

AN5152

UD1. Fundamentos de seguridad web y análisis de riesgos

Ciberseguridad en el Desarrollo de Aplicaciones · DAW

Duración	Teoría	Prácticas	Enfoque
5 horas	2 horas	3 horas	Aplicado y orientado al desarrollo web

Finalidad de la unidad

Adquirir el vocabulario y los conceptos mínimos para analizar la seguridad de una aplicación web y aplicarlos desde el primer momento sobre entornos educativos reales: OWASP Juice Shop y PortSwigger Web Security Academy.

1. Resultados de aprendizaje y criterios de evaluación

La unidad se centra fundamentalmente en el RA1. Además, se incorpora de forma transversal RA4.ce1 por su utilidad profesional para el alumnado de DAW.

RA	Resultado de aprendizaje	CE	Criterio de evaluación
RA1	Identifica vulnerabilidades comunes en aplicaciones y propone soluciones.	RA1.ce1 (a)	Se ha analizado ejemplos reales de vulnerabilidades como inyecciones SQL.
RA1	Identifica vulnerabilidades comunes en aplicaciones y propone soluciones.	RA1.ce3 (c)	Se ha clasificado vulnerabilidades según su severidad.
RA1	Identifica vulnerabilidades comunes en aplicaciones y propone soluciones.	RA1.ce4 (d)	Se ha propuesto medidas correctivas específicas.
RA1	Identifica vulnerabilidades comunes en aplicaciones y propone soluciones.	RA1.ce5 (e)	Se ha documentado riesgos asociados a vulnerabilidades detectadas.
RA4*	Implementa prácticas seguras en el desarrollo de aplicaciones multiplataforma.	RA4.ce1 (a)*	Se ha analizado el impacto de no seguir buenas prácticas.

* RA4 y RA4.ce1 son elementos propuestos seleccionados por su especial utilidad para un desarrollador DAW.

RA1.ce2 (revisión manual de código fuente) se trabajará en unidades posteriores, cuando el alumnado disponga de conocimientos suficientes para analizar código vulnerable con mayor profundidad y sobre todo en la fase de EMPRESAS.

2. Índice de contenidos

- Introducción a la seguridad de aplicaciones web: activos, amenazas, vulnerabilidades, ataques, impacto y riesgo.
- Principios fundamentales: confidencialidad, integridad, disponibilidad, autenticación, autorización y mínimo privilegio.
- Superficie de ataque de una aplicación web: frontend, backend, bases de datos, APIs, formularios, parámetros, cookies, sesiones, archivos, dependencias y servicios externos.
- Introducción al análisis de riesgos: probabilidad, impacto, severidad, priorización e introducción a NIST.
- Laboratorios de seguridad: OWASP Juice Shop, PortSwigger Web Security Academy, DevTools e identificación inicial de vulnerabilidades.

Prácticas

- P1. Instalación, exploración y análisis de OWASP Juice Shop.
- P2. Score Board e identificación de vulnerabilidades en Juice Shop.
- P3. Introducción a PortSwigger Web Security Academy y primer laboratorio de SQL Injection.

3. Teoría

3.1. ¿Qué significa que una aplicación web sea segura?

Una aplicación web segura no es simplemente una aplicación que «no ha sido hackeada». La seguridad consiste en identificar qué debemos proteger, determinar qué podría ocurrir, conocer las debilidades del sistema y aplicar medidas que reduzcan el riesgo hasta un nivel aceptable.

En una tienda online, por ejemplo, existen usuarios, contraseñas, direcciones, productos, precios, pedidos, información de pago, sesiones y roles. Algunos problemas posibles serían el acceso a pedidos ajenos, el robo de credenciales, la modificación de precios, el acceso como administrador o la extracción de la base de datos.

Cadena conceptual del curso

ACTIVO → AMENAZA → VULNERABILIDAD → ATAQUE → IMPACTO → RIESGO → MITIGACIÓN

3.2. Activos

Un activo es cualquier elemento que tenga valor para la organización y que, por tanto, necesitemos proteger. No todos los activos tienen la misma importancia; por ello la seguridad debe priorizarse en función del riesgo.

Activo	Ejemplo
Datos personales	Nombre, DNI, dirección
Credenciales	Usuarios y contraseñas
Sesiones	Cookies y tokens
Información empresarial	Pedidos, precios y facturas
Base de datos	Usuarios, productos y pedidos
Código fuente	Repositorio del proyecto
Secretos	Claves API y certificados
Infraestructura	Servidores y servicios
Disponibilidad	Aplicación accesible y operativa

3.3. Amenaza, vulnerabilidad y ataque

Una amenaza es una circunstancia, persona o evento que potencialmente puede provocar un daño: un atacante externo, un usuario malicioso, un bot, malware o incluso un error humano.

Una vulnerabilidad es una debilidad susceptible de ser aprovechada. Por ejemplo:

```
SELECT *  
FROM usuarios  
WHERE email='$email';
```

Si el valor de email procede directamente del usuario y se concatena en la consulta, podría existir una vulnerabilidad de SQL Injection.

```
GET /api/pedidos/123
```

El usuario cambia 123 por 124 y obtiene el pedido de otra persona.

En el segundo caso existiría un problema de control de acceso. El ataque o explotación es la acción mediante la cual se intenta aprovechar la vulnerabilidad.

Idea clave

Una vulnerabilidad puede existir aunque todavía nadie la haya explotado. Que un sistema no haya sufrido un incidente conocido no significa que sea seguro.

3.4. La tríada CIA

Tres propiedades fundamentales de la seguridad de la información son Confidentiality, Integrity y Availability: confidencialidad, integridad y disponibilidad.

Confidencialidad

La información solo debe ser accesible para las personas autorizadas. Si cambiar /pedidos/853 por /pedidos/854 permite ver el pedido de otra persona, se compromete la confidencialidad.

Integridad

Los datos solo deben poder modificarse de manera autorizada. Si el navegador envía un precio y el servidor lo acepta sin comprobarlo, un usuario podría alterarlo.

```
{  
  "producto": 152,  
  "precio": 1.99  
}
```

Regla para desarrolladores

Nunca debemos confiar en el navegador para tomar decisiones de seguridad. El navegador está bajo control del usuario y sus peticiones pueden manipularse.

Disponibilidad

Los sistemas y la información deben estar disponibles cuando sean necesarios. Una funcionalidad que permita consumir recursos de forma desproporcionada puede convertirse en un problema de disponibilidad porque consume todos los recursos y otros usuarios no pueden acceder a esos recursos (Denegación de servicios)

3.5. Autenticación y autorización

Autenticación responde a «¿quién eres?». Autorización responde a «ya sé quién eres, ¿qué puedes hacer?». Son procesos distintos.

```
CLIENTE → consultar SUS pedidos  
ADMINISTRADOR → consultar TODOS los pedidos
```

Si un usuario autenticado solicita GET /api/pedidos/154, cambia 154 por 155 y obtiene el pedido de otro usuario, la autenticación puede haber funcionado correctamente. El fallo está en la autorización del servidor.

3.6. Principio de mínimo privilegio

Un usuario, proceso o servicio debería disponer únicamente de los permisos que necesita para realizar su función. Un cliente puede comprar y consultar sus pedidos, pero no debería eliminar usuarios, modificar roles ni consultar todos los pedidos.

3.7. Superficie de ataque

La superficie de ataque es el conjunto de puntos por los que un atacante puede intentar interactuar con el sistema. Una aplicación DAW moderna incluye frontend, backend, base de datos, APIs y servicios externos.



La aplicación puede recibir información mediante URL, parámetros GET, formularios POST, JSON, cabeceras HTTP, cookies, tokens, archivos, APIs o WebSockets. Todo dato procedente del exterior debe considerarse inicialmente NO confiable.

3.8. Riesgo

Encontrar una vulnerabilidad no es suficiente: debemos determinar su importancia. En esta unidad utilizaremos una aproximación sencilla basada en probabilidad e impacto.

	Impacto bajo	Impacto medio	Impacto alto
Probabilidad baja	Bajo	Bajo	Medio
Probabilidad media	Bajo	Medio	Alto
Probabilidad alta	Medio	Alto	Crítico

Ejemplo: si modificar una URL permite descargar fácilmente las facturas de cualquier cliente, la probabilidad puede ