



Reconocimiento-CompartirIgual 3.0  
España (CC BY-SA 3.0 ES)

---

## Introducción a la Ciberseguridad

### Práctica 3.1: Redes e Internet

Marta Beltrán Pardo  
Miguel Calvo Matalobos

Agradecimientos (versiones anteriores): Isaac Martín de Diego y Alberto Fernández Isabel

## Contenidos

---

Contenidos .....	2
1. Introducción.....	3
2. Material de la práctica.....	4
3. Normativa y evaluación .....	5
4. Enunciado de la práctica.....	6
4.1 Programar un escáner de puertos .....	6
Ejercicio 1 .....	6
Ejercicio 2 .....	8
Ejercicio 4 .....	11
Ejercicio 5 .....	12
4.2 Montar una máquina virtual con LAMP.....	14
Ejercicio 6 .....	15
Ejercicio 7 .....	18
5. Referencias de interés.....	20
6. Anexo I .....	21

## 1. Introducción

---

En la tercera práctica de la asignatura vamos a realizar varias tareas relacionadas con las Redes e Internet. Esta práctica, se divide en dos partes ([Practica3.1\\_Guion.pdf](#) y [Practica3.2\\_Guion.pdf](#)):

- En la primera parte de la práctica ([Practica3.1\\_Guion.pdf](#)), programaremos un escáner de puertos en C. Posteriormente montaremos una máquina virtual con LAMP (Linux + Apache + MySQL + PHP) para probar nuestro escáner de puertos contra un servidor controlado.
- En la segunda parte ([Practica3.2\\_Guion.pdf](#)), montaremos nuestra primera página web con HTML y PHP para añadir una zona privada empleando una base de datos MySQL. El objetivo de esta segunda parte será entender la diferencia entre código estático y dinámico. Además, probaremos nuestra primera inyección SQL para poder acceder a la zona privada sin tener cuenta de usuario.

## 2. Material de la práctica

---

A continuación, se exponen los archivos necesarios para la realización de esta práctica, que pueden descargarse desde el Aula Virtual:

- **Practica3.1\_Guion.pdf:** este documento. Se corresponde con el guion de la primera parte de la práctica y en él están contenidas todas las explicaciones necesarias para su realización.
- **Practica3.1\_Myportscanner.c:** pequeño esquema o estructura del escáner de puertos para que no haya que partir de cero en su programación durante la práctica.

Además, será necesario descargar el programa **VirtualBox** en su última versión. Este programa, servirá para crear la máquina virtual que se utilizará a lo largo del desarrollo de esta práctica. Puede descargarse desde el [siguiente enlace](#). También será necesario el pack de extensiones “Oracle VM VirtualBox”, que permitirá la configuración y el uso de ciertos parámetros y características en las máquinas virtuales (puede descargarse [aquí](#)).

Para el desarrollo de esta práctica, hará falta también la descarga de una distribución de un sistema operativo Linux. Puede utilizarse la [máquina virtual “Kali Linux”](#) utilizada en prácticas anteriores o una máquina virtual creada a partir de una ISO de [“Kali Linux”](#), [“Ubuntu”](#) o cualquier otra distribución de Linux que conozcas. Este sistema operativo será el utilizado a lo largo de esta práctica.

### 3. Normativa y evaluación

---

En este apartado se detalla el formato de entrega de la práctica y la forma en la que se evaluará la misma:

- El porcentaje de la nota final de la asignatura al que corresponde esta práctica puede consultarse en la Guía docente de la propia asignatura.
- La práctica deberá realizarse, de forma obligatoria, en grupos de dos personas. Para la asignación de los grupos se deberán seguir las indicaciones del profesor.
- Cada grupo deberá:
  - Realizar una única memoria (puede utilizarse esta misma guía como plantilla) en la que responda las preguntas planteadas y/o en la que se exponga, de forma argumentada, las decisiones tomadas para la realización de la práctica.
  - Desarrollar y/o modificar tantos archivos de código como se soliciten en los diferentes ejercicios que forman la práctica.
- El resultado de la realización de la práctica consistirá en un fichero .zip llamado **Practica3.zip** que deberá entregarse a través del Aula Virtual en el espacio habilitado para ello y en la fecha límite allí expuesta. Este fichero debe contener:
  - La memoria completa de la práctica 3 en formato PDF. En ella debe indicarse, en la primera página y de forma clara, el nombre y apellidos de los integrantes del grupo.
  - Los archivos de código resultantes de la realización de los ejercicios solicitados, de momento (habrá más asociados a la segunda parte de la práctica, se indicará en el siguiente guion):
    - Practica3\_myportscanner\_resuelto.c

