Sistemas confiables

Máster Universitario de Investigación en Ciberseguridad

Ángel Manuel Guerrero Higueras
am.guerrero@unileon.es

GRUPO DE **ROBÓTICA**
Universidad de León

Parte 2
Asegurando redes de computadoras

1. Conceptos básicos sobre redes de comunicaciones
2. Filtrado de paquetes
3. Multi-container environments
   - Docker Compose
4. Securización de servidores web. Apache
5. Otros servicios: proxy, DHCP, DNS, VPN, LDAP

5. Conceptos básicos sobre redes de comunicaciones

1. Conceptos básicos sobre redes de comunicaciones
   - Internet: estructura y funcionamiento
   - Redes y protocolos de comunicaciones

5.1. Internet: estructura y funcionamiento

Estructura de internet

- No es una red centralizada ni está regida por un solo organismo.
- Su estructura es semejante a una tela de araña en la cual unas redes se conectan con otras.
- Hay una serie de organizaciones responsables de la adjudicación de recursos y el desarrollo de los protocolos necesarios para que Internet evolucione.
  - Internet Engineering Task Force (**IETF**): se encarga de redactar los protocolos usados en Internet.
  - Internet Corporation for Assigned Names and Numbers (nombres de dominio, direcciones IP, etc.): coordina la asignación de identificadores únicos en Internet, incluyendo...
    - nombres de dominio,
    - direcciones IP,
    - etc.

Direcciones en internet

En Internet se emplean varios formatos para identificar máquinas, usuarios o recursos en general.

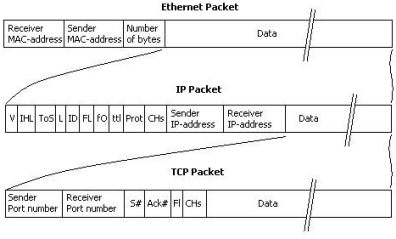
- Direcciones numéricas para identificar **máquinas**: las direcciones IP.
  - Se representan por cuatro números, de 0 a 255, separados por puntos.
    - Un servidor puede identificarse, por ejemplo, con la dirección IP 66.230.200.100.
  - Como es más sencillo recordar un nombre, las direcciones se **traducen** a nombres.
    - Los trozos "traducidos"se denominan nombres de dominio.
    - El servicio encargado de la traducción es DNS
- Para identificar a usuarios de correo electrónico se emplean las **direcciones de correo electrónico**.
  - `usuarioservidor_de_correo.dominio`
- Para identificar **recursos en Internet**, se emplean direcciones URL (Uniform Resource Locator.
  - Una dirección URL tiene la forma: `http://identificador.dominio/abc.htm`
    - `http://` es el protocolo,
    - `identificador.dominio` es el dominio (que se traduce a una dirección IP por los servicios DNS),
    - y `abc.htm` es la localización del recurso al que se accede.

Flujo de información I

- Arquitectura **cliente-servidor**:
  - El procedimiento empleado para intercambiar información en Internet sigue el modelo cliente-servidor.
  - Los servidores son computadoras donde se almacenan datos.
  - El cliente es la computadora que realiza la petición al servidor para que éste le muestre alguno de los recursos almacenados.
- **Paquetes** de información:
  - En Internet la información se transmite en pequeños trozos llamados *paquetes*.
  - La prioridad es la reconstrucción en el destino del mensaje emitido, no el camino seguido por los paquetes que lo componen.
  - Si se destruye un nodo de la red, los paquetes encontrarán caminos alternativos.

Flujo de información II

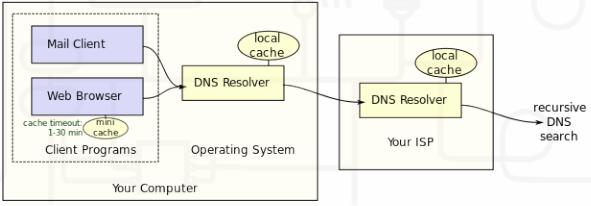
- **TCP/IP**:
  - Pila de protocolos que controla el flujo de datos en Internet.
    - El protocolo TCP (y también el UDP), se encarga de fragmentar el mensaje emitido en paquetes. En el destino, se reorganizan los paquetes para formar de nuevo el mensaje, y entregarlo a la aplicación correspondiente.
    - El protocolo IP enruta los paquetes. Esto hace posible que los distintos paquetes que forman un mensaje pueden viajar por caminos diferentes hasta llegar al destino.



  - La unión de varias redes (ARPANET y otras) en Estados Unidos, en 1983, siguiendo el modelo TCP/IP, se considera como el nacimiento de Internet.

Flujo de información III

- **Servicio de Nombres**
  - El servicio Domain Name System (**DNS**) proporciona la correspondencia entre una dirección IP y su nombre de dominio, y viceversa.
  - Cada vez que se inicia una comunicación con un nombre de dominio, el ordenador realiza una petición a su servidor DNS para que le proporcione la IP asociada a ese nombre.



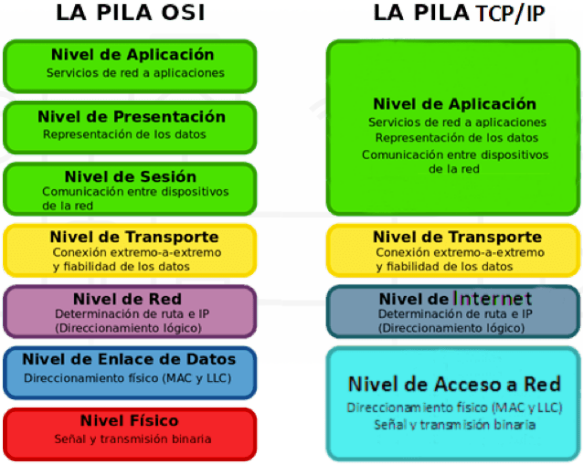
  - DNS es jerárquico.
    - Cada subdominio de Internet suele tener su propio servidor DNS, responsable de los nombres bajo su dominio.
    - A su vez, hay un servidor encargado de cada dominio (por ejemplo un nivel nacional (.es)).
    - Hay una serie de servidores raíz, que conocen toda la estructura.

5.2. Redes y protocolos de comunicaciones

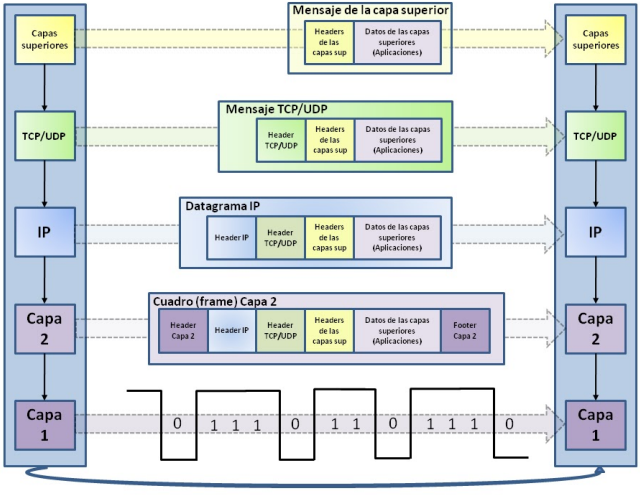
Redes y protocolos de comunicaciones

- **Red de comunicaciones** $\longrightarrow$ colección de elementos conectados entre sí para intercambiar información. Incluye todos los medios técnicos (materiales o lógicos) necesarios para lograr esa interconexión.
- La comunicación se realiza por medio del envío/recepción de mensajes construidos de acuerdo a un código común de señales.
- **Protocolo de comunicaciones** $\longrightarrow$ conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información.
- **Pila de protocolos** $\longrightarrow$ colección ordenada de protocolos organizados en capas que se ponen unas encima de otras y en donde cada protocolo implementa una abstracción encuadrada en la funcionalidad que proporciona cada capa.
  - Los protocolos encuadrados en la capa inferior proporcionan sus servicios a los protocolos de la capa superior para que estos puedan realizar su propia funcionalidad.
  - ejemplos:
    - Modelo de interconexión de sistemas abiertos (ISO/IEC 7498-1), también llamado **modelo OSI**.
    - La **pila TCP/IP**, denominada así debido a los protocolos más importantes que lo componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

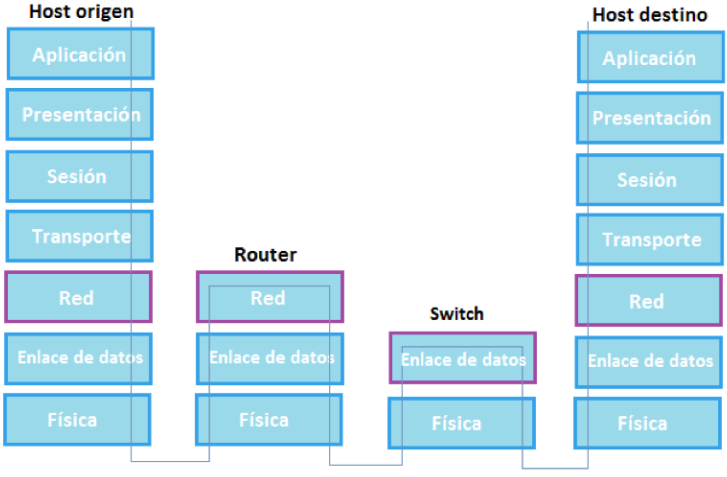
Modelo OSI vs. TCP/IP



Encapsulamiento TCP/IP



Transiciones en el modelo OSI



Direcciones IP (red) I

- La dirección IP es un número que identifica, de manera lógica y jerárquica, a una Interfaz en red (elemento de comunicación/conexión) de un dispositivo que utilice Internet Protocol (IP) en nivel de red del modelo TCP/IP.
- La dirección IP no debe confundirse con la dirección MAC, que es un identificador de 48 bits expresado en código hexadecimal, para identificar de forma única la tarjeta de red y no depende del protocolo de conexión utilizado en la red.
- Las direcciones IPv4 se expresan mediante un número binario de 32 bits permitiendo un espacio de direcciones de hasta 4.294.967.296 (2^{32}) direcciones posibles.
- Se pueden expresar como números de notación decimal: se dividen los 32 bits de la dirección en cuatro octetos.
- En la expresión de direcciones IPv4 en decimal se separa cada octeto por un carácter ".".
- Cada uno de estos octetos puede estar comprendido entre 0 y 255.

Direcciones IP (red) II

- Clases de direcciones:

Clase	Bits iniciales	Intervalo	# de redes	# de direcciones	# de hosts	Máscara	Broadcast
A	0	0.0.0.0 - 127.255.255.255	126	16.777.216	16.777.214	255.0.0.0	x.255.255.255
B	10	128.0.0.0 - 191.255.255.255	16.384	65.536	65.534	255.255.0.0	x.x.255.255
C	110	192.0.0.0 - 223.255.255.255	2.097.152	256	254	255.255.255.0	x.x.x.255
D (Multicast)	1110	224.0.0.0 - 239.255.255.255					
E (experimental)	1111	240.0.0.0 - 255.255.255.255					

- Direcciones privadas:

Nombre	Bloque CIDR	Raango	# de direcciones	Clase
bloque de 24-bit	10.0.0.0/8	10.0.0.0 – 10.255.255.255	16.777.216	Clase A
bloque de 20-bit	172.16.0.0/12	172.16.0.0 – 172.31.255.255	1.048.576	16 bloques de clase B
bloque de 16-bit	192.168.0.0/16	192.168.0.0 – 192.168.255.255	65.536	256 bloques de clase C

Puertos de comunicaciones (transporte)

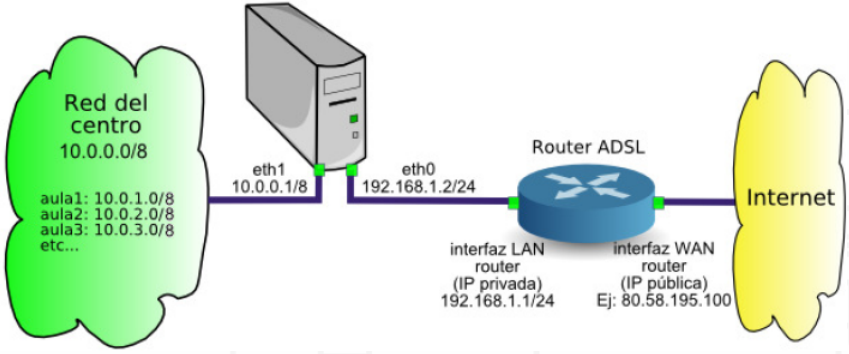
- Cuando un proceso desea establecer una conexión con un servidor de aplicación remoto, debe especificar a cuál se conectará.
- El método que normalmente se emplea es definir direcciones de transporte en las que los procesos pueden estar a la escucha de solicitudes de conexiones.
- Estos puntos terminales se denominan **puertos**.
- Esta asignación permite a una máquina establecer simultáneamente diversas conexiones con máquinas distintas, ya que todos los paquetes que se reciben tienen la misma dirección, pero van dirigidos a puertos diferentes.
- Los puertos se numeran del 0 al 65535.
  1. Bien conocidos: 0—1023.
  2. Registrados: 1024—49151.
  3. Dinámicos o privados: 49152—65535.

6. Filtrado de paquetes

2 Filtrado de paquetes
- Routing
- Firewall
- UFW
- iptables
- De-Militarized Zone (DMZ)

6.1. Routing

Routing on Linux



- Tipos de tráfico: **INPUT**, **OUTPUT** y **FORWARD**.
- Enable routing:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -A FORWARD -j ACCEPT
iptables -t nat -A POSTROUTING -s 10.0.0.0/8 -o eth0 -j MASQUERADE
```

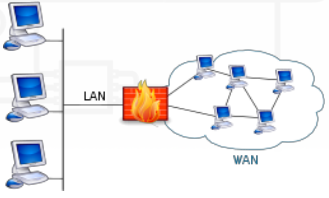
6.2. Firewall

Características básicas

Firewall

Dispositivo que realiza un filtrado de paquetes de datos a partir de unas reglas definidas por el administrador de la red, teniendo en cuenta...

- las direcciones IP fuente o destino (es decir, de qué ordenador provienen y a qué ordenador van dirigidos los paquetes de datos)
- y el servicio de red al que se corresponden (puerto).



- Recuerda, para habilitar el enrutamiento hay activar bit de *forwarding*:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Servicios ofrecidos

1. **Bloqueo del tráfico** no autorizado por la organización:
   - servicios de Internet que se desaen restringir,
   - determinadas direcciones de equipos o de ciertas páginas web,
   - etc.
2. **Ocultación** de los equipos internos de la organización, de forma que estos puedan resultar invisibles ante posibles ataques provenientes del exterior.
   - Un cortafuegos permite ocultar información sobre la topología de la red interna, los nombres de los equipos, los dispositivos de red utilizados, etc.
     - Traducción de direcciones utilizando el protocolo NAT (*Network Address Translation*).
     - Mapeo de puertos utilizando el protocolo PAT (*Port Address Translation*).
3. **Registro** de todo el tráfico entrante y saliente de la red.
4. Redirección del tráfico entrante hacia determinadas zonas restringidas o especialmente vigiladas (**zonas DMZ**).

6.3. UFW

Uncomplicated Firewall (UFW) I

UFW (Uncomplicated Firewall), es una interfaz para ipatables orientado a simplificar el proceso de configuración de un firewall.

- Instalación de UWF (instalado por defecto desde Ubuntu 16):

```
$ sudo apt-get install ufw
```

- Configuración de UFW: `/etc/default/ufw`
- Establecimiento de las **políticas por defecto**:

```
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
```

- Permitir conexiones SSH:

```
$ sudo ufw allow ssh
$ sudo ufw allow 22
$ sudo ufw allow 2222
```

Uncomplicated Firewall (UFW) II

- Habilitar UFW:

```
$ sudo ufw enable
```

- Permitir otras conexiones:

```
$ sudo ufw allow http
$ sudo ufw allow 80
$ sudo ufw allow https
$ sudo ufw allow 443
$ sudo ufw allow ftp                          # you probably shouldn't use
$ sudo ufw allow 21/tcp
$ sudo ufw allow 6000:6007/tcp                # Specific Port Ranges
$ sudo ufw allow 6000:6007/udp                # Specific Port Ranges
$ sudo ufw allow from 15.15.15.51             # Specific IP Addresses
$ sudo ufw allow from 15.15.15.51 to any port 22
$ sudo ufw allow from 15.15.15.0/24           # Subnets
$ sudo ufw allow from 15.15.15.0/24 to any port 22
```

Conexiones a una interfaz de red específica:

```
$ ip addr
$ sudo ufw allow in on eth0 to any port 80
$ sudo ufw allow in on eth1 to any port 3306
```

Uncomplicated Firewall (UFW) III

- Denegación de conexiones:

```
$ sudo ufw deny http
$ sudo ufw deny from 15.15.15.51
```

- Borrado de reglas mediante número:

```
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    15.15.15.0/24
[ 2] 80                         ALLOW IN    Anywhere
```

Borrado de reglas:

```
$ sudo ufw delete 2
```

- Borrado de reglas mediante descripción:

```
$ sudo ufw delete allow http
$ sudo ufw delete allow 80
```

Uncomplicated Firewall (UFW) IV

- Comprobar el estado de UFW y las reglas:

```
$ sudo ufw status verbose
```

Salida:

```
Status: inactive
```

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
```

- Inhabilitar o resetear UFW:

```
$ sudo ufw disable
$ sudo ufw reset
```

6.4. iptables

Conceptos iptables I

iptables

Aplicación que permite a un administrador configurar las **tablas** provistas por el firewall del kernel de Linux (implementado como diferentes módulos de Netfilter), y las **cadenas** y **reglas** que almacena.

- Tablas (**tables**) $\rightsquigarrow$ defined by the general aim of the rules they hold.
  - `filter`*: filtrado paquetes.
  - `nat`*: traducción de direcciones y redirecciones de puertos.
  - `raw`: configuración de exenciones de seguimiento de conexiones en combinación con el target NOTRACK.
  - `mangle`: alteración de los paquetes.

Conceptos iptables II

- Cadenas (**chains**) $\rightsquigarrow$ represent the netfilter hooks which trigger them.
  - `PREROUTING`: Los paquetes pasan por esta cadena antes de que se decida su encaminamiento.
  - `INPUT`: Paquetes dirigidos a la propia máquina.
  - `OUTPUT`: Paquetes enviados desde la propia máquina.
  - `FORWARD`: Paquetes encaminados a través de la máquina.
  - `POSTROUTING`: La decisión de encaminamiento ha sido tomada. Los paquetes pasan por esta cadena justo antes de abandonar la máquina.

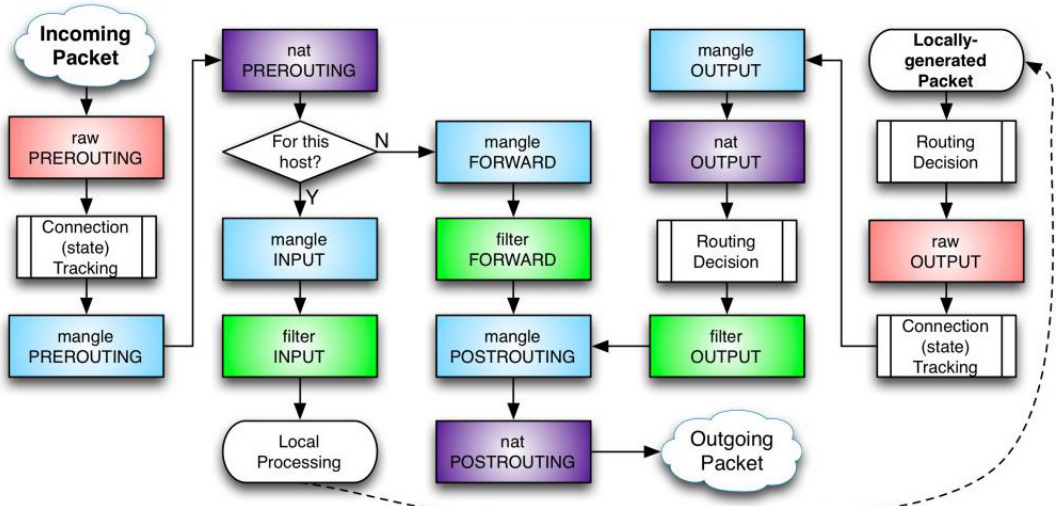
Conceptos iptables III

- Objetivos (**targets**):
  - `ACCEPT`*
  - `DROP`*
  - `DNAT`*, `SNAT`*
  - `MASQUERADE`*
  - `QUEUE`
  - `RETURN`
  - `REJECT`
  - `LOG`, `ULOG`
  - `NOTRACK`

Conceptos iptables IV

- Seguimiento de conexiones (**states**):
  - `NEW`
  - `ESTABLISHED`
  - `RELATED`
  - `INVALID`

iptables procress flow



IPTables. Opciones generales y reglas I

- Sintaxis general del comando `iptables`:

```
$ iptables { -A | --append | -D | --delete } chain rule-specification [ options ]
```

- Opciones generales:

```
-t table        # default filter
-L, --list      # list rules
-v              # Modo verboso
-n              # Salida numérica
--line-numbers  # Mostrar números en las reglas
```

- Mostrar reglas:

```
$ sudo iptables -t filter -L -n -v --line-numbers
$ sudo iptables -t nat -L -n -v --line-numbers
```

IPTables. Opciones generales y reglas II

- Argumentos para la definición de reglas:

```
-j destino, --jump destino
-i [!] in-interface, --in-interface [!] in-interface
-o [!] out-interface, --out-interface [!] out-interface
-p [!] protocol, --protocol [!] protocol
-s [!] source[/prefix], --source [!] source[/prefix]
-d [!] destination[/prefix], --destination [!] destination[/prefix]
--destination-port [!] [port[[:port]], --dport [!] [port[[:port]]
--source-port [!] [port[[:port]], --sport [!] [port[[:port]]
-m state [!] --state [state[[:state]]
-m multiport [!] --sports [port[[:port]]
-m multiport [!] --dports [port[[:port]]
-m limit --limit packets-number/minute --limit-burst packets-number
```

Opciones generales: ejemplos I

- Activar bit de *forwarding* (sólo si queremos que nuestra máquina encamine tráfico). Afecta a los paquetes que pasen por la cadena `FORWARD`:

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

O bien:

```
$ sudo vi /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

```
$ sudo sysctl -p
```

- Borrado de reglas:

```
$ iptables -t filter -F
$ iptables -t nat -F
$ iptables -t mangle -F
$ iptables -t raw -F
```

Opciones generales: ejemplos II

- Borrado de cadenas definidas por el usuario (aplicable sólo si están vacías):

```
$ iptables -X
```

- Resetear contadores:

```
$ iptables -t filter -Z
$ iptables -t nat -Z
$ iptables -t mangle -Z
$ iptables -t raw -Z
```

Cada tabla mantiene un contador con el número paquetes que procesa.

- Establecer **políticas por defecto**:

```
$ iptables -P INPUT ACCEPT     # ACCEPT para permitir el tráficho por la cadena, DROP para descartar
$ iptables -P OUTPUT ACCEPT    # ACCEPT para permitir por la cadena, DROP para descartar
$ iptables -P FORWARD ACCEPT   # ACCEPT para permitir por la cadena, DROP para descartar
```

Reglas: ejemplos I

- Bloquear una dirección IP específica:

```
$ BLOCK_THIS_IP="x.x.x.x"
$ iptables -A INPUT -s "$BLOCK_THIS_IP" -j DROP                 # Blocks any traffic from x.x.x.x
$ iptables -A INPUT -i eth0 -s "$BLOCK_THIS_IP" -j DROP         # blocks on eth0
$ iptables -A INPUT -i eth0 -p tcp -s "$BLOCK_THIS_IP" -j DROP  # blocks only TCP traffic on eth0
```

- Permitir tráfico **SSH entrante** en la interfaz eth0:

```
$ iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

- Permitir tráfico **SSH entrante** en la interfaz eth0 desde la red 192.168.100.0:

```
$ iptables -A INPUT -i eth0 -p tcp -s 192.168.100.0/24 --dport 22 -m state --state NEW,ESTABLISHED -j
      ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

Reglas: ejemplos II

- Permitir tráfico **HTTP y HTTPS entrante** en la interfaz eth0:

```
$ iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
```

- Combinar reglas usando multiport (incoming SSH, HTTP and HTTPS)

```
$ iptables -A INPUT -i eth0 -p tcp -m multiport --dports 22,80,443 -m state --state NEW,ESTABLISHED -
    j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp -m multiport --sports 22,80,443 -m state --state ESTABLISHED -j
    ACCEPT
```

- Permitir tráfico **SSH saliente** en la interfaz eth0:

```
$ iptables -A OUTPUT -o eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

Reglas: ejemplos III

- Permitir tráfico **SSH saliente** en la interfaz eth0 hacia la red 192.168.100.0:

```
$ iptables -A OUTPUT -o eth0 -p tcp -d 192.168.100.0/24 --dport 22 -m state --state NEW,ESTABLISHED -
    j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
```

- Permitir tráfico **HTTP y HTTPS saliente** en la interfaz eth0:

```
$ iptables -A OUTPUT -o eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
```

- Balanceo de conexiones HTTPS entrantes:

```
$ iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state NEW -m nth --counter 0 --every 3
      --packet 0 -j DNAT --to-destination 192.168.1.101:443
$ iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state NEW -m nth --counter 0 --every 3
      --packet 1 -j DNAT --to-destination 192.168.1.102:443
$ iptables -A PREROUTING -i eth0 -p tcp --dport 443 -m state --state NEW -m nth --counter 0 --every 3
      --packet 2 -j DNAT --to-destination 192.168.1.103:4
```

Reglas: ejemplos IV

- Permitir *ping* desde fuera a dentro:

```
$ iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
$ iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

- Permitir *ping* desde dentro a fuera:

```
$ iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
$ iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

- Permitir todo el tráfico en la interfaz de Loopback:

```
$ iptables -A INPUT -i lo -j ACCEPT
$ iptables -A OUTPUT -o lo -j ACCEPT
```

- Permitir tráfico de una red interna a otra externa:

```
$ iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

Reglas: ejemplos V

- Permitir tráfico **DNS saliente**:

```
$ iptables -A OUTPUT -p udp -o eth0 --dport 53 -j ACCEPT
$ iptables -A INPUT -p udp -i eth0 --sport 53 -j ACCEPT
```

- Permitir rsync desde la red 192.168.101.0:

```
$ iptables -A INPUT -i eth0 -p tcp -s 192.168.101.0/24 --dport 873 -m state --state NEW,ESTABLISHED -
    j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 873 -m state --state ESTABLISHED -j ACCEPT
```

- Permitir conexiones MySQL desde la red 192.168.100.0:

```
$ iptables -A INPUT -i eth0 -p tcp -s 192.168.100.0/24 --dport 3306 -m state --state NEW,ESTABLISHED
    -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 3306 -m state --state ESTABLISHED -j ACCEPT
```

- Permitir tráfico **SMTP entrante**:

```
$ iptables -A INPUT -i eth0 -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT
```

Reglas: ejemplos VI

- Permitir conexiones **IMAP e IMAPS entrantes**:

```
$ iptables -A INPUT -i eth0 -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 143 -m state --state ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --dport 993 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 993 -m state --state ESTABLISHED -j ACCEPT
```

- Permitir conexiones **POP3 y POP3S entrantes**:

```
$ iptables -A INPUT -i eth0 -p tcp --dport 110 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 110 -m state --state ESTABLISHED -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --dport 995 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 995 -m state --state ESTABLISHED -j ACCEPT
```

- Prevenir ataques DoS:

```
$ iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT
```

  - `-m limit`: extension iptables limit
  - `--limit 25/minute`: Máximo 25 conexiones por minuto.
  - `--limit-burst 100`: Este valor indica que el limite de conexiones por minuto se aplica solo si se alcancan las 100 conexiones.

Reglas: ejemplos VII

- Redirección de puertos (conexiones SSH desde el puerto 2222):

```
$ iptables -t nat -A PREROUTING -p tcp -d 192.168.102.37 --dport 2222 -j DNAT --to 192.168.102.37:22
$ iptables -A INPUT -i eth0 -p tcp --dport 2222 -m state --state NEW,ESTABLISHED -j ACCEPT
$ iptables -A OUTPUT -o eth0 -p tcp --sport 2222 -m state --state ESTABLISHED -j ACCEPT
```

- Redirección enmascarada del tráfico procedente de la subred 192.168.100.0 hacia la interfaz eth0:

```
$ iptables -t nat -A POSTROUTING -o eth0 -s 192.168.100.0/24 -j MASQUERADE
```

  - Este tipo de reglas se definen cuando nuestra máquina hace de firewall para una subred (p.e. 192.168.100.0)
  - y redirije todas las peticiones (enmascarando la dirección IP origen) a una interfaz con salida a internet (p.e. eth0).

Reglas: ejemplos VIII

- Logging de paquetes descartados (debería ser de las últimas):

```
$ iptables -N LOGGING              # create a new chain called LOGGING
$ iptables -A INPUT -j LOGGING     # make all the remaining incoming connections jump to the LOGGING
    chain
$ iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables Packet Dropped: " --log-
    level 7                         # log
$ iptables -A LOGGING -j DROP      # drop these packets
```

- Bloqueo de direcciones tras sucesivos intentos de login:

```
$ iptables -A INPUT -p tcp --dport 22 -m recent --update --seconds 600 --hitcount 3 --name SSH --
    rsource -j REJECT
$ iptables -A INPUT -p tcp --dport 22 -m recent --set --name SSH --rsource -j ACCEPT
```

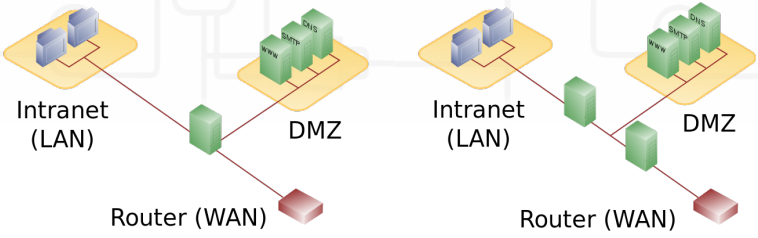
  - The first rule says that for any incoming connection on port 22,
  - iptables first checks to see if this same IP address has already tried to connect 3+ times over the last 600 seconds.
  - If it has, the newest incoming connection is rejected.
  - If it hasn't, the second rule is matched, accepting the connection.

6.5. De-Militarized Zone (DMZ)

De-Militarized Zone (DMZ)

Una DMZ (zona desmilitarizada) es una subred física o lógica que contiene y expone los servicios externos de una organización a una red que no es de confianza, generalmente una red más grande como Internet.

- **Objetivo**: añade una capa adicional de seguridad a la LAN de una organización; un nodo de red externo puede acceder solo a lo que está expuesto en la DMZ, mientras que el resto de la red de la organización está protegida por el cortafuegos.



Posibles implementaciones.

7. Multi-container environments

Docker Compose

- Compose is a tool that is used for defining and running multi-container Docker apps.
- It provides a configuration file called **docker-compose.yml** that can be used to bring up an application and the suite of services it depends on with just one command.

Docker Compose Install

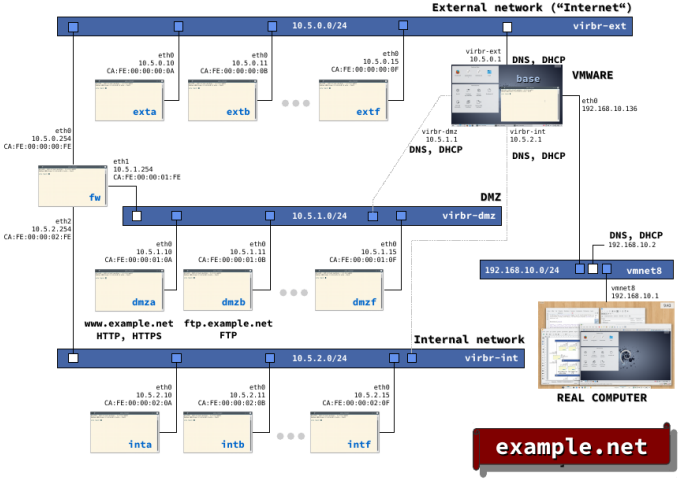
- Windows or Mac: Docker Compose is already installed as it comes in the Docker Toolbox.
- Linux:
  - Compose is written in Python, you can also simply do:

```
$ pip3 install docker-compose
```

  - Test your installation with:

```
$ docker-compose --version
docker-compose version 1.25.4, build 8d51620a
```

Caso de uso: example.net I



Source: `https://informatica.uv.es/~carlos/docencia/netinvm/netinvm.html`

Caso de uso: example.net II

- Docker Compose file (`docker-compose.yml`):

```
version: "2.2" # La versión 3 no permite asignar IPs estáticas
services:
  ...
networks:
  ...
```

- Services:
  - Firewall:

```
fw:                             # id del servicio
  image: bin/fw                 # nombre de la imagen
  build: ./fw                   # ruta del Dockerfile
  container_name: fw            # identificador
  networks:                     # id. de la red/es a la/s que pertenece el contenedor
    external-network:
      ipv4_address: 10.5.0.1    # ip estática
    DMZ:
      ipv4_address: 10.5.1.1
    internal-network:
      ipv4_address: 10.5.2.1
  privileged: true              # permisos extendidos
  tty: true
```

Caso de uso: example.net III

- Servicios en la red externa:

```
external:
  image: bin/external
  build: ./external
  networks:
    - external-network              # ip dinámica
  depends_on:                       # Dependencias
    - fw
  scale: 3                          # número de ejemplares, incompatible con container_name
  privileged: true
  tty: true
```

- Servicios en la DMZ:

```
dmz:
  image: bin/dmz
  build: ./dmz
  networks:
    - DMZ
  depends_on:
    - fw
  scale: 3
  privileged: true
  tty: true
```

Caso de uso: example.net IV

- Servicios en la red interna:

```
internal:
  image: bin/internal
  build: ./internal
  networks:
    - internal-network
  depends_on:
    - fw
  scale: 3
  privileged: true
  tty: true
```

Caso de uso: example.net V

- Networks:

```
external-network:           # id. de la red
  driver: bridge            # modo desde el host
  ipam:
   config:
    - subnet: 10.5.0.0/24   # subred
      gateway: 10.5.0.254   # encaminador

DMZ:
  driver: bridge
  ipam:
    config:
    - subnet: 10.5.1.0/24
      gateway: 10.5.1.254

internal-network:
  driver: bridge
  ipam:
    config:
    - subnet: 10.5.2.0/24
      gateway: 10.5.2.254
```

Caso de uso: example.net VI

- Sample Dockerfile (internal):

```
FROM ubuntu:18.04
RUN apt-get update
RUN apt-get install -y openssh-server net-tools iputils-ping traceroute vim iptables iptables-dev pkg
    -config
RUN mkdir /var/run/sshd
RUN echo 'root:root' |chpasswd
RUN sed -ri 's/^#?PermitRootLogin\s+.*/PermitRootLogin yes/' /etc/ssh/sshd_config
RUN sed -ri 's/UsePAM yes/#UsePAM yes/g' /etc/ssh/sshd_config
RUN mkdir /root/.ssh
RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
EXPOSE 22
COPY ./start.sh .
RUN chmod +x start.sh
CMD ["./start.sh"]
```

Caso de uso: example.net VII

- Sample start script (internal):

```
#!/bin/bash

# Workarround para solventar el error que aparece si en la definición de la red ponemos como gateway
     la IP de máquina que queremos que ejerza como tal (en este caso, fw). No podemos poner IPs
     repetidas en el fichero docker-compose.yml
route add default gw 10.5.2.1
route del default gw 10.5.2.254

/usr/sbin/sshd -D
```

Caso de uso: example.net VIII

- Build and run:

```
$ docker-compose build
...
$ docker-compose up
Starting fw ... done
Starting src_dmz_3      ... done
Starting src_internal_2 ... done
Starting src_internal_3 ... done
Starting src_external_2 ... done
Creating src_dmz_4      ... done
Creating src_dmz_5      ... done
Creating src_external_3 ... done
Creating src_external_4 ... done
Creating src_internal_4 ... done
Attaching to fw, src_dmz_3, src_dmz_5, src_dmz_4, src_external_2, src_external_4, src_external_3,
     src_internal_2, src_internal_3, src_internal_4
```

Caso de uso: example.net IX

- Accessing containers:

```
$ docker-compose ps
     Name          Command     State   Ports
-----------------------------------------------
fw                ./start.sh   Up      22/tcp
src_dmz_3         ./start.sh   Up      22/tcp
src_dmz_4         ./start.sh   Up      22/tcp
src_dmz_5         ./start.sh   Up      22/tcp
src_external_2    ./start.sh   Up      22/tcp
src_external_3    ./start.sh   Up      22/tcp
src_external_4    ./start.sh   Up      22/tcp
src_internal_2    ./start.sh   Up      22/tcp
src_internal_3    ./start.sh   Up      22/tcp
src_internal_4    ./start.sh   Up      22/tcp
$ docker-compose exec fw bash
root@6872c99ea415:/# exit
$ docker-compose exec --index=2 internal bash
root@4ae6c81508bf:/#
```

8. Securización de servidores web. Apache

8.1. Instalación y configuración

Apache HTTP Server I

Apache HTTP Server project...

is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows.

The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

- Install:

```
$ sudo apt update
$ sudo apt-get install apache2
```

- Check status:

```
$ sudo systemctl status apache2
```

```
$ curl http://localhost
```

Apache HTTP Server II

- Firewall settings:

```
$ sudo ufw app list
Available applications:
  Apache           # este perfil abre solo el puerto 80 (tráfico web normal no cifrado)
  Apache Full      # este perfil abre el puerto 80 (tráfico web normal no cifrado) y el puerto 443 (trá
     fico TLS/SSL cifrado)
  Apache Secure # este perfil abre solo el puerto 443 (tráfico TLS/SSL cifrado)
  OpenSSH
```

```
$ sudo ufw allow 'Apache'
$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache (v6)                ALLOW       Anywhere (v6)
```

Apache HTTP Server III

- Stop/start/reload/disable/enable:

```
$ sudo systemctl stop apache2
```

```
$ sudo systemctl start apache2
```

```
$ sudo systemctl restart apache2
```

```
$ sudo systemctl reload apache2 # Si solo se realizan cambios de configuración, Apache puede
    recargarse sin cerrar las conexiones activas.
```

```
$ sudo systemctl disable apache2 # Deshabilitar el servicio
$ sudo systemctl enable apache2  # Habilitarlo de nuevo
```

Virtual hosts I

- The term Virtual Host refers to the practice of running more than one web site (such as `company1.example.com` and `company2.example.com`) on a single machine.
  - Virtual hosts can be “IP-based”, meaning that you have a different IP address for every web site, or “‘
  - name-based”, meaning that you have multiple names running on each IP address.
- Define a Virtual Host:

```
$ sudo mkdir /var/www/your_domain
$ sudo chown -R $USER:$USER /var/www/your_domain
$ sudo chmod -R 755 /var/www/your_domain
```

Virtual hosts II

```
$ sudo vi /var/www/your_domain/index.html
```

```
<html>
    <head>
        <title>Welcome to Your_domain!</title>
    </head>
    <body>
        <h1>Success! The your_domain virtual host is working!</h1>
    </body>
</html>
```

```
$ sudo vi /etc/apache2/sites-available/your_domain.conf
```

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName your_domain              # dominio de base
    ServerAlias www.your_domain         # nombres adicionales
    DocumentRoot /var/www/your_domain
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Virtual hosts III

- Enable site:

```
$ sudo a2ensite your_domain.conf
```

- Disable default site:

```
$ sudo a2dissite 000-default.conf
```

- Check settings:

```
$ sudo apache2ctl configtest
Syntax OK
$ sudo systemctl restart apache2
```

Apache settings I

- Get Apache info:

```
$ apache2ctl -V
```

- Contents: `/var/www/html`
- Settings:
  - `/etc/apache2` ⟶ Directorio con los archivos de configuración de Apache.
  - `/etc/apache2/apache2.conf` ⟶ archivo principal de configuración.
  - `/etc/apache2/ports.conf` ⟶ este archivo especifica los puertos en los que Apache escuchará.
    - Por defecto, Apache escucha en el puerto 80.
    - De forma adicional, lo hace en el 443 cuando se habilita un módulo que proporciona capacidades SSL.
  - `/etc/apache2/sites-available/` y `/etc/apache2/sites-enabled/` ⟶ `a2ensite` y `a2dissite`.
  - `/etc/apache2/mods-available/` y `/etc/apache2/mods-enabled/` ⟶ `a2enmod` y `a2dismod`.

Apache settings II

- Logs:
  - `/var/log/apache2/access.log`
  - `/var/log/apache2/error.log`
- Post-install settings
  - Disable unused modules.
  - Permissions on Apache logs and configuration files:

```
$ chmod -R go-r /etc/apache2
$ chmod -R go-r /var/log/apache2
```

    - Por defecto otros usuarios tienen permisos de lectura sobre los archivos de configuración y de logs de Apache.

8.2. Hardening Apache HTTP Server

Control de los archivos publicados I

- **Deny default access**:

```
<Directory />
  Order Deny,Allow
  Deny from all
  Options None
  AllowOverride None
</Directory>
<Directory /var/www/html>
  Order Allow,Deny
  Allow from all
</Directory>
```

  - Las directivas `Allow` y `Deny` son utilizadas para permitir o denegar respectivamente el acceso a un directorio.
  - `Order` especifica el orden en el que serán evaluadas.
  - La directiva `Directory` sirve para aplicar una serie de directivas sobre la carpeta especificada y a todas sus subcarpetas.
    - Es común que existan varias de estas directivas y que, como también se aplican a los subdirectorios, existan directivas contradictorias aplicadas a un directorio.
    - En este caso, la directiva del `Directory` más específica es la que prevalece.

Control de los archivos publicados II

- **Check Alias directives**:
  - Apache permite mediante la directiva **alias** asignar cualquier tipo de ruta en el sistema de ficheros para que sea accesible desde la web.
  - Se deben revisar estas reglas para verificar que no existan directorios asociados al servicio web que puedan ser mostrados.
- **Links**:

```
Options -FollowSymLinks
Options -FollowSymLinks +SymLinksIfOwnerMatch
```

- **Hide directory contents**:

```
Options -Indexes
```

Control de los archivos publicados III

- **Limit file and directory access**:

```
<Files ~ "^\.ht">
  Order allow,deny
  Deny from all
</Files>
<FilesMatch "(\.bak$|\.BAK$)">
  Order Allow,Deny
  Deny from all
</FilesMatch>
<DirectoryMatch /CVS/>
  Order Allow,Deny
  Deny from all
</DirectoryMatch>
```

- **Server header info** (`/etc/apache2/conf-enabled/security.conf`)

```
ServerSignature Off
ServerTokens Prod
```

Control de los archivos publicados IV

- **Error pages**:

```
ErrorDocument 404 /error/404.html
ErrorDocument 500 "Error en el servidor"
```

- **Security options**:

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
Options -ExecCGI
```

  - **ScriptAlias** especifica el patrón de las URLs que corresponden a scripts y el directorio en el que se encuentran, de manera que los scripts que estén contenidos en ese directorio serán ejecutados.
  - El directorio al que hace referencia ScriptAlias ha de encontrarse fuera del DocumentRoot, sino se podrían mostrar su código fuente accediendo al directorio final sin usar el alias.
  - Para evitar la ejecución de scripts en el DocumentRoot se puede utilizar la directiva **Options**.

Control de los archivos publicados V

- **Authentication**:

```
$ htpasswd -c /usr/local/apache/passwords user
```

.htaccess:

```
<Directory /var/www/htdocs/privado/>
  AuthType Basic AuthName "Zona privada"
  AuthBasicProvider file
  AuthUserFile /usr/local/apache/passwords
  Require valid-user
</Directory>
```

  - Los archivos **.htaccess** contienen directivas de configuración que se aplican al directorio en el que están contenidos dentro del sistema de ficheros.

- **Group authentication**:

```
Require group administradores
Require user jcmengano pcortes
```

Control de los archivos publicados VI

- **Others requirements**:

```
Require ip 192.168.205
```

  - **Require** se ha ampliado, entre otras cosas, permite especificar rangos de IPs autorizadas.
  - **RequireAll** y **RequireNone** permiten la composición de requisitos de acceso.
  - En el siguiente ejemplo, se permite el acceso a todo el mundo excepto si pertenecen a la red 192.168.205.255, su host es “*.phishers.example.com”, “*.moreidiots.example” o “*.ke”:

```
<RequireAll>
  Require all granted
  <RequireNone>
    Require ip 192.168.205
    Require host phishers.example.com moreidiots.example
    Require host ke
  </RequireNone>
</RequireAll>
```

Control de los archivos publicados VII

- En el ejemplo siguiente se exige que los usuarios pertenezcan al grupo alpha o beta y que no estén incluidos en el grupo reject:

```
<Directory /www/docs>
  <RequireAll>
    Require group alpha beta
    Require not group reject
  </RequireAll>
</Directory>
```

Control de los archivos publicados VIII

- **mod_security** y **mod_evasive**:
  - mod_security $\longrightarrow$ Acts as a firewall for web servers and applications, providing protection against brute force attacks.
  - mod_evasive $\longrightarrow$ Detects and provides protection against DDOS and HTTP brute force attacks. It detects attacks whenever:
    - so many requests are directed to a page several times per second;
    - temporarily blacklisted IP still tries to make a new request;
    - child process attempts making more than 50 concurrent requests.

```
$ sudo apt install libapache2-mod-security2
$ sudo apt install libapache2-mod-evasive
$ sudo systemctl restart apache2
```

Enable HTTPS I

1. Enable `mod_ssl`:

```
$ sudo a2enmod ssl
$ sudo systemctl restart apache2
```

2. **Create a certificate** (OpenSSL):

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned
    .key -out /etc/ssl/certs/apache-selfsigned.crt
```

- `openssl` permite crear y administrar certificados y claves.
- `req -x509` → Tipo de certificado X.509 (estándar de infraestructura de claves).
- `-nodes` → omita la protección del certificado con una passphrase. Apache necesita leerlo sin intervención del usuario.
- `-days 365` → tiempo de validez del certificado.
- `-newkey rsa:2048` → se creará una clave RSA de 2048 bits junto con el certificado.
- `-keyout` → ruta del archivo de clave privada generado que estamos creando.
- `-out` → ruta del certificado.

Enable HTTPS II

- Durante el proceso, se solicitan algunos datos:

```
Country Name (2 letter code) [US]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

3. **Configure Apache HTTP Server**:
   - Update Virtual Hosts:

```
$ sudo nano /etc/apache2/sites-available/your_domain.conf
```

```
<VirtualHost *:443>
   ServerName your_domain              # Debe coincidir con el Common Name del certificado
   DocumentRoot /var/www/your_domain

   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
   SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

Enable HTTPS III

- Redirect default HTTP host (`/etc/apache2/sites-available/000-default.conf`) to HTTPS (recommended).

```
<VirtualHost *:80>
    ...
    Redirect "/" "https://your_domain_or_IP/"
    ....
</VirtualHost>
```

4. After reloading Apache, **check** `https://server_domain_or_IP` **in your browser**.
   - Aparecerá una advertencia debido a que el certificado no está firmado por una de las autoridades de certificados de confianza del navegador.

9. Otros servicios: proxy, DHCP, DNS, VPN, LDAP

5 Otros servicios: proxy, DHCP, DNS, VPN, LDAP
- Proxy
- DNS y DHCP
- LDAP
- VPN

9.1. Proxy

Proxy

Servidor proxy

Dispositivo o programa que hace de **intermediario** entre un cliente y un servidor.

Servicios ofrecidos:

- control de acceso,
- registro del tráfico,
- restricción a determinados tipos de tráfico,
- mejora de rendimiento,
- anonimato de la comunicación,
- caché web,
- etc.

Squid I

Squid is a caching and forwarding HTTP web proxy.

- Installing squid:

```
$ sudo apt-get install squid
```

- Configuration (`/etc/squid/squid.conf`):
  - Options for authentication.
  - Access control:

```
acl blacklist src 10.0.0.0/8
http_access deny blacklist
acl blacklist2 src "/etc/squid/blacklist"
http_access deny blacklist2
acl forbidden-urls dst "/etc/squid/forbidden-urls"
http_access deny forbidden-urls
http_access allow all
```

  - Network options:

```
http_proxy 3128
```

Squid II

```
http_proxy 3128 transparent
sudo iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

- Memory cache options:

```
cache_mem 192 MB
```

- Disk cache options:

```
cache_dir ufs /var/spool/squid 40000 16 256
```

- Tracking connections: `/var/log/squid/access.log`

Configuración automática del proxy

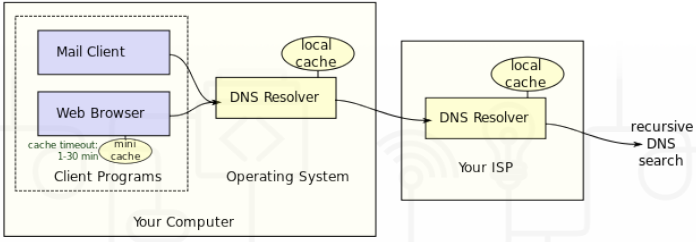
- e.g. `/var/www/proxy.pac`:

```
function FindProxyForURL(url,host){
  if (isInNet(host, "10.0.0.0", "255.0.0.0"))
    return "DIRECT";
  else if (isInNet(host, "127.0.0.1", "255.255.255.255"))
    return "DIRECT";
  else return "PROXY 192.168.1.239:3128";
}
```

- Accessing configuration file: `http://192.168.1.239/proxy.pac`

9.2. DNS y DHCP

DNS



- The **Domain Name System** (**DNS**) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.
- It associates various information with domain names assigned to each of the participating entities.
- Translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.
- In use since 1985.

Bind9 I

Ubuntu ships with **BIND** (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.

- Install DNS server:

```
$ sudo apt install bind9
$ sudo apt install dnsutils
```

  - 3 modos de funcionamiento
    1. **maestro**: DNS para nuestra red local. Atenderá directamente a las peticiones de resolución de direcciones pertenecientes a la red local y reenviará a servidores DNS externos las peticiones del resto de direcciones de Internet.
    2. **esclavo**: Un servidor esclavo actuará como un servidor espejo de un servidor DNS maestro. Permanecerá sincronizado con el maestro. Se utilizan para repartir las peticiones entre varios servidores aunque las modificaciones solo se realicen en el maestro
    3. **caché DNS**: cuando recibe una petición de DNS por parte de un cliente de nuestra red, la trasladará a un DNS maestro que puede estar en nuestra red o fuera, almacenará en una memoria caché la respuesta y a la vez la comunicará a quien hizo la petición.

- Configuración como caché DNS (`/etc/bind/named.conf`):

Bind9 II

```
forwarders {
    1.2.3.4; 5.6.7.8;
};
```

- Configuración como maestro (`/etc/bind/named.conf`):

```
# Archivo para búsquedas directas
zone "example.com" {
  type master;
  file "/etc/bind/example.db";
};
# Archivo para búsquedas inversas
zone "0.168.192.in-addr.arpa" {
  type master;
  file "/etc/bind/192.rev";
};
```

  - `example.db`

Bind9 III

```
;
; BIND data file for example.com
;
@ IN SOA example.com. root.example.com. (
1 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Default TTL

IN NS dns.example.com.     # Name Server
IN MX 10 mail.example.com. # Mail eXchange

pc1 IN A 192.168.0.101
pc2 IN A 192.168.0.102
```

- `/etc/bind/192.rev`

Bind9 IV

```
;
; BIND reverse data file for 192.168.0.0
;
@ IN SOA example.com. root.example.com. (
1 ; Serial
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Default TTL

IN NS dns.example.com.

101 IN PTR pc1.example.com.
102 IN PTR pc2.example.com.
```

- Podemos comprobar la configuración con los comandos `named-checkconf` y `named-checkzone`.
- Debemos indicar a nuestro servidor Linux que el servidor DNS es él mismo (`/etc/resolv.conf`).

```
nameserver 127.0.0.1 # nosotros mismos somos servidores DNS
search example.com # y por defecto buscamos en nuestro dominio
```

Bind9 V

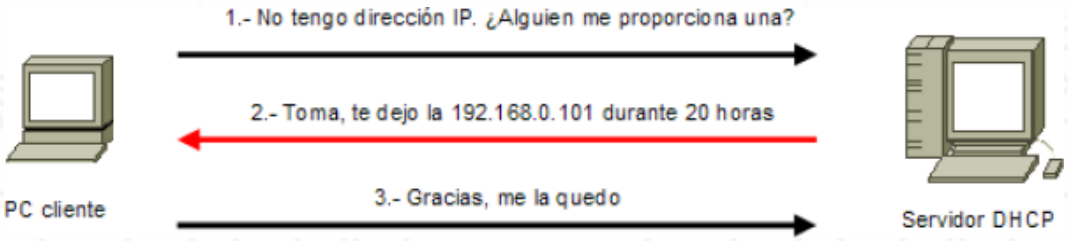
- En el resto de PCs de la red indicamos quién es el DNS (`/etc/resolv.conf`).

```
nameserver 192.168.0.112
```

- Start/Stop DNS server:

```
$ sudo systemctl stop bind9.service
$ sudo systemctl start bind9.service
$ sudo systemctl restart bind9.service
```

DHCP



- The **Dynamic Host Configuration Protocol** (**DHCP**) is a standardized network protocol used on Internet Protocol (IP) networks.
- The DHCP is controlled by a DHCP server that dynamically distributes network configuration parameters, such as IP addresses, for interfaces and services.

isc-dhcp-server I

- Install dhcp server:

```
$ sudo apt-get install isc-dhcp-server
```

  - You need to edit /etc/default/isc-dhcp-server to specify the interfaces dhcpd should listen to.
  - You have to assign a static ip to the interface that you will use for dhcp.

- Configure dhcp server (`/etc/dhcp/dhcpd.conf`):

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.example";

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
range 192.168.1.150 192.168.1.200;
}
```

isc-dhcp-server II

- client IP addresses from the range 192.168.1.10-100 or 192.168.1.150-200.
- It will lease an IP address for 600 seconds if the client doesn't ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds.
- The server will also "advise" the client that it should use 255.255.255.0 as its subnet mask, 192.168.1.255 as its broadcast address, 192.168.1.254 as the router/gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers

- Start/Stop dhcp server:

```
$ sudo service isc-dhcp-server restart
$ sudo service isc-dhcp-server start
$ sudo service isc-dhcp-server stop
```

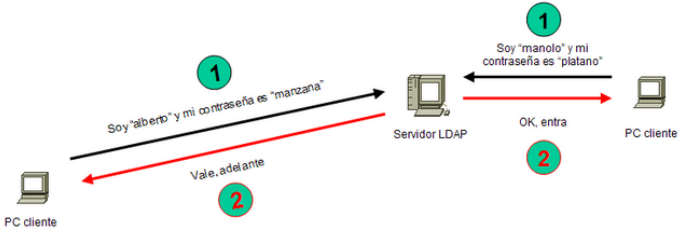
9.3. LDAP

LDAP

Lightweight Directory Access Protocol (LDAP)

Protocolo del nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido.

- Directorio
  - Conjunto de objetos con atributos organizados en una manera lógica y jerárquica (árbol).
  - Habitualmente, almacena la información de autenticación (usuario y contraseña) aunque es posible almacenar otra información (datos de contacto, ubicación, permisos, etc.).
- La versión actual es LDAPv3, y se encuentra definido en el RFC 4511.



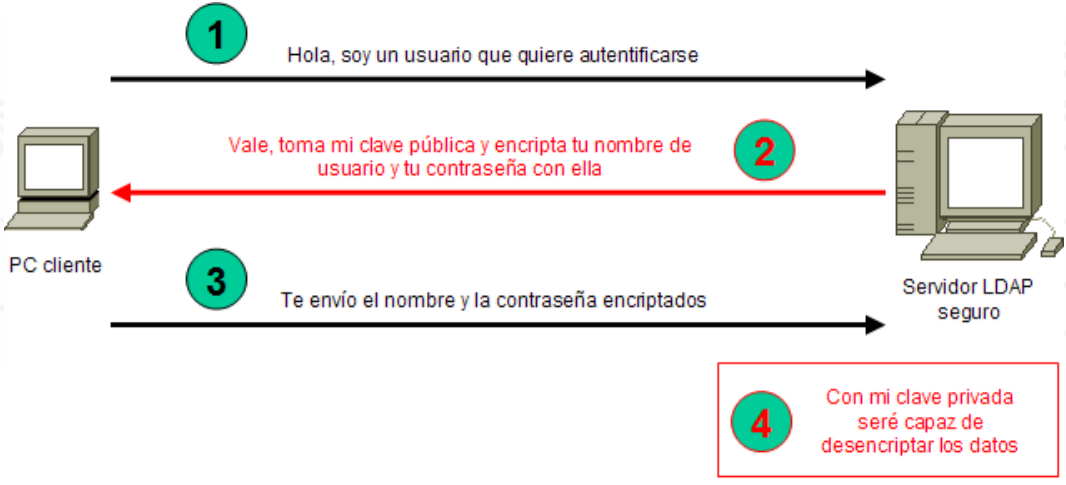
OpenLDAP

- Free, open source implementation of LDAP developed by the OpenLDAP Project.
- It is released under its own BSD-style license called the OpenLDAP Public License.
- Installing OpenLDAP server:

```
$ sudo apt-get install slapd ldap-utils
```

  - During the install you were prompted to define administrative credentials.
  - These are LDAP-based credentials for the rootDN of your database instance.
  - By default, this user's DN is `cn=admin,dc=example,dc=com`.
  - Also by default, there is no administrative account created for the slapd-config database and you will therefore need to authenticate externally to LDAP in order to access it.
  - Some classical schemas (*cosine*, *nis,* inetorgperson) come built-in with slapd nowadays.
  - There is also an included *core* schema, a pre-requisite for any schema to work.
- Post-install: `https://help.ubuntu.com/lts/serverguide/openldap-server.html#openldap-server-postinstall`

OpenSSL + OpenLDAP



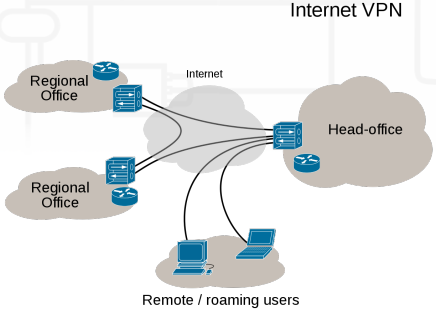
9.4. VPN

VPN

VPN

A **virtual private network** (**VPN**) extends a private network across a public network, and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network.

Applications running across the VPN may therefore benefit from the functionality, security, and management of the private network.



OpenVPN server side I

OpenVPN is a full featured secure network tunneling VPN software solution that integrates server capabilities and client software packages that accommodate Windows, MAC, Linux, Android, and iOS environments.

1. **Installing OpenVPN**:

```
$ sudo apt-get update
$ sudo apt-get install openvpn easy-rsa
```

2. **Create and configure CA directory**:

```
$ make-cadir ~/openvpn-ca
$ cd ~/openvpn-ca
```

OpenVPN server side II

3. **Configure CA variables**:

```
$ vi vars # inside ~/openvpn-ca
```

Edit `vars` contents (do not leave them blank):

```
export KEY_COUNTRY="ES"
export KEY_PROVINCE="LE"
export KEY_CITY="LEON"
export KEY_ORG="Universidad de leon"
export KEY_EMAIL="me@unileon.es"
export KEY_OU="RIASC"
```

```
export KEY_NAME="server"
```

OpenVPN server side III

4. **Build CA**: Run `vars`:

```
$ cd ~/openvpn-ca
$ source vars
```

Output:

```
NOTE: If you run ./clean-all, I will be doing a rm -rf on /home/sammy/openvpn-ca/keys
```

Build:

```
$ ./clean-all
$ ./build-ca
```

Press *Enter* to confirm selections.

OpenVPN server side IV

5. **Create server certificate and keys**:

```
$ ./build-key-server server # KEY_NAME
```

Press *Enter* to confirm default values. Do not enter a challenge password. Answer *y* to the questions for signing and confirm certificate.
Create a Diffie-Hellman key for Key exchange:

```
$ ./build-dh
```

Create an HMAC sign to strengthen TLS integrity checking capabilities

```
$ openvpn --genkey --secret keys/ta.key
```

OpenVPN server side V

6. **Create a client certificate and keys**:
   Without password:

```
$ cd ~/openvpn-ca
$ source vars
$ ./build-key client1
```

   With password:

```
$ cd ~/openvpn-ca
$ source vars
$ ./build-key-pass client1
```

OpenVPN server side VI

7. **Configure OpenVPN service**:
   Move files to `/etc/openvpn`:

```
$ cd ~/openvpn-ca/keys
$ sudo cp ca.crt ca.key server.crt server.key ta.key dh2048.pem /etc/openvpn
```

   Copy sample configuration file:

```
$ gunzip -c /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz | sudo tee /etc/
    openvpn/server.conf
```

OpenVPN server side VII

Edit `/etc/openvpn/server.conf` contents:

```
$ vi /etc/openvpn/server.conf
```

```
tls-auth ta.key 0 # This file is secret
key-direction 0
```

```
cipher AES-128-CBC
auth SHA256
```

```
user nobody
group nogroup
```

OpenVPN server side VIII

8. **Network configuration**:
   Allow forwarding:

```
$ sudo vi /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

```
$ sudo sysctl -p
```

Get public network interface:

```
$ ip route | grep default #
```

OpenVPN server side IX

Adjust UFW rules. Edit `/etc/ufw/before.rules` contents:

```
$ sudo vi /etc/ufw/before.rules
```

```
#
# rules.before
#
# Rules that should be run before the ufw command line added rules. Custom
# rules should be added to one of these chains:
#   ufw-before-input
#   ufw-before-output
#   ufw-before-forward
#

# START OPENVPN RULES
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]
# Allow traffic from OpenVPN client to eth0
-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE
COMMIT
# END OPENVPN RULES

# Don't delete these required lines, otherwise there will be errors
```

OpenVPN server side X

```
*filter
```

Edit `/etc/default/ufw` contents:

```
$ sudo vi /etc/default/ufw
```

```
DEFAULT_FORWARD_POLICY="ACCEPT"
```

OpenVPN server side XI

Open OpenVPN port:

```
$ sudo ufw allow 1194/udp
$ sudo ufw allow OpenSSH
```

Restart UFW:

```
$ sudo ufw disable
$ sudo ufw enable
```

OpenVPN server side XII

9. **Restart OpenSSH server**:

```
$ sudo systemctl start openvpn@server
$ sudo systemctl status openvpn@server
```

enable service to start on reboot:

```
$ sudo systemctl enable openvpn@server
```

OpenVPN server side XIII

10. **Client Configuration Infrastructure**:

```
$ mkdir -p ~/client-configs/files
$ chmod 700 ~/client-configs/files
$ cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf ~/client-configs/base.conf
$ vi ~/client-configs/base.conf
```

Edit `base.conf` contents:

```
remote server_IP_address 1194
proto udp
# Downgrade privileges after initialization (non-Windows only)
user nobody
group nogroup
#ca ca.crt # Comment
#cert client.crt # Comment
#key client.key # Comment
cipher AES-128-CBC # Same as /etc/openvpn/server.conf
auth SHA256
key-direction 1 # New
# script-security 2 # Add commented
# up /etc/openvpn/update-resolv-conf # Add commented
# down /etc/openvpn/update-resolv-conf # Add commented
```

OpenVPN server side XIV

Create a shell script called `make_config.sh` at `/client-configs`:

```
#!/bin/bash
# First argument: Client identifier
KEY_DIR=~/openvpn-ca/keys
OUTPUT_DIR=~/client-configs/files
BASE_CONFIG=~/client-configs/base.conf

cat ${BASE_CONFIG} \
    <(echo -e '<ca>') \
    ${KEY_DIR}/ca.crt \
    <(echo -e '</ca>\n<cert>') \
    ${KEY_DIR}/${1}.crt \
    <(echo -e '</cert>\n<key>') \
    ${KEY_DIR}/${1}.key \
    <(echo -e '</key>\n<tls-auth>') \
    ${KEY_DIR}/ta.key \
    <(echo -e '</tls-auth>') \
    > ${OUTPUT_DIR}/${1}.ovpn
```

Grant appropriate permissions:

```
$ chmod 700 ~/client-configs/make_config.sh
```

OpenVPN server side XV

Create client configuration files:

```
$ cd ~/client-configs
$ ./make_config.sh client1
$ ls ~/client-configs/files
client1.ovpn
```

OpenVPN server side XVI

1. **Configure linux clients**:

```
client$ sftp sammy@openvpn_server_ip:client-configs/files/client1.ovpn ~
client$ sudo apt-get install openvpn
client$ ls /etc/openvpn
update-resolve-conf
client$ vi ~/client1.ovpn
```

Edit `client1.ovpn` contents:

```
script-security 2 # Uncomment
up /etc/openvpn/update-resolv-conf # Uncomment
down /etc/openvpn/update-resolv-conf # Uncomment
```

Run openvpn command:

```
$ sudo openvpn --config client1.ovpn
```

Sistemas confiables

Máster Universitario de Investigación en Ciberseguridad

Ángel Manuel Guerrero Higueras
am.guerrero@unileon.es

GRUPO DE **ROBÓTICA**
Universidad de León