
Apuntes de Seguridad y alta disponibilidad.

Versión 1.0

Oscar GG

25 de septiembre de 2025

1. Adopción de pautas de seguridad informática	1
1.1. Fiabilidad, confidencialidad, integridad y disponibilidad.	1
1.2. Elementos vulnerables en el sistema informático; hardware, software y datos.	2
1.3. Análisis de las principales vulnerabilidades de un sistema informático.	2
1.4. Amenazas. Tipos.	3
1.5. Amenazas físicas.	3
1.6. Amenazas lógicas.	3
1.7. Seguridad física y ambiental	4
1.8. Sistemas de alimentación ininterrumpida.	4
1.9. Seguridad lógica.	5
1.10. Criptografía.	5
1.10.1. Autenticación	6
1.10.2. Privacidad	6
1.11. Cifrado de ficheros en línea de comandos	7
1.12. Conceptos generales sobre permisos	7
1.13. Propiedad de archivos	8
1.14. Regulando los permisos en ficheros	8
1.15. Permitiendo el acceso a otros usuarios	8
1.16. Permisos en directorios	9
1.17. Cambiando permisos	9
1.18. Curiosidades	9
1.19. Listas de control de acceso.	9
1.19.1. Ejercicio con permisos	10
1.19.2. Ejercicio resuelto con listas de acceso (II)	11
1.20. Establecimiento de políticas de contraseñas.	12
1.21. Políticas de almacenamiento.	12
1.22. Copias de seguridad e imágenes de respaldo.	12
1.23. Medios de almacenamiento.	13
1.24. Copias de seguridad incrementales y diferenciales con tar	14
1.24.1. Copia de seguridad incremental	14
1.24.2. Copia de seguridad diferencial	15
1.25. Sistemas biométricos. Funcionamiento. Estándares.	16
1.26. Análisis forense en sistemas informáticos	17
1.27. Funcionalidad y fases de un análisis forense.	17
1.28. Respuesta a incidentes.	17
1.29. Análisis de evidencias digitales.	17

1.30.	Herramientas de análisis forense.	17
1.31.	El sistema operativo Unix	17
1.32.	Anexo: Guest additions	18
2.	Instalación y configuración de cortafuegos	19
2.1.	Definición de cortafuegos	19
2.2.	Recordatorio de los conceptos básicos de redes	19
2.3.	Utilización de cortafuegos.	20
2.4.	Filtrado de paquetes de datos.	20
2.5.	Tipos de cortafuegos. Características. Funciones principales.	20
2.6.	Instalación de cortafuegos. Ubicación.	21
2.7.	Reglas de filtrado de cortafuegos.	21
2.7.1.	Comandos y fichero de configuración	21
2.7.2.	El cortafuegos NfTables	22
2.7.3.	Gestión de tablas	23
2.7.4.	Gestión de cadenas	23
2.7.5.	Gestión de reglas	24
2.7.6.	Acciones sobre paquetes	25
2.8.	Pruebas de funcionamiento. Sondeo.	25
2.9.	Registros de sucesos de un cortafuegos.	25
2.10.	Cortafuegos integrados en los sistemas operativos.	25
2.11.	Cortafuegos libres y propietarios.	26
2.12.	Distribuciones libres para implementar cortafuegos en máquinas dedicadas.	26
2.13.	Cortafuegos hardware.	26
2.14.	Anexo: configuración de IP en Linux con Netplan	26
2.15.	Anexo: ejercicio de configuración	27
2.15.1.	Ejemplo: denegación del servidor web al exterior	27
2.16.	Anexo: resolución de problemas	28
2.17.	Un ejercicio comentado	28
2.17.1.	Solución 1 (errónea)	28
2.17.2.	Una solución un poco mejor	29
2.17.3.	Una solución (un poco mejor)	29
2.17.4.	Otra solución	30
2.17.5.	Otra solución (más segura)	30
2.18.	Ejercicio resuelto con ficheros	30
2.18.1.	Resolución	31
3.	Implantación de soluciones de alta disponibilidad	33
3.1.	Definición y objetivos.	33
3.2.	Virtualización de sistemas.	33
3.2.1.	Modos de red en VirtualBox	34
3.3.	Posibilidades de la virtualización de sistemas.	35
3.4.	Herramientas para la virtualización.	35
3.5.	Configuración y utilización de máquinas virtuales.	35
3.6.	Alta disponibilidad y virtualización.	35
3.6.1.	El fichero Vagrantfile	36
3.6.2.	Operaciones con el interior de la máquina: cambiar la IP a una tarjeta pública	37
3.6.3.	Operaciones con el interior de una máquina Virtual: MySQL	38
3.6.4.	Problemas típicos con Vagrant	40
3.7.	Simulación de servicios con virtualización.	41
3.8.	Análisis de configuraciones de alta disponibilidad	41
3.8.1.	Hardware duplicado	41
3.8.2.	Virtualización	42
3.9.	Docker	42

3.9.1.	Contenedores	42
3.10.	Imágenes y procesos Docker	42
3.10.1.	Gestión de contenedores	43
3.11.	Los elementos básicos de Docker	43
3.11.1.	La consola y los contenedores	44
3.11.2.	Gestión de imágenes	44
3.11.3.	Instalando Docker	44
3.11.4.	Conexiones de red en Docker	45
3.11.5.	Creando nuestra propia red en Docker	46
3.11.6.	Ejercicios de redes Docker	46
3.11.7.	Almacenamiento con Docker	47
3.11.8.	Montaje de directorios	47
3.11.9.	Almacenamiento en memoria	47
3.11.10.	Volúmenes	48
3.11.11.	Usando volúmenes	48
3.11.12.	Un ejemplo simple de Docker	48
3.11.13.	Un ejemplo más avanzado de Docker usando MySQL	48
3.11.14.	Ejercicio	49
3.12.	Otros ejercicios con Docker	49
3.12.1.	Ejercicio (I)	49
3.12.2.	Solución al ejercicio I de Docker	50
3.12.3.	Ejercicio II de Docker	50
3.12.4.	Fichero PHP	50
3.12.5.	Ejercicio III de Docker	51
3.12.6.	Parte 1	51
3.12.7.	Parte 2	51
3.12.8.	Parte 3	52
3.12.9.	Parte 4	52
3.12.10.	Ejercicio con Docker (IV)	53
3.12.11.	Enunciado	53
3.12.12.	Pista	53
3.12.13.	Solución	53
3.13.	Funcionamiento ininterrumpido.	55
3.14.	Integridad de datos y recuperación de servicio.	55
3.15.	Servidores redundantes.	55
3.16.	Sistemas de clusters.	55
3.17.	SAN, NAS, FiberChannel	55
3.18.	Balanceadores de carga.	55
3.19.	Instalación y configuración de soluciones de alta disponibilidad.	55
3.20.	Ejercicio: recuperando una web con Vagrant	55
3.21.	Solución a la recuperación de la web	56
3.22.	Solución al ejercicio de alojar una base de datos en Docker	56
4.	Instalación y configuración de servidores proxy	59
4.1.	Material para la unidad	59
4.2.	Tipos de proxy . Características y funciones.	59
4.2.1.	Tipos de proxy en función de la arquitectura de red.	60
4.2.2.	Tipos de proxy en función de la aplicación	61
4.3.	Instalación y configuración de clientes proxy.	61
4.4.	Instalación de servidores proxy	61
4.5.	Instalación de servidores proxy con apt source	62
4.6.	Instalación de servidores proxy desde cero.	63
4.6.1.	Ficheros de interés	64
4.6.2.	Iniciando y parando el servicio	64

4.6.3.	Squid y SSL/TLS	65
4.6.4.	ACLs en Squid. ACLS de origen.	66
4.7.	Configuración de filtros.	66
4.7.1.	ACLs en Squid. ACLS de destino.	66
4.7.2.	ACLs basadas en la URL	67
4.7.3.	ACLs basadas en la ruta	68
4.7.4.	ACLs basadas en el tipo de archivo	68
4.8.	Métodos de autenticación en un proxy	68
4.9.	Configuración del almacenamiento en la caché de un proxy	70
4.10.	Proxys inversos.	70
4.11.	Proxys encadenados.	72
4.12.	Pruebas de funcionamiento. Herramientas gráficas.	72
5.	Legislación y normas sobre seguridad	73
5.1.	Legislación sobre protección de datos.	73
5.2.	La Ley Orgánica de Protección de Datos	73
5.2.1.	Artículo 1. Objeto de la Ley	73
5.2.2.	Artículo 2. Ámbito de aplicación	73
5.2.3.	Artículo 3. Datos de fallecidos	73
5.2.4.	Artículo 4. Exactitud de los datos	74
5.2.5.	Artículo 5. Deber de confidencialidad.	74
5.2.6.	Artículo 6. Tratamiento basado en el consentimiento del afectado.	74
5.2.7.	Artículo 7. Consentimiento de los menores de edad.	74
5.2.8.	Artículo 8. Tratamiento de datos por obligación legal, interés público o ejercicio de poderes públicos.	74
5.2.9.	Artículo 9. Categorías especiales de datos.	74
5.2.10.	Artículo 10. Tratamiento de datos de naturaleza penal.	74
5.2.11.	Artículo 11. Transparencia e información al afectado.	75
5.2.12.	Artículo 12. Disposiciones generales sobre ejercicio de los derechos.	75
6.	Implantación de técnicas de acceso remoto. Seguridad perimetral	77
6.1.	Elementos básicos de la seguridad perimetral.	77
6.2.	Perímetros de red. Zonas desmilitarizadas. Router frontera.	77
6.3.	Arquitectura débil de subred protegida.	77
6.4.	Arquitectura fuerte de subred protegida.	77
6.5.	Políticas de defensa en profundidad.	77
6.6.	Defensa perimetral.	78
6.7.	Defensa interna.	78
6.8.	Factor Humano.	78
6.9.	Redes privadas virtuales. VPN.	78
6.9.1.	VPN a nivel de enlace.	78
6.9.2.	VPN a nivel de red. SSL, IPSec	78
6.9.3.	VPN a nivel de aplicación. SSH.	78
6.10.	Infraestructura de clave pública o PKI	78
6.10.1.	Paso 0: instalar Easy-RSA	78
6.10.2.	Paso 1: Crear una autoridad	79
6.10.3.	Paso 2: crear la infraestructura de claves	79
6.10.4.	Paso 3: construir los ficheros de la CA	79
6.10.5.	Paso 4: generar un certificado para un servidor	79
6.10.6.	Paso 5: generar un certificado para un cliente	79
6.10.7.	Paso 5: precalcular parámetros de claves	80
6.10.8.	Paso 6: configurar el servidor y arrancarlos	80
6.10.9.	Paso 7: configurar el cliente y arrancarlo.	81
6.11.	Beneficios y desventajas con respecto a las líneas dedicadas.	82

6.12. Técnicas de cifrado. Clave pública y clave privada.	82
6.13. Intérprete de comandos SSH.	82
6.14. Gestión de archivos SSH.	82
6.15. Servidores de acceso remoto	82
6.16. Protocolos de autenticación.	82
6.17. Configuración de parámetros de acceso.	82
6.18. Servidores de autenticación.	82

Adopción de pautas de seguridad informática

1.1 Fiabilidad, confidencialidad, integridad y disponibilidad.

A continuación definimos los siguientes términos

- **Fiabilidad:** la capacidad de conseguir que un SI ofrezca la información sin pausas entre peticiones.
- **Confidencialidad:** capacidad de conseguir que la información se muestre solo a las personas que estén autorizadas para ello.
- **Integridad:** capacidad de conseguir que la información no se altere por causas involuntarias.
- **Disponibilidad:** capacidad de respuesta a una peticiones con las mínimas pausas por causas involuntarias.

En relación con el último punto, se mide la disponibilidad de un SI en «nueves».

- Se dice que un SI ofrece una disponibilidad de «2 nueves», si está disponible el 99 % del tiempo.
- Se dice que un SI ofrece una disponibilidad de «3 nueves», si lo está al 99.9 %.
- Se dice que un SI ofrece una disponibilidad de «4 nueves» si lo está al 99.99 %.
- Se dice que un SI ofrece una disponibilidad de «5 nueves» si lo está al 99.999 %

Ejercicio: Si un año tiene 365 días, calcular cuanto tiempo podría estar como «no disponible» cada uno de los sistemas que hemos enumerado en el punto anterior.

1.2 Elementos vulnerables en el sistema informático; hardware, software y datos.

«Vulnerable»: medida de la capacidad de un sistema para fallar de manera inesperada. En pocas palabras una vulnerabilidad es un punto débil.

Como resulta evidente en un SI hay tres grandes elementos que son susceptibles de ser vulnerables:

- **Hardware.**
 - Fluido eléctrico.
 - Placa base.
 - RAM: errores muy difíciles de detectar.
 - Discos: hay abundantes estadísticas acerca de sus tasas de fallos.
 - Tarjetas gráficas.
 - Interconexiones, cables y/o soldaduras
- **Software.**
 - Sistema operativo: es importante tener activada la actualización automática que aplica «parches» sin necesidad de recordar aplicarlas manualmente. Para evitar la aplicación de actualizaciones que puedan estropear otras partes se suele aconsejar NO TENER ACTIVADA LA ACTUALIZACIÓN AUTOMÁTICA en equipos críticos y sí tenerla en otros equipos que actúen como «cobayas.»
 - Aplicaciones. También pueden mostrar fallos que den lugar a consecuencias muy desagradables especialmente con los datos.
- **Datos.** Hoy en día son casi con total seguridad el activo más valioso de la empresa. Para protegerlos habrá que tomar muchas medidas de seguridad.

1.3 Análisis de las principales vulnerabilidades de un sistema informático.

Cuando se habla de vulnerabilidad, se asocia este término con problemas software. Una vulnerabilidad puede conllevar una serie de problemas muy graves:

- Que un intruso consiga permisos de administración en un sistema.
- Que un virus informático consiga tomar el control de los equipos de la empresa.
- Que un software o individuo consiga borrar/alterar/cifrar datos de la empresa.

En Internet todas las vulnerabilidades detectadas se publican como un informe CVE (Common Vulnerability Exposure)

Recientemente se han descubierto **vulnerabilidades a nivel de microprocesador**. En entornos muy sofisticados existen unas vulnerabilidades llamadas TEMPEST.

Algunas aplicaciones basadas en bases de datos son susceptibles de sufrir «ataques SQL» o «inyecciones SQL» o «SQL injects».

Otro tipo de ataque común son los HTML/JS injects.

En líneas generales, ningún programa web debe confiar en lo que escriben sus usuarios.

1.4 Amenazas. Tipos.

Clasificando por lugar

- Interna: los problemas originados dentro de la propia empresa son **los más frecuentes y los de impacto más grave**
- Externa: son las originadas fuera de la propia empresa.

Clasificando por mecanismo

- Físicas
- Lógicas

1.5 Amenazas físicas.

Son todas aquellas que hacen uso de algún mecanismo tangible, ya sea por acción efectiva o por fallo, para perjudicar el funcionamiento de los sistemas informáticos.

- Rotura intencionada.
- Desastre natural: terremotos, inundaciones, incendios, etc...
 - Se debe disponer de la protección antiincendios adecuada.
 - No todos los extintores son apropiados para todo.
 - Los seguros no suelen cubrir eventos de este tipo.
 - Se desaconseja la instalación de centros de datos en bajos o sótanos.

En relación con todos estos sucesos se recomiendan algunas medidas básicas de protección.

- Barreras físicas.
 - Los servidores deberían estar cerrados con llaves y con acceso restringido.
 - Controles de acceso con tarjeta y/o guardia de seguridad.
 - En relación con el punto anterior a veces se llegan a utilizar mecanismos biométricos.
 - Puertas con apertura programada.
- Protección eléctrica.

1.6 Amenazas lógicas.

¿Qué problemas podrían causarse por motivos de un uso inapropiado de software?

- Ataques a nivel de red IP. P.ej ataques de tipo «spoofing». Phishing. MITM
- A nivel de SO. Buffer overflow. Errores humanos.
- A nivel de aplicación. Un problema muy común es el SQL/HTML/JS injection y/o los errores humanos que provoquen fugas de datos.
- Malware: spyware, ransomware, virus, DOS (Denial of service).

1.7 Seguridad física y ambiental

La seguridad física y ambiental implica controlar tres grandes tipos de posibles acciones:

- Engaños/fraudes.
- Robos/pérdidas.
- Sabotajes.

Para evitarlos se suele recurrir a una o varias medidas de las siguientes:

- Sistemas biométricos.
- Personal de seguridad.
- Protección electrónica como sensores de presencia, infrarrojos, de movimiento.

1.8 Sistemas de alimentación ininterrumpida.

Un sistema de alimentación ininterrumpida o SAI protege contra problemas eléctricos comunes que pueden afectar al funcionamiento normal de un sistema informático.

- Bajadas de tensión. Produce daños a largo plazo.
- Interrupciones del suministro. Da lugar a perjuicios económicos.
- Subidas de tensión puntuales. Menores o iguales de 4 milisegundos y producen daños en días/semanas.
- Subidas de tensión sostenidas. Dura mas de 4 milisegundos y produce daños en escasos minutos e instantáneos.

En líneas generales el parámetro principal que debemos mirar en un SAI es su «potencia aparente».

La potencia aparente de un SAI se mide en «voltio-amperios» o «kilo-voltio-amperios» o «KVA» (también pronunciado como «kabeas» o «kivas»)

Lo que nos interesa es la potencia eficaz que se obtiene multiplicando la aparente por 0,75. En algunos SAI nos indican el factor de potencia. En ese caso, sí podemos saber directamente la potencia eficaz multiplicando la potencia aparente por ese factor.

Pot eficaz (Wattios) = Pot. apar (VA) Factor de pot.

Supongamos un SAI en el que la caja simplemente indica 2000VA (o 2KVAS). Si no nos dicen nada, asumiremos que en realidad ese SAI ofrece $2000 \times 0,75 = 1500 \text{ W}$

Si tuviésemos 3 ordenadores y cada uno consumiese 650W está claro que no podríamos conectar los 3.

Si un SAI se anuncia indicando que ofrece 1500KVA y 850W de potencia ¿qué factor de potencia ofrece?

- Los 1500KVA son la «nominal/máxima/aparente»
- Los 850W son la «eficaz/de salida»

Si Eficaz=Aparente * FdP entonces

FDP=Eficaz/aparente

FDP=850/1500

Cuando se diseña un edificio con instalaciones informáticas es frecuente que con el tiempo haya cambios y finalmente sea necesario ampliar. Por ello, se recomienda incrementar nuestros cálculos en torno a un 20-30 %

Supongamos que deseamos instalar servidores que en su conjunto consumen 1300W.

¿Que potencia aparente deberíamos buscar al comprar un SAI?

No nos dicen el factor de potencia así que usaremos 0,75. Así la potencia aparente debería ser $E_{\text{ficaz}}/0,75$ es decir 1733 VA. Como dicha potencia podría resultar insuficiente en el futuro, incrementaremos, por ejemplo un 20 % multiplicando los VA por 1,20. Así, $1733 * 1,20 = 2080$ VA.

1.9 Seguridad lógica.

Implica restringir el acceso a datos en función de la persona que lo intente:

- Claves de acceso.
- Tarjetas de identificación.
- Copias de seguridad.
- Listas de control de acceso.
- Control horario.
- Roles.
- Cortafuegos.
- Distribución de carga.
- Redundancia de sistemas

1.10 Criptografía.

La Criptografía es la técnica que transforma mensajes en otros mensajes cuyo contenido no se puede conocer. Un mecanismo muy básico es por ejemplo el «cifrado César».

ABCDEFGHIJKLMNOPQRSTUVWXYZ XYZABCDEFGHIJKLMNOPQRSTUVWXYZ

ATACAD AL AMANECER XQXZXA X...

El cifrado César es un «mecanismo de sustitución». Existen otros mecanismos basados en la «transposición».

Si hacemos una transposición de 4 columnas del mensaje «ATACAD AL AMANECER» se obtiene esto.

ATAC AD AL AM ANEC ER

AALAETD NRA AE CAMC

Algunos sistemas de cifrado «combinan otros sistemas». Supongamos que alguien aplica un «César desplazamiento 3» con un «transposición de 4 columnas».

ABCDEFGHIJKLMNOPQRSTUVWXYZ XYZABCDEFGHIJKLMNOPQRSTUVWXYZ

ATACAD AL AMANECER XQXZXA XI XJXKBZBO

XQXZ XA XI XJ XKBZ BO

XXIXBQA KOX XB ZXJZ

A veces una sustitución puede usar una clave como «2527»

ABCDEFGHIJKLMNOPQRSTUVWXYZ 2527252725272527252725272527 DH...

Existen diversas técnicas llamadas «criptoanálisis» que investigan como descifrar mensajes cifrados.

- Averiguar el idioma en que se escribió.
- Buscar palabras comunes: «el» «la» «los» «las», «con»

- Usar «fuerza bruta»

Como el uso de la informática ha simplificado muchísimo el atacar claves se investigaron nuevos mecanismos de criptografía: los asimétricos.

En los viejos sistemas se podía descifrar un mensaje si alguien obtenía la clave, solo había que hacer el proceso inverso

Los sistemas asimétricos utilizan una clave de cifrado y otra de descifrado. Aunque se tenga una clave es matemáticamente imposible averiguar la otra clave por lo que se puede dar a todo el mundo una de las claves (llamada habitualmente clave pública) y conservar la otra (llamada clave privada). Además, podemos usar las claves para lo que queramos y por ejemplo en unos casos cifraremos con la clave pública y en otros tal vez cifremos con la clave privada.

En los puntos siguiente veremos como usar la criptografía asimétrica para dos cosas distintas: la autenticación y la privacidad.

1.10.1 Autenticación

La autenticación consiste en «comprobar que alguien es quien dice ser», ¿como conseguirlo?. Muy sencillo.

En primer lugar tendremos tres elementos:

1. Un servidor (como por ejemplo Amazon) que desea ofrecer garantías a sus clientes de que cuando se conectan a Amazon realmente se conectan a un servidor de Amazon.
2. Por otro lado tendremos clientes que desean obtener la garantía de que cuando escriben `http://amazon.es` **realmente se están conectando a un servidor de Amazon**
3. Por último tendremos un tercero que se encarga de verificar el proceso para ambos llamada CA o «autoridad de certificación».

Así, el proceso es el siguiente:

1. Amazon envía a la CA una «petición de firma de certificado».
2. La CA lo recibe y lo cifra con su clave privada.
3. La CA da su clave pública (que se usará para descifrar) a todos los navegadores, que lo incorporan de serie en la instalación.
4. Amazon pone en sus servidores el certificado «firmado» por la CA
5. Cuando el cliente se conecta a Amazon, el servidor le envía el certificado.
6. El cliente descarga el certificado y lo descifra con la clave pública de la CA, obteniendo un fichero válido que le garantiza que esa máquina realmente es Amazon.

1.10.2 Privacidad

Una vez que Amazon ha ofrecido garantías a su cliente ahora se necesita usar la criptografía para que el usuario pueda hacer sus compras sin que nadie espíe. Ahora las claves se usarán al revés.

1. El cliente se conecta a Amazon (después de haber comprobado que el certificado es correcto)
2. Amazon envía al cliente su clave de cifrado.
3. El cliente la recibe y cifra el pedido con la clave pública de cifrado de Amazon.
4. El mensaje viaja por la red pero nadie podrá descifrarlo.
5. El mensaje llega a Amazon y usa su clave privada para descifrar.

1.11 Cifrado de ficheros en línea de comandos

Existe una utilidad de libre distribución llamada **gpg** que existe para muchos sistemas operativos distintos y que permite trabajar con criptografía asimétrica. Este programa asume que usaremos la clave pública para cifrar y la privada para descifrar.