Recuerda lo que has aprendido con la práctica 1 acerca de la redacción de memorias de prácticas.

## 4. Enunciado de la práctica

---

En este apartado se describirán las distintas actividades, programas y ejercicios que deberá realizar cada grupo de prácticas, así como la información a completar y/o rellenar en la memoria.

### 4.1 Programar un escáner de puertos

En la primera parte de la práctica, vamos a programar un sencillo escáner de puertos en el lenguaje de programación C.

#### Ejercicio 1

Un **puerto** identifica a una aplicación que recurre a la capa de transporte para conectarse con la red. Por ejemplo, los programas de correo electrónico POP3, como Outlook, envían y reciben correos electrónicos a través de puertos específicos, 110 y 995 para recibir correos electrónicos, 25, 2525 y 443 para enviar correos electrónicos, y puertos 143 y 993 para conectarse a servidores IMAP. El modelo OSI (*Open System Interconnection*) es el encargado de definir estos puertos y su uso. Los números de puerto se indican mediante 16 bits (por tanto, existen  $2^{16} = 65536$  puertos). La entidad IANA (*Internet Assigned Numbers Authority*) es la encargada de asignar a cada puerto su número específico. En la Tabla 1 puede encontrarse una pequeña muestra de puertos utilizados habitualmente.

**Pregunta 1.1.** Empleando como referencia al organismo IANA, averigua el nombre del puerto 993, su protocolo de transporte, su descripción y la fecha de la última modificación en su definición.

Nombre	Puerto	Comentario
ftp	20	FTP - data
frp	21	FTP – control
ssh	22	SSH Remote Login Protocol
telnet	23	TELNET
smtp	25	Simple Mail Transfer Protocol
domain	53	Domain Name Server
bootps	67	Bootstrap Protocol – server
bootpc	68	Bootstrap Protocol – client
tftp	69	Trivial File Transfer
http	80	World Wide Web
pop3	110	Post Office Protocol – version 3
sunprc	111	SUN Remote Procedure Call
netbios-ssn	139	NETBIOS Session Service (SMB)
imap	143	Internet Message Access Protocol
snmp	161	Simple Network Management Protocol
bgp	179	Border Gateway Protocol
irc	194	Internet Relay Chat Protocol
ldap	389	Lightweight Directory Access Protocol
https	443	Http secure
ipp	631	Internet Printing Protocol
wins	1512	Windows Internet Name Service
nfsd	2049	NFS server
squid	3128	Squid Web Proxy
mysql	3306	MySQL

*Tabla 1. Puertos utilizados habitualmente.*

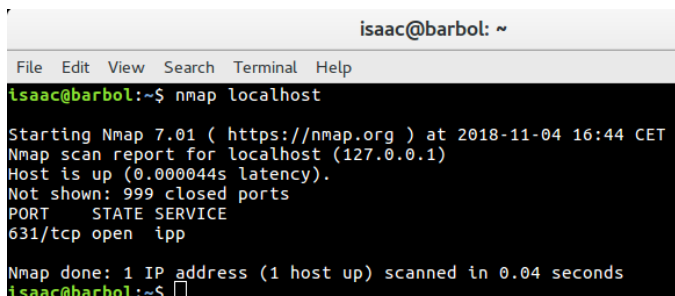
## Ejercicio 2

Por otro lado, un **escáner de puertos** es un programa que sirve para examinar los puertos de un equipo o servidor red en busca de uno de los tres estados posibles: abierto, cerrado o filtrado. Estas herramientas son muy útiles para los administradores en relación con el diagnóstico de problemas de red y la conectividad, sin embargo, también suelen utilizarse por los atacantes para, por ejemplo, detectar posibles puntos de acceso a una máquina y/o identificar que dispositivos se están conectados y a la escucha en una red (como firewalls, proxis, servidores VPN, etc.).

Como ejemplo de escáner de puertos, podemos encontrar el software “**nmap**”. Esta herramienta, disponible tanto para Linux como para Windows, es una de las más conocidas y utilizadas tanto por administradores de sistemas como por atacantes. Será necesaria su instalación para la realización de este ejercicio.

- Windows: Descargar desde [este](#) enlace.
- Linux: Instalar mediante el comando `sudo apt-get install nmap`.

Un ejemplo de uso de esta herramienta es mostrar los puertos abiertos en local (mediante el comando `nmap localhost`). En la *Figura 1* puede observarse que el único puerto abierto en local (o lo que es lo mismo, que no necesariamente sale a Internet o es visible desde Internet), es el 631 (correspondiente al IIP).



```
isaac@barbol: ~  
File Edit View Search Terminal Help  
isaac@barbol:~$ nmap localhost  
Starting Nmap 7.01 ( https://nmap.org ) at 2018-11-04 16:44 CET  
Nmap scan report for localhost (127.0.0.1)  
Host is up (0.000044s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
631/tcp  open  iip  
Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds  
isaac@barbol:~$
```

*Figura 1. Mostrando los puertos locales con nmap.*



**Pregunta 2.1.** ¿Qué es el localhost? ¿A qué dirección IP corresponde este nombre? ¿Qué es un *loopback*?

### Ejercicio 3

Además de ver los puertos locales de nuestra máquina, también pueden observarse los que están abiertos desde Internet. Para ello, también puede utilizarse la herramienta “nmap”, pero esta vez con la dirección IP asignada a nuestro equipo.

Existen diferentes formas de averiguar la dirección IP, por ejemplo, en Linux, mediante “ifconfig” (esta herramienta no está instalada por defecto en todas las distribuciones Linux. Para instalarla, puede utilizarse el comando `sudo apt-get install net-tools`).

Lo más probable es que tengas asignada una dirección privada dentro de tu red doméstica o del aula (compruébalo con el comando `ifconfig`). Tras averiguar la dirección IP asignada a tu máquina, prueba a realizar un escaneo con “nmap” (`nmap IP-de-tu-maquina`). Puedes escanear también la dirección privada de tu router y la pública, a ver qué diferencias observas.

Cuidado, no lances escaneos de puertos indiscriminadamente contra direcciones IP reales públicas. Ese proceso puede tardar bastante tiempo en completarse y generará un tráfico considerable en la red. Además, según la legislación del país, el escaneo de puertos puede ser **ilegal**, ya que es posible que degrade el rendimiento de la máquina escaneada.

**Pregunta 3.1.** Menciona alguna página web que puedas consultar para averiguar y geolocalizar la dirección IP pública de tu router doméstico. Saca conclusiones acerca de los resultados de tus escaneos, tanto del ordenador como del router.

## Ejercicio 4

El escaneo de puertos puede darte mucha información (puertos abiertos, protegidos tras un firewall, sistemas operativos instalados, servicios corriendo, etc.), “nmap” es una herramienta muy potente y con muchas opciones.

SSH es un acrónimo de *Secure Shell*. Este protocolo de red permite que dos ordenadores se comuniquen de forma segura y puedan compartir datos, información o incluso realizar operaciones desde una máquina en otra. Una característica inherente de SSH es que la comunicación está cifrada. Para realizar estas acciones, se necesita un cliente (cliente SSH; por ejemplo, instalado en nuestro equipo), que será el encargado de conectarse al servicio (servidor SSH) y, de esta forma transferir los datos hacia/desde un equipo a otro utilizando una interfaz gráfica o la línea de comandos (dependiendo del cliente SSH elegido).

En tu máquina virtual con SO Linux, instala, desde la línea de comandos, un servidor SSH. Para ello, desde la terminal, ejecuta los siguientes comandos:

- `sudo apt-get install openssh-server`
- `sudo systemctl enable ssh`
- `sudo systemctl start ssh`

**Pregunta 4.1.** Con ayuda del manual de nmap (`man nmap`), investiga que comando o comandos necesitarías para extraer la versión utilizada en el servicio SSH de tu máquina virtual. ¿Qué comando te ha servido para realizar esta tarea? ¿Qué información, aparte de la versión del servicio SSH, has conseguido extraer?

## Ejercicio 5

Antes de comenzar a realizar este ejercicio, es necesario revisar el código proporcionado junto al enunciado ("Practica3\_myportscanner.c"). Se trata de que te familiarices con el código y comprendas como se ejecutan las diferentes instrucciones, que finalidad tiene el programa o su estructura, etc.

Para programar el escáner de puertos, se proporciona, junto a este documento, un esquema del código en C que debe completarse ("**Practica3\_myportscanner.c**"). Este código NO COMPILA; en él, existen 10 huecos que deben rellenarse (indicados por XXXi, donde "i" es un indicador de 1 hasta 10). No es necesario utilizar el esquema proporcionado si no se quiere, puedes programar tu propio escáner desde cero, siempre y cuando lo justifiques y expliques convenientemente.

**Pregunta 5.1.** Podemos decir, de modo muy simple, que un socket es una manera de hablar con otro computador usando descriptores de ficheros estándar en Linux. Para inicializar un socket se emplea el siguiente formato:

```
int sockfd = socket(int familia, int tipo, int protocolo)
```

¿Qué significado tienen los valores de entrada de la función socket "familia", "tipo" y "protocolo"? Identifica esta llamada en el código proporcionado. Explica los valores concretos que se emplean en el código. ¿Qué significado tiene cuando la función devuelve un -1? ¿Qué significa si devuelve un valor distinto de -1?

**Pregunta 5.2.** Completa el código para hacer funcional tu escáner de puertos. Una vez completado el código en C, compílalo (`gcc Practica3_myportscanner.c -o myportscanner`), Pruébalo (`./myportscanner`) y comenta los resultados. Compara con los que obtuviste con nmap (obviamente nuestro escáner es mucho más sencillo, pero por lo menos compara que detecta los mismos estados para los puertos o que si encuentras diferencias las puedes explicar).

## 4.2 Montar una máquina virtual con LAMP

**LAMP** (Linux, Apache, MySQL, PHP) es un conjunto de software de código abierto que se instala normalmente en conjunto para habilitar un servidor para alojar sitios y aplicaciones web dinámicas. Esta plataforma, utiliza el sistema operativo Linux como base. En él, se despliega Apache como servidor web, MySQL como gestor de base de datos relacionales y PHP como lenguaje de programación. Existen otras pilas similares como WAMP (Windows, Apache, MySQL, PHP), MAMP (Mac OS, Apache, MySQL, PHP) o XAMPP (Windows/Linux/Solaris/MacOS, Apache, MySQL, PHP, PERL).

Como ya se montó una [máquina virtual con “Kali Linux”](#) en prácticas anteriores, puede utilizarse esta máquina para el despliegue de LAMP. Otra opción es instalar una nueva máquina virtual con sistema operativo Linux (creada a partir de una ISO de [“Kali Linux”](#), [“Ubuntu”](#) o cualquier otra distribución Linux que conozcas), siendo esta nueva máquina uno de nuestros servidores vulnerables desde ahora (independiente de la máquina “Kali Linux”). Una vez instalado y configurado LAMP, esta parte de la práctica consiste en probar el escáner puertos, previamente programado (“Programar un escáner de puertos”), desde la máquina virtual con “Kali Linux”.

**NOTA:** Para habilitar y configurar Internet en “Kali Linux”, una vez instada la máquina virtual, debemos dirigirnos a la pestaña de Configuración en VirtualBox. Una vez allí, elegimos el apartado “Red”, donde se seleccionará la opción “Conectad a adaptador puente” y se elegirá el adaptador de red que haya conectado a Internet (por ejemplo, el adaptador Wifi). Desplegando la opción “Avanzadas”, se debe elegir “permitir todo en modo promiscuo”. Tras aceptar, se debe iniciar “Kali Linux” y probar la nueva configuración (por ejemplo, comprobando que puede navegarse por la web).

## Ejercicio 6

En este ejercicio, debes crear un servidor LAMP en una máquina virtual con Linux. Para ello, sigue los siguientes pasos:

- Actualiza los paquetes y librerías instalados en tu máquina Linux:
  - `sudo apt-get update`
  - `sudo apt-get upgrade`
- Instala Apache2 e inícialo:
  - `sudo apt-get install apache2`
  - `sudo /etc/init.d/apache2 start` ó `sudo service apache2 start` (para pararlo se puede utilizar la opción “stop” y para reiniciarlo, “restart”).
  - `sudo /etc/init.d/apache2 status` ó `sudo service apache2 status` (para comprobar el estado del servicio).
- Instala MySQL o MariaDB e inícialo:
  - **NOTA:** Dependiendo del SO en el que estés haciendo la instalación del motor de base de datos, puede que no sea MySQL sino MariaDB. Todo es muy similar, busca un manual específico si te surgen dudas con la sintaxis. Por lo tanto, si no funciona la opción de MySQL, prueba con MariaDB.
  - `sudo apt-get install mysql-server` ó `sudo apt-get install mariadb-server`
  - `sudo /etc/init.d/mysql start` ó `sudo service mysql start` (para pararlo se puede utilizar la opción “stop” y para reiniciarlo, “restart”).
  - `mysql_secure_installation` (Enter + n + y + Contraseña que queramos).

- Instala PHP, algunas utilidades e inícialo:
  - `sudo apt-get install php libapache2-mod-php php-mysql php-cgi php-curl`
  - `sudo /etc/init.d/apache2 restart` ó `sudo service apache2 restart` (se reinicia el servicio “Apache2” para que la configuración incluya la instalación de PHP).
- Comprueba si está funcionando:
  - `cd /var/www/html`
  - `nano info.php`
    - Escribir en el fichero: `<?php phpinfo(); ?>`
    - Guardar: Ctrl+o (+ Enter).
    - Salir: Ctrl+x (+ Enter).
  - Abre el navegador y visita la página <http://localhost/info.php>. Si ves algo similar a lo que se muestra en la Figura 2, has tenido éxito en la instalación de LAMP.



PHP Version 7.4.21	
System	Linux kali 5.10.0-kali9-686-pae #1 SMP Debian 5.10.46-1kali1 (2021-06-25) i686
Build Date	Jul 2 2021 03:59:48
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqld.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-curl.ini, /etc/php/7.4/apache2/conf.d/20-dom.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-fdi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-simplexml.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xmlreader.ini, /etc/php/7.4/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.4/apache2/conf.d/20-xsl.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled

Figura 2. Comprobación de la instalación de LAMP.

RECUERDA ARRANCAR LOS SERVICIOS (Apache2 y MySQL) CADA VEZ QUE INICIES/REINICIES LA MÁQUINA.

**Pregunta 6.1.** Una vez instalado y configurado LAMP, muestra, mediante un pantallazo, el resultado de acceder a la página <http://localhost/info.php>.

--

## Ejercicio 7

Tras finalizar con la instalación y configuración de LAMP, debes crear una base de datos (BBDD) sencilla y un usuario de MySQL con las características que se muestran en la Tabla 2.

<b>Nombre de la BBDD</b>	<i>accesos</i>
<b>Nombre de la tabla</b>	<i>user_pass</i>
<b>Campos de la tabla</b>	<i>user; password</i>
<b>Usuario</b>	<i>app</i>
<b>Permisos del usuario</b>	<i>ALL PRIVILEGES (sobre la tabla “user_pass” de la BBDD “accesos”)</i>

Tabla 2. Características solicitadas para la base de datos.

### AYUDA:

- Para acceder a MySQL: `mysql -u root -p`
- Para crear una base de datos: `CREATE DATABASE nombre_de_la_bbdd;`
- Para usar una base de datos: `USE nombre_de_la_bbdd;`
- Para crear una tabla en la base de datos: `CREATE TABLE nombre_de_la_tabla (campo1 tipo_campo1, campo2 tipo_campo2, campo3 tipo_campo3, ...);`
- Para crear un nuevo usuario: `CREATE USER 'nombre-usuario'@'localhost' IDENTIFIED BY 'password-usuario';`
- Para dar permisos a un usuario: `GRANT TIPO-DE-PERMISO ON nombreBBDD.nombreTabla TO 'nombre-usuario'@'localhost';`
- Para aplicar los cambios de permisos de un usuario: `FLUSH PRIVILEGES;`
- Para salir de MySQL: `exit`

**Pregunta 7.1.** Una vez creada la base de datos y la tabla con los campos requeridos (véase la Tabla 2), inserta mediante comandos (con “INSERT INTO”), dos usuarios con sus correspondientes contraseñas. ¿Qué comando o comandos has utilizado?

## 5. Referencias de interés

---

How to Install Ubuntu 20.04 on VirtualBox - [https://linuxhint.com/install\\_ubuntu\\_virtualbox\\_2004/](https://linuxhint.com/install_ubuntu_virtualbox_2004/)

Cómo instalar la pila Linux, Apache, MySQL y PHP (LAMP) en Ubuntu 20.04 - <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04-es>

Instalar LAMPP *Stack* en Ubuntu 20.04 - <https://www.librebyte.net/series/instalar-lampp-stack-en-ubuntu-20-04/>

## 6. Anexo I

En este apartado se incluyen los códigos, scripts y útiles necesarios para la realización de los ejercicios de esta práctica.

### Practica3.1\_myportscanner.c

```
#include "stdio.h"
#include "sys/socket.h"
#include "errno.h"
#include "netdb.h"
#include "string.h"
#include "stdlib.h"

// Cuidado, está diseñado para Linux. Si se quiere compilar para ejecutar en sistemas Windows, las librerías
// y funciones de sockets cambian un poco.
int main(int argc , char **argv)
{
    struct hostent *host;
    int err, port , connection ,start , end;
    char hostname[100];
    struct sockaddr_in sa;

    //Host que se desea escanear
    printf("Introduce hostname o dirección IP : ");
    scanf("%s", &XXX1);

    //Puerto de inicio para el escaneo
    printf("\nPuerto de inicio: ");
    scanf("%d" , XXX2);

    //Puerto final para el escaneo
    XXX3
    XXX4

    //Inicializar la estructura sockaddr_in
    strncpy((char*)&sa , "" , sizeof sa);
    sa.sin_family = XXX5;

    //Dirección IP
    if(isdigit(hostname[0]))
    {
        printf("Doing inet_addr...");
        sa.sin_addr.s_addr = inet_addr(hostname);
        printf("Hecho\n");
    }
    //Resolución de nombre a IP
    else if( (host = gethostbyname(hostname)) != 0)
    {
        printf("Doing gethostbyname...");
        strncpy((char*)&sa.sin_addr , (char*)host -> h_addr , sizeof sa.sin_addr);
        printf("Hecho\n");
    }
}
```

```
else
{
    perror(hostname);
    exit(2);
}

//Bucle de escaneo de puertos
printf("Comienza el escaneo de puertos... \n");
for(XXX6)
{
    //Convertir el identificador de puerto
    sa.sin_port = htons(port);
    //Crear el socket
    connection = XXX7(AF_INET , SOCK_STREAM , 0);

    //Comprobar si el socket se ha creado correctamente
    if(connection < 0)
    {
        perror("\nSocket");
        exit(1);
    }
    //Conectarse al socket creado
    err = connect(XXX8 , (struct sockaddr*)&sa , sizeof sa);

    //Si no se consigue realizar la conexión
    if( err < XXX9 )
    {
        //printf("%s %-5d %s\n" , hostname , i, strerror(errno));
        fflush(stdout);
    }
    //Si sí se consigue realizar la conexión
    else
    {
        printf("%-5d open\n", port);
    }

    //Cerramos el socket
    close(XXX10);
}

printf("\n");
fflush(stdout);
return(0);
}
```