puede controlar). Las redes perimetrales internas representan los límites de los recursos que necesitan **seguridad adicional.**

Cortafuegos de *host dual-homed*

Un **equipo *dual-homed*** es aquel que tiene **dos tarjetas de red.** Si el *host* ejecuta un proceso de *firewall,* las interfaces se pueden aprovechar para una **arquitectura** de seguridad concreta: una interfaz está conectada a la **red interna** y otra interfaz está conectada a **Internet,** o a alguna **otra red no confiable** (Figura 2). Por lo tanto, todo el tráfico IP de Internet debe pasar por el *firewall* antes de llegar a un *host* en la red privada. Del mismo modo, un *host* interno puede comunicarse con *hosts* externos a través del *host dual-homed.*

Cualquier **comunicación indirecta** que intente esquivar el cortafuegos está **bloqueada** por diseño, ya que no hay otra conectividad entre las redes más que la que atraviesa el cortafuegos. El *host dual-homed* puede funcionar como un rúter para garantizar que Internet y la red privada están **lógicamente desconectadas,** de modo que, incluso cuando haya problemas en el sistema, el cortafuegos no falle.

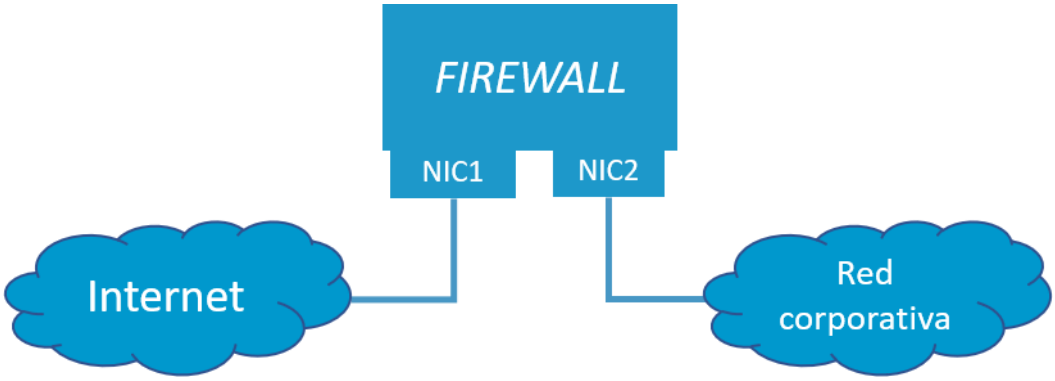


Figura 2. *Firewall dual-homed.* Fuente: elaboración propia.

Cortafuegos de *host* de rastreo

En esta arquitectura, el *host* que actúa como cortafuegos, llamado **bastión,** solo se conecta a **la red privada.** Un rúter de rastreo (o de *screening*) adicional es colocado entre el bastión e Internet (Figura 3). Por lo tanto, esta arquitectura **combina** un **enrutador de filtrado de paquetes** y un ***gateway* de aplicación.**

El rúter de rastreo realiza una **función de filtrado** de paquetes, y está configurado para que el bastión sea el único *host* de la red privada al que se puede acceder desde Internet. Se puede proporcionar **seguridad extra** para que el rastreo permita el tráfico solo a **puertos específicos** del bastión, bloqueando el resto por defecto.

Dado que el anfitrión del bastión es el **huésped más expuesto** en la red privada, suele ser el **más protegido.** Generalmente, no hay un único bastión, sino varios. Estos suelen actuar como **servidores *proxy*** para servicios públicos como FTP, HTTP o SMTP.

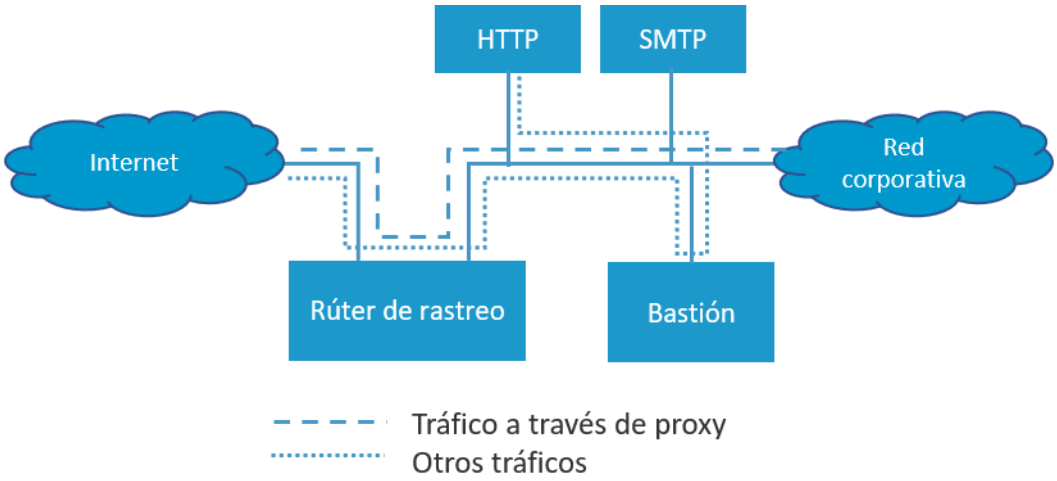


Figura 3. *Firewall* con rúter de rastreo. Fuente: elaboración propia.

Cortafuegos de subred o DMZ

El cortafuegos de subred se puede considerar una **extensión del cortafuegos de *host* de rastreo.** También incorpora un rúter de rastreo, denominado externo, y un *host* bastión. Sin embargo, este cortafuegos crea una **capa adicional de seguridad** añadiendo una **red de perímetro** que aísla a la red privada de Internet.

Esta capa define una *Demilitarized Zone* o **zona desmilitarizada** (DMZ) demarcada por el rúter externo y un rúter interno, tal como muestra la Figura 4. Este último es localizado más cerca de la red privada que del enrutador externo. El **bastión** y los servidores de acceso público se encuentran, entonces, dentro de la DMZ.

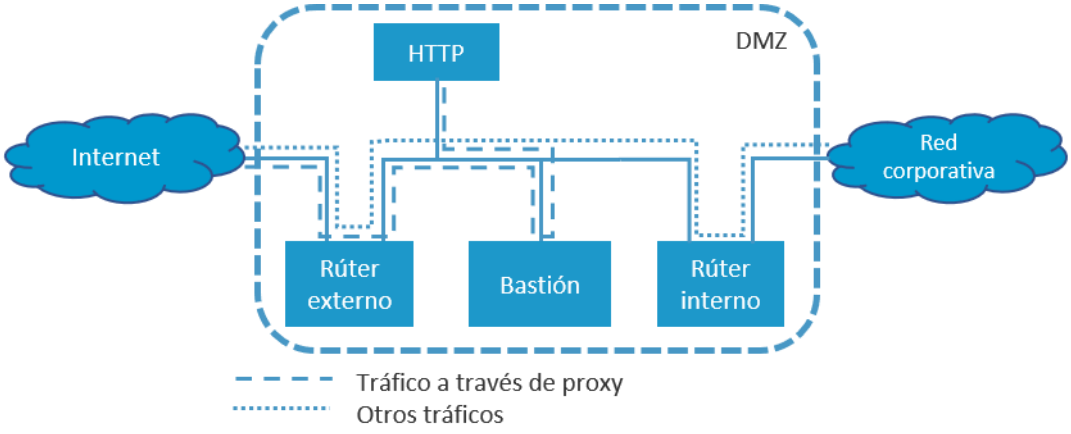


Figura 4. Arquitectura con DMZ. Fuente: elaboración propia.

La DMZ puede considerarse una **red aislada** entre el sector privado de red e Internet. Para que un ataque llegue a cualquier *host* interno localizado fuera de DMZ, el atacante debe acceder a **ambos rúters.** Además, su arquitectura **revela** solo la red DMZ al mundo exterior y mantiene la red privada oculta.

5.5. Redes privadas virtuales

Una **red privada virtual** (VPN) proporciona una **conexión segura** entre un origen y un destino, a través de una red pública no segura, como Internet.

> Las VPN no son las únicas tecnologías capaces de conectar dos redes de manera segura: las operadoras de telecomunicaciones suelen hacer uso de MPLS tanto internamente, para conectar sus redes, como en modo servicio a clientes externos. Una red MPLS se parece a una red de conmutación de circuitos, ya que ofrece una conexión punto a punto entre dos extremos, encapsulando los paquetes IP en una trama con su propia cabecera. (Tanenbaum y Wetherall, 2011).

Las VPN **reducen** los costes de acceso remoto. En comparación con otras soluciones, incluidas las redes privadas, una VPN es **barata.**

Una VPN utiliza el **cifrado de datos** y otros mecanismos de seguridad para que los usuarios no autorizados no puedan acceder a los datos (es decir, leer el contenido de los paquetes), y para **garantizar** que los datos no se puedan **modificar** sin detección (es decir, que no alteren los datos en tránsito) a medida que **fluyen** a través de Internet.

Las VPN **encapsulan un tráfico** en otro, a modo de túnel. En el contexto de Internet, el proceso de tunelización permite encapsular **protocolos** tales como IPX, AppleTalk e IP a través de IP. Efectivamente, IP puede ir encapsulado en IP: el contenido sería

el tráfico IP de la **red privada,** y la capa IP externa sería el flujo a través de la red no segura. El proceso de encapsulado **cifra el contenido del tráfico interno** antes de transportarlo por el túnel y lo descifra una vez en el destino.

Ventajas de VPN

Las empresas pueden **explotar la naturaleza global de Internet** y utilizar VPN para enlazar todas las sucursales en redes privadas. Una organización puede hacer que ciertas secciones de su intranet sean accesibles a sus proveedores estratégicos por medio de la *extranet.*

Los **túneles VPN** permiten transmitir subredes no enrutables entre segmentos específicos de LAN en la intranet corporativa. Esta **característica** es útil, especialmente en aplicaciones *legacy,* o heredadas: **aplicaciones antiguas,** a veces desarrolladas como proyectos de llave en mano, sin soporte y difíciles de mantener.

Tipos de VPN

Se pueden citar **tres tipos** de VPN en función de la necesidad que cubren dentro de una organización:

VPN de acceso	Proporcionan acceso seguro a las redes corporativas a los usuarios remotos, teletrabajadores y sucursales de oficinas.
VPN de intranet	Permiten que las oficinas remotas y sucursales estén vinculadas a las sedes corporativas de forma segura.
VPN de extranet	Permiten que los clientes, proveedores y socios puedan acceder a la intranet corporativa de una manera segura.

Tabla 2. Tipos de VPN según la necesidad en una organización. Fuente: elaboración propia.

Arquitecturas VPN

Una VPN consiste en los siguientes **componentes:**

- Un cliente VPN.
- Un servidor de acceso a la red o *Network Access Server* (NAS).
- Un dispositivo que termina en un túnel o servidor VPN.
- Un protocolo VPN.

En una **conexión tradicional** de VPN de acceso, el cliente VPN inicia una conexión PPP con el NAS del ISP a través de la **red telefónica.** Un NAS es un dispositivo que termina las llamadas de marcación en **circuitos analógicos o digitales** (RDSI). El NAS es propiedad del ISP, y se instala, generalmente, en el punto del servicio del ISP. Después de que el usuario haya sido autenticado, el NAS dirige el paquete al túnel que conecta tanto al NAS como al servidor VPN.

El servidor VPN puede residir en el **POP** del **ISP** o en el **sitio corporativo,** dependiendo del modelo de VPN que se implemente. El servidor VPN recupera el paquete del túnel, lo desenrolla y lo entrega a la red corporativa.

Hay **cuatro protocolos de tunelización** utilizados para establecer VPN. Pueden clasificarse de forma general en **dos grupos:** PPTP, L2F y L2TP son protocolos de tunelización de **capa 2,** mientras que IPSec es un protocolo de tunelización de **capa 3.**

Protocolos de capa 2

Estos protocolos operan en la **capa de enlace de datos.** Encapsulan paquetes de capa 3 en PPP de capa 2 antes de encapsularlos en IP. Utilizan la seguridad proporcionada por PPP, por lo tanto, realizan la **autenticación** de usuario con los protocolos de autenticación propios de PPP: PAP y CHAP.

No existen **disposiciones específicas** para el cifrado de datos, que puede ser realizado por el usuario antes de encapsular los datos en la VPN.

- PPTP

  PPTP, o *Point-to-Point Tunneling Protocol,* encapsula los paquetes IPX o IP dentro de paquetes IP. Por tanto, **facilita la interconexión de redes no IP** a través de Internet. Es una extensión de PPP y no soporta conexiones punto a multipunto.

Para obtener más información, puedes acceder al documento *Point-to-Point Tunneling Protocol (PPTP)* a través del aula virtual o desde la siguiente dirección web: https://datatracker.ietf.org/doc/html/rfc2637

  PPTP no proporciona **encriptación** paquete a paquete. En su lugar, se basa en el cifrado nativo de PPP con PAP y CHAP. Un **paquete PPTP** es encapsulado en *Generic Routing Encapsulation* (GRE), el cual es, entonces, transportado por IP. PPTP separa los **canales de control y de datos:** el flujo de control corre sobre TCP y el flujo de datos se encapsula sobre GRE, que se transmite directamente sobre IP.

- L2F

  L2F es un **protocolo propietario** desarrollado por Cisco Systems. Soporta una gran variedad de protocolos: puede encapsular tráfico IP, IPX y AppleTalk sobre X.25, IP o ATM. Usa UDP para la **encapsulación** sobre IP.

  A diferencia de PPTP, L2F **define** su propio encabezado encapsulado, que no es dependiente de IP y GRE. Esta capacidad permite a L2F trabajar en **diferentes** tipos de redes.

- L2TP

Cuando PPTP y L2F se presentaron a la IETF, la organización decidió **combinar** las características de ambos protocolos en el protocolo L2TP. A diferencia de PPTP, que se ejecuta a través de TCP, L2TP se ejecuta a través de UDP y **no utiliza GRE.** Debido a que muchos *firewalls* no son compatibles con GRE, L2TP es más fácil de integrar que PPTP.

En L2TP, el **NAS** se llama concentrador de acceso L2TP o LAC, y el **servidor VPN** se llama servidor de red L2TP o LNS. L2TP utiliza enlaces PPP *dial-up* y, por ende, también utiliza PAP y CHAP para autenticación, aunque soporta el uso de **RADIUS.**

L2TP se basa en IPSec para realizar el cifrado de datos. Si L2TP descubre que IPSec **no es compatible** en el **extremo remoto,** utiliza el cifrado PPP menos seguro. El cifrado puede realizarse desde el puesto de trabajo del usuario, o por LAC, dependiendo de la VPN que se utilice.

- IPSec

IPSec fue diseñado originalmente para incorporar seguridad en la **pila TCP/IP.** Proporciona autenticación, integridad y confidencialidad a nivel de paquete mediante la adición de **dos protocolos:** *Authentication Header* (AH) o **encabezado de autenticación**, que proporciona integridad de encabezado y autenticación sin confidencialidad; y *Encapsulating Security Payload* (ESP) o **contenido con encapsulado de seguridad**, que proporciona integridad, autenticación, y confidencialidad a la carga útil.

Una VPN de IPSec se puede establecer con AH o ESP, o ambos.

ESP permite el **encriptado paquete a paquete** y utiliza un **protocolo de gestión de claves de cifrado** basada en estándares. AH no proporciona el cifrado de datos y es útil en aquellos entornos donde solo se requiere autenticación. También tiene una sobrecarga de procesamiento más baja que ESP.

Un **inconveniente** de usar IPSec es que solo admite IP, mientras que los protocolos de nivel 2 permiten encapsular más protocolos de nivel 3.

Firewalls y VPN

Los *firewalls* y las VPN van de la mano. Mientras que los *firewalls* **controlan el acceso** a los recursos de la red corporativa, las VPN proporcionan **privacidad** entre dos redes cuando el camino entre ellas no es seguro. Muchos productos de *firewall* proporcionan acceso seguro desde un **cortafuegos** hasta otro, siguiendo el esquema de la Figura 5.

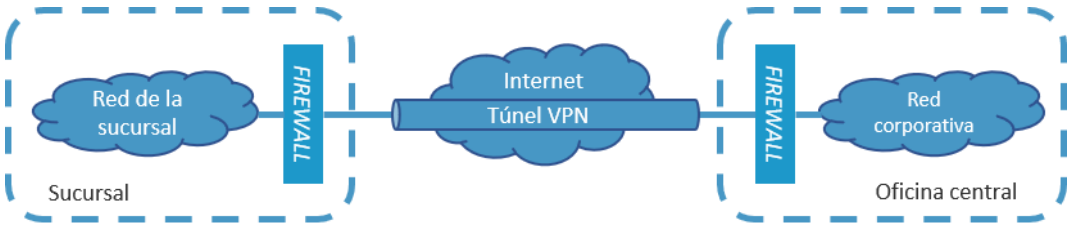


Figura 5. Redes corporativas conectadas por VPN y protegidas por firewalls. Fuente: elaboración propia.

5.6. Configuración de un *firewall* con *iptables*

El **diseño** y **configuración** de un buen cortafuegos es un **paso esencial** para la seguridad de cualquier sistema operativo moderno. La mayoría de las distribuciones de Linux incorporan herramientas de *firewall* a nivel de *host.* El **software *iptables*** es uno de ellos. Más que un firewall, *iptables* es la **utilidad de línea de comandos** del subsistema de proceso de paquetes del núcleo de Linux, Netfilter. Funciona en la **capa 3** a nivel de paquete.

Los siguientes apartados mostrarán cómo configurar un *host* con ***iptables*** para permitir solo tráfico SSH y HTTP en Ubuntu.

Comandos básicos de *iptables*

Un buen punto de partida es listar las **reglas** actuales con el **flag -L.** Cualquier comando requiere permisos de administración y debe ejecutarse con **sudo.**

```
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target   prot opt source        destination

Chain FORWARD (policy ACCEPT)
target   prot opt source        destination

Chain OUTPUT (policy ACCEPT)
target   prot opt source        destination
```

El listado muestra **tres cadenas** por defecto: `INPUT`, `FORWARD` y `OUTPUT`. La política predeterminada de cada cadena es `ACCEPT`. Aunque **aparecen** los **encabezados** de columna, no hay ninguna regla real.

También es posible imprimir el listado de reglas en un formato de **comandos de configuración con el flag -S.** El resultado es menos legible para un ser humano, pero facilita la aplicación de reglas mediante un **fichero de configuración.** Para llegar a la configuración actual, no hay más que ejecutar `iptables` seguida de cada una de las **líneas.** Las reglas de la configuración actual son simplemente las **políticas** por defecto de cada cadena.

```
$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

El comando `iptables -F` **elimina** todas las reglas existentes **salvo** las políticas por defecto. Si un administrador remoto elimina todas las reglas, incluso las que permiten **SSH,** y la política por defecto es **DROP,** la sesión remota terminará inmediatamente.

Reglas básicas

Para facilitar el trabajo de los **administradores remotos,** la primera regla será la que **acepte,** explícitamente, el tráfico SSH en la cadena INPUT. El comando será:

```
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Paso por paso, el significado de cada modificador es el siguiente:

`A INPUT`	Agrega una regla al final de la cadena INPUT.
`m conntrack`	Tiene un conjunto de funcionalidades básicas, pero también tiene un conjunto de extensiones o módulos que proporcionan capacidades adicionales. Esta parte del comando configura el módulo `conntrack`. Este módulo da acceso a comandos que se pueden utilizar para tomar decisiones basadas en la relación del paquete con las conexiones anteriores, convirtiendo iptables en un *firewall* con estado.
`ctstate`	Este es uno de los comandos del módulo `conntrack`. Permite seleccionar los paquetes basándose en cómo están relacionados con los paquetes de conexiones anteriores. El parámetro `ESTABLISHED` permite paquetes que forman parte de una conexión existente, mientras que `RELATED` permite paquetes que están asociados con una conexión establecida. Esta es la parte de la regla que coincide con nuestra sesión actual de SSH.
`j ACCEPT`	Especifica la acción sobre los paquetes seleccionados.

Tabla 3. Significado de modificadores. Fuente: elaboración propia.

Esta regla está al principio para garantizar que los **paquetes de las conexiones establecidas** sean aceptados. El listado de reglas pasa a ser:

```
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target   prot opt source        destination
ACCEPT   all -- anywhere       anywhere   ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target   prot opt source        destination

Chain OUTPUT (policy ACCEPT)
target   prot opt source        destination
```

Otras conexiones necesarias

El primer protocolo que permitirá explícitamente será **SSH.** De lo contrario, no aceptará **conexiones remotas** nuevas si se cambiar la política por defecto a DROP. El segundo será el protocolo **HTTP en el puerto 80,** asumiendo que el servidor tiene un servicio a tal efecto.

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Las **nuevas opciones** son:

- p tcp: esta opción selecciona los paquetes con protocolo TCP.
- dport 22: como continuación del parámetro anterior, la regla seleccionará paquetes TCP con el puerto de destino 22. La segunda regla es equivalente, pero para el puerto 80.

Hace falta una regla más para asegurar que el servidor pueda **funcionar correctamente:** la comunicación con la interfaz de *loopback* debe estar permitida.

> A menudo, los servicios se comunican entre sí mediante el envío de paquetes de red una pseudo-interfaz de red, llamada *loopback* o dispositivo de bucle invertido, que dirige el tráfico de nuevo a sí mismo, en lugar de a otros equipos. En cualquier equipo, *localhost* apunta a la IP `127.0.0.1`, que es la IP por defecto de la interfaz de *loopback.* Por lo tanto, si un servicio desea comunicarse con otro servicio, que está escuchando conexiones en el puerto `4555`, puede enviar un paquete al puerto `localhost:4555` o `127.0.0.1:4555`.

La **regla necesaria** es:

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

Los parámetros de este comando son:

- `I INPUT 1`: mientras que el **flag -A** añade una regla al final de la lista, el **flag -I** inserta la regla en una posición concreta. En este caso, la regla será la primera de la cadena INPUT. Con esto se consigue que el tráfico de *loopback* no se vea afectado por otras reglas.

- `i lo`: la regla seleccionará paquetes que usen la **interfaz lo**, que es el nombre interno de la interfaz de *loopback.*

Una nueva **consulta de reglas,** esta vez con `iptables -S`, muestra todas las reglas que se han **añadido** hasta el momento. La lista en formato tabla no incluye **parámetros** como la interfaz, así que, aunque es fácil de leer, puede **despistar,** por no incluir toda la información.

```
$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

Regla de omisión

Las cuatro reglas aceptan, explícitamente, paquetes basados en ciertos **criterios,** pero, hasta el momento, el *firewall* **no bloquea** nada de tráfico. Si un paquete entra en la cadena INPUT, y **no coincide** con una de las **cuatro reglas,** se le aplica la política predeterminada. Hay **dos maneras** de descartar todos los paquetes que no cumplan alguna de las reglas específicas.

La **primera** es modificar la política por defecto de la cadena INPUT con:

```
$ sudo iptables -P INPUT DROP
```

Esto **captura** y **descarta** todos los paquetes que caen a través de nuestra cadena INPUT. Si las reglas desaparecen, y se mantiene esta política, no será posible acceder al **servidor** remotamente (siempre sería posible a través de la consola virtual, equivalente a una pantalla tradicional ya que, en ese caso, el inicio de sesión es local, no a través de red).

La **segunda manera** es mantener la política predeterminada como `ACCEPT`, y agregar una **regla** que **capture** todos los paquetes restantes al final de la cadena con:

```
$ sudo iptables -A INPUT -j DROP
```

El **resultado,** bajo condiciones normales, es exactamente el mismo que una **política DROP por defecto.** Básicamente, esto se utiliza para que la política predeterminada acepte tráfico. De esta forma, si hay algún **problema,** y las reglas se vacían, todavía podrá acceder a la máquina a través de la red.

Esta es una forma de implementar una acción predeterminada sin alterar la política que se aplicará a una cadena vacía.

Esto también significa que **cualquier regla adicional** debe preceder a la regla de DROP general. Esto puede conseguirse borrando la regla temporalmente y añadiéndola de nuevo tras la regla adicional.

```
$ sudo iptables -D INPUT -j DROP
$ sudo iptables -A INPUT nueva_regla_aqui
$ sudo iptables -A INPUT -j DROP
```

También es posible **insertar las reglas adicionales** especificando el número de línea, como se hizo con la regla específica para el *loopback.* El listado de reglas puede incluir el número de línea con `--line-numbers`.

```
$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target   prot opt source        destination
1  ACCEPT   all -- anywhere       anywhere
2  ACCEPT   all -- anywhere       anywhere ctstate RELATED,ESTABLISHED
3  ACCEPT   tcp -- anywhere       anywhere tcp dpt:ssh
4  ACCEPT   tcp -- anywhere       anywhere tcp dpt:http
5  DROP    all -- anywhere       anywhere

Chain FORWARD (policy ACCEPT)
num target   prot opt source        destination

Chain OUTPUT (policy ACCEPT)
num target   prot opt source        destination
```

Guardar la configuración

De forma predeterminada, las reglas agregadas con `iptables` son **efímeras.** Esto quiere decir que las reglas **desaparecerán** durante un **reinicio** del servidor. El paquete `iptables-persistent` facilita esta tarea. Si no está disponible, es posible **instalarlo** con apt-get.

```
$ sudo apt-get update && sudo apt-get -y install iptables-persistent
```

El **proceso de instalación** preguntará si se desea guardar las reglas actuales para cargarlas automáticamente, tanto de **IPv4** como **IPv6.** Una vez termina la instalación, el nuevo servicio `netfilter-persistent` arranca. Este servicio cargará las **reglas** y las **aplicará** cuando se inicie el servidor. Los cambios de configuración posteriores a la instalación de `iptables-persistent` se pueden guardar con:

```
$ sudo netfilter-persistent save
```

Comprobación

La manera más rápida de **comprobar** que las reglas **funcionan** es probar tanto que las reglas de ACCEPT **permiten** el tráfico correctamente, como que **cualquier otro tráfico** es rechazado. Probar todos los puertos y todos los protocolos es complicado, pero se pueden llevar a cabo **análisis reducidos.**

La Figura 6 muestra un **ejemplo** sencillo de cómo un ***ping,*** que no es más que un **paquete de control ICMP,** es rechazado con el conjunto de reglas definido anteriormente. Si la última regla de DROP se **elimina,** el *ping* vuelve a ser aceptado.

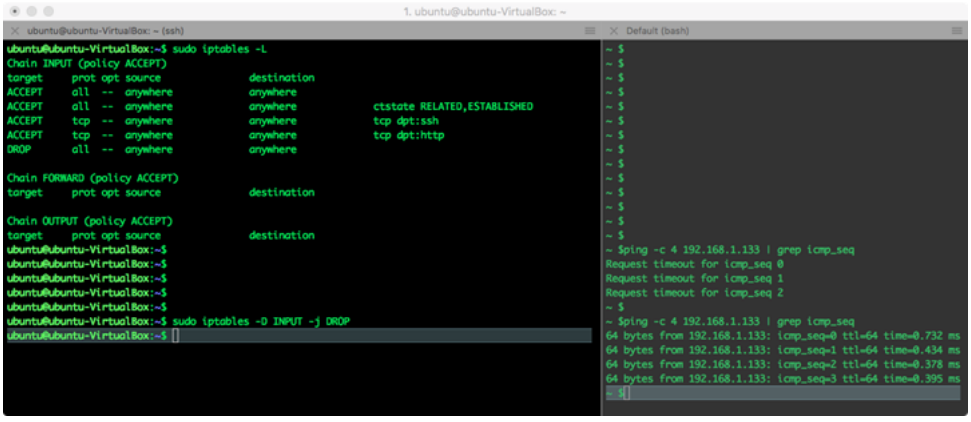


Figura 6. Prueba de reglas de *iptables*. Fuente: elaboración propia.

5.7. Grupos de seguridad en AWS EC2

Los **entornos de nube** ofrecen funcionalidades de seguridad que no suelen estar presentes en entornos tradicionales. Los **grupos de seguridad** o *security groups* de **AWS** son un ejemplo. Cada instancia del servicio de computación, EC2, tienen un **grupo de seguridad asociado** que actúa como un *firewall,* que limita los protocolos y puertos de la instancia.

En la sección A fondo encontrarás un artículo para ampliar información sobre *Amazon Elastic Compute Cloud* (EC2).

Mientras que *iptables* funciona como parte del **sistema operativo,** los grupos de seguridad se ejecutan fuera de la instancia, por lo que no consumen recursos de CPU de esta. Varias instancias pueden **compartir** un mismo grupo de seguridad, de forma que la configuración se aplica una única vez y se aplica en múltiples sitios. Además, una misma instancia puede tener hasta **cinco grupos** de seguridad asociados.

El siguiente vídeo, titulado **«Grupos de seguridad en AWS»**, demuestra cómo crear un grupo de seguridad y cómo asignarlo a una instancia EC2.

 Accede al vídeo

Cada instancia tiene un **grupo de seguridad predeterminado**. También es posible crear un grupo de seguridad desde cero y asociarlo a las instancias durante la creación, o una vez arrancada. Como cualquier otra **operación de AWS,** esta configuración se puede realizar con la herramienta de línea de comandos `awscli`, desde un SDK o en la consola de administración web de **Amazon EC2.** El grupo de seguridad predeterminado tendrá la configuración de la Figura 7.

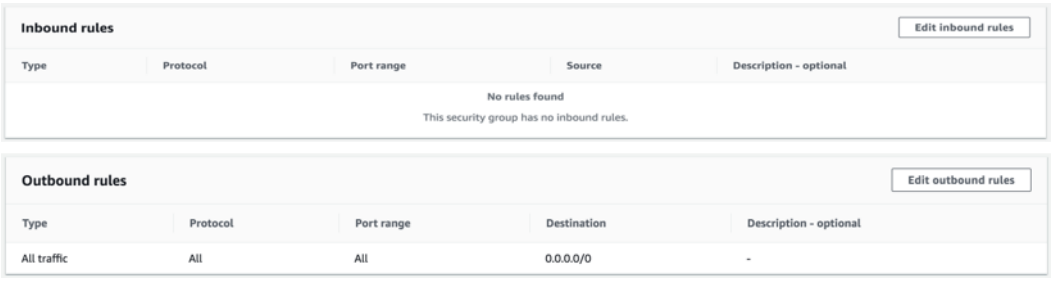


Figura 7. Grupo de seguridad por defecto en AWS EC2. Fuente: elaboración propia.

Los grupos de seguridad **bloquean tráfico** por defecto y las reglas solo son permisivas, no restrictivas. Por tanto, un **grupo sin reglas** bloquea todo el tráfico y cada regla permite un tráfico específico. El grupo de seguridad predeterminado no permite tráfico de entrada y tiene una **única regla de salida** que permite todo el tráfico.

Cada regla acepta los siguientes **parámetros** (Figura 8):

- Dirección: entrada o salida. Las reglas de cada tipo se definen en una pestaña diferente.

- Tipo: lista predefinida de protocolos. Facilita la selección del protocolo de transporte y del puerto a partir del protocolo de aplicación. También se puede seleccionar *Custom Protocol* y definir el resto de los parámetros manualmente.

- Protocolo: TCP, UDP o ICMP.

- Rango de puertos: puerto individual o rango de puertos.

- Origen: dirección IP en formato CIDR. Puede permitir el acceso a cualquier origen con `0.0.0.0/0` a una subred con `139.47.101.0/24`, o a una IP concreta con `139.47.101.40/32`. También permite especificar otro grupo de seguridad de la misma cuenta de AWS, o de otra cuenta.

- Descripción.

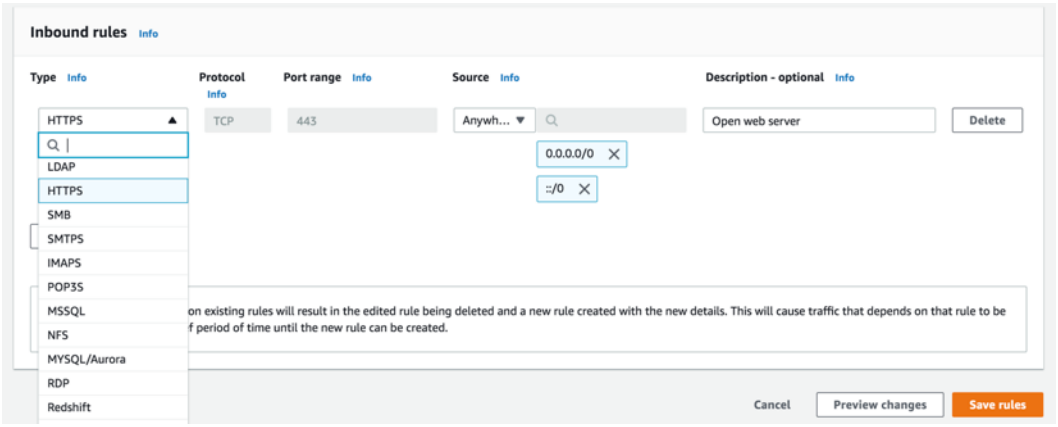


Figura 8. Edición de regla en un grupo de seguridad. Fuente: elaboración propia.

Cada regla solo permite **un protocolo y un origen.** Por tanto, un grupo de seguridad que permita el **acceso abierto** a un servidor web por HTTP y HTTPS, el *ping* y el acceso por SSH para administradores desde dos oficinas diferentes tendrá las reglas de la Figura 9.

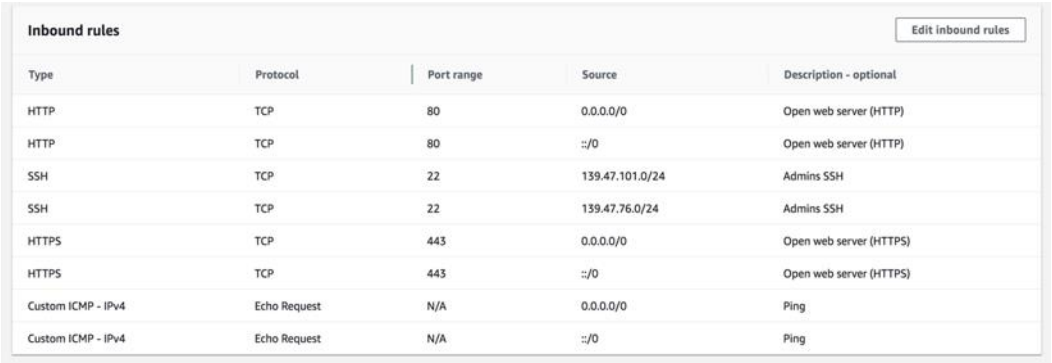


Figura 9. Reglas de servidor web con acceso para administradores. Fuente: elaboración propia.

La imagen muestra **más reglas** que las citadas anteriormente. Eso se debe a que cada regla solo permite un parámetro de cada tipo: para **aceptar paquetes** de cualquier origen, hace falta una regla para **IPv4** con la red de origen `0.0.0.0/0`, y otra para IPv6 con `::/0`.

5.8. Referencias bibliográficas

Amazon Web Services. (S. f.). *Security groups for your VPC.* https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

IETF. (S. f.). *Cisco Layer Two Forwarding (Protocol) "L2F".* https://tools.ietf.org/html/rfc2341

IETF. (S. f.). *Generic Routing Encapsulation (GRE).* https://tools.ietf.org/html/rfc1701

IETF. (S. f.). *Layer Two Tunneling Protocol "L2TP".* https://tools.ietf.org/html/rfc2661

IETF. (S. f.). *Point-to-Point Tunneling Protocol (PPTP).* https://tools.ietf.org/html/rfc2637

Ingham, K. y Forrest, S. (2002). *A History and Survey of Network Firewalls.* University of New Mexico. https://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf

Purdy, G. (2004). *Linux Iptables Pocket Reference.* O'Reilly Media.

Tanenbaum, A. y Wetherall, D. (2011). *Computer Networks.* (5ª ed.). *Pearson New International*.

A fondo

Linux Iptables Pocket Reference

Purdy, G. (2004). *Linux Iptables Pocket Reference.* O'Reilly Media.

Aunque algo antiguo, este libro sigue tan vigente como el primer día. El primer capítulo explica en detalle todas las funcionalidades y conceptos esenciales de *iptables,* sin entrar en el detalle de la sintaxis de los comandos. Es muy útil para ampliar los contenidos de la sección correspondiente. Te sugerimos la introducción.

Grupos de seguridad de Azure

Microsoft. (2020, agosto 9). *Network security groups*. https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview

AWS no es el único proveedor de servicios en la nube con la funcionalidad *firewall* a nivel de instancia. Azure ofrece su propia versión, los grupos de seguridad o *network security groups.* Aunque la interfaz sea diferente, el concepto es similar. Para un administrador, es fundamental poder abarcar conocimientos de diferentes tecnologías.

Amazon EC2

Amazon Web Services. (2020, agosto 9). *Amazon EC2.* https://aws.amazon.com/es/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

En este recurso puedes profundizar sobre las posibilidades de *Amazon Elastic Compute Cloud* (Amazon EC2).

Test

1. ¿Qué diferencia hay entre un equipo *dual-homed* y uno *single-homed*?

A. Ninguna, son iguales.

B. El *dual-homed* tiene dos o más tarjetas de red, cada una conectada a una red diferente, mientas que el *single-homed* solo tiene una.

C. El *dual-homed* tiene dos tarjetas de red, ambas en la misma red para soportar alta disponibilidad, mientas que el *single-homed* solo tiene una.

D. El *dual-homed* tiene una tarjeta de red, mientas que el *single-homed* tiene dos o más.

2. ¿Cómo se denomina la sección entre dos rúters con servicios públicos?

A. DMZ.

B. Rúter externo.

C. Rúter de rastreo.

D. Bastión.

3. ¿Qué reglas se procesan primero en *iptables*?

A. Las primeras de la lista, en orden ascendente, seguidas de la política por defecto.

B. Las últimas de la lista, en orden decreciente.

C. Las últimas de la lista, en orden decreciente, seguidas de la política por defecto.

D. No se sigue un orden: solo una regla puede identificar unívocamente un paquete.

4. ¿Cómo se puede averiguar en qué interfaz se aplica una regla?

A. `iptables -L.`

B. `iptables -L --line-numbers.`

C. `iptables -S.`

D. Ninguna de las anteriores.

5. Relaciona cada protocolo con la capa en la que trabaja.

L2F	1	A	Nivel 2
IPSec	2	B	Nivel 3

6. ¿Cuándo se considera que una VPN conecta dos subredes de manera segura?

A. Cuando evita que los datos se puedan leer en tránsito.

B. Cuando evita que los datos se puedan modificar en tránsito.

C. Todas las anteriores.

D. Siempre, por el mero hecho de ser un túnel.

7. ¿Por qué es más seguro como *firewall* un *gateway* de aplicación que un filtro de paquetes?

A. Son igual de seguros.

B. Porque soporta más reglas.

C. Porque un filtro de paquetes no tiene estado.

D. Porque es capaz de analizar el contenido de los paquetes IP y TCP, lo que permite implementar políticas más específicas.

8. ¿Qué tráfico permite un grupo de seguridad predeterminado de AWS?

A. Todo el tráfico de salida.

B. Todo el tráfico de entrada y de salida.

C. Tráfico de entrada a cualquier origen en el puerto 80 y todo el tráfico de salida.

D. Todo el tráfico de entrada desde otros grupos de seguridad de la misma red y todo el tráfico de salida.

9. ¿Qué comando añade una regla en *iptables* al final de la lista?

A. `sudo iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT.`

B. `sudo iptables -D INPUT -p tcp --dport 22 -j ACCEPT.`

C. `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT.`

D. `sudo iptables -I INPUT "end" -p tcp --dport 22 -j ACCEPT.`

10. ¿De qué se encarga un equipo bastión?

A. Actúa como *proxy* de aplicación de servicios públicos como HTTP y FTP.

B. Es un *firewall* sin estado.

C. Actúa como rúter interno de una DMZ.

D. Ninguna de las anteriores.