Cuando se trabaja con claves públicas todo se guarda en un «almacén de claves» al cual se debe acceder con otra clave distinta. Cuando listamos la clave pueden verse entre corchetes cosas como estas:

- [SC] Esto significa que la clave puede «firmar» (S) y para «crear un certificado».
- [SCEA] La clave puede firmar, generar un certificado, puede «encriptar» (E) y puede «autenticar» (A)
- También veremos el nivel de confianza de una clave que puede ser algo como [ultimate] (confianza absoluta) [full] (confianza completa) y otros términos que indican hasta qué punto confiamos en la clave. Hay varios niveles: «undefined», «never», «marginally», «full» y «ultimate».
- Se debe empezar por generar una pareja de claves usando el comando **gpg --full-generate-key** (el proceso de generación de claves puede ser muy lento, se recomienda tener paciencia y a ser posible abrir otra consola y trabajar en ella).
- Una vez generado tendremos un directorio llamado **.gnupg** en el que se almacenan las claves. Podemos listar las claves de nuestro almacén con **gpg --list-keys**
- Una vez se tenga generada la clave la costumbre es tener preparado un «certificado de revocación». Se utilizará si creemos que nos han robado alguna clave y distribuiremos el fichero para avisar de que no se debe confiar en nuestras claves. Esto se hace con el comando **gpg --gen-revoke "usuario" --output ClaveRevocada.asc**. Se pueden usar otros nombres de fichero pero la costumbre es usar la extensión **.asc**
- A continuación se suele extraer nuestra clave pública del almacén de claves y ponerla en un fichero con el comando **gpg --export <usuario> > ClavePublicaUsuario.gpg**. Se generará un fichero binario en **ClavePublicaUsuario.gpg**. Si deseamos generar un fichero con ASCII normal podemos hacer esto **gpg --armor --export <usuario> --output ClavePublicaUsuario.gpg**
- Una vez que alguien nos haya pasado su clave pública deberemos incorporarla a nuestro almacén usando **gpg --import <fichero.gpg>**
- Para cifrar un fichero y que solo lo pueda descifrar la otra persona usaremos **gpg --encrypt --recipient pepito@gmail.com ficheroparacifrar**. El comando generará un fichero nuevo cifrado.
- Cuando tengamos la clave de alguien podemos enviarle un fichero cifrado con su clave pública que **solo esa persona podrá descifrar**. Para ello indicaremos el fichero y la persona que va a recibir dicho fichero cifrado con **gpg --output ficherooriginal.doc.gpg --recipient persona@mail.com ficherooriginal.doc**
- Finalmente podremos descifrar un fichero que nos hayan enviado usando **gpg --decrypt <ficheroparanosotros>**.

1.12 Conceptos generales sobre permisos

1) Comprobamos nuestro usuario con el comando **whoami** 2) Sabemos que podemos crear usuarios con el comando **adduser**. Por ejemplo, creemos el usuario **usu01** con el comando **adduser usu01** 3) Podemos iniciar sesión en otra consola usando la teclas **Alt+F2**, **Alt+F3**. Iniciemos sesión con ese usuario **usu01**. Si ejecutamos **ls -la** podremos ver los ficheros y permisos asociados. 4) Detalle importante: existe el concepto de «grupo». Un grupo puede contener muchos usuarios. Para evitar problemas de seguridad, cuando creamos un usuario, Linux crea automáticamente un grupo con el mismo nombre y en ese grupo solo está ese usuario. Es decir, que al crear el usuario **usu01** se ha creado automáticamente un grupo llamado **usu01** en el que hay un solo usuario, el usuario **usu01** 5) Volvemos al tema de los ficheros. Todo fichero tiene un permisos asociados y la estructura de permisos, sigue un patrón muy conocido: * Primero podemos ver uno de estos símbolos: «-», «d», «l», «s» * Después podemos ver tres símbolos, cada uno de

los cuales puede ser «-«,»r«,»w» o «x». Por ejemplo podríamos ver «-x«,»rw-«, «-w-». Esta primera columna de 3 símbolos son los *permisos que se le aplican al usuario*

Así por ejemplo, en esta entrada:

```
-rw-r--r-- 1 usu01 usu01 3771 sep 25 07:59 .bashrc
```

El usuario «usu01» tiene asignados los permisos «rw-». Es decir, que el usuario «usu01» puede LEER el contenido (r) del fichero y además puede ESCRIBIR en el fichero (w). Sin embargo, el usuario no puede EJECUTAR el fichero (no hay x)

La siguiente columna de 3 símbolos de esa entrada son la cadena «r--»: eso significa que «los usuarios que estén metidos en el grupo llamado usu01» podrán LEER el contenido del fichero, pero no pueden ESCRIBIR en él ni pueden EJECUTARLO.

Por último hay una tercera columna de 3 permisos, que también pueden ser «r», «w», «x» o «-». El significado de los permisos es el mismo. Esta tercera columna indica qué permisos se deben aplicar a usuarios que ni son el propietario ni están en el grupo del fichero.

1.13 Propiedad de archivos

El primer texto que vemos en una secuencia de permisos es el propietario del archivo («owner»). Sin embargo, el dueño puede «transferir» la propiedad usando el comando `chown`. Sin embargo esta puede estar limitado por los permisos de los directorios padre.

También puede cambiarse el grupo de un archivo usando `chown`

1.14 Regulando los permisos en ficheros

Si creamos un usuario `usu02` y ese usuario intenta ver un fichero del usuario `usu01` nos vamos a topar con una sorpresa: ¡no podrá verlo!. ¿Por qué? porque no sirve de nada que el fichero del usuario `usu01` tenga el permiso «r» para «otros». Si el fichero está dentro de un directorio protegido, el fichero del usuario `usu01` no será visible.

1.15 Permitiendo el acceso a otros usuarios

Para que el usuario `usu01` dé permisos para que el usuario `usu02` pueda ver sus ficheros se pueden hacer tres cosas:

- El usuario `usu01` da permisos a otros usuarios para que puedan «explorar» su directorio. Este paso es fácil y funciona, pero puede que sin querer «demos permisos de más»
- La otra posibilidad es usar un directorio específico vacío en el que solo esté el fichero a compartir.
- Usar listas de control de acceso o ACLs.

1.16 Permisos en directorios

Los símbolos «r», «w» y «x» significan cosas distintas cuando hablamos de directorios:

- El «r» significa que ese usuario o grupo podrá ver los nombres de fichero que hay en ese directorio.
- El permiso «w» significa que se podrán escribir o no «nombres de fichero nuevos»
- El permiso «x» permite hacer `cd` para entrar en ese directorio.
- El permiso «t» es bastante especial y cambia las reglas de borrado. Si tenemos permisos «w» y «x» en un directorio podremos borrar archivos aunque no seamos los propietarios. Si alguien activa el permiso «t» solo el dueño del archivo, del directorio (o el administrador) podrán borrar el fichero.

1.17 Cambiando permisos

Se puede usar el comando `chmod` para cambiar los permisos de un fichero o directorio. La sintaxis es más o menos así:

```
chmod [augo][+][rwx] <nombre fichero>
```

Así, por ejemplo, el usuario `usu01` puede QUITAR el permiso para ver nombres de fichero a otros usuario en su directorio con `chmod o-r /tmp/compartida`

1.18 Curiosidades

Cambiando permisos en directorios se pueden dar situaciones tan curiosas como que alguien puede «entrar» en un directorio pero no puede hacer `ls` y no puede ver los ficheros que hay (lo cual no sirve de mucho)

1.19 Listas de control de acceso.

Antes de empezar instala las herramientas para ACLs (Access Control Lists) en el caso de que no las tuvieras. El comando necesario es `sudo apt-get install acl`

En los sistemas UNIX (como GNU/Linux) tradicionalmente se han usado permisos basados en usuarios y grupos. Así, cuando se crear un usuario (con `sudo adduser nombreusuario`) tradicionalmente se crea un grupo con el mismo nombre y en el que está solo ese usuario.

Cuando un usuario cualquiera crea un fichero, ese fichero tiene asignados automáticamente unos permisos que pueden ser

- `r` si se puede leer el fichero
- `w` si se puede escribir/modificar el fichero.
- `x` si se puede ejecutar.

Estos permisos pueden ser del usuario, del grupo al que pertenece o de otros usuarios en general. Así, un fichero cualquiera puede mostrar unos permisos como estos (necesitaremos el comando `ls -l` para ver los permisos).

```
-rw-rw-r-- 1 profesor profesor 171 oct 3 2016 Makefile
-rw-rw-r-- 1 profesor profesor 11 sep 30 2016 postinst
-rw-rw-r-- 1 profesor profesor 58 sep 30 2016 README
```

Figura 1: Ejemplos de permisos en un sistema GNU/Linux

Si examinamos el fichero Makefile veremos que tiene unos permisos como estos `-rw-rw-r--` y veremos también que pone `profesor profesor`. Por este orden, esto significa

- El usuario propietario del fichero se llama **profesor**. El grupo asignado a este fichero es **profesor** (recuérdese que puede cambiarse el propietario con `chown` y el grupo con `chgrp`)
- El primer permiso tiene un `-`. Este primer permiso indica el tipo de fichero, que puede ser «fichero normal» (`-`), «directorio» (veríamos «`d`»), «enlace» (`l`),...
- Despues vemos `rw-`. Este primer grupo de tres permisos es el aplicado al propietario (que este caso es **profesor**). Este grupo significa que el propietario puede leer y escribir en este fichero, pero no ejecutar.
- Despues vemos `rw-`. Estos son los permisos que se aplicarán al grupo, que en este caso es el grupo «profesor» (no pasa nada porque un grupo se llame igual que un usuario). Esto significa que cualquier usuario asignado al grupo «profesor» también podrá leer y escribir el fichero.
- Por último vemos `r--`. Esto significa que cualquier otro usuario que ni sea **profesor** ni pertenezca al grupo **profesor** podrá hacer nada que no sea leer en el fichero.

Este sistema de permisos ha funcionado muy bien durante mucho tiempo, sin embargo con el tiempo ha mostrado algunas flaquezas.

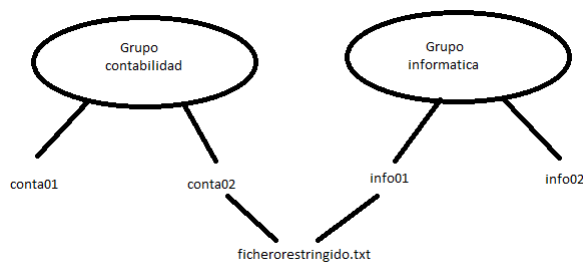


Figura 2: Un ejemplo de problema con los permisos

1.19.1 Ejercicio con permisos

Crea los usuario `info01`, `info02`, `conta01` y `conta02`. Inicia sesión con cada uno de ellos y haz que cada uno de ellos cree un fichero con su mismo nombre, es decir `info01.txt`, `info02.txt`, `conta01.txt` y `conta02.txt`.

Los comandos serían `sudo adduser info01`, `sudo adduser info02`, `sudo adduser conta01` y `sudo adduser conta02`.

Una vez hecho esto, nos salimos con el comando `exit` e iniciamos sesión con, por ejemplo, `conta01`. El sistema nos dejará en el directorio `/home/conta01` y en él podremos crear el fichero. Puedes asegurarte de que estás en el directorio correcto con `pwd`. Puedes crear el fichero con `nano conta01.txt` y rellenando el fichero con el texto que quieras. Sal de la consola y repite el proceso con el resto de usuario.

Ahora hay que establecer permisos, por ejemplo usaremos la configuración `-rw-r-----` que hace lo siguiente:

- Permite leer y escribir (`rw-`) al propietario.
- Permite leer a los usuarios que estén en el grupo del fichero (`r-`).
- No deja hacer **nada** a los otros (`---`).

¿Que harías si deseas permitir algo como lo siguiente?

- Que el fichero `conta01.txt` sea de lectura y escritura para `conta02`
- Que el fichero `conta01.txt` de lectura y escritura para `info01`.

- Que el fichero `conta01.txt` sea de lectura para `info02`.

La solución a este problema sería compleja. Por ejemplo podríamos hacer esto:

- Siendo administradores crear un grupo: `sudo addgroup info01conta02`.
- Siendo administradores modificar los usuarios `info01` y `conta02` para que pertenezcan al nuevo grupo con `sudo usermod -a -G info01conta02 info01` y `sudo usermod -a -G info01conta02 conta02`.
- Cambiar el grupo del fichero `conta01.txt` con `sudo chgrp info01conta02 conta01.txt`
- Dar al fichero `conta01` permisos de lectura y escritura para el grupo con `sudo chmod g+w conta01.txt`
- Dar permisos de lectura a **otros usuarios** con `chmod o+w conta01.txt`

Sin embargo el último caso es **un agujero de seguridad**. Sin querer vamos a dar permisos de lectura a `info02` y a *todos los demás usuarios*

Se necesita usar el comando `setfacl` que funciona de esta manera:

- Podemos añadir permisos con `setfacl -m u:info01:rw conta01.txt`. Podemos hacer esto desde el usuario normal `conta01`.
- Ejecutamos `setfacl -m u:conta02:rw conta01.txt`.
- Ejecutamos `setfacl -m u:info02:w conta01.txt`

Para consultar los permisos de un archivo usaremos `getfacl conta01.txt`. Si nos equivocamos y deseamos borrar una entrada de la lista usaremos cosas como `setfacl -x u:conta02 conta01.txt`

1.19.2 Ejercicio resuelto con listas de acceso (II)

Hacer que el usuario «usuario» fabrique un fichero llamado «ficherodatosespeciales.txt» con un texto cualquiera y que ocurra esto:

Datos especiales:

- El `info01` sí puede leerlo, pero no escribir
- El `info02` puede leerlo y escribirlo (modificarlo).
- El `conta01` no puede hacer NADA.
- El `conta02` puede leerlo pero no escribirlo.

Iniciar sesión con todos los usuarios y verificar que efectivamente solo ocurre lo que nos han indicado.

1. Iniciar sesión con «usuario».
2. Creamos el fichero «fichero.txt».
3. Una buena medida es quitar todos los permisos, `chmod a-rwx fichero.txt`
4. Para que el `info01` sí pueda leer ejecutamos `setfacl -m u:info01:r fichero.txt`.
5. Para que el `info02` pueda tanto leer como escribir, ejecutamos `setfacl -m u:info02:rw fichero.txt`.
6. Para que el `conta01` no pueda hacer nada, no necesitamos nada especial, ya se quitaron todos los permisos como medida de seguridad.
7. Para que el `conta02` pueda leer y solo leer, solo necesitamos `setfacl -m u:conta02:r fichero.txt`

1.20 Establecimiento de políticas de contraseñas.

Por incómodo que resulte, las contraseñas:

- Deben ser largas (de 8 símbolos o más)
- Deben mezclar todos los siguientes conjuntos posibles, o al menos el máximo posible: mayúsculas, minúsculas, números y símbolos especiales.
- Deben cambiarse con la máxima periodicidad posible.
- Deben ser lo más distintas posibles a las claves antiguas.
- Deberían caducar automáticamente.
- No deberían almacenarse como «texto plano» en ningún sitio. Lo típico es almacenar contraseñas «cifradas».

1.21 Políticas de almacenamiento.

Se debe determinar lo siguiente en cuanto a los datos:

- ¿Qué datos se va a almacenar? Hay que recordar que la LOPD marca las principales directrices a tener en cuenta sobre la información almacenada.
- ¿Donde se va a almacenar? Los distintos medios tienen distintas características.
- ¿Qué mecanismos de copia se van a implementar? Por su excesivo tamaño tal vez no siempre podamos hacer una copia entera de todo el disco duro.

En cuanto al primer punto debemos recordar lo básico sobre los datos según la LOPD.

- Datos de nivel básico: en general información como nombre, apellidos, datos postales, información laboral...
- Datos de nivel medio: información financiera, infracciones administrativas, multas...
- Datos de nivel alto: filiación política, confesiones y/o religiones, sexualidad, datos sanitarios

¿Qué ocurre con las IP, datos sobre navegador, sistema operativo, etc...?

1.22 Copias de seguridad e imágenes de respaldo.

No es lo mismo una copia de seguridad que una imagen.

En cuanto a las copias de seguridad podemos hablar de:

- Copias completas. Son muy fáciles de aplicar y muy fáciles de recuperar pero pueden consumir muchísimo espacio.
- Copias incrementales. Una copia incremental siempre se fijará en la última copia que se hizo (da igual si la última fue una incremental o una completa). Esto ahorra mucho espacio pero si hay ue recuperar una copia hay que recuperar la última completa **más todas las incrementales** lo cual puede ser muy lento.
- Copias diferenciales. Son copias en las que solo se guarda lo que haya cambiado **con respecto a la última copia completa** . Así, si hay que recuperar una copia solo necesitamos la última completa y la última diferencial. Lo malo es que las copias intermedias ocupan más que las copias intermedias incrementales.

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Día||| |+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

En UNIX las copias se hacen de otra manera.

- En primer lugar necesitaremos el comando `tar`. Podemos crear un archivo llamado `usuarios.tar` que almacene todos los directorios de todos los usuarios ejecutando `sudo tar -cf /home/usuario.tar /home/`
- Para programar la ejecución de la copia habrá que «editar la tabla de trabajos programados» que se hace con `sudo crontab -e`. En realidad cada usuario tiene su propia tabla de trabajos pero como queremos leer directorios de otros usuarios ejecutaremos la tarea de copia desde la tabla de trabajos del administrador.
- Una vez que ejecutemos `crontab -e` veremos un fichero que permite indicar tareas y el horario de ejecución usando el formato `minutos horas dia-del-mes mes dia-de-la-semana comando`.

Así podemos escribir algo como esto:

```
* * * * * tar -cf /home/usuarios$(date '+\%d-\%M-\%Y\%H\%M').tar /home/*
```

1.23 Medios de almacenamiento.

- La «nube». Se debe recordar que «la nube» es «el ordenador de otra persona sobre el cual no tengo ningún control».
- Otros ordenadores. Ventajas: tenemos control sobre ellos, fácil accesibilidad, coste medio...
- NAS: Network Attached Storage. Coste medio, facilidad de uso, alta disponibilidad.
- Dispositivos USB: muy versátiles, pero muy sensibles a campos magnéticos/golpes.
- Medios ópticos. Ofrecen una mayor tasa de supervivencia que los USB extraíbles.
- Discos magnéticos: ofrecen la mejor tasa coste/supervivencia.
- Cinta magnética. Ofrecen de lejos, el mejor coste. Sin embargo son muy lentas de recuperar y rellenar
- Discos SSD. Son variantes de los dispositivos USB, pero normalmente usan una conexión al dispositivo que es más rápida que un USB.
- Imprimir los datos. No es tan inútil como podría parecer, la duración de los datos impresos puede ser muy alta y además son muy difíciles de «robar».

1.24 Copias de seguridad incrementales y diferenciales con tar

En ambos casos `tar` va a necesitar guardar información en un archivo propio. Esto lo hace para poder saber qué archivos se guardaron en la copia completa y distinguirlos de los nuevos archivos que aparezcan y que deberá guardar en la diferencial o incremental.

1.24.1 Copia de seguridad incremental

Fabriquemos un directorio `datos_importantes`:

```
mkdir datos_importantes
```

Entremos en él:

```
cd datos_importantes
```

Añadamos un par de archivos:

```
nano Archivo01.txt
nano Archivo02.txt
```

Salgamos del directorio:

```
cd ..
```

Y fabriquemos nuestra copia completa:

```
tar -cf /home/usuario/copia_completa.tar -g /home/usuario/info_copia /home/usuario/datos_
↪importantes
```

Obsérvese que hay tres «campos»:

- El `-cf /home/usuario/copia_completa.tar` indica el nombre que va a tener el archivo de copia.
- El `-g /home/usuario/info_copia` indica el nombre de archivo donde permitimos que `tar` guarde información sobre lo que contiene la copia (la siguiente copia incremental lo necesitará)
- El `/home/usuario/datos_importantes` indica el directorio que contiene los datos de los cuales queremos hacer la copia de seguridad.

Ahora añadamos un archivo a `datos_importantes`:

```
cd datos_importantes
nano Archivo03.txt
```

Ahora hay un archivo nuevo pero no queremos volver a hacer una copia completa. Nos salimos del directorio:

```
cd ..
```

Y lanzamos *casi el mismo comando*

```
tar -cf /home/usuario/copia_incremental_lunes.tar -g /home/usuario/info_copia /home/
↪usuario/datos_importantes
```

De esta manera tendremos muchos archivos `.tar` pero cada uno de ellos **solo tendrá los archivos nuevos con respecto a la última copia**. Para restaurar la copia habrá que ir haciendo:

- `tar -xf copia_completa.tar`
- `tar -xf copia_incremental_lunes.tar`
- `tar -xf copia_incremental_martes.tar`, y así sucesivamente.

1.24.2 Copia de seguridad diferencial

Haremos en gran parte lo mismo que antes. Empezaremos borrando nuestros directorios:

```
cd
rm -rf datos_importantes
rm -rf restaurar
```

Fabriquemos un directorio `datos_importantes`:

```
mkdir datos_importantes
```

Entremos en él:

```
cd datos_importantes
```

Añadamos un par de archivos:

```
nano Archivo01.txt
nano Archivo02.txt
```

Salgamos del directorio:

```
cd ..
```

Y fabriquemos nuestra copia completa (esta vez no indicaremos rutas absolutas, por variar):

```
tar -cf Copia_completa.tar datos_importantes/
```

Ahora añadamos algún fichero con datos importantes:

```
nano datos_importantes/Archivo03.txt
```

Y ahora hacemos una copia diferencial, indicando que queremos que en ella aparezcan solo los archivos que sean más nuevos que nuestra copia:

```
tar -cf Copia_diferencial_1.tar -N ./Copia_completa.tar datos_importantes/
```

Añadimos algún fichero más:

```
nano datos_importantes/Archivo04.txt
```

Y hacemos otra copia diferencial:

```
tar -cf Copia_diferencial_2.tar -N ./Copia_completa.tar datos_importantes/
```

Ahora nos fabricamos un directorio para probar a restaurar:

```
mkdir restaurar
```

Metemos en él la copia completa y solo la última diferencial::

```
cp Copia_completa.tar Copia_diferencial_2.tar restaurar
```

Entramos en el directorio:

```
cd restaurar
```

Y probamos a extraer todos los archivos .tar:

```
tar -xf Copia_completa.tar  
tar -xf Copia_diferencial_2.tar
```

Veremos que nos ha aparecido un directorio **con todos los archivos importantes**

1.25 Sistemas biométricos. Funcionamiento. Estándares.

Por desgracia la siguiente figura ilustra muy bien el problema actual con los estándares biométricos:

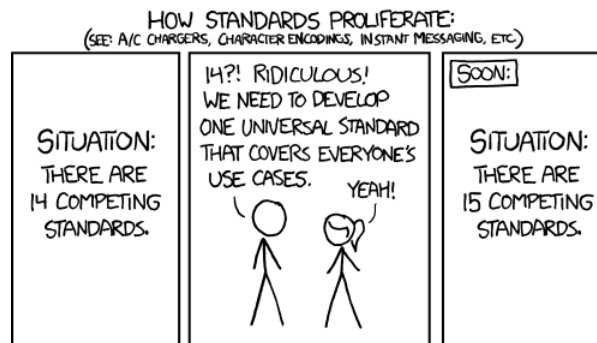


Figura 3: El problema actual con los estándares.

De hecho podemos nombrar los siguientes estándares:

- BioAPI: con el apoyo de IBM y Hewlett-Packard.
- BAPI: propiedad de I/O Software pero utilizado por Microsoft.
- ANSI X.9: pensada para la industria financiera e impulsado por los Estados Unidos.
- CBEFF: pensado para ficheros que almacenen información biométrica.
- Estándares del NIST estadounidense (National Institute of Standards and Technology).

A fecha de hoy ninguno de ellos ha triunfado sobre los demás y de hecho por el momento cada fabricante tiene sus propios mecanismos, software y drivers (con el consiguiente problema para los administradores de sistemas).

1.26 Análisis forense en sistemas informáticos

1.27 Funcionalidad y fases de un análisis forense.

1.28 Respuesta a incidentes.

1.29 Análisis de evidencias digitales.

1.30 Herramientas de análisis forense.

1.31 El sistema operativo Unix

A lo largo del curso usaremos GNU/Linux, un sistema operativo de tipo UNIX de libre distribución. Aunque GNU/Linux suele empaquetarse en «distribuciones» que suelen incluir un entorno gráfico en este módulo aprenderemos a movernos por el sistema utilizando los comandos.

- `mkdir` nos permite crear directorios.
- `cd` nos permite movernos a un directorio.
- `rm` nos permite borrar ficheros.
- `rmdir` nos permite borrar un directorio **siempre y cuando esté vacío**.
- `ls` muestra los ficheros del directorio actual.
- `cat` nos permite imprimir un fichero por pantalla.
- `man <comando>` nos permite obtener ayuda sobre un cierto comando (o incluso fichero de configuración).
- `pwd` nos muestra el nombre del directorio actual.
- `nano` nos da acceso a un pequeño editor de texto que nos permitirá editar, entre otras cosas, los ficheros de configuración del sistema.
- Cuando se manipulan ficheros se puede ocultar un fichero *usando el punto como primer carácter de un fichero*.
- `apt-get` nos permitirá instalar software de los repositorios del empaquetador de la distribución.
- Se puede ejecutar un comando escribiendo el nombre de dicho comando. Si el comando no está en las rutas de búsqueda se puede escribir la ruta completa.
- Se puede redirigir la salida de un comando hacia un fichero (usando `<comando> > <fichero>` o redirigir la salida de un comando hacia otro comando con `<comando> | <comando>`
- Un comando de cualquier tipo podría necesitar **permisos de administrador**. En ese caso tendremos que usar el comando `sudo` de esta manera: `sudo <comando>`.
- Para construir ficheros que almacenen un conjunto de ficheros usaremos un comando llamado `tar`. Podremos comprimir un fichero usando compresores como `gzip` o `bzip`

**** El comando `tar` acepta una serie de opciones por medio de un guión.**

*** Por ejemplo podemos usar `tar -cf copiasseguridad.tar .gnupg` * Para extraer el contenido de un fichero se usa `tar -xf copiasseguridad.tar`**

**** El comando `gzip` o `bzip2` permiten comprimir un fichero.**

1.32 Anexo: Guest additions

Las «Guest Additions» permiten que VirtualBox pueda «conectar directorios» entre el sistema operativo anfitrión y el invitado. Por ello, a menudo resulta útil tenerlas instaladas.

Si bien en los sistemas operativos invitado con entorno gráfico la instalación es muy sencilla en los sistemas basados en comandos (como Ubuntu Server) el proceso puede resultar un poco más largo.

En primer lugar las Guest Additions instalan módulos en el núcleo y es posible que para ello requiera de ciertos paquetes. Tendremos que instalar estos paquetes con «`sudo apt-get install dkms build-essential`»

- En el menú «Dispositivos» de VirtualBox elegir «Insertar CD de VirtualBox Guest Additions».
- Enganchamos el dispositivo `/dev/cdrom` a algún punto del sistema, por ejemplo dentro de `/media`.
- Usando el comando `sudo mkdir /media/cdrom` podremos crear un directorio «cdrom» dentro de «media».
- Una vez creado el directorio usamos el comando `sudo mount /dev/cdrom /media/cdrom`.
- Nos vamos al directorio con `cd /media/cdrom` y ejecutamos `ls`.
- Deberíamos ver un fichero llamado `VBoxLinuxAdditions.run`. Lo ejecutamos como administrador con `sudo ./VBoxLinuxAdditions.run`
- Existe un comando para apagar/reiniciar el sistema y es `sudo shutdown -r now` (-r para reiniciar o -h para parar).
- Una vez ejecutado, debemos ir al menú de VirtualBox y elegir una carpeta del sistema anfitrión para «conectarla» con el sistema operativo invitado.
- Reiniciamos la máquina y ejecutamos el comando `mount`.
- Puede ser necesario añadir nuestro usuario al grupo «vboxsf» que es el grupo con el que se «monta» el directorio compartido. Para hacer esto usaremos el comando `sudo usermod -a -G vboxsf pepito`
- Si en el momento de la instalación no indicamos correctamente nuestro país es posible que la hora del sistema no sea la hora local. Podemos ajustar la hora usando el comando `sudo timedatectl set-timezone Europe/Madrid`.

Instalación y configuración de cortafuegos

2.1 Definición de cortafuegos

Un cortafuegos o «firewall» es un programa que examina los paquetes de red entrantes y salientes y decide autorizar o no el paso en función de las reglas que dictamine el administrador.

A pesar de ser un software existen fabricantes de hardware que venden «dispositivos cortafuegos». Estos dispositivos suelen ser routers que incluyen de serie el software cortafuegos.

2.2 Recordatorio de los conceptos básicos de redes

- Dirección IP: todo paquete de datos lleva una IP de origen y una de destino.
- Puerto: número asociado a un cierto programa o servicio. Todo paquete lleva un puerto de origen y un puerto de destino.

Cuando un paquete llega a una máquina es posible que haya que tomar una decisión como:

- Permitir el paso del paquete.
- Denegar el paso.
- Redirigirlo a otra máquina

Para ello podemos usar cualquier parámetro de TCP/IP, como por ejemplo.

- Autorizar o no dependiendo de la IP de origen.
- Autorizar o no dependiendo de la IP de destino.
- Basarnos en el puerto de destino. Dado que los puertos de origen suelen ser aleatorios **es muy poco probable que nos basemos en el puerto de origen**. Debe recordarse también que hay que indicar claramente si el puerto será **TCP o UDP**.
- También es posible que usemos otros protocolos para tomar la decisión, por ejemplo <prohibir todos los paquetes ICMP>

- Aunque es poco probable es posible que haya que examinar algún flag del protocolo (RST, SYN, ACK...)

2.3 Utilización de cortafuegos.

Un cortafuegos es un programa que se ejecuta en los niveles más básicos del sistema operativo. Para utilizar esta capacidad es posible que tengamos dos grandes tipos de interfaz.

- Un interfaz gráfico (estilo Windows): en este curso analizaremos como usar el cortafuegos de Windows 2016.
- Un interfaz basado en comandos (estilo Unix): en este curso veremos como funciona `nftables`

2.4 Filtrado de paquetes de datos.

Una de las utilidades principales de un cortafuegos es denegar la entrada o salida de ciertos tipos de tráfico basándonos en distintos parámetros que ya se han mencionado antes:

- La IP de origen y/o destino.
- El puerto de origen y/o destino.
- El protocolo.
- Otros parámetros...

En los apartados posteriores veremos como usar un cortafuegos de red para permitir o denegar tráfico basándonos en estos datos.

2.5 Tipos de cortafuegos. Características. Funciones principales.

Hay muchas maneras de organizar la arquitectura del cortafuegos de una red. En este curso usaremos la más simple, reflejada en la figura siguiente. En ella, un dispositivo controla todo el tráfico de entrada y salida de una red. Por desgracia tiene el inconveniente de que si un intruso logra «saltarse» el cortafuegos tendrá acceso total a toda la red.

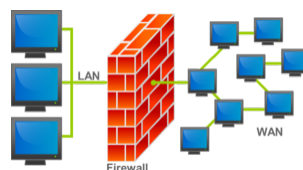


Figura 1: Estructura de un cortafuegos (Fuente:Wikipedia)

El mecanismo más utilizado hoy en día es la creación de una «zona desmilitarizada» o DMZ y requiere varios dispositivos. También se conoce por el nombre de «screened host».

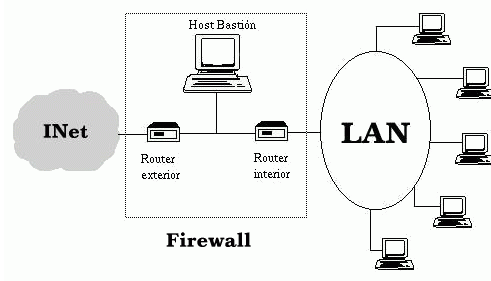


Figura 2: Arquitectura de una DMZ (Fuente: RedIris).

2.6 Instalación de cortafuegos. Ubicación.

En GNU/Linux podemos instalar **NFTables**, una arquitectura de filtrado de paquetes que se integra con el núcleo de Linux y reemplaza al anterior **IPTables**. Para instalarlo podemos hacer lo siguiente:

- Ejecutar `sudo apt-get update` y `sudo apt-get upgrade` para asegurarnos de recargar la información de interés sobre paquetes.
- Ejecutar `sudo apt-get remove iptables` para asegurarnos de que no tengamos varios sistemas de filtrado ejecutándose a la vez.
- Ejecutar `sudo apt-get install nftables` para instalar **NFTables**.

Una vez hecho esto deberíamos tener disponible el comando `nft` que permite el acceso al sistema de filtrado.

Advertencia: El comando `nft` debe ejecutarse **SIEMPRE** como administrador. Si olvidamos poner `sudo nft` ... es probable que el comando *no devuelva ningún error* ya que es posible permitir a otros usuarios ejecutar el comando. Así, aunque no se diga expresamente **DEBEREMOS EJECUTAR SIEMPRE ESTE COMANDO COMO SUPERUSUARIOS**. Otra alternativa es convertirnos primero en superusuarios con `sudo su` y a partir de ahí lanzar todos los comandos que necesitemos.

En este tema usaremos unas de las configuraciones más habituales: situar el cortafuegos entre nuestra red y el resto de Internet, haciendo que nuestro equipo no solo haga NAT sino que también permita filtrar el tráfico entre ambas redes.

2.7 Reglas de filtrado de cortafuegos.

2.7.1 Comandos y fichero de configuración

La opción más cómoda para configurar el cortafuegos suele ser la edición del fichero `/etc/nftables.conf`. Sin embargo, se puede configurar cualquier cosa tanto desde dicho fichero como desde la línea de comandos.

2.7.2 El cortafuegos NFTables

NFTables es la nueva arquitectura de filtrado y manipulación de paquetes de red en el núcleo de Linux. Como NFTables puede procesar muchos datos de red es importante saber que distingue entre varias «familias de protocolos de red». En concreto podemos usar estas familias:

- «ip»: se refiere a IPv4
- «ip6»: para IPv6
- «inet»: para tratar tanto con IPv4 como IPv6.
- «arp»: para tramas Ethernet. No lo usaremos en este tema.
- «bridge»: para tramas Ethernet que «cruzan» este equipo cuando esté siendo usado como switch. No lo usaremos en este tema.
- «netdev»: en general solo para programadores que deseen examinar todo el tráfico que entre en la tarjeta.

El comando básico de acceso a todas las operaciones es `nft`. En líneas generales `nft` examina los paquetes y analiza las reglas que se le dan para decidir que hacer con un paquete de red. Para ello es importante tener claro que `nft` usa «hooks», «tablas», «cadenas» y «reglas».

Un «hook» es una etapa en la que está un paquete. En el dibujo siguiente se puede ver las etapas en las que está un paquete IP

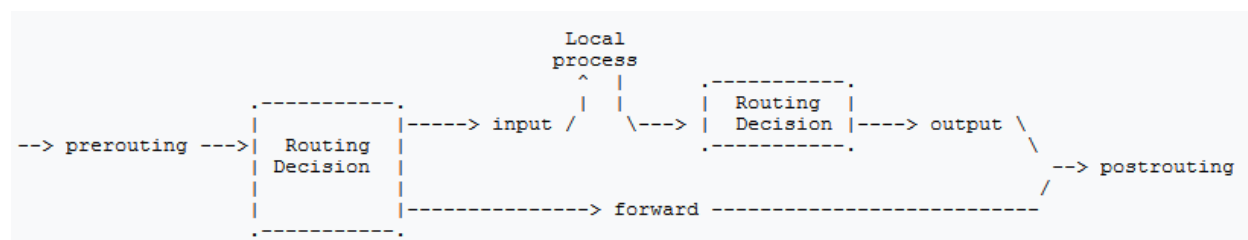


Figura 3: «Hooks» o etapas de un paquete de red. (Figura sacada de la web de NFTables)

- «Prerouting»: el paquete aún no ha entrado en la tabla de enrutamiento.
- «Input»: el paquete era para nosotros pero aún no ha entrado en ningún programa.
- «Output»: un paquete sale de un programa nuestro pero aún no se ha decidido qué hacer con él.
- «Postrouting»: un paquete que sale de un programa nuestro ya ha cruzado la tabla de enrutamiento.
- «Forwarding»: se usa para paquetes que «cruzan» nuestra máquina pero no van dirigidas a «esta máquina».

Es decir, el tráfico que «entre» verá los hooks «prerouting» e «input». El que sale de nuestro ordenador verá «output» y «postrouting». El tráfico que atraviesa nuestro equipo Linux cuando esté funcionando como router verá «forwarding».

- Una «tabla» indica el protocolo que queremos analizar, puede ser «ip», «ip6», «arp», «bridge» y otros. Así, tenemos comandos como `nft list tables` o `nft list tables ip` que nos permiten examinar qué hemos hecho con los distintos protocolos. Por ejemplo, podemos crear una tabla con el comando `nft add table ip TablaFiltradoSQL` y la podemos borrar con `sudo nft delete table ip TablaFiltradoSQL`. Es decir la pauta es `sudo nft (add|remove) table <familia> <NombreDeLaTabla>`.
- Una «cadena» indica un conjunto de reglas que `nft` irá examinando por orden para decidir qué hacer con un paquete. Una cadena puede ser «base» o «no base». Una cadena «base» puede ver TODO el tráfico TCP y una «no base» al principio no ve nada. En una cadena hay que indicar:
 - Qué tipo de manipulación queremos aplicar en un protocolo. Las posibles manipulaciones son «filter», «route» y «nat».

- En una cadena también hay que indicar la etapa o hook.
- Se debe indicar la prioridad, que es un número que determina el orden de las reglas. Una regla con la prioridad 20 se examina antes que una con prioridad 30.
- Se debe indicar la política que será una de dos: `accept` o `drop`. Si no se pone nada se asume que la política es «accept».
- Una «regla» siempre va metida dentro de una cadena. Toda regla tiene:
 - Un identificador o «handle», que podríamos también llamar el «código de regla».
 - Una posición dentro de la cadena. Si por ejemplo creamos primero la regla 10, y luego la 20, podemos después insertar una regla entre medias con la posición 15. De hecho es aconsejable no usar números de regla consecutivos.
 - Una regla PUEDE llevar un «match» que permite crear «condiciones» para saber si una regla se aplica o no. Por ejemplo una regla se podría aplicar solo a paquetes que superen una cierta longitud.
 - Si ponemos un «match» entonces DEBE haber una sentencia, que indique lo que se tiene que hacer en ese caso.

Una vez configurado todo podemos hacer lo siguiente:

- `sudo nft list ruleset` muestra **toda la configuración del cortafuegos**
- `sudo nft list ruleset > ficheronftables.conf`
- Si el fichero anterior lo ponemos «encima» del fichero `/etc/nftables.conf` el sistema operativo recargará esta configuración. El comando sería algo como `sudo cp ficheronftables.conf /etc/nftables.conf`

2.7.3 Gestión de tablas

En los puntos siguientes vemos algunas operaciones básicas con tablas:

- Crear una tabla que trabaje con el protocolo IPv4: `nft add table ip filtradoBD`
- Crear una tabla que trabaje con IP en general (ya sea version 4 o 6): `nft add table inet filtradoWeb`
- Ver las tablas del cortafuegos: `nft list tables`
- Borrar una tabla: `sudo nft delete table ip filtradoBD` o `sudo nft delete table inet filtradoWeb`

Peligro: Se puede borrar **absolutamente todas las tablas** usando el comando `nft flush ruleset`.

2.7.4 Gestión de cadenas

Supongamos una tabla cualquiera de tipo «ip» y llamada por ejemplo «filtradoUsuarios». Podemos trabajar con ella con comandos como los siguientes:

- Creamos una cadena para examinar por ejemplo el tráfico de entrada, pero refiriéndonos «**tráfico cuyo destino sera algún servidor que esté instalado en el mismo ordenador del cortafuegos**». Dicha cadena se llamara «tráficoEntrada»: `sudo nft add chain ip filtradoUsuarios traficoEntrada {type filter hook input priority 0\; }`

2.7.5 Gestión de reglas

Supongamos una tabla cualquiera de tipo «ip» y llamada por ejemplo «filtradoUsuarios» y supongamos que dentro hay una cadena llamada «traficoEntrada» y que dicha regla examina el tráfico de entrada.

- Podemos por ejemplo rechazar que alguien se conecte desde fuera poniendo una regla que prohíba el tráfico de entrada que intente conectarse a nuestro puerto 22 (el de SSH). Recordemos que dichas personas usarán como puerto de destino el 22.
- Podemos borrar toda la cadena con `nft flush chain ip filtradoUsuarios traficoEntrada`
- Podemos borrar una regla si conocemos su handle. Para averiguarlo podemos listar la tabla con `nft list table ip filtradoUsuarios`

En general, en todos los casos necesitamos un mecanismo para **determinar qué campos de un paquete queremos examinar, los «matches»**. En NFTables, un match puede ser algo como esto:

- Comprobar la IP de origen; `sudo nft add rule ip filtradoUsuarios traficoEntrada ip saddr 192.168.47.5 drop`. Esta regla **descarta** (drop) paquetes cuya IP de origen (Source ADDRESS) sea 192.168.47.5.
- Comprobar la IP de destino; `sudo nft add rule ip filtradoUsuarios traficoEntrada ip daddr 10.45.10.10 drop`. Esta regla **descarta** (drop) paquetes cuya IP de destino (Destination ADDRESS) sea 10.45.10.10.
- Se pueden usar rangos de IP: `sudo nft add rule ip filtradoUsuarios traficoEntrada ip daddr 10.45.10.0/24 drop`. Esta regla **descarta** (drop) paquetes cuya IP de destino (Destination ADDRESS) sea del tipo 10.45.10.xxx.
- Comprobar ambos campos a la vez: `sudo nft add rule ip filtradoUsuarios traficoEntrada ip saddr 192.168.47.5 daddr 10.45.10.10 drop`. Esta regla descarta los paquetes cuya IP de origen sea 192.168.47.5 y vayan destinados a la IP 10.45.10.10.
- Comprobar el puerto de origen: `sudo nft add rule ip filtradoUsuarios traficoEntrada tcp dport 80 drop`. Esta regla descarta **TODO EL TRÁFICO** cuyo puerto de destino sea el 80.
- Mezclar campos: `sudo nft add rule ip filtradoUsuarios traficoEntrada tcp dport 80 drop ip saddr 192.168.47.5 tcp dport 443`. Esta regla **descarta todo el tráfico cuyo IP de origen sea 192.168.47.5 y cuyo puerto de destino sea el 443**
- Usar rangos de puertos incluso mezclando con rangos de IP: `sudo nft add rule ip filtradoUsuarios traficoEntrada tcp dport 80 drop ip saddr 192.168.1.0/24 tcp dport 1-1024`. Esta regla **descarta todo el tráfico cuyo IP de origen sea 192.168.1.xxx y cuyo puerto de destino esté entre 1 y 1024**
- Usar cantidad de tráfico: examinemos las siguientes reglas (nota, es importante que las reglas que limitan la cantidad de tráfico vayan en la cadena correcta, en concreto en prerouting:
 - `sudo nft add rule ip filtradoUsuarios traficoEntrada ip saddr 192.168.1.45 limit rate 100kbytes/second accept` : se autoriza el tráfico desde la IP que se mantenga en un ratio de hasta 100kbytes por segundos.
 - La regla de arriba no basta para limitar el tráfico, pero si ahora añadimos esto `sudo nft add rule ip filtradoUsuarios traficoEntrada ip saddr 192.168.1.45 limit rate over 100kbytes/second drop` ahora tenemos una segunda regla que indica que **si se excede el límite de 100kbytes/seg entonces el tráfico se descarta**. Esto constituye un mecanismo excelente para «regular el tráfico».
- Podemos «abrir puertos» en el cortafuegos con NAT usando reglas como esta en la tabla que haga NAT: `sudo nft add rule ip tablaNAT natEntrada tcp dport 80 dnat to 192.168.100.10:80` que significa algo como «cuando llegue una conexión al puerto 80 de la ip de este cortafuegos redirigir la conexión hacia el puerto 80 de la IP 192.168.100.10»

- Podemos poner límites o «cuotas». Para poner un límite y por ejemplo no aceptar más de 100MBytes descargados haremos algo como `tcp sport {80,443} quota until 100 mbytes accept`. Es importante recordar que el orden de esta regla puede ser muy importante.

2.7.6 Acciones sobre paquetes

Hemos visto hasta ahora la acción drop (descartar), pero se pueden usar también:

- **counter**: nos permite hacer un recuento de bytes/paquetes que cumplen una cierta regla (y por ejemplo «llevar la contabilidad de descargas».
- **accept** : si una cadena utiliza por defecto la acción drop es posible que nos interese permitir algunos paquetes.

2.8 Pruebas de funcionamiento. Sondeo.

Para comprobar el funcionamiento de un cortafuegos se pueden usar varias técnicas:

- Usar una herramienta genérica de gestión de redes, como `netcat`.
- Usar una herramienta específica de comprobación de puertos como `nmap`
- Usar los ficheros de log para registrar la actividad de la red.

2.9 Registros de sucesos de un cortafuegos.

Se puede registrar el tráfico que entra en un cortafuegos como `nft` . Para ello, se debe usar la acción `log` por ejemplo mediante reglas como las siguientes:

- `sudo nft add rule ip tablaFiltrado cadenaEntrada log`. Esto registra *absolutamente todo el tráfico* que circule por esa cadena. Aunque puede ser de mucha utilidad, esto puede ser excesivo.
- `sudo nft add rule ip tablaFiltrado cadenaEntrada tcp dport 80 log` . Esto registra solo el tráfico de entrada HTTP.

Los registros del cortafuegos van al fichero `/etc/syslog`

2.10 Cortafuegos integrados en los sistemas operativos.

Windows incluye un cortafuegos como parte integral de su sistema operativo pero debe recordarse que **el cortafuegos de Windows es solo y exclusivamente de host**. Recordemos que eso significa que puede regular el tráfico que sale de él y el tráfico que llega a él pero **no puede analizar el tráfico que pasa a través de él**. Y esto incluye el cortafuegos de Windows Server.

El cortafuegos de Windows tiene tres «modos de funcionamiento»

- Perfil dominio: es el conjunto de reglas a aplicar cuando Windows 7 está unido a un dominio.
- Perfil público: es el conjunto de reglas a aplicar cuando Windows 7 está unido a una red que se ha marcado como «pública», es decir «no confiable». El perfil público trae por defecto reglas bastante restrictivas y lo más probable es que marquemos como públicas redes Wi-Fi de cafeterías, bibliotecas...
- Perfil privado: es el conjunto de reglas a aplicar cuando Windows 7 está unido a una red que se ha marcado como «pública», es decir «confiable». Este perfil tiene reglas menos restrictivas.

Este cortafuegos permite crear reglas a medida sobre el tipo de tráfico que entra, que sale, que use un cierto protocolo o direccion o direcciones IP y la posibilidad de aplicar la regla a uno o varios perfiles. Sin embargo, recordemos que no podrá tener reglas aplicadas al tráfico que pasa a través de él, así que si ponemos un Windows Server con dos tarjetas de red NO TENDREMOS NADA QUE HACER.

2.11 Cortafuegos libres y propietarios.

Existen otros muchos sistemas de cortafuegos:

- El sistema iptables para Linux: **sigue siendo muy utilizado**
- Packet Filter para sistemas OpenBSD.
- Cortafuegos de host integrados en antivirus...

2.12 Distribuciones libres para implementar cortafuegos en máquinas dedicadas.

Probablemente las más usadas sean Ubuntu y Debian, siendo la posibilidad mas potente el sistema nftables que hemos visto en este tema.

2.13 Cortafuegos hardware.

2.14 Anexo: configuración de IP en Linux con Netplan

La herramienta netplan utiliza ficheros YAML para configurar la IP en Linux. En distribuciones Linux orientadas a servidores es la herramienta que se usará en el futuro para configurar todos los parámetros de red. La estructura de estos ficheros permite indicar parámetros y subparámetros de configuración usando 4 espacios. Así, el fichero típico de netplan es como sigue:

```
network:
  version: 2 #Version de YAML que se usan
  ethernets: #Configuración de tarjetas Ethernet
    enp0s3: #Nombre de la tarjeta a configurar
      #Se pueden poner muchas direcciones
      #usando corchetes y separando por comas
      addresses: [192.168.100/24]
      #Dirección del router que nos permitirá
      #salir al exterior
      gateway4: 192.168.100.1
      nameservers:
        addresses: [10.15.0.220, 8.8.8.8]
```

2.15 Anexo: ejercicio de configuración

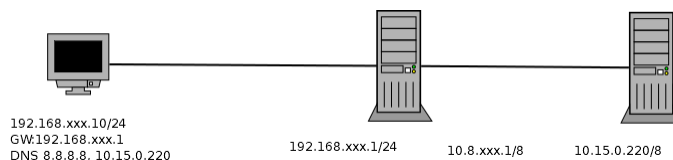


Figura 4: Ejemplos de configuración de una red.

Se supone que tenemos dos máquinas

- La máquina de la izquierda es un cliente. Todo su tráfico pasa por la máquina central que no solo hace NAT sino que también regula el tráfico de entrada y salida por medio de un cortafuegos. La máquina de la izquierda puede ser de cualquier tipo, en nuestro caso usaremos una máquina con Windows 7.
- La máquina central será una Ubuntu Server con dos tarjetas de red configuradas mediante `netplan`.

En la Ubuntu Server habrá que hacer algunas operaciones:

- Se debe activar el enrutamiento. Esto puede hacerse con el comando `echo 1 > /proc/sys/net/ip_forward` pero se desactivará en el siguiente reinicio. Por eso es mejor dejar activado el enrutamiento en el arranque editando el fichero `/etc/sysctl.conf`. Se debe escribir una línea en el fichero que ponga «`net.ipv4.ip_forward=1`» (es muy posible que ya exista la línea y solo haya que quitar el comentario inicial). Una vez modificado se puede ejecutar `sysctl -p` para iniciar el servicio (que quedará activado para siempre).
- Se debe instalar en Ubuntu el servidor web con `sudo apt-get install apache2`
- Se debe poner en marcha el NAT. Para ello podemos añadir algo como lo siguiente al fichero `/etc/nftables.conf`. Por favor, recuerda cambiar el nombre de la interfaz de salida, es posible que en tu máquina no se llame `enp0s8`

```
table ip nat {
    chain prerouting {
        type nat hook prerouting priority 0; policy accept;
    }
    chain postrouting {
        type nat hook postrouting priority 100; policy accept;
        oifname "enp0s8" masquerade
    }
}
```

2.15.1 Ejemplo: denegación del servidor web al exterior

Se nos pide lo siguiente: «*permitir que los clientes de dentro de la red SÍ puedan ver el servidor web instalado en el propio servidor donde está el cortafuegos*»

Advertencia: En los párrafos siguientes se muestra una decisión que «técnicamente funciona» pero que puede ser una mala decisión por motivos que se comentarán después.

Supongamos que creamos una tabla donde meteremos todas las decisiones que afectan al servidor web. Usaremos `nft add table ip filtradoweb`.

Supongamos también que creamos una cadena que se ocupará de denegar el tráfico web en la etapa de «prerouting» con `nft add chain ip filtradoweb denegacionApache {type filter hook prerouting priority 0 \; policy accept \;}`

Comentario: antes hemos dicho que hay malas decisiones.

- Por ejemplo, la política por defecto es «accept». Quizá fuera más seguro poner por defecto «drop» y luego permitir solo lo que nos interese.
- Examinamos el tráfico en la etapa de «prerouting» pero eso obliga al sistema operativo a comprobar muchísimos paquetes, algunos de los cuales no irán dirigidos a nosotros. Otra posibilidad sería vincular nuestra cadena a la etapa «input»

2.16 Anexo: resolución de problemas

Si algo va mal comprueba lo siguiente:

- Lanza el comando `nft` como superusuario, es decir en realidad debes hacer `sudo nft`.
- Si has aplicado una regla y parece que no funciona, asegúrate de que la pones en el «hook» correcto.
- Si no puedes enrutar, asegúrate de que has activado el enrutado o «forwarding» en el núcleo. Repasa el fichero y ábrelo con `sudo nano /etc/sysctl.conf`. La línea que controla el forwarding debe ser así `net.ipv4.forward=1`.

2.17 Un ejercicio comentado

Supongamos que tenemos la misma configuración de siempre, que volvemos a mostrar en la figura siguiente:

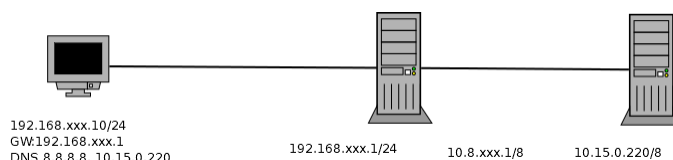


Figura 5: Ejemplos de configuración de una red.

Y supongamos que nos piden lo siguiente:

Asumiendo que los ordenadores de la izquierda son una «red interna» y que el resto son «el exterior» instalar un servidor web en el equipo del cortafuegos y conseguir que solo «el interior» pueda conectarse al servidor web

2.17.1 Solución 1 (errónea)

1. Ejecutamos `sudo nft flush ruleset` para limpiar el cortafuegos.
2. Construimos una tabla con `sudo nft add table ip filtradoWeb` y comprobamos su creación con `sudo nft list tables`
3. Comprobamos que realmente ni siquiera hay cadenas en la tabla creada con `sudo nft list table ip filtradoWeb`

Peligro: A partir de aquí lo vamos a hacer mal simplemente para hacer la prueba

4. Construimos una cadena que examine el tráfico en prerouting con `sudo nft add chain ip filtradoWeb filtradoApache { type filter hook prerouting priority 0\; policy accept\; }`
5. Comprobamos que la cadena se ha creado con `sudo nft list table ip filtradoWeb`
6. Prohibimos el tráfico cuyo puerto de destino sea el 80 con `sudo nft add rule ip filtradoWeb filtradoApache tcp dport 80 drop`

Por desgracia aquí hay algunos problemas:

- En primer lugar, esta configuración **NO FUNCIONA**. Se ha prohibido *todo el tráfico web* incluyendo, sin querer, a los de la red interna.
- Además, hemos añadido comprobaciones en la etapa prerouting lo que sobrecarga mucho el cortafuegos, al obligarle a examinar **TODO EL TRÁFICO**. En realidad solo nos interesaba el tráfico de entrada, así que probablemente deberíamos haber usado la etapa input

2.17.2 Una solución un poco mejor

1. Borramos las reglas con `sudo nft flush ruleset`
2. Reconstruimos la tabla con `sudo nft add table ip filtradoWeb`
3. Reconstruimos la cadena con `sudo nft add chain ip filtradoWeb filtradoApache { type filter hook prerouting priority 0\; policy accept\; }`
4. Prohibimos el tráfico de las direcciones de la red «externa» con `sudo nft add rule ip filtradoWeb filtradoApache ip saddr 10.0.0.0/8 drop`

Esto funciona un poco mejor, pero en realidad «fuera» podría haber muchos rangos distintos, por lo que quizá hubiera que poner muchos (y quizá demasiados) rangos de IPs.

2.17.3 Una solución (un poco mejor)

1. Borramos las reglas con `sudo nft flush ruleset`
2. Reconstruimos la tabla con `sudo nft add table ip filtradoWeb`
3. Reconstruimos la cadena con `sudo nft add chain ip filtradoWeb filtradoApache { type filter hook prerouting priority 0\; policy accept\; }`
4. Ejecutamos `sudo nft add rule ip filtradoWeb filtradoApache iifname enp0s8 tcp dport 80 drop`

Esta última regla es un poco mejor, porque ahora descartamos indicando que «prohibimos el tráfico que intente entrar al puerto 80 usando como interfaz de entrada (iifname o «input interface name») la tarjeta enp0s8» (o la que sea)

2.17.4 Otra solución

1. Borramos las reglas con `sudo nft flush ruleset`
2. Vamos a crear una tabla con `sudo nft add table ip filtradoWeb`
3. Examinamos el tráfico de entrada con una cadena que examine la etapa de red input con el comando `sudo nft add chain ip filtradoWeb filtradoEntrada {type filter hook input priority 0\; policy accept\;}`
4. Cerramos el tráfico que entra por la tarjeta que nos conecta al exterior con `sudo nft add rule ip filtradoWeb filtradoEntrada iifname enp0s8 tcp dport 80 drop`

2.17.5 Otra solución (más segura)

1. Borramos las reglas con `sudo nft flush ruleset`
2. Vamos a crear una tabla con `sudo nft add table ip filtradoWeb`
3. Ahora creamos una cadena en la que la política por defecto **va a ser mucho más segura**. Usamos el comando `sudo nft add chain ip filtradoWeb prohibicionEntrada {type filter hook input priority 0\; policy drop\;}`
4. Ahora todo el tráfico web está prohibido, así que podríamos empezar a dar permiso a quien corresponda.
 - Podríamos dar permiso solo a una cierta IP con el comando `sudo nft add rule ip filtradoWeb prohibicionEntrada ip saddr 192.168.100.10 tcp dport 80 accept`
 - Podríamos dar permiso a todo el tráfico que entre por una cierta tarjeta con `sudo nft add rule ip filtradoWeb prohibicionEntrada iifname enp0s3 tcp dport 80 accept`
 - Se puede dar permiso a un rango de IPs usando una ip de red con su máscara con `sudo nft add rule ip filtradoWeb prohibicionEntrada saddr 192.168.100.0/24 tcp dport 80 accept`

2.18 Ejercicio resuelto con ficheros

Supongamos que nos han asignado una red como 172.25.xxx.xxx/16. Se han establecido los requisitos siguientes:

- Los ordenadores de la oficina (Windows 7 por ejemplo), tienen direcciones como 172.25.xxx.10 (o 172.25.xx.11 o 172.25.xxx.12, así sucesivamente)
- Queremos tener un cortafuegos que interconecta dos redes. Por la tarjeta enp0s3 tendremos una dirección como 172.25.xxx.1/16. Por la tarjeta enp0s8 tendremos una dirección como 192.168.xxx.10/16. Esa tarjeta tiene conexión con un router que nos lleva al exterior y cuya IP es 192.168.200.1
- Teniendo esos datos, es evidente que el Windows 7 llevará como gateway al 172.25.xxx.1 y como DNS pondremos (por ejemplo 8.8.8.8 y 8.8.4.4)

Dada esta configuración:

1. Prohibir al Windows 7 que navegue por la Web.
2. Permitir al Windows 7 que navegue por la Web pero limitando la velocidad a 250KBytes/segundo
3. Permitir al Windows 7 que navegue por la Web pero cuando llegue a los 10MBytes descargados, debe detenerse todo su tráfico.
4. Permitir al Windows 7 que navegue por el servidor web instalado en el Ubuntu Server pero no que navegue por ningún otro sitio.

5. Dentro de Windows 7 hay un servidor Web con XAMPP. Conseguir que dicho XAMPP sea accesible desde el exterior.

2.18.1 Resolución

Empezaremos por asignar los parámetros IP al cortafuegos. Una vez hecho eso deberíamos tener ping al interior de la oficina y al router asignado. El fichero de netplan del Ubuntu Server es algo así:

```
network:
  ethernets:
    enp0s3:
      addresses: [172.25.200.1/16]
    enp0s8:
      addresses: [192.168.200.10/16]
      gateway4: 192.168.200.200
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
  version: 2
```

Y a continuación se muestra, con las soluciones relevantes comentadas, el fichero nftables.conf:

```
#!/usr/sbin/nft -f
flush ruleset
table inet enrutadoNAT {
  chain procesado_paquetes {
    type nat hook prerouting priority 0;
    #Punto 5: permitir el acceso web
    #al XAMPP de Windows 7 desde el exterior
    #ip daddr 192.168.200.10 tcp dport 80 \
    #   dnat 172.25.200.10:80
    policy accept;
  }
  chain traduccion_nat{
    type nat hook postrouting priority 1;
    oifname "enp0s8" masquerade;
    policy accept;
  }
}
table inet filtradoEmpresaACME{
  chain filtradoAbusoWeb{
    type filter hook prerouting priority 2;
    #Punto 1: prohibir la web
    #ip saddr 172.25.200.10 tcp dport {80,443} drop;
    policy accept;

    #Punto 2: limitar la velocidad a 250KBytes/seg
    #ip daddr 172.25.200.10 tcp sport {80,443} \
    #   limit rate      250 kbytes/second accept;
    #ip daddr 172.25.200.10 tcp sport {80,443}
    #   limit rate over 250 kbytes/second drop;

    #Punto 3: limitar la cantidad de tráfico
    #ip daddr 172.25.200.10 tcp sport {80,443} \
```

(continúe en la próxima página)

(proviene de la página anterior)

```
#          quota      10 mbytes accept;
#ip daddr 172.25.200.10 tcp sport {80,443} \
#          quota over 10 mbytes drop;

#Punto 4: permitir a Windows 7 que navegue por
#          el servidor web de este Ubuntu pero
#          prohibirle que navegue por ningún
#          otro sitio
# Manera 1: ir a otra cadena con el hook "input"
#          y prohibir todo el tráfico de
#          172.25.200.10 en "postrouting"

#ip saddr 172.25.200.10 ip daddr 172.25.200.1 accept;
#ip saddr 172.25.200.10 tcp dport {80,443} drop;

#¡Cuidado!, si hubiéramos puesto
#esta regla en segundo lugar, lo habríamos
#hecho mal
#ip saddr 172.25.200.10 ip daddr 172.25.200.1 accept;
} #Fin de la cadena
} #Fin de la tabla
```

Implantación de soluciones de alta disponibilidad

3.1 Definición y objetivos.

En el tema «Pautas de seguridad informática» definíamos disponibilidad de la siguiente manera: *capacidad de respuesta a una peticiones con las mínimas pausas por causas involuntarias* .

También decíamos que se mide la disponibilidad de un SI en «nueves».

- Se dice que un SI ofrece una disponibilidad de «2 nueves», si está disponible el 99 % del tiempo.
- Se dice que un SI ofrece una disponibilidad de «3 nueves», si lo está al 99.9 %.
- Se dice que un SI ofrece una disponibilidad de «4 nueves» si lo está al 99.99 %.
- Se dice que un SI ofrece una disponibilidad de «5 nueves» si lo está al 99.999 %

Evidentemente lograr una disponibilidad del 100 % es imposible pero en este bloque analizaremos como lograr la máxima disponibilidad en un entorno informático.

3.2 Virtualización de sistemas.

Muchos sistemas operativos que tengan que trabajar como invitados pueden beneficiarse de ciertas posibilidades instalando un software que VirtualBox llama «Guest additions».

Las «Guest additions» («añadidos para el sistema operativo invitado») son un conjunto de drivers y programas que mejoran la experiencia de uso y el rendimiento de los sistemas operativos invitados. En general, es buena idea instalarlas ya que ofrecen:

- Mejor soporte para el ratón, tarjeta de vídeo y comunicación entre el anfitrión el invitado.
- Capacidad para compartir carpetas entre el anfitrión y el invitado.
- Mejor sincronización de la hora entre anfitrión e invitado.
- Posibilidad de compartir datos entre el portapapeles del anfitrión e invitado.
- Inicio de sesión automático.

3.2.1 Modos de red en VirtualBox

A la hora de virtualizar un servicio es importante elegir correctamente el modo de funcionamiento del subsistema de red, ya que cada uno de ellos tiene sus ventajas e inconvenientes. En concreto VirtualBox ofrece los siguientes modos:

- No conectado.
- NAT: Network Address Translation es el proceso por el cual una máquina intercepta las peticiones de red de otra y las efectúa en su lugar (sustituyendo la IP). Cuando llega la respuesta, la máquina interceptora modifica esa respuesta para que la IP de destino sea la de la máquina interceptada. En el caso de VirtualBox el modo NAT hace que el programa VirtualBox “intercepte” las peticiones que salen desde el SO “invitado”. Si algún ordenador de fuera desea iniciar una conexión hacia el SO invitado, VirtualBox prohibirá dicha conexión. Será necesario abrir puertos.
- Red NAT: Facilita la creación de servidores protegidos detrás de un servicio NAT. Supongamos que queremos un servidor HTTP y uno FTP. Podríamos ponerlos en dos máquinas virtuales cada una con su NAT. Pero esto implicaría «tratar a las máquinas por separado». Creando una red NAT podemos simplificar un poco la apertura de puertos trabajando con una sola red NAT. De alguna manera esto implica poder fabricar grupos de máquinas virtuales gestionados por la misma red NAT, son máquinas que compartirán ese «router firewall NAT» virtual.
- Bridge/adaptador puente: el SO invitado no tendrá ninguna restricción y se portará como uno más de la red. El SO invitado necesitará su propia IP separada y distinta del anfitrión.
- Red interna: En este modo podemos crear «redes ficticias que no se ven desde fuera del anfitrión». Consiste en crear redes con un cierto nombre y los distintos invitados que estén asociados a esa «red ficticia» podrán verse entre sí pero no podrán salir al exterior. De hecho, ni siquiera el sistema operativo anfitrión puede ver el tráfico usando un *sniffer*.
- Solo anfitrión: el invitado solo «ve» a otras máquinas en modo «host-only» que estén en el mismo anfitrión. La diferencia con el modo «red interna» es que en «host-only» se utiliza un interfaz de red *loopback* **que no es ningún interfaz físico real**. Por tanto, el anfitrión *puede capturar el tráfico de estas máquinas virtuales usando un sniffer*.
- Red genérica: solo se usará cuando virtualicemos sistemas operativos que no tengan drivers para alguna de las tarjetas

En la figura siguiente, sacada de la web de VirtualBox se ilustra «quien puede ver a quien»

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	-	-
Internal	-	-	+	-	-
Bridged	+	+	+	+	+
NAT	+	Port forward	-	+	Port forward
NATservice	+	Port forward	+	+	Port forward

Figura 1: Modos de red en VirtualBox

3.3 Posibilidades de la virtualización de sistemas.

- Posibilidad de mover entornos a distintos lugares (remotos o no)
- Facilidad de recuperación de un entorno corrupto.
- Fácil replicación de entornos.

3.4 Herramientas para la virtualización.

- VirtualBox
- VMWare

Y para gestionar la virtualización tenemos:

- Vagrant
- Docker
- Kubernetes

3.5 Configuración y utilización de maquinas virtuales.

Durante el primer curso ya se ha explicado el funcionamiento básico de este software por lo que aquí no volveremos a repetir lo ya visto.

3.6 Alta disponibilidad y virtualización.

En pocas palabras podemos reconstruir un sistema virtualizado previamente usando solo estos comandos:

- `vagrant init usuario/maquina ``` : Inicializa un directorio con la configuración de esa máquina (cuidado, en Windows hay que cambiar y en lugar de escribir cosas como ```vagrant init d:\directorio\maquina.box` usar `vagrant init d:/directorio/maquina.box`, es decir cambiar la barra por la /).
- `vagrant up` : «Levanta» la máquina, instalándola, recuperando su estado tal y como se hubiera quedado y configurándola desde cero. Por defecto, las máquinas suelen tener el usuario «vagrant» con la clave «vagrant».

Para «exportar» nuestra máquina y facilitar su gestión con Vagrant se debe:

- Instalar un sistema operativo invitado como Windows 7 o superior o alguna variante de Linux.
- Al principio como mínimo se debe tener una tarjeta en modo NAT y además se debe anotar la MAC de dicha tarjeta.
- Si estamos en Linux se deben haber instalado los elementos que permiten añadir módulos al núcleo del sistema con `sudo apt-get install linux-headers-$(uname -r) build-essential dkms`
- Se deben instalar las «Guest Additions» en el anfitrión.
- Se debe instalar OpenSSH con `sudo apt-get install openssh-server`.
- Es recomendable crear el usuario «vagrant» y ponerle la clave Vagrant. También es importante permitir que ese usuario pueda ser administrador y que además no necesite indicar su clave de administrador cada vez. Esto

puede hacerse editando los parámetros de administración con `visudo` y poniendo la línea `vagrant ALL=(ALL) NOPASSWD: ALL`. En concreto y leyendo palabra a palabra esto significa que:

- **vagrant** **ALL**=(ALL) NOPASSWD: ALL (La regla se aplica al usuario vagrant)
 - vagrant **ALL** =(ALL) NOPASSWD: ALL (La regla se aplica a todos los host)
 - vagrant ALL=(**ALL**) NOPASSWD: ALL (vagrant puede ejecutar algo como si fuese cualquier usuario)
 - vagrant ALL=(ALL) **NOPASSWD**: ALL (no se necesita indicar contraseña)
 - vagrant ALL=(ALL) NOPASSWD: **ALL** (puede ejecutar cualquier comando)
- Se debe iniciar sesión en la máquina virtual con el usuario «vagrant» y la clave «vagrant». Nos conectaremos a nuestra propia máquina con `ssh localhost` y despues nos salimos (eso permite que se cree el directorio `.ssh`). Se debe meter la clave pública de Vagrant dentro del directorio `ssh` con `cat vagrant.pub > .ssh/authorized_keys`. Las claves públicas de *Vagrant* pueden encontrarse en (<https://raw.githubusercontent.com/hashicorp/vagrant/master/keys/vagrant.pub>){}<https://raw.githubusercontent.com/hashicorp/vagrant/master/keys/vagrant.pub>]
 - Vamos a hacer que solo el propietario pueda leer ese fichero y ese directorio de claves usando `chmod 0700 .ssh`
 - Una vez hecho todo esto podemos apagar la máquina virtual, cerrar VirtualBox y abrir la línea de comandos y crear un directorio vacío. Dentro de él inicializaremos el directorio para que sea un directorio inicializado por Vagrant con el comando `vagrant init` y luego exportaremos la máquina con `vagrant package --base <nombredemaquina> --output Maquina.box`.

3.6.1 El fichero Vagrantfile

Este fichero controla como se inicializará la máquina virtual y ofrece un completo script con parámetros comentados, mencionamos algunos de los más utilizados. Como curiosidad utiliza un lenguaje de programación llamado «Ruby». Cada línea del fichero configura algo y suele indicar distintos parámetros usando las comas como separador.

Por defecto, las máquinas virtuales tienen una sola tarjeta en modo «NAT». A menudo queremos «abrir puertos» y conseguir que alguien pueda conectarse a un servicio virtualizado. Para ello podemos editar la configuración y poner algo como esto:

```
#Esto hace que la tarjeta de red del invitado esté
#en modo NAT y que use DHCP para configurarse.
#Probablemente la dirección que se nos asigne sea
#algo como 10.0.2.15
config.vm.network "private_network", type: "dhcp"
#Con esto conseguimos que cuando se conecte al 8000 del anfitrión
#en realidad se redirija la conexión al 80 del invitado
config.vm.network "forwarded_port", guest:80, host:8000
#Podemos también forzar a que el puerto se redija hacia un ip exacta
#de invitado o a una ip exacta de host
config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1", guest_
→ip:"10.0.2.15"
```

3.6.2 Operaciones con el interior de la máquina: cambiar la IP a una tarjeta pública

Es posible copiar un fichero o carpeta entera desde el anfitrión al interior de la máquina virtual usando esto

```
config.vm.provision "file", source: "C:/archivo_con_slashes.txt", destination: "/vagrant_
↪compartida"
```

Podemos aprovecharnos de esta técnica e insertar ficheros de configuración netplan dentro de la máquina virtual y así por ejemplo configurar tarjetas en modo puente con los datos IP que queramos.

Supongamos que tenemos un fichero de netplan como este. Supongamos que :

```
network:
  version: 2
  ethernets:
    #¡Cuidado! El nombre de la tarjeta IMPORTA
    enp0s8: #Nombre de la tarjeta a configurar
      addresses: [10.8.100.110/24]
      gateway4: 10.8.0.254
      nameservers:
        addresses: [10.1.0.1, 8.8.8.8]
```

Podemos configurar el Vagrantfile de esta manera

```
#Esto añade una segunda tarjeta de red, Ubuntu suele llamarla "enp0s8"
config.vm.network "public_network"
#Necesitaremos una carpeta compartida donde inyectar
#nuestro fichero de configuración de netplan
config.vm.synced_folder "H:/oscar/maquinas/compartida_vagrant", "/vagrant_data"
#Esto copiará el fichero (¡no se puede hacer directamente en el fichero /etc
#ya que esta copia la hace un usuario sin permisos)
config.vm.provision "file", source: "C:/midirectorio/minetplan.yaml", destination: "/"
↪vagrant_data/00-installer-config.yaml"
#Y esto pone el fichero de la máquina en /etc (como esto sí lo ejecuta un
#usuario con permisos sí es posible poner cosas en /etc)
config.vm.provision "shell", inline: <<-SHELL
  #Borramos el fichero viejo de netplan
  #y ponemos el que antes se inyectó en la máquina
  cp /vagrant_data/00-installer-config.yaml /etc/netplan/00-installer-config.yaml
  #Y por supuesto aplicamos los cambios
  netplan apply
SHELL
```

Cuidado: si estamos en Windows y queremos usar una opción de Vagrant llamada bridge deberemos poner en bridge el nombre de la tarjeta de red a la que queramos vincular la máquina virtual. Probablemente en Windows el nombre del «bridge» o tarjeta de red sea algo como «Conexión de área local» o «Conexión de área local 1» .

También podemos hacer que una cierta máquina instale software en el momento de ser recuperada haciendo algo como esto

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y apache2
SHELL
```

3.6.3 Operaciones con el interior de una máquina Virtual: MySQL

Supongamos que queremos tener virtualizado un servicio de bases de datos. Se asume que tenemos los ficheros SQL que reconstruyen la base de datos, por ejemplo, algo como esto:

```
#Más abajo se crea un usuario llamado "usuario"
#con la clave '1234' que tiene acceso
#a todos los objetos de esta tabla proyectos
create database proyectos;

use proyectos;

create table proveedores (
    numprov varchar(3) primary key,
    nombreprov varchar(8),
    estado tinyint,
    ciudad varchar(15)
) ;

create table partes (
    numparte varchar(3) primary key,
    nombreparte varchar(9),
    color varchar(6),
    peso tinyint,
    ciudad varchar(8)
);

create table proyectos (
    numproyecto varchar(3) primary key,
    nombreproyecto varchar(13),
    ciudad varchar(8)
);

create table suministra (
    numprov varchar(3)
        references proveedores(numprov),
    numparte varchar(3)
        references partes(numparte),
    numproyecto varchar(3)
        references proyectos(numproyecto),
    cantidad int,
    primary key (numprov,numparte, numproyecto)
);

create user 'usuario'@'%' identified by "1234";
grant all on proyectos.* to 'usuario'@'%';

insert into proveedores values ("v1", "Smith", 20, "Londres");
insert into proveedores values ("v2", "Jones", 10, "Paris");
insert into proveedores values ("v3", "Blake", 30, "Paris");
```

(continúe en la próxima página)

(proviene de la página anterior)

```

insert into proveedores values ("v4", "Clarke", 20, "Londres");
insert into proveedores values ("v5", "Adams", 30, "Atenas");

insert into partes values ("p1", "Tuerca", "Rojo", "12", "Londres");
insert into partes values ("p2", "Perno", "Verde", "17", "Paris");
insert into partes values ("p3", "Tornillo", "Azul", "17", "Roma");
insert into partes values ("p4", "Tornillo", "Rojo", "14", "Londres");
insert into partes values ("p5", "Leva", "Azul", "12", "Paris");
insert into partes values ("p6", "Engranaje", "Rojo", "19", "Londres");

insert into proyectos values ("y1", "Clasificador", "Paris");
insert into proyectos values ("y2", "Monitor", "Roma");
insert into proyectos values ("y3", "OCR", "Atenas");
insert into proyectos values ("y4", "Consola", "Atenas");
insert into proyectos values ("y5", "RAID", "Londres");
insert into proyectos values ("y6", "EDS", "Oslo");
insert into proyectos values ("y7", "Cinta", "Londres");

insert into suministra values ("v1", "p1", "y1", 200);
insert into suministra values ("v1", "p1", "y4", 700);
insert into suministra values ("v2", "p3", "y1", 400);
insert into suministra values ("v2", "p3", "y2", 200);
insert into suministra values ("v2", "p3", "y3", 300);
insert into suministra values ("v2", "p3", "y4", 500);
insert into suministra values ("v2", "p3", "y5", 600);
insert into suministra values ("v2", "p3", "y6", 400);
insert into suministra values ("v2", "p3", "y7", 600);
insert into suministra values ("v2", "p5", "y2", 100);
insert into suministra values ("v3", "p3", "y1", 200);
insert into suministra values ("v3", "p4", "y2", 500);
insert into suministra values ("v4", "p6", "y3", 300);
insert into suministra values ("v4", "p6", "y7", 300);
insert into suministra values ("v5", "p2", "y2", 200);
insert into suministra values ("v5", "p2", "y4", 100);
insert into suministra values ("v5", "p5", "y5", 500);
insert into suministra values ("v5", "p6", "y2", 200);
insert into suministra values ("v5", "p1", "y4", 100);
insert into suministra values ("v5", "p3", "y4", 200);
insert into suministra values ("v5", "p4", "y4", 800);
insert into suministra values ("v5", "p5", "y4", 400);
insert into suministra values ("v5", "p6", "y4", 500);

```

Este fichero crea una base de datos llamada proyectos y un usuario MySQL llamado usuario con la clave `1234. Desde el exterior podremos hacer consultas MySQL usando este usuario.

Para conseguirlo necesitamos un fichero `mysqld.cnf` que incluya esta línea:

```

#Esto permite que MySQL acepte
#conexiones desde cualquier punto de la red.
bind-address                = 0.0.0.0

```

Dado estos dos ficheros, podríamos crear un Vagrantfile como este:

```

Vagrant.configure("2") do |config|
  config.vm.box = "oscarmaestre/ubuntu-server20"

  #Importante, necesitamos que el 3306 en el
  #anfitrión redirija al 3306 del invitado
  config.vm.network "forwarded_port", guest: 3306, host: 3306
  #Necesitaremos compartir una máquina entre anfitrión e invitado
  config.vm.synced_folder "H:/oscar/maquinas/compartida-vagrant", "/vagrant_data"

  config.vm.provider "virtualbox" do |vb|
    #Copiamos el script que crea todo lo relacionado
    #con la base de datos al interior de
    #la máquina virtual
    config.vm.provision "file", source:"H:/oscar/maquinas/compartida-vagrant/creacion.sql",
    ↪ destination:"/vagrant_data/creacion.sql"
    config.vm.provision "file", source:"H:/oscar/maquinas/compartida-vagrant/mysql.cnf",
    ↪ destination:"/vagrant_data/mysql.cnf"
    vb.gui = true
  end

  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y mysql-server
    #Este fichero "abre" las conexiones de MySQL
    cp /vagrant_data/mysql.cnf /etc/mysql/mysql.conf.d/mysql.cnf
    #Reiniciamos el servicio para que
    #coja los cambios...
    service mysql restart
    #Y reconstruimos la base de datos
    #Ejecutamos el script de creación
    #de la base de datos y listo
    mysql -u root < /vagrant_data/creacion.sql
  SHELL
end

```

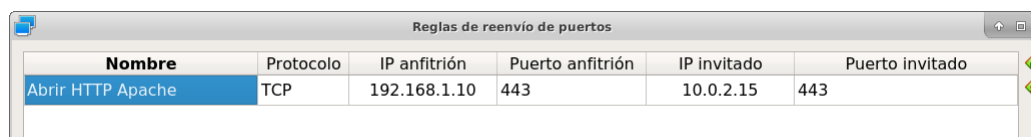
3.6.4 Problemas típicos con Vagrant

1. Cuando se cambia la memoria, se ejecutan comandos o se desea mostrar el GUI de la máquina se debe recordar activar el bloque correspondiente. Prestar atención a que unos bloques terminan con `end` pero el de comandos suele terminar con `SHELL`
2. Si usamos rutas Windows se debe cambiar el backslash (`\`) por un slash (`/`)

3.7 Simulación de servicios con virtualización.

A continuación explicamos como virtualizar un servidor web «oculto» detrás del NAT de VirtualBox.

- Una vez instalado el sistema operativo dentro de VirtualBox deberemos configurar la red de dicho sistema operativo.
- Cuando estamos dentro de VirtualBox y con la tarjeta en modo NAT, VirtualBox se convierte en «router NAT» para sus invitados y les asigna una IP como 10.0.2.15/24 con gateway 10.0.2.2. Si nuestro invitado tiene la red en modo DHCP tomará esa IP aunque si queremos podemos modificarla.
- Un sistema operativo que esté dentro de una red con NAT **no puede recibir conexiones iniciadas en el exterior** por lo que habrá que abrir puertos dentro de VirtualBox.
- Para abrir puertos deberemos tener apagado el sistema operativo invitado.
- Una vez apagado, nos vamos a la configuración de la máquina virtual y en la categoría «Red» veremos que con la tarjeta en modo NAT podemos abrir un menú «Avanzado» que ofrece un botón «Reenvío de puertos».
- Si deseamos por ejemplo tener un servidor web seguro virtualizado podemos pedirle a VirtualBox que cuando alguien se conecte a la IP del anfitrión usando el puerto seguro redirija dicha conexión al sistema operativo invitado usando datos como los siguientes:



Nombre	Protocolo	IP anfitrión	Puerto anfitrión	IP invitado	Puerto invitado
Abrir HTTP Apache	TCP	192.168.1.10	443	10.0.2.15	443

Figura 2: Apertura de puertos en VirtualBox en modo NAT

3.8 Análisis de configuraciones de alta disponibilidad

Para lograr la máxima disponibilidad podemos recurrir a distintas técnicas:

- Hardware duplicado.
- Virtualización.
- Tecnologías de contenedores.

3.8.1 Hardware duplicado

Un determinado servicio, p. ej. de bases de datos, podría estar replicado en varios equipos distintos. Diversos SGBD pueden hacer que cualquier inserción o borrado se replique automáticamente en todas las copias. Si se produce algún fallo en algún equipo, el resto de equipos pueden «repartirse» la carga extra de trabajo y conseguir así que los datos no dejen de estar disponibles en ningún momento.

Entre las ventajas podemos contar con que el rendimiento es el mejor de todas las configuraciones. Dado que los servicios se ejecutan directamente sobre el hardware tenemos casi la total garantía de que la ejecución y procesamiento de datos se harán con la máxima eficiencia, al no haber ninguna capa intermedia como las que veremos en los apartados siguientes.

El inconveniente más destacado es el coste. El hardware de servidores suele tener un coste muy alto, el cual puede multiplicarse aún más si necesitamos aumentar el número de equipos.

3.8.2 Virtualización

Programas como VirtualBox o VMWare permiten instalar un servicio dentro de un sistema operativo llamado «invitado». Esta «máquina virtual» puede copiarse y moverse con facilidad pero la tenemos en ejecución en un solo equipo. Si hay un problema de hardware podemos mover esta máquina virtual en poco tiempo y así lograr una alta disponibilidad.

La mayor ventaja es que ahorramos mucho. Podemos tener un solo servidor de gama alta ejecutando dicha máquina virtual. Si este equipo falla, podemos mover la máquina virtual a otro ordenador (aunque sea un poco menos potente) que permita cubrir las necesidades hasta que reparemos/sustituamos el otro equipo.

El inconveniente es que en realidad estamos «ejecutando un sistema operativo dentro de otro sistema operativo» con la enorme pérdida de rendimiento que esto supone

3.9 Docker

3.9.1 Contenedores

Los contenedores son un software del sistema operativo capaz de «encerrar y aislar otros programas o ficheros», consiguiendo que la ejecución de los mismos sea muy segura pero sin necesitar otro sistema operativo. Además los contenedores son programables mediante scripts lo que nos facilita mucho la tarea de desplegar servicios sin necesidad de perder rendimiento. La comparación entre arquitecturas es la siguiente (imagen tomada de la web de Docker)

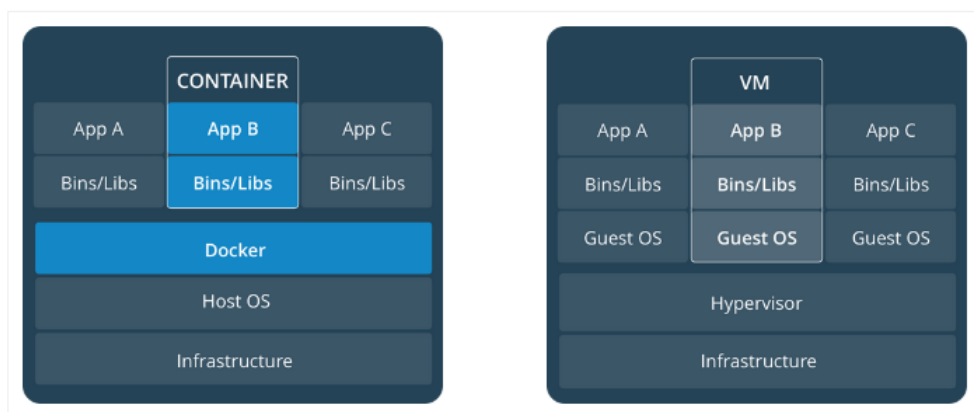


Figura 3: Comparativa entre arquitectura de virtualización y contenedores

3.10 Imágenes y procesos Docker

En primer lugar hay que distinguir entre imágenes y contenedores.

- Una «imagen» contiene lo necesario para ejecutar un programa o servicio.
- Un contenedor es una «imagen en marcha», como un proceso, y es la ejecución de una o más imágenes.

Así, si por ejemplo tenemos una imagen que contenga, por ejemplo, el servidor web Apache podríamos lanzar muchas ejecuciones de esa imagen. Una vez que descargamos una imagen, dicha imagen se queda en el catálogo de Docker. Como puede verse, el concepto de «imagen» es muy similar al de «boxes» de Vagrant.

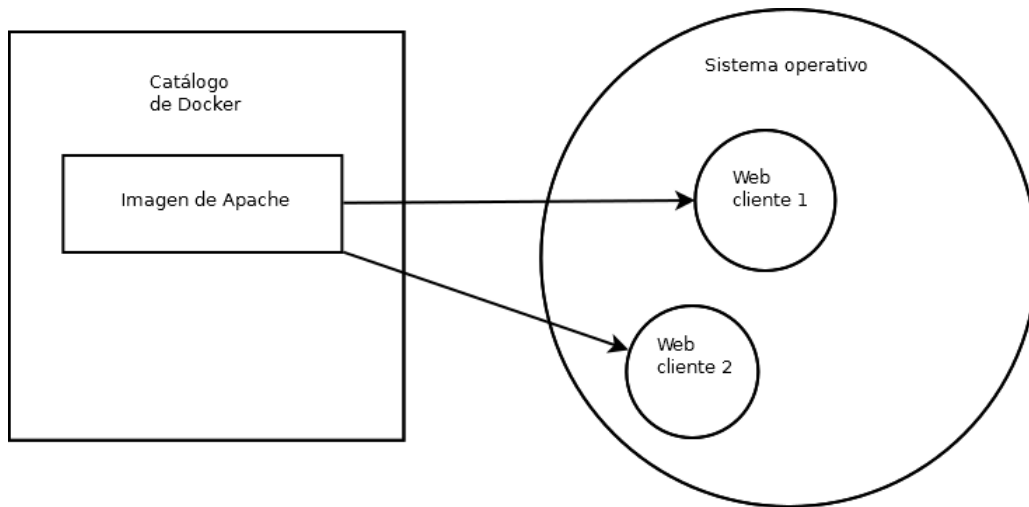


Figura 4: Imágenes y procesos Docker

3.10.1 Gestión de contenedores

- `sudo docker ps` : permite ver qué contenedores están activos.
- `sudo docker ps -a` : permite ver qué contenedores existen, estén activos o inactivos.
- `sudo docker stop <identificador|nombre>` : permite detener la ejecución de un programa en un contenedor. Se puede usar el identificador numérico asignado por Docker o el nombre que hayamos dado al contenedor.
- `sudo docker start <identificador|nombre>` : inicia un contenedor.
- `sudo docker restart <identificador|nombre>` : se asegura de detener primero el contenedor y después arranca el contenedor.
- **`sudo docker create <nombredeimagen>`**
[hace varias cosas a la vez:]
 - Descarga la imagen en caso de que no esté en el repositorio local.
 - Crea el contenedor
 - Arranca su ejecución.

3.11 Los elementos básicos de Docker

Docker permite tener por separado distintos elementos y combinarlos como queramos en un contenedor, estos elementos son:

- La consola de E/S: podemos conectar nuestra consola a la de un contenedor o no. Además podemos conectar solo la entrada, solo la salida o ambos.
- La red: podremos crear redes virtuales y enganchar el contenedor que queramos a la red que queramos.
- El almacenamiento: podemos crear discos virtuales y enganchar varios contenedores a un mismo disco o hacer que un contenedor tenga distintos discos.

3.11.1 La consola y los contenedores

Antes de examinar como funcionan las imágenes es importante comprender como funciona la E/S por consola. Nuestro sistema operativo tiene un *shell* (en Linux por defecto suele ser *bash*) pero ese *shell* **no tiene absolutamente nada que ver con lo que hay dentro del contenedor**. Si por ejemplo alguien mete un proceso que escriba simplemente «hola mundo» dentro de un contenedor y ejecutamos ese contenedor veremos la cadena, pero una vez impresa **el contenedor se detiene**.

1. Probemos a ejecutar `sudo docker run dockerinaction/hello_world`. Veremos el mensaje «hello world».
2. Si volvemos a iniciar el contenedor (`sudo docker start <id>`) veremos que **no aparece nada**. Nuestra salida (lo que vemos en pantalla) no está conectada con la salida del contenedor.
3. Para que un contenedor conecte su salida con nuestra pantalla necesitamos la opción `--attach` o `-a` de esta manera `sudo docker start -a <id>`
4. De la misma manera, si queremos que el contenedor acepte entrada desde nuestro teclado deberemos usar `--interactive` o `-i` como por ejemplo `sudo docker start -a -i <id>`

La pregunta lógica es **¿por qué docker run sí muestra cosas en la consola pero docker start no lo hace**. La respuesta es que `sudo docker run` (que sabemos que equivale a ejecutar `create+start`) vincula por defecto la entrada y salida estándar del contenedor con nuestra consola y teclado. Sin embargo, `docker start` no hace nada de eso por defecto

3.11.2 Gestión de imágenes

Algunas operaciones básicas son estas:

- `sudo docker images` : permite ver las imágenes que tenemos en nuestro repositorio local.
- `sudo docker pull <nombreimagen>` : permite descargar una imagen del registro de Docker, por ejemplo `docker pull mysql`
- `sudo docker rmi <nombreimagen>` : elimina una imagen de nuestro repositorio local.

Advertencia: No se puede borrar una imagen de nuestro registro si algún contenedor la está usando. Ni siquiera aunque el contenedor esté detenido.

3.11.3 Instalando Docker

Ubuntu tiene su propio paquete Docker que puede instalarse usando `sudo apt-get install docker.io`, sin embargo podemos instalar la versión oficial en Linux añadiendo sus repositorios a la lista de repositorios de nuestro sistema. Para ello podemos usar estos comandos.

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get -y install docker-ce docker-ce-cli containerd.io
```

Docker incluye un repositorio (que en Docker se llama registro) con imágenes de muchos servicios listos para descargar y ejecutarse simplemente usando scripts. Por ejemplo, ejecutemos un programa simple que se limita a saludar en

pantalla con `sudo docker run dockerinaction/hello_world` (Se dice que `dockerinaction` es un «espacio de nombres», en concreto es del autor de un libro llamado precisamente «Docker in action»).

El programa «se ha ejecutado dentro de un contenedor». Después ha terminado y ha salido. Como programa es bastante simple, sin embargo, podemos ejecutar un Apache dentro de un contenedor con algo como esto (cuidado, si ya se tiene instalado Apache en Ubuntu esta ejecución fallará, se debe desinstalar primero). Si ejecutamos `docker run httpd` veremos como Docker descarga e «instala una imagen de Apache».

En este último ejemplo no hemos puesto espacio de nombres, así que Docker asume que se debe buscar en los «repositorios oficiales de imágenes». Una vez ejecutado **Apache se queda en ejecución y se «apodera» de la consola**. Esto es normal, así que si queremos que el servidor Web se vaya a un segundo plano deberemos cerrar el programa (Ctrl-C) y ejecutar `sudo docker run --detach httpd` o `sudo docker run -d httpd`.

Podemos ver que Apache se está ejecutando en un contenedor con `sudo docker ps` y «apagar» el contenedor con `sudo docker stop <identificador>` o incluso «terminarlo» `sudo docker kill <identificador>` (no hace falta escribir todo el ID del container, basta con escribir las primeras letras).

También podemos reiniciar un servicio con `sudo docker restart <id_container>` e incluso ver los logs del servicio con `sudo docker logs <id_container>`.

Si queremos tener el mismo servicio para distintos clientes está claro que no podremos usar el mismo nombre, podemos lanzar un servicio con distintos nombres usando algo como `sudo docker run -d --name ApacheCliente1 httpd` lo que **crea y ejecuta un contenedor llamado ApacheCliente1**. Hay que recordar que aunque lo paremos no podremos volver a ejecutarlo con `sudo docker run -d --name ApacheCliente1 httpd` ya que eso intentaría volver a crear el contenedor (cosa imposible porque ya existe). Un contenedor puede volver a ejecutarse con `sudo docker restart ApacheCliente1`.

3.11.4 Conexiones de red en Docker

Advertencia: En clase usaremos Docker dentro de un VirtualBox, lo que nos complicará la gestión de servicios al tener que interactuar tanto con el subsistema de red de VirtualBox como con el subsistema de red de Docker.

Igual que VirtualBox, Docker tiene distintos modos de red, Docker ofrece tres «redes por defecto» con distintos comportamientos para los servicios alojados en él. En concreto existen estos tipos de redes (podemos ver los primeros con `sudo docker network ls`):

- **Bridge:** Es el modo por defecto. Cualquier imagen que se ejecute en este modo puede ver a las otras imágenes que estén en ese host físico. Las direcciones por defecto son 172.16.0.0/16. Aunque se llama «bridge» se parece al modo NAT de VirtualBox.
- **Host:** Se parecen al modo «puente» de VirtualBox. Un contenedor en modo «red host» no tiene su propio sistema de red, sino que usa el del host. **A fecha de Febrero este sistema no funciona en Docker para Windows.** Este sistema de red permite a los contenedores compartir la tarjeta de red del anfitrión. Esto significa que es necesario poner IP a los contenedores, en el caso de que la necesiten.
- **Overlay:** Está pensado para crear lo que Docker llama «enjambres», no los veremos en este tema, pero ofrecen mucha potencia al permitir crear redundancia y así tener servicios que tomen el trabajo de otros servidores caídos.
- **Macvlan:** permiten asignar una MAC distinta a nuestros contenedores y obtener acceso total a la red. Aunque puede parecer que son iguales que las redes Docker en «modo host» en el modo host no podemos cambiar la MAC (cosa que sí podemos hacer siempre en VirtualBox).
- **None:** permite deshabilitar la red de un contenedor.

3.11.5 Creando nuestra propia red en Docker

Podemos crear nuestra propia red para un grupo separado de servidores usando `sudo docker network create --driver bridge <nombredered> --subnet <IP/Mascara>`. Docker creará una red separada con el prefijo IP que hayamos indicado. Por ejemplo, tecleemos esto:

```
sudo docker network create --driver bridge red_clientes --subnet 172.30.20.0/24
```

Si deseamos trabajar con la red «host» en ese caso los contenedor **no tienen su propia IP separada**, es como si estuvieran ejecutándose en el host y entonces **usaremos la ip del host**. En este tipo de redes no se crean redes de tipo `--driver host`. Solo hay una red de tipo host y cuando creamos el contenedor podremos indicar que su red es de tipo host.

Dicho esto, supongamos que queremos crear un contenedor que ejecute Apache y que vaya conectado a la nueva red llamada «red_clientes». El comando sería este:

```
sudo docker run --network red_clientes httpd
```

Si ejecutamos este último comando veremos que Apache utiliza una IP de la red 172.30.20.0.

Cuando hayamos terminado de usar una red podemos borrarla con:

```
sudo docker network rm <nombre o id>
```

Un detalle importante es que no podemos crear dos o más redes en las que las IP se solapen.

3.11.6 Ejercicios de redes Docker

- 1) Crea una red de tipo «bridge» que use la dirección 192.168.16.0/24.
- 2) Crea una red de tipo «bridge» que use la dirección 10.0.20/24.
- 3) Crea una red de tipo «bridge» que use la dirección 10.161.0.0/16.
- 4) Crea una red de tipo «bridge» que use la dirección 172.84.128.0/18
- 5) Crea una red de tipo «bridge» que use la dirección 10.192.0.0/28
- 6) Crea una red de tipo «bridge» que use la dirección 192.168.65.128/26.

Soluciones a los ejercicios

- 1) Crea una red de tipo «bridge» que use la dirección 192.168.16.0/24:

```
sudo docker network create --driver bridge red_1 --subnet 192.168.16.0/24
sudo docker network inspect red_1
```

- 2) Crea una red de tipo «bridge» que use la dirección 10.0.20/24.:

```
sudo docker network create --driver bridge red_2 --subnet 10.0.20/24
sudo docker network inspect red_2
```

- 3) Crea una red de tipo «bridge» que use la dirección 10.161.0.0/16:

```
sudo docker network create --driver bridge red_3 --subnet 10.161.0.0/16
sudo docker network inspect red_3
```

- 4) Crea una red de tipo «bridge» que use la dirección 172.17.84.0/18:

```
sudo docker network create --driver bridge red_4 --subnet 172.84.128.0/18
sudo docker network inspect red_4
```

5) Crea una red de tipo «bridge» que use la dirección 10.192.0.0/28:

```
sudo docker network create --driver bridge red_5 --subnet 10.192.0.0/28
sudo docker network inspect red_5
```

6) Crea una red de tipo «bridge» que use la dirección 192.168.65.128/26:

```
sudo docker network create --driver bridge red_6 --subnet 192.168.65.128/26
sudo docker network inspect red_6
```

3.11.7 Almacenamiento con Docker

En Docker podemos crear almacenamiento para los contenedores usando tres posibles elementos:

- Montaje de directorios (*bind mounts* en la terminología de Docker)
- Almacenamiento en memoria.
- Volúmenes.

3.11.8 Montaje de directorios

Esto consiste simplemente en conectar un directorio del «anfitrión» con otro directorio del contenedor Docker. Los directorios que usemos son *parte de nuestro sistema operativo anfitrión* así que si algún proceso del sistema operativo los modifica sin querer, nuestro contenedor se verá afectado. Por otro lado, un proceso Docker maligno podría modificar los archivos del sistema operativo anfitrión, lo que también es un riesgo para la seguridad. Los directorios pueden crearse simplemente con `mkdir` y montarse en cualquier contenedor en ejecución.

Por ejemplo, podríamos conectar un directorio del anfitrión llamado `/home/usuario/web_cliente` con uno del invitado llamado `/usr/local/apache2/htdocs` usando un comando como este (se muestra en varias líneas):

```
sudo docker run --mount type=bind,
  src=/home/usuario/web_cliente,
  dst=/usr/local/apache2/htdocs/
  httpd
```

Al hacer esto, el servidor web tomará los ficheros del directorio del anfitrión, lo que nos permitirá modificar la web cómodamente.

3.11.9 Almacenamiento en memoria

Docker puede usar un almacenamiento de tipo `tmpfs` que aloja los archivos en memoria. Esto es especialmente rápido y sobre todo útil para almacenar secretos solo para estos dos casos de uso. Si se necesita almacenamiento lo más seguro es que se desee usar directorios o volúmenes.

3.11.10 Volúmenes

Al contrario que los directorios montados, son archivos *gestionados por Docker*. Esto los hace más seguros y más eficientes a la hora de trabajar con contenedores. Por ello, la documentación oficial de Docker recomienda trabajar con ellos. Para los volúmenes usaremos estos comandos:

- `sudo docker volume create <nombre_volumen>` para crear un volumen.
- `sudo docker volume ls` para ver los volúmenes creados.
- `sudo docker volume rm <nombre_volumen>` para borrar un volumen.
- `sudo docker volume prune` borra **todos los contenedores** que no estén conectados a un contenedor. Usar con cuidado.

3.11.11 Usando volúmenes

Podemos arrancar un contenedor cualquiera y ofrecerle espacio de almacenamiento con la opción `--volume <nombre_volumen>:/ruta`. Esto hará que el contenedor pueda acceder a `/ruta`, por ejemplo:

```
sudo docker create --volume -it mi_volumen01:/app ubuntu
```

Con esto tendremos un contenedor Ubuntu que puede guardar cosas en el directorio `/app`

3.11.12 Un ejemplo simple de Docker

Docker también se puede automatizar con fichero `Dockerfile`

```
FROM httpd
COPY index.html /usr/local/apache2/htdocs/index.html
EXPOSE 80
ENTRYPOINT ["apachectl", "start"]
```

- Construyamos una imagen con `sudo docker build . -t ImagenPropia`
- Creemos un contenedor de prueba con `sudo docker run -dti --name Servidor1 ImagenPropia /bin/bash`
- Este contenedor ahora ejecuta Apache usando como HTML el fichero que le hayamos pasado.
- Cuando queramos, podemos detener el contenedor y borrar con `sudo docker stop Servidor1`; `sudo docker rm Servidor1`

Este ejemplo tan simple reconstruye un servidor Apache con el HTML que necesitamos.

3.11.13 Un ejemplo más avanzado de Docker usando MySQL

En el ejemplo siguiente deseamos disponer de una pequeña base de datos almacenada dentro de un contenedor que ejecuta MySQL. En primer lugar, debemos saber que MySQL es una base de datos cliente/servidor y que dado que queremos ofrecer un conjunto de datos lo que haremos será usar una imagen Docker que ejecute el servidor MySQL con una base de datos como la siguiente:

- Nombre de la base de datos: `ventas`.
- Tabla:
 - Nombre: `clientes`.

- Campo dni, de tipo varchar(10) y clave primaria.
- Campo nombre, de tipo varchar(80).

En la única tabla de esta base de datos almacenaremos estos dos clientes:

- Cliente 1, dni “5111222C” y nombre “Juan Ruiz”
- Cliente 2, dni “5222333Z” y nombre “Carmen Diaz”

En primer lugar, necesitamos el SQL que meteremos dentro del servidor y que nos construye esta base de datos, llamaremos a este fichero, por ejemplo `clientes.sql`:

```
use ventas;
create table clientes (dni varchar(10), nombre varchar(40));
insert into clientes values ('5111222C', 'Juan Ruiz');
insert into clientes values ('5222333Z', 'Carmen Diaz');
```

Como vemos estos datos se meten en una base de datos llamada «ventas». Sin embargo la base de datos, el usuario y la clave los indicaremos en el momento de la creación del contenedor.

Ahora creamos un fichero `Dockerfile` donde indicamos que usaremos MySQL, indicaremos como se llamará esta base de datos e indicaremos que nuestro script SQL debe ejecutarse al comienzo.

```
FROM mysql
ENV MYSQL_DATABASE ventas
COPY clientes.sql /docker-entrypoint-initdb.d/
```

Con esto ya podemos preparar nuestra propia imagen que sirva nuestros datos. Podemos construirla con `sudo docker build -t bdempresaacme ..`

Si ahora ejecutamos `sudo docker images` podremos ver nuestra imagen. Una vez hecho esto ya podemos lanzar nuestro propio servicio de datos con `sudo docker run -e MYSQL_ROOT_PASSWORD=clave1234 -e MYSQL_USER=admin -e MYSQL_PASSWORD=1234`.

El comando anterior lanza el servidor de base de datos accesible solo en nuestro equipo (no hemos expuesto puertos ni nada por el estilo) y si queremos podemos consultar estos datos averiguando la ip de nuestro contenedor y usando un cliente como `mysql -u admin -h 172.17.0.2 -p`. Se nos preguntará la clave del usuario «admin» (hemos puesto arriba «1234») y podremos usarla.

3.11.14 Ejercicio

Pensar alguna manera de usar imágenes y contenedores para conseguir que los datos sean **persistentes**. El objetivo es que cuando se modifiquen los datos, dichos datos pueda estar disponibles para algún otro sistema MySQL.

3.12 Otros ejercicios con Docker

3.12.1 Ejercicio (I)

- Fabricar un pequeño archivo HTML en Ubuntu Server.
- Insertar ese archivo dentro del contenedor «httpd» y ponerle un nombre de imagen como «webempresa»
- Al ejecutar «`sudo docker run webempresa`» debería arrancar un contenedor que muestre la web de nuestra empresa

3.12.2 Solución al ejercicio I de Docker

- a) Fabricamos un directorio vacío
- b) Dentro de ese directorio fabricamos un html
- c) Dentro de ese directorio vacío ponemos un Dockerfile como este

Contenido:

```
FROM httpd
COPY index.html /usr/local/apache2/htdocs/index.html
```

- d) Construimos nuestra propia imagen con esto:

```
sudo docker build -t webempresa .
```

- e) Lanzamos la imagen con:

```
sudo docker run webempresa
```

- f) Se habrá abierto un servidor web que muestra una IP a la que nos podemos conectar para ver que realmente ahora la imagen sirve la web de la empresa

3.12.3 Ejercicio II de Docker

1. Crear un fichero PHP que escriba en un fichero.
2. Ponerlo en una carpeta carpeta_compartida
3. Conseguir que una imagen Docker de Ubuntu sirva ese fichero, que llamaremos «index.php» y que desde Windows 10 podamos acceder a él.
4. Si todo va bien, deberíamos poder ver como el fichero de la carpeta compartida va incluyendo más y más texto.

3.12.4 Fichero PHP

Creamos este fichero que por ejemplo pondremos en un directorio vacío llamado «ejemplo»

```
<?php
$fecha=date("l jS \of F Y h:i:s A");
$fichero = fopen("registros.txt", "a") or die("Unable to open file!");
fwrite($fichero, "Acceso registrado:".$fecha);
echo "Hemos registrado su acceso en un fichero con fecha $fecha<br>";
?>
```

La imagen httpd NO CONTIENE EL MÓDULO PHP. Necesitaremos alguna imagen que incluya PHP y Apache. Nos descargamos, por ejemplo, esta:

```
sudo docker pull php:apache
```

Ahora podremos usar esa imagen para servir ficheros PHP.

¡Cuidado! Esta imagen espera que trabajemos en el contenedor usando el directorio /var/www/html

Si queremos que esta pequeña aplicacion web trabaje en una red separada le podemos asignar una:

```
sudo docker network create --driver bridge --subnet 10.167.140.0/24 red_empresa_con_php
```

Ahora podemos arrancar todo:

-Necesitaremos `sudo docker run` -Necesitaremos vincularlo a la red `red_empresa_con_php` y usaremos `--network red_empresa_con_php` -Si queremos abrir un puerto en el anfitrión lo haremos, por ejemplo con `-p3200:80` -Uniremos nuestra carpeta `./compartida` (es del anfitrión) con la `/var/www/html` (es del invitado). Necesitaremos escribir `--mount type=bind,src=./ejemplo,dst=/var/www/html` -Por último indicaremos que queremos lanzar la imagen `php:apache`

Así, el comando entero sería:

```
sudo docker run --network red_empresa_con_php -p3200:80 --mount type=bind,src=./ejemplo,  
↪dst=/var/www/html php:apache
```

Si no funciona: Docker tiene un usuario propio. Comprueba los permisos de tu directorio y si es necesario permite que otros usuarios escriban en tu directorio:

```
chmod o+w ejemplo
```

3.12.5 Ejercicio III de Docker

El siguiente enunciado intenta unir varios de los conceptos que hemos visto en un solo ejercicio.

3.12.6 Parte 1

Crear un volumen llamado «almacenamiento_redes».

La solución sería el comando:

```
sudo docker volume create almacenamiento_redes
```

3.12.7 Parte 2

Crear un contenedor que haga las pruebas siguientes y las guarde en el directorio `/datos/resultados.txt`. El contenedor debe tener asociado el directorio `/datos` con el volumen «almacenamiento_redes». Las pruebas son:

- Escribir en el fichero la fecha.
- Escribir en el fichero si 10.14.0.254 funciona (responde a ping).
- Escribir en el fichero si 192.168.1.1 funciona (responde a ping).
- Escribir en el fichero si 8.8.8.8 funciona (responde a ping)

El comando sería este:

```
sudo docker run -it --volume almacenamiento_redes:/datos ubuntu
```

Pero claro, no hay editor, no hay ping. Tenemos que instalarlos:

```
apt-get update; apt-get install -y iputils-ping nano
```

En este contenedor crearemos un script con:

```
nano pruebas.sh
```

Y el script será:

```
#!/bin/bash
echo "Fecha: $(date)" > /datos/resultados.txt
echo "-----">> /datos/resultados.txt
ping -c 3 10.14.0.254 >> /datos/resultados.txt
ping -c 3 192.168.1.1 >> /datos/resultados.txt
ping -c 3 8.8.8.8 >> /datos/resultados.txt
```

3.12.8 Parte 3

Crear un contenedor que lea las pruebas y las muestre en pantalla. El contenedor debe leer del fichero /informes/resultados.txt. El contenedor debe tener asociado el directorio /informes con el volumen «almacenamiento_redes»

Insertaremos este script:

```
#!/bin/bash

echo "=====
echo "= Resultados de las pruebas de red="
echo "=====

cat /informes/resultados.txt

echo "=====
echo "=   Fin de los informes   ="
echo "=====
```

3.12.9 Parte 4

Automatizar todo el proceso de pruebas y muestra de informes con dos Dockerfile y un script que lance todo.

Primero vamos a crear una IMAGEN que se llamará por ejemplo pruebas_red. Nuestra imagen usará Ubuntu. Tendrá que tener dentro el paquete iputils-ping. Copiaremos en ella el script de pruebas y fabricaremos la imagen:

```
mkdir pruebas_red
nano Dockerfile
```

Dentro del Dockerfile:

```
FROM ubuntu
RUN apt-get update ; apt-get install -y iputils-ping
COPY pruebas.sh /pruebas.sh
CMD bash /pruebas.sh
```

Y construimos la imagen con:

```
sudo docker build -t pruebas_red .
```

Ahora hay que construir la segunda imagen. En esa imagen pondremos nuestro script de «informes», pero ya no necesita ping ni nano ni nada más. Creamos un directorio vacío con el nombre `informes_red`:

```
mkdir informes_red
```

Ahora metemos dentro el script de pruebas y el Dockerfile:

```
FROM ubuntu
COPY informes.sh /informes.sh
CMD bash /informes.sh
```

3.12.10 Ejercicio con Docker (IV)

3.12.11 Enunciado

Crear una infraestructura de contenedores que permita hacer pruebas desde distintas redes.

- Se necesita una red 192.168.230.0/24 con el nombre «red_c» (sin comillas) y una red 10.165.0.0/16 con el nombre «red_a» (sin comillas)
- Se necesita crear una imagen propia, basada en Ubuntu, que al ser puesta en marcha indique si está encendido el enrutamiento. Deberá guardar esta información en `/routers/informacion.txt`. Esta imagen debe llamarse «enrutador» (sin comillas)
- Se necesita crear una imagen propia, basada en Ubuntu, que al ser puesta en marcha recoja la información que hayan dejado las máquinas. Este contenedor espera encontrarlo todo en `/datos/informacion.txt`. Esta imagen debe llamarse «info_red»,
- Lanzar dos contenedores «enrutador» uno asociado a «red_c» y otro con «red_a»
- Lanzar un contenedor «info_red»

El resultado final es que el contenedor lanzado en e) debería decirnos el estado de enrutamiento de los routers.

3.12.12 Pista

Para saber si el enrutamiento está encendido podemos lanzar esto:

```
cat /etc/sysctl.conf | grep "ip_forward"
```

3.12.13 Solución

Primero creamos las redes con:

```
sudo docker network create --driver bridge --subnet 192.168.230.0/24 red_c
sudo docker network create --driver bridge --subnet 10.165.0.0/16 red_a
```

Para crear la imagen «enrutador» tendremos que hacer dos cosas, crear el script que lanzará la imagen al ser ejecutada y crear el Dockerfile asociado. Empezamos por el script que llamaremos, por ejemplo, `info_routers.sh`:

```
#!/bin/bash
cat /etc/sysctl.conf | grep "ip_forward" >> /routers/informacion.txt
```

Y ahora creamos el Dockerfile:

```
FROM ubuntu
COPY info_routers.sh /info_routers.sh
CMD bash /info_routers.sh
```

Ahora construimos nuestra imagen con:

```
sudo docker build -t enrutador .
```

Para la segunda imagen «info_red» crearemos primero un script que imprima la información en pantalla. El script podría ser algo así y llamarse, por ejemplo, `info.sh`:

```
#!/bin/bash
echo "Información sobre el estado de los routers"
cat /datos/informacion.txt
```

Ahora creamos el Dockerfile:

```
FROM ubuntu
COPY info.sh /info.sh
CMD bash /info.sh
```

Y construimos la imagen:

```
sudo docker build -t info_red .
```

El ejercicio no lo dice, pero se necesitará espacio en disco para intercomunicar las máquinas. Podemos usar un directorio compartido o un volumen. Crearemos un volumen:

```
sudo docker volume create disco_enrutadores
```

Ahora podemos lanzar nuestra imagen «enrutador» asociando a la `red_c` y luego a `red_a` haciendo que escriba en el volumen:

```
sudo docker run --volume disco_enrutadores:/routers --network red_a enrutador
sudo docker run --volume disco_enrutadores:/routers --network red_c enrutador
```

Y lanzar nuestra imagen «info_red» haciendo que «lea» del volumen apropiado y el directorio apropiado:

```
sudo docker run --volume disco_enrutadores:/datos info_red
```

3.13 Funcionamiento ininterrumpido.

3.14 Integridad de datos y recuperación de servicio.

3.15 Servidores redundantes.

3.16 Sistemas de clusters.

3.17 SAN, NAS, FiberChannel

3.18 Balanceadores de carga.

3.19 Instalación y configuración de soluciones de alta disponibilidad.

3.20 Ejercicio: recuperando una web con Vagrant

Una empresa desea poder recuperar su sitio web con rapidez, por lo que ha decidido intentar automatizar la recuperación con Vagrant. Su web tiene un solo archivo, llamado `index.html` y su contenido es el siguiente:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Empresa ACME</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Bienvenido</h1>
    <p>
      Esta es la web de la empresa ACME
    </p>
  </body>
</html>
```

En concreto se ha pensado en tener una máquina virtualizada con una tarjeta en modo NAT. Se desea que cuando alguien se conecte a la IP del anfitrión y puerto 80 se redirija la conexión al interior de la máquina virtual (también a su puerto 80) pero por supuesto se desea que se vea la web de la empresa y no el archivo `index.html` que suele mostrar Apache sobre Ubuntu.

3.21 Solución a la recuperación de la web

- Sabemos que podemos instalar Apache en la máquina virtual recuperada usando los scripts de aprovisionamiento.
- Sabemos que Apache tiene un directorio `/var/www/html`. En dicho directorio se deben poner los archivos de web.
- Sabemos que el archivo de la empresa está en `C:\Users\admin\Documents\index.html`

Teniendo eso en mente podemos hacer lo siguiente:

En primer lugar usamos `vagrant init e:/maquinas/UbuntuServerBase.box`. Esto nos creará un fichero `Vagrantfile`. Si lo editamos podremos poner en él éstas líneas (se han omitido partes no relevantes):

```
Vagrant.configure("2") do |config|
  config.vm.box = "e:/maquinas/UbuntuServerBase.box"
  config.vm.network "forwarded_port", guest: 80, host: 80
  config.vm.synced_folder "e:/directorio_auxiliar", "/var/www/html"
  config.vm.provision "shell", inline: <<-SHELL
    systemctl disable apt-daily.timer
    systemctl disable apt-daily.service
    apt-get update
    apt-get install -y apache2
SHELL
end
```

Con esto, recuperamos la máquina, instalamos Apache y sobre todo **conectamos el directorio del Apache virtualizado con un directorio del anfitrión donde están los archivos web.**

Una vez hecho esto, podemos crear un fichero `.BAT` **que copie el HTML de la web al directorio auxiliar**. Si tenemos el `Vagrantfile` y este fichero `.BAT` podremos recuperar la web con toda comodidad

```
vagrant up
copy C:\Users\admin\Documents\index.html e:/directorio_auxiliar
```

3.22 Solución al ejercicio de alojar una base de datos en Docker

En primer lugar se necesita el fichero SQL, que también mostramos aquí:

```
drop database proyectos;
create database proyectos;

use proyectos;

create table proveedores (
  numprov varchar(3) primary key,
  nombreprov varchar(8),
  estado tinyint,
  ciudad varchar(15)
) ;

create table partes (
  numparte varchar(3) primary key,
```

(continúe en la próxima página)

(proviene de la página anterior)

```

nombreparte varchar(9),
color varchar(6),
peso tinyint,
ciudad varchar(8)
);

create table proyectos (
numproyecto varchar(3) primary key,
nombreproyecto varchar(13),
ciudad varchar(8)
);

create table suministra (
numprov varchar(3)
    references proveedores(numprov),
numparte varchar(3)
    references partes(numparte),
numproyecto varchar(3)
    references proyectos(numproyecto),
cantidad int,
primary key (numprov,numparte, numproyecto)
);

insert into proveedores values ("v1", "Smith", 20, "Londres");
insert into proveedores values ("v2", "Jones", 10, "Paris");
insert into proveedores values ("v3", "Blake", 30, "Paris");
insert into proveedores values ("v4", "Clarke", 20, "Londres");
insert into proveedores values ("v5", "Adams", 30, "Atenas");

insert into partes values ("p1", "Tuerca", "Rojo", "12", "Londres");
insert into partes values ("p2", "Perno", "Verde", "17", "Paris");
insert into partes values ("p3", "Tornillo", "Azul", "17", "Roma");
insert into partes values ("p4", "Tornillo", "Rojo", "14", "Londres");
insert into partes values ("p5", "Leva", "Azul", "12", "Paris");
insert into partes values ("p6", "Engranaje", "Rojo", "19", "Londres");

insert into proyectos values ("y1", "Clasificador", "Paris");
insert into proyectos values ("y2", "Monitor", "Roma");
insert into proyectos values ("y3", "OCR", "Atenas");
insert into proyectos values ("y4", "Consola", "Atenas");
insert into proyectos values ("y5", "RAID", "Londres");
insert into proyectos values ("y6", "EDS", "Oslo");
insert into proyectos values ("y7", "Cinta", "Londres");

insert into suministra values ("v1", "p1", "y1", 200);
insert into suministra values ("v1", "p1", "y4", 700);
insert into suministra values ("v2", "p3", "y1", 400);
insert into suministra values ("v2", "p3", "y2", 200);
insert into suministra values ("v2", "p3", "y3", 300);
insert into suministra values ("v2", "p3", "y4", 500);

```

(continúe en la próxima página)

(proviene de la página anterior)

```
insert into suministra values ("v2", "p3", "y5", 600);
insert into suministra values ("v2", "p3", "y6", 400);
insert into suministra values ("v2", "p3", "y7", 600);
insert into suministra values ("v2", "p5", "y2", 100);
insert into suministra values ("v3", "p3", "y1", 200);
insert into suministra values ("v3", "p4", "y2", 500);
insert into suministra values ("v4", "p6", "y3", 300);
insert into suministra values ("v4", "p6", "y7", 300);
insert into suministra values ("v5", "p2", "y2", 200);
insert into suministra values ("v5", "p2", "y4", 100);
insert into suministra values ("v5", "p5", "y5", 500);
insert into suministra values ("v5", "p6", "y2", 200);
insert into suministra values ("v5", "p1", "y4", 100);
insert into suministra values ("v5", "p3", "y4", 200);
insert into suministra values ("v5", "p4", "y4", 800);
insert into suministra values ("v5", "p5", "y4", 400);
insert into suministra values ("v5", "p6", "y4", 500);
```

Instalación y configuración de servidores proxy

4.1 Material para la unidad

Para esta unidad vas a necesitar dos máquinas virtuales.

- Una de las máquinas virtuales ejecutará Ubuntu 20 para escritorio y tendrá una tarjeta de red en modo puente. Si quieres puedes construir una con rapidez yendo a un directorio vacío y ejecutando los comandos `vagrant init oscarmaestre/ubuntu20desktop` y después `vagrant up`. No olvides añadir una segunda tarjeta en modo puente y una carpeta compartida con el anfitrión.
- La otra máquina virtual usará Ubuntu Server (versión 22 o superior). No olvides añadir una segunda tarjeta en modo puente y una carpeta compartida con el anfitrión.

Habrás que configurar la IP en ambos casos y recuerda que es **imprescindible** que ambas máquinas puedan hacerse ping.

4.2 Tipos de proxy . Características y funciones.

Antes de explicar los tipos de proxy es importante entender el concepto: básicamente, se puede decir que un proxy es un software que actúa como intermediario entre las peticiones de un cliente y un servidor. Si examinamos la figura siguiente veremos que ahora no hay solo una petición y una respuesta, sino que hay varios pasos más.

1. Un cliente solicita, por ejemplo, una web como <http://acme.com>
2. La petición pasa primero por el proxy, que la intercepta, la analiza y puede tomar decisiones sobre si permitirle o no. Si está permitida **es el proxy quien realiza la petición en el nombre del usuario**
3. El servidor contesta a la petición y dicha respuesta llega al proxy, quien a menudo conservará la respuesta en caché por si en el futuro alguien necesita la misma página web.
4. La respuesta finalmente llega al cliente.

Una ventaja añadida de este caso del proxy es que si en el futuro otra máquina de la red necesita la misma web obtendrá la misma página pero a más velocidad, al no ser necesario hacer una conexión extra al exterior.

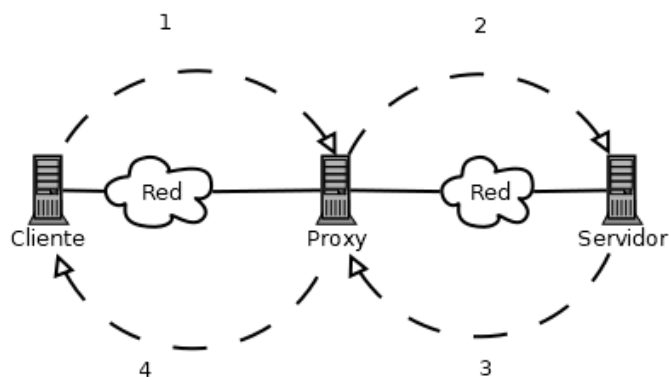


Figura 1: Comportamiento de un proxy

Una vez analizado el concepto de proxy se pueden encontrar distintos tipos de proxy

4.2.1 Tipos de proxy en función de la arquitectura de red.

Podemos distinguir entre proxy directo (o simplemente proxy), proxy abierto y proxy inverso.

El término proxy suele reservarse para esta arquitectura de red. En este caso, el proxy está dentro de nuestra red y nosotros como administradores tenemos el control y podemos tomar todas las decisiones que deseemos.

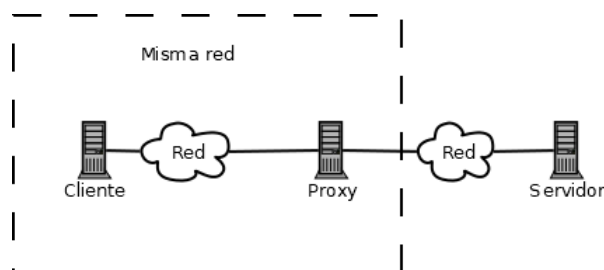


Figura 2: Proxy (en su configuración más típica)

En un proxy abierto el proxy no está bajo nuestro control, sino que está en alguna red intermedia y suele ofrecer servicios como ocultación de ip, privacidad y similares. No tenemos garantías de que realmente el proxy no registre nuestra actividad.

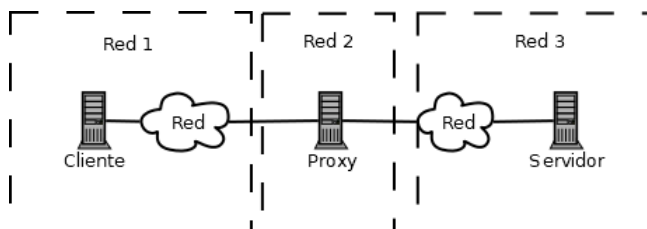


Figura 3: Proxy abierto

Un proxy inverso es aquel que se sitúa dentro de la red del servidor de manera que actúa como caché, distribuidor de carga o simplemente como respaldo.

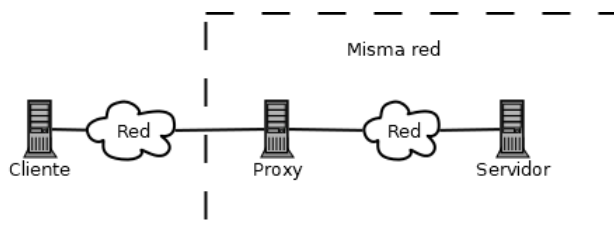


Figura 4: Proxy inverso

4.2.2 Tipos de proxy en función de la aplicación

Aunque mucho menos conocidos que los proxies web existen proxies para otras aplicaciones como FTP e IRC. En este módulo no se tratan estos programas aunque su funcionamiento en esencia es muy similar a los proxies web.

Un tipo de proxy muy habitual fuera de HTTP es el proxy SOCKS. Estos proxies *trabajan a nivel de TCP* y no en la capa de aplicación. La versión 5 de SOCKS (llamada SOCKS5) permite incluso autenticación a nivel de TCP.

ICAP es otro protocolo para proxies que permite trabajar de varias maneras:

- Modo solicitud: el proxy redirige la petición y si los filtros lo permiten redirige la petición hacia el servidor. Esto lo hace útil para **filtrar conexiones**
- Modo de respuesta: en este modo la petición se redirige y cuando llega la respuesta podemos procesarla con filtros y decidir qué hacer con el contenido que llega. Esto permite **filtrar contenidos**

En el resto del tema veremos como configurar SQUID, un proxy muy sofisticado con capacidad de procesar tráfico HTTP y FTP y capaz de actuar tanto en modo solicitud como en modo respuesta.

4.3 Instalación y configuración de clientes proxy.

Configurar un cliente para que utilice un proxy es bastante sencillo. En la imagen siguiente se muestra una captura de Firefox en el que se indica que la conexión debe hacerse a través de un proxy. Como puede apreciarse basta con rellenar la IP del servidor proxy y el puerto en el que escucha (Squid suele hacerlo en el 3128).

Sin embargo, aunque hayamos instalado Squid en la segunda máquina virtual veremos que el Firefox de la máquina cliente no funciona y muestra un mensaje como «El servidor proxy está rechazando las conexiones entrantes». Aún se tiene que configurar el servidor, cosa que haremos en los pasos siguientes.

4.4 Instalación de servidores proxy

La opción más sencilla es simplemente ejecutar `sudo apt-get install squid-openssl`. Esto descarga, instala y arranca el servicio. Observa que es importante instalar `squid-openssl` y no `squid` ya que el primero permite trabajar también con conexiones HTTPS.

En los dos puntos siguientes se ilustra, solo como curiosidad, como reconstruir Squid desde su código fuente. Si `sudo apt-get install squid-openssl` te ha funcionado no es necesario que recompiles el programa.

Configurar acceso proxy a Internet

☐ Sin proxy
☐ Autodetectar configuración del proxy para esta red
☐ Usar la configuración del proxy del sistema
☒ Configuración manual del proxy

Proxy HTTP: 192.168.1.120 Puerto: 3128
☒ Usar también este proxy para FTP y HTTPS
 Proxy HTTPS: 192.168.1.120 Puerto: 3128
 Proxy FTP: 192.168.1.120 Puerto: 3128

Host SOCKS:
☐ SOCKS v4 ☒ SOCKS v5

☐ URL de configuración automática del proxy

No usar proxy para:

Ayuda Cancelar Aceptar

Figura 5: Configuración de proxies en Firefox

4.5 Instalación de servidores proxy con apt source

Advertencia: Este paso lo ilustramos solo para mostrar al menos una vez como recompilar un programa desde cero. Salvo problemas, por favor instala Squid en clase con `sudo apt-get install squid-openssl`

En general las distribuciones de Linux no solo permiten descargar programas listos para ejecutar sino también su *código fuente*. Para ello, las herramientas como apt utilizan listas de repositorios con un aspecto como este:

```
deb http://archive.ubuntu.com/ubuntu/ jammy main restricted universe multiverse
# deb-src http://archive.ubuntu.com/ubuntu/ jammy main restricted universe multiverse

deb http://archive.ubuntu.com/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src http://archive.ubuntu.com/ubuntu/ jammy-updates main restricted universe
↪multiverse

deb http://archive.ubuntu.com/ubuntu/ jammy-security main restricted universe multiverse
# deb-src http://archive.ubuntu.com/ubuntu/ jammy-security main restricted universe
↪multiverse
```

Como vemos, este fichero está configurado solamente para descargar programas, pero si queremos poder usar código fuente de programas podemos descomentar las líneas con `deb-src` o simplemente copiar líneas reemplazando `deb` por `deb-src`. Una vez hecho podremos usar `apt-get update` y tener actualizados los repositorios de nuestras máquinas.

Compilar Squid con soporte para hacer operaciones SSL requiere también instalar algunos paquetes extra:

```
sudo apt-get install openssl devscripts
```

Una vez hecho esto podremos partir del código fuente de Squid que hay en los repositorios de Ubuntu y compilar Squid. Para ello podemos hacer esto:

```
#Descargar todo lo que sea necesario para
#compilar el paquete Squid
sudo apt-get build-dep squid
#Descargar el código fuente de Squid. NO NECESITAS SER root
apt-get source squid
#Entramos en el directorio de squid
cd squid-5.2
#Añadimos las opciones --enable-ssl-crtld y --with-openssl
nano debian/rules
#Y fabricamos los paquetes .deb
sudo debuild -b -uc -us
#Por último instalamos estos paquetes
#No nos harán falta todos los que construye
#apt-get, solo ponemos estos y en este orden
sudo dpkg -i squid-common.deb
```

4.6 Instalación de servidores proxy desde cero.

Advertencia: Este paso es todavía más extremo que el anterior. Lo recomendable en clase es instalar Squid con `sudo apt-get install squid-openssl`.

El paso anterior se puede llevar todavía más lejos y trabajar directamente con el código fuente del programa en lugar de con el código fuente que almacena Ubuntu. Squid permite descargarse el código fuente y recompilarlo usando la secuencia típica de comandos en GNU/Linux:

1. `configure`

2. `make`

3. `make install`

- Lo primero que debemos hacer es descargar el código fuente de squid con algo como `wget http://www.squid-cache.org/Versions/v5/squid-5.3.tar.gz`. Si no tenemos `wget` lo instalamos con `sudo apt-get install wget`.
- Después descomprimos el archivo con `tar -xzf squid-5.3.tar.gz`
- El programa necesita el compilador `g++` y el intérprete `perl`. Los instalamos con `sudo apt-get install g++ perl`.
- Nos metemos en el directorio de squid con `cd squid-5.3`.
- Ejecutamos `./configure`. El programa analizará nuestro sistema y preparará todo para la recompilación de Squid. Si falta algo por instalar nos lo dirá.
- Ejecutamos `make`. Esto reconstruirá el programa desde 0 y lo optimizará para la ejecución en nuestra máquina. Este proceso puede necesitar bastante tiempo.
- Lo instalamos con `sudo make install`

- Una vez hecho esto, Squid se habrá instalado en el directorio `/usr/local/squid` y allí tendremos todos los ficheros.
- Squid se ejecuta con su propio usuario. Debemos crearlo con `sudo adduser squid`.
- Squid necesitará un directorio donde dejar los logs. Lo creamos con `sudo mkdir -p /usr/local/squid/log`. La opción `-p` significa «crear con los mismos permisos del directorio padre».
- Squid necesitará un directorio donde dejar datos durante la ejecución. Lo creamos con `sudo mkdir -p /usr/local/squid/run`.
- El usuario `squid` debe ser el propietario de los ficheros y directorios donde se instaló el programa. ejecutaremos esto:
 - `sudo chown -R squid:squid /usr/local/squid/log`
 - `sudo chown -R squid:squid /usr/local/squid/var`
 - `sudo chown -R squid:squid /usr/local/squid/run`
- Para ejecutarlo podemos usar convertirnos en el usuario `squid` con su `squid` y luego ejecutar `/usr/local/squid/sbin/squid`. Podremos ver el número de proceso de Squid con `ps -e | grep squid`.

Todo este proceso ofrece más eficiencia, al adaptar el programa a la máquina donde lo vamos a ejecutar. Sin embargo, dado que compilar es un proceso lento, en sistemas tipo Debian/Ubuntu también puede usarse `sudo apt-get install squid-openssl`, que instalará el programa y todas sus dependencias. Si hay algún problema probablemente se resuelva después de actualizar los repositorios e instalar actualizaciones pendientes:

```
sudo apt-get install update
sudo apt-get install upgrade -y
sudo apt-get install squid-openssl
```

4.6.1 Ficheros de interés

- El fichero `/etc/squid/squid.conf` contiene la configuración del proxy, se hablará detenidamente de él en seguida. Este fichero acepta procesar otros ficheros de configuración que estén en el directorio `/etc/squid/conf.d`. Por tanto, **no es necesario meterlo todo siempre dentro del mismo fichero**. Esto permite trabajar más organizadamente, ya que como podrá apreciarse pronto, el fichero `squid.conf` es muy grande y localizar ciertos parámetros puede ser difícil.
- El directorio `/var/spool/squid` contiene los directorios que actuarán como caché de Squid.

4.6.2 Iniciando y parando el servicio

- Se puede arrancar Squid usando `sudo service squid start`, detenerlo con `sudo service squid stop` y hacer un reinicio del servicio con `sudo service squid restart`. Sin embargo, antes de arrancar puede ser útil ejecutar `sudo squid -k parse`, que analizará el fichero de configuración y nos dirá si hay algún fallo en alguna línea. También está la posibilidad de usar `sudo squid -k check` que nos mostrará solo los errores.

Advertencia: Squid siempre muestra mucha información durante el análisis, así que puede ser interesante ejecutar algo como `squid -k parse 2> errores.txt` para poder leer los resultados tranquilamente con algo como `nano errores.txt`. Si se prueba a introducir un error veremos como el fichero muestra no solo el error, sino también todo lo que funciona (lo que complica el localizar el error)

- Si hacemos un cambio en la configuración y deseamos que Squid tome la nueva configuración *sin reiniciar el servicio* se puede usar `sudo squid -k reconfigure`.

4.6.3 Squid y SSL/TLS

Squid es básicamente un proxy HTTP, lo que significa que en principio no puede manejar HTTPS. Sin embargo, se puede recurrir a **certificados falsos** emitidos por una **autoridad certificadora falsa** los cuales al ser instalados en los navegadores de los usuarios darán por buena una conexión HTTPS aunque sea interceptada por Squid. Para ello hay que dar varios pasos.

En primer lugar fabricaremos un directorio para poner los certificados y generaremos los datos necesarios para poder generar certificados con rapidez:

```
cd /etc/squid
#Esto genera una tabla de números primos
#que pueden ser usados a la hora de generar los
#certificados falsos
sudo openssl dhparam -outform PEM -out /etc/squid/parametros_dh.pem 2048
#Esto genera un certificado autofirmado
sudo openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout ClaveCertificado.
↪key -out Certificado.crt
```

En segundo lugar generamos una base de datos de certificados (en realidad será un directorio) con un programa llamado `/usr/lib/squid/security_file_certgen` que está incluido al instalar squid (podría estar también en `/usr/lib64`):

```
#Esto crea (-c) una base de datos (o directorio) llamada (-s) base_de_datos_ssl
#y no permitirá que la base de datos ocupe más de (-M) 50MB
#Puede que necesites usar sudo
/usr/lib/squid/security_file_certgen -c -s /etc/squid/certificados_ssl -M 50MB
#El propietario de los directorios también podría ser squid
sudo chown proxy:proxy /etc/squid/certificados_ssl
```

En tercer lugar nos vamos al fichero de configuración que estemos usando e indicamos que está permitido traer certificados añadiendo estas líneas **al principio del fichero** (si no lo hacemos así es posible que nunca se autorice el inicio de la conexión segura)

```
acl traer_certificados transaction_initiator certificate-fetching
http_access allow traer_certificados
```

En cuarto lugar nos vamos **al final del fichero de configuración** e indicamos qué programa va a gestionar los certificados y el directorio donde puede trabajar:

```
sslcrtd_program /usr/lib/squid/security_file_certgen -s /etc/squid/certificados_ssl -M
↪50MB
#Esto indica que Squid debe pasar por alto los errores de validación
#que es justo lo que queremos
sslproxy_cert_error allow all
#Y esto hace que Squid observe todos los certificados
#tanto de emisor como de cliente
ssl_bump stare all
```

Y por último buscamos en `squid.conf` la línea «`http_port 3128`» y la reemplazamos por esto (los símbolos *backslash* sirven para poder continuar en la línea siguiente):

```
http_port 3128 tcpkeepalive=60,30,3 ssl-bump \
generate-host-certificates=on \
dynamic_cert_mem_cache_size=50MB \
```

(continúe en la próxima página)

(proviene de la página anterior)

```
tls-cert=/etc/squid/Certificado.crt \  
tls-key=/etc/squid/ClaveCertificado.key
```

Si cogemos el fichero `Certificado.crt` y lo importamos en el navegador podremos navegar por Internet con normalidad, pero permitiendo que Squid observe las conexiones SSL.

4.6.4 ACLs en Squid. ACLS de origen.

Squid *no permite su uso a cualquier cliente*. Puede usar listas de control de acceso para determinar exactamente lo que se quiere hacer:

- Se puede restringir el uso a solo ciertas IPs origen.
- Se puede restringir el acceso a determinados sitios web destino.
- Se pueden combinar ambos mecanismos para permitir el acceso solo a ciertas web y solo por parte de ciertos usuarios de la empresa.

Por defecto **Squid no permite a nadie la conexión**. Así que es necesario crear una ACL donde indiquemos una lista de máquinas y después tendremos que dar permiso a esa lista de máquinas.

Como hemos dicho antes, no es necesario meter todo en el fichero `/etc/squid.conf`, así que vamos a definir nuestro propio acceso en un fichero como `/etc/squid/conf.d/accesopropio.acl`. Supongamos que la red de nuestra empresa tiene el prefijo `192.168.1.0/24`...

```
acl red_empresa src 192.168.1.0/24  
http_access allow red_empresa
```

Si ponemos esto en el fichero `/etc/squid/conf.d/accesopropio.acl` y ejecutamos `sudo squid -k parse` podremos ver si hay algún error. Si lo hay lo corregiremos y si no podremos ejecutar `sudo service squid restart` para que el proxy empiece a funcionar. Si nos vamos a la máquina cliente y probamos alguna URL veremos que ahora sí estamos navegando a través del proxy. Si se desea comprobar si realmente navegamos a través del proxy podemos detener el proxy en el servidor con `sudo service squid stop` y ver que Firefox deja de funcionar. Por supuesto, si reiniciamos el proxy Firefox volverá a poder navegar con normalidad.

4.7 Configuración de filtros.

4.7.1 ACLs en Squid. ACLS de destino.

Una vez que hemos visto como procesar las IPs de origen que pueden navegar nos interesan otros parámetros de Squid como las listas `dstdomain`. Estas listas permiten tomar decisiones sobre dominios de destino por los cuales quieren navegar los clientes (y normalmente queremos saberlo para denegarles el permiso). Supongamos que hay un dominio llamado `http://marca.com` al cual deseamos prohibir el acceso. Podemos poner un fichero como `/etc/squid/conf.d/accesopropio.acl` en el que escribamos

```
acl prohibidos dstdomain .marca.com  
http_access deny prohibidos
```

Advertencia: El orden de las ACLS es **importantísimo**. Si en el apartado anterior habíamos dado permiso a ciertos usuarios ese «permiso para salir» ya fue concedido así que intentar denegar no funcionará.

Este fichero **no deniega el acceso al periódico**

```
acl red_empresa src 192.168.1.0/24
http_access allow red_empresa
acl prohibidos dstdomain .marca.com
http_access deny prohibidos
```

Este fichero **sí deniega el acceso al periódico**

```
acl prohibidos dstdomain .marca.com
http_access deny prohibidos
acl red_empresa src 192.168.1.0/24
http_access allow red_empresa
```

En el caso de las restricciones a sitios web es frecuente que haya varios, así que un fichero podría ser algo así:

```
acl prohibidos dstdomain .marca.com .sport.es
http_access deny prohibidos
acl red_empresa src 192.168.1.0/24
http_access allow red_empresa
```

Pero es habitual tener muchos nombres de dominio. Para simplificar esto, Squid permite cargar datos desde ficheros externos usando las comillas. Por ejemplo, supongamos que queremos tener todos los dominios prohibidos en un fichero llamado por ejemplo «/etc/squid/sitios_prohibidos.txt». Podemos usar este fichero:

```
acl prohibidos dstdomain "/etc/squid/sitios_prohibidos.txt"
http_access deny prohibidos
acl red_empresa src 192.168.1.0/24
http_access allow red_empresa
```

Y por supuesto poner en el fichero /etc/sitios_prohibidos.txt una lista de dominios no permitidos, como:

```
.marca.com
.sport.es
.mundodeportivo.com
...
```

En general, se pueden usar las comillas en todas las listas de acceso. Si Squid encuentra algo entre comillas, asumirá que es la ruta de un fichero y que debe tomar todas las líneas de dicho fichero.

4.7.2 ACLs basadas en la URL

A veces no es suficiente con fabricar una lista de páginas web prohibidas porque simplemente puede haber demasiadas. En esos casos se pueden usar ACLs que examinan el nombre de dominio de la web y si dicho nombre se ajusta una regla entonces denegarlo.

Por ejemplo, supongamos que hay una serie de páginas que se desea prohibir como `http://violencia.com`, `http://violencia.net`, `http://masviolencia.com`, `http://todoviolenencia.com`, `http://muchaviolenencia.es`, etc... Como vemos, la lista de páginas podría ser enorme. En ese caso, podemos usar una regla como esta:

```
acl prohibicion_violencia url_regex violen
http_access deny prohibicion_violencia
```

Así, todas página que tengan un nombre de dominio que incluya de alguna manera la cadena «violen» serán denegadas.

4.7.3 ACLs basadas en la ruta

Si el sistema anterior no es suficiente se puede utilizar el análisis de las rutas. Por ejemplo, si examinamos una página como `http://acme.com/violencia` veremos que claramente contiene un término que deseamos prohibir. Para prohibir páginas Web en las que ocurra esto se pueden usar las ACLs basadas en ruta de esta manera:

```
acl prohibicion_ruta_violencia urlpath_regex violen
http_access deny prohibicion_ruta_violencia
```

Advertencia: Hoy en día cada vez más páginas, incluidos los buscadores usan cifrado en las conexiones. Eso significa que estos bloqueos podrían no funcionar ya que lo primero que hace el navegador es establecer una conexión cifrada (que Squid no puede descifrar) y después solicitar la página concreta

4.7.4 ACLs basadas en el tipo de archivo

En Internet hay una definición general de tipos de archivo llamada «tipos MIME» (Multimedia Internet Mail Extensions, se definieron para transportar archivos en el correo electrónico). Si lo que nos interesa en bloquear el acceso a ciertos tipos de contenido como vídeo o audio podemos bloquear usando el análisis del tipo de petición MIME al servidor de esta manera. En este caso bloqueamos los «webm» que es un tipo de vídeo muy utilizado, aunque no el único (de hecho YouTube ofrece diversos tipos). Obsérvese que usamos la «denegación de respuestas», ya que el tráfico que Squid «descifra» es el del servidor:

```
acl video_webm rep_mime_type video/webm
http_reply_access deny video_web
acl video_mp4 rep_mime_type video/mp4
http_reply_access deny video_mp4
acl video_flv rep_mime_type video/flv
http_reply_access deny video_flv
```

4.8 Métodos de autenticación en un proxy .

Aparte de usar direcciones IP podemos hacer que un proxy exija a un usuario el proporcionar un usuario y contraseña si desea navegar por Internet. Existen diversos como mecanismos como NTLM (que usar autenticación Windows) o LDAP (que usa un servidor LDAP). Dado que estamos usando Unix y que en este módulo no se menciona LDAP usaremos el mecanismo NCSA que usará un fichero de usuarios y claves externo gestionado por una herramienta externa, en concreto usaremos la herramienta `htpasswd` (si no la tenemos habrá que instalarla con `sudo apt-get install -y apache2-utils`

Una vez la tengamos instalada tenemos que decidir donde ubicar el fichero de credenciales, que por supuesto debe ser un lugar protegido de los usuarios normales. Elegiremos `/etc/squid/credenciales` y empezaremos insertando un usuario llamado «contabilidad» de esta manera: `sudo htpasswd -c /etc/squid/credenciales contabilidad`. La herramienta nos pedirá que indiquemos la clave de este usuario.

Advertencia: La opción `-c` es para *crear el fichero*, así que solo la usaremos una vez.

A continuación crearemos por ejemplo otro usuario llamado «gerencia» con `sudo htpasswd /etc/squid/credenciales gerencia`. En este punto ya tenemos un fichero con dos usuarios. Puede mostrarse el contenido de este fichero con `sudo cat /etc/squid/credenciales` y veremos que aparecen los usuarios y su clave cifrada.

Una vez hecho esto debemos configurar la autenticación de Squid usando estos parámetros:

1. «program»: se usa un módulo de Squid llamado `nlsa_auth` que debería estar dentro de `/usr/lib/squid3` probablemente con la ruta `/usr/lib/squid3/basic_nlsa_auth`.
2. «children»: indica el número de módulos hijo que deben estar listos para atender peticiones de autenticación. Este valor dependerá de cuantas personas como máximo se vayan a conectar al proxy. Para nuestro ejemplo un valor de 10 será suficiente.
3. «realm»: un texto que aparecerá a los usuarios que inicien sesión.
4. «credentialsttl»: tiempo máximo que se almacena la sesión de alguien antes de volver a pedirle la clave.

Así, si queremos crear una lista en la que estén los usuarios autenticados podríamos configurar y crear todo con algo como esto:

```
#El orden en estos ficheros es fundamental
#Primero creamos permisos básicos para usuarios
#que no tengan que proporcionar una clave de acceso
acl marca_permitido src 10.8.0.250
acl resto_personas src 10.8.0.0/24
acl periodicos dstdomain .marca.com .as.com

#Esto permite a una IP (¿del jefe?) acceder
#a periódicos
http_access allow marca_permitido periodicos
#El jefe también entra en este grupo y también
#puede ver otras cosas que no sean periódicos
http_access allow resto_personas !periodicos

#A partir de aquí creamos una configuración de seguridad

#Esto indica que para un esquema de configuración de nivel básico
#usaremos el programa basic_nlsa_auth con el fichero de credenciales /etc/squid/
↪ credenciales
auth_param basic program /usr/lib/squid3/basic_nlsa_auth /etc/squid/credenciales

#Se deben poder autenticar hasta a 10 usuarios a la vez
auth_param basic children 10

#Texto que se envía a los usuarios que inicien sesión
auth_param basic realm Indique sus credenciales

auth_param basic credentialsttl 2 hours

#Esto fabrica una lista para indicar "usuarios que están autenticados".
acl autenticados proxy_auth REQUIRED

#Esto indica la lista de periodicos deportivos
acl deportes dstdomain .marca.com

#Y aquí está la clave, COMBINAR listas
#Se deniega el acceso al periodico a
#aquellos usuarios que NO estén autenticados
http_access deny deportes !autenticados
```

```
auth_param basic program /usr/lib/squid3/basic_nlsa_auth /etc/squid/credenciales.txt auth_param basic realm Indique
```

```
su clave por favor auth_param basic children 10 auth_param basic credentialsttl 4 hours
```

```
acl programadores proxy_auth REQUIRED
```

```
http_access allow programadores http_access deny resto_personas
```

4.9 Configuración del almacenamiento en la caché de un proxy .

En general los valores por defecto de Squid suelen considerarse bastante apropiados, pero pueden modificarse algunos de ellos si se desea obtener más rendimiento.

```
cache_dir ufs /var/spool/squid 20000 64 1024
```

Esta caché acepta hasta 20.000MB de datos organizándonos en 64 directorios de hasta 1024 subdirectorios cada uno

```
cache_mem 16MB
```

Indica que se deben tener preparados unos 16MB de RAM para los objetos populares en cache (páginas o archivos muy solicitados).

Advertencia: Hay que tener cuidado con cuanta memoria RAM consume Squid ya que si tenemos muy poca la caché empezará a desviar objetos a «swap» y el rendimiento bajará en picado. En general la `cache_mem` debe ser como un tercio de la RAM disponible y luego añadir unos 14MB por cada GB de caché en disco. Esto significa que si en nuestro ejemplo tenemos 20GB de caché en disco y una `cache_mem` de 16MB debemos asegurarnos de tener como mínimo $(16*3)+(20*14)=48+280=328\text{MB}$ como mínimo de RAM.

Squid rota los ficheros de caché automáticamente y borra los ficheros menos usados para dar cabida a los nuevos. Si de todas maneras se desea saber cuanto espacio está ocupando la cache puede usarse el comando `du -h /var/spool/squid` para ver cuanto espacio en disco ocupa la caché.

4.10 Proxys inversos.

Como ya hemos visto antes, un proxy inverso es aquel que se sitúa delante de un servidor Web con el objetivo de «descargarle de trabajo», como señala la figura siguiente:

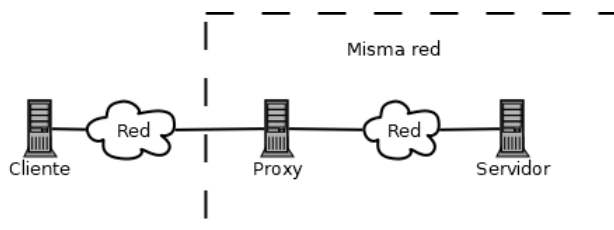


Figura 6: Proxy inverso

Para experimentar con esta configuración necesitaremos esto:

- Dos máquinas virtuales, ambas con Ubuntu Server.
- En una de ellas (que llamaremos «Proxy») pondremos el proxy Squid (si no se tiene, se debe instalar con `sudo apt-get install squid`)

- En la otra (que llamaremos «Servidor Web») pondremos el servidor Web Apache, PHP y Links (se puede instalar con `sudo apt-get install apache2 php`). Asegúrate de que tenga una carpeta compartida con el anfitrión para que podamos pasar con comodidad ficheros entre ambas máquinas.
- Las dos máquinas deben tener una tarjeta de red en modo puente, deben tener un IP de la misma red y deben poder hacerse ping entre ellas.

Dentro del servidor Web pondremos una pequeña página PHP que simplemente muestre información de la fecha y hora, como esta. La llamaremos `index.php` y debe estar en el directorio `/var/www/html` (asegúrate también de que la página la puede leer todo el mundo con `sudo chmod a+r /var/www/html/index.php` :

```
<!DOCTYPE html>
<html>

  <head>
    <title>Página HTML para Servidor web con proxy inverso</title>
    <meta charset="utf-8">
  </head>
  <!--Cuerpo-->
  <body>
    <h1>Bienvenido a nuestra página</h1>
    <p>
      <?php

        $fecha=date("d/m/Y");
        $hora =date("h:i:s");

        echo "Esta página se generó en el servidor el $fecha a las $hora.";
      ?>
    </p>
  </body>
</html>
```

Si está todo bien, en el Servidor Web podremos ejecutar links `http://127.0.0.1` y veremos la página generada en el servidor donde nos dirá la fecha y la hora.

Una vez configurado el Servidor Web toca configurar Squid. En Squid un «proxy inverso» se denomina un «acelerador». En realidad, la configuración básica es muy sencilla y aun así ya ofrece mucha mejora en el rendimiento:

```
http_port 80 accel defaultsite=192.168.1.130 no-vhost
cache_peer 192.168.1.30 parent 80 0 no-query originserver name=AceleradorWebLocal
refresh_pattern -i \.php 2 50% 9
```

¿Qué significa todo esto?

- La primera línea indica que Squid va aceptar peticiones en el puerto 80 pero esto va a servir para acelerar un sitio web en el que no hay hosting virtual.
- La segunda línea indica que vamos a cachear tales peticiones conectando con el sitio web principal («parent») y que cuando se conecten a nuestro puerto 80 no nos conectaremos a ningún otro puerto ICP (sirve para proxies encadenados), y que como no haremos encadenamiento de proxies no hay que hacer consultas («no-query») sino que esto es el servidor real de origen («originserver»). A este servidor cacheado le hemos puesto un nombre que luego podríamos proteger con ACLs.
- La tercera línea indica que cualquier petición que termine en PHP debe ser cacheada durante al menos 2 minutos y como máximo 9 minutos. Si alguna petición *no lleva indicado el tiempo máximo por parte del navegador* se

asume que se conserva en caché hasta que llega a la mitad de su edad. Por ejemplo si alguien pidió un PHP hace 6 horas y han pasado 3 se considera que el objeto ya no debe estar en caché.

Advertencia: En algunos Squid hay una línea como esta en el fichero `/etc/squid.conf` que hace que por defecto

Para comprobar que esto funciona abre varias pestañas en tu navegador (no vale pulsar F5 porque el navegador solicita entonces actualizar las cachés) y verás que todas llevan la misma hora de generación.

Advertencia: Lo siguiente es **una violación del protocolo HTTP**: Si realmente queremos ignorar a los usuarios que pulsen F5 se puede añadir una opción al final de la línea y dejarla como `refresh_pattern php$ 2 50% 9 ignore-reload` Con ello, Squid ignorará todas las peticiones de recarga incluso aunque se pulse F5

Para aprender realmente todo lo que envía y recibe el navegador utiliza las herramientas del desarrollador (usa la tecla F12) y explora las distintas cabeceras que envía Firefox/Chrome/Edge.

4.11 Proxys encadenados.

4.12 Pruebas de funcionamiento. Herramientas gráficas.

En Ubuntu es posible instalar un programa llamado `squidclient` que permite hacer consultas sencillas al servidor Squid y obtener muchas estadísticas sobre el uso de memoria, CPU y disco. Para instalarlo puede usarse `sudo apt-get install squidclient` y una vez instalado usar `squidclient mgr:info` para obtener los valores de uso.

5.1 Legislación sobre protección de datos.

5.2 La Ley Orgánica de Protección de Datos

5.2.1 Artículo 1. Objeto de la Ley

5.2.2 Artículo 2. Ámbito de aplicación

Ficheros automatizados pero sin incluir información clasificada y que estén dentro del ámbito de la UE (si están fuera de la UE se estará a lo dispuesto en «su legislación específica»). Los datos resultado de procesos judiciales se incluyen en este fichero sin incluir otras disposiciones que se puedan aplicar debido a la Ley del Poder Judicial.

5.2.3 Artículo 3. Datos de fallecidos

- Los familiares de un fallecido pueden acceder a sus datos. Hay una excepción: el fallecido podría haber prohibido dicho acceso.
- El fallecido puede permitir acceso a otros, aparte de sus familiares.

5.2.4 Artículo 4. Exactitud de los datos

Los datos deberán ser exactos y actualizados. Si no es así se puede imputar al responsable excepto que:

- Los datos vengan del propio afectado (es decir, mintió)
- Los datos provengan de un mediador (se imputará al mediador).
- Los datos fueran procesados por otro responsable.
- Los datos se obtuvieran de un registro público.

5.2.5 Artículo 5. Deber de confidencialidad.

Se debe guardar el secreto profesional **incluso despues de dejar de ser el responsable**

5.2.6 Artículo 6. Tratamiento basado en el consentimiento del afectado.

El afectado debe dar su consentimiento claramente. Si se pide el consentimiento para muchas finalidades, se debe indicar claramente que se pide para todas. **No podrá supeditarse la ejecución del contrato a que el afectado consienta el tratamiento de los datos personales para finalidades que no guarden relación con el mantenimiento, desarrollo o control de la relación contractual.**

5.2.7 Artículo 7. Consentimiento de los menores de edad.

Solo se acepta el consentimiento de los mayores de 14. Solo se acepta el consentimiento de los menores de 14 si el consentimiento de sus tutores también aparece.

5.2.8 Artículo 8. Tratamiento de datos por obligación legal, interés público o ejercicio de poderes públicos.

El reglamento de la UE es el que dice cuando un tratamiento de datos es una obligación legal o un interés público. P.ej, impuestos o censos.

5.2.9 Artículo 9. Categorías especiales de datos.

En general: ideología, afiliación sindical, religión, orientación sexual, creencias u origen racial o étnico. Hay algunas excepciones, como por ejemplo una fundación, ONG o sindicato.

5.2.10 Artículo 10. Tratamiento de datos de naturaleza penal.

Los datos penales no podrán usarse para otros fines y su tratamiento solo podrá hacerse por abogados y procuradores.

5.2.11 Artículo 11. Transparencia e información al afectado.

El responsable debe cumplir con el deber de información y decir al afectado:

1. La identidad del responsable o su representante.
2. La finalidad del tratamiento.
3. La posibilidad de ejercer sus derechos a modificación, borrado o cancelación.

5.2.12 Artículo 12. Disposiciones generales sobre ejercicio de los derechos.

- El responsable del tratamiento estará obligado a informar al afectado sobre los medios a su disposición para ejercer los derechos que le corresponden.
- La prueba del cumplimiento del deber de responder a la solicitud de ejercicio de sus derechos formulado por el afectado recaerá sobre el responsable
- Las actividades del responsable serán gratuitas.

Implantación de técnicas de acceso remoto. Seguridad perimetral

6.1 Elementos básicos de la seguridad perimetral.

- Cortafuegos
- Routers frontera
- Zonas desmilitarizadas.
- VPNs
- Host bastión.
- Sistemas IDS.
- VLAN.

6.2 Perímetros de red. Zonas desmilitarizadas. Router frontera.

6.3 Arquitectura débil de subred protegida.

6.4 Arquitectura fuerte de subred protegida.

6.5 Políticas de defensa en profundidad.

- Permitir lo que no esté prohibido
- Prohibir lo no permitido.

6.6 Defensa perimetral.

- Administrar contenido
- Filtrar tráfico.
- Redirigir tráfico.

6.7 Defensa interna.

6.8 Factor Humano.

6.9 Redes privadas virtuales. VPN.

6.9.1 VPN a nivel de enlace.

6.9.2 VPN a nivel de red. SSL, IPSec .

6.9.3 VPN a nivel de aplicación. SSH.

6.10 Infraestructura de clave pública o PKI

En administración de sistemas suele ser necesario la configuración de mecanismos seguros que impliquen cifrado y confianza. Aunque el proceso que se va a mencionar ahora suele requerir que se pida la intervención de una autoridad de certificación externa en otros casos no va a ser necesario. A continuación se muestra como crear nuestra propia autoridad certificadora que pueda firmar certificados de otros. El objetivo de estos puntos es conseguir lo siguiente:

- Una clave raíz (que estará en un fichero llamado `ca.key`)
- Un certificado raíz que podremos pasar a todos nuestros clientes (fichero `ca.crt`).
- Una clave privada para un servidor (fichero `servidor.key`)
- Un certificado emitido para un cierto servicio o servidor (fichero `servidor.crt`)
- Una clave privada para un servidor (fichero `cliente.key`)
- Un certificado emitido para un cierto servicio o servidor (fichero `cliente.crt`)
- Un fichero con números primos precalculados (`Precalculados.pem`)

6.10.1 Paso 0: instalar Easy-RSA

Usando `sudo apt-get install easy-rsa` podremos instalar un software que nos automatizará el proceso de crear una autoridad de certificación con su propio certificado y un certificado de servidor firmador por esa autoridad propia.

6.10.2 Paso 1: Crear una autoridad

Usando `make-cadir <directorio>` podremos crear un directorio con los ficheros de configuración necesarios para crear nuestra propia autoridad de certificación. Después entramos en él y editamos el fichero `vars` para indicar los siguientes datos:

- País.
- Estado/Provincia.
- Ciudad.
- Organización.
- Email.
- Unidad organizativa.

6.10.3 Paso 2: crear la infraestructura de claves

Toda autoridad de certificación tiene como mínimo una clave raíz que se necesitará en todos los procesos. Podemos usar el comando `./easysrsa init-pki` para construir los ficheros necesarios (pero aún no se generarán las claves).

6.10.4 Paso 3: construir los ficheros de la CA

Con los datos rellenos en el paso 1 y la clave privada del paso 2 se puede crear el certificado raíz de nuestra CA usando `./easysrsa build-ca`. Se nos pedirá una clave de acceso para custodiar la clave raíz que se va a generar y se nos pedirá un nombre de usuario o de servidor para incorporar al certificado raíz.

6.10.5 Paso 4: generar un certificado para un servidor

Usando `./easysrsa build-server-full <nombre_de_servidor_o_servicio>` se generarán dos cosas:

- Una clave privada para el servicio (estará en `pki/private/<nombre>.key`)
- Un certificado para ese servidor que irá firmado por nuestra CA (estará en `pki/issued/<nombre>.crt`)

6.10.6 Paso 5: generar un certificado para un cliente

Usando `./easysrsa build-cliente-full <nombre_de_fichero_cliente>` se generarán otra vez dos cosas:

- Una clave privada para el cliente (estará en `pki/private/<nombre_fichero_cliente>.key`)
- Un certificado para ese cliente que irá firmado por nuestra CA (estará en `pki/issued/<cliente>.crt`)

Advertencia: Es importante no olvidar las claves de acceso a los ficheros, los necesitaremos después para conectar los clientes la VPN.

6.10.7 Paso 5: precalcular parámetros de claves

Cuando se establezca una conexión se van a utilizar algunos números para cifrar los datos. Estos valores pueden tenerse precalculados en un fichero para acelerar el inicio de las conexiones. Esto puede hacerse con el comando `./easyrsa gen-dh` o también con `openssl dhparam -dsaparam 2048 -out Parametros.pem`

Este comando genera números primos aceptables para el establecimiento de una conexión, usando 2048 bits como longitud de clave pero evitando (con el parámetro `dsaparam` una serie de números que no aportan más seguridad).

6.10.8 Paso 6: configurar el servidor y arrancarlos

En el servidor podemos crear un fichero como este:

```
proto udp #OpenVPN usará UDP para la comunicación
port 1194 #OpenVPN escuchará en este puerto
dev tun #Se creará un dispositivo de red de tipo túnel
#Los usuarios que se conecten usarán direcciones de esta subred
server 10.100.0.0 255.255.255.0
#OpenVPN podía usar otras topologías como
#punto a punto, pero hoy en día no se recomiendan
topology subnet
#Si la conexión VPN sufre un reinicio no
#hace falta volver a leer los ficheros de claves
persist-key
#Si la conexión VPN sufre un reinicio no
#hay que re-crear el dispositivo de red
persist-tun
#Enviar un paquete si el cliente no envía nada
#en 10*2=20 segundos (el doble es por el tiempo
#de ida y vuelta) y reiniciar la conexión VPN
#si el servidor no recibe nada en 60 segundos.
keepalive 10 60
#Fichero con los parámetros de intercambio de claves
dh /home/usuario/autoridad/ParametrosDH.pem
#Fichero con el certificado del servidor
cert /home/usuario/autoridad/pki/issued/ServidorOpenVPN.crt
#Fichero con la clave privada del servidor
key /home/usuario/autoridad/pki/private/ServidorOpenVPN.key
#Fichero con el certificado de la autoridad que firmó
#el certificado del servidor
ca /home/usuario/autoridad/pki/ca.crt

log-append /var/log/openvpn.log
```

Y arrancar OpenVPN con `sudo openvpn --config servidor.conf`

6.10.9 Paso 7: configurar el cliente y arrancarlo.

Estando en el mismo directorio del servidor podemos usar el comando `./easysrsa -build-cliente-full Cliente01` y generar todo lo necesario para que se conecte un cliente. En concreto nos interesa esto:

- El certificado del cliente (debe estar en `pki/issued/Cliente01.crt`)
- La clave privada del cliente (debe estar en `pki/private/Cliente01.key`)
- El certificado de la autoridad de certificación (en `pki/ca.crt`)

Si es necesario, usaremos una carpeta compartida para meter estos ficheros dentro de la máquina virtual y copiarlos a algún directorio donde tengamos permisos. Una vez los tengamos dentro, solo hay que abrir el menú de configuración de VPN del entorno de escritorio (si queremos usar Windows deberemos instalar OpenVPN).

Dentro del entorno de escritorio podemos indicar la IP del servidor OpenVPN así como los tres ficheros que hemos indicado. En la figura siguiente se muestra una captura de pantalla de un cliente Linux. En dicha figura puede observarse que la traducción no es muy correcta y que quizá los términos correctos que deberían verse son:

1. Certificado de la CA.
2. Certificado de usuario.
3. Fichero de clave privada de usuario.
4. Clave de acceso al fichero de clave privada de usuario.

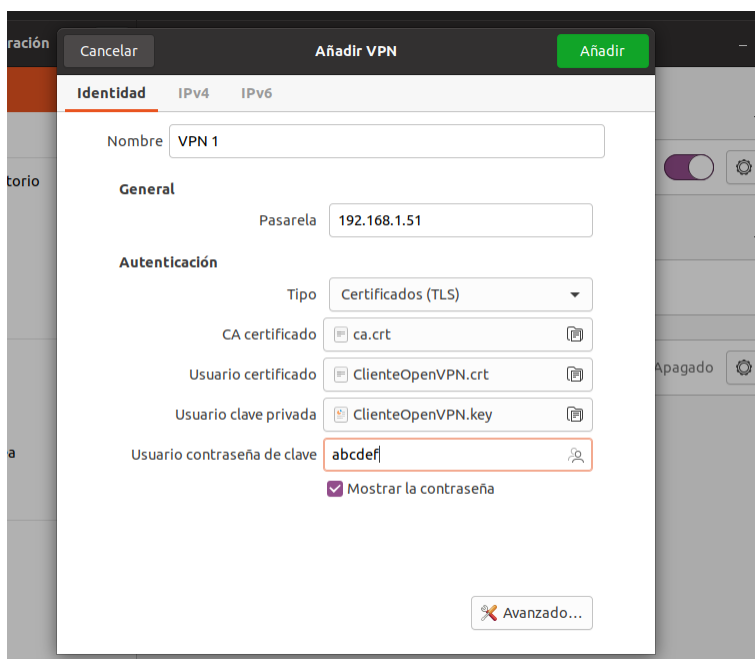


Figura 1: Configuración de un cliente Linux con VPN

Si hemos hecho todo correctamente podremos ver en el servidor que se recibe una conexión y en el cliente veremos que ha aparecido una nueva IP y que ahora podemos hacer ping a la IP de la VPN del servidor. Todo el tráfico que fluye entre cliente y servidor ahora circula cifrado.

Pregunta: *¿por qué ahora el cliente no puede navegar por Internet?*. La respuesta probablemente sea porque en el servidor aún no se han hecho muchas cosas, como activar el enrutamiento, tal vez configurar el NAT o quizá incluso ni siquiera hayamos añadido una tarjeta de red a la máquina virtual del servidor.

6.11 Beneficios y desventajas con respecto a las líneas dedicadas.

6.12 Técnicas de cifrado. Clave pública y clave privada.

6.13 Intérprete de comandos SSH.

6.14 Gestión de archivos SSH.

6.15 Servidores de acceso remoto

6.16 Protocolos de autenticación.

6.17 Configuración de parámetros de acceso.

6.18 Servidores de autenticación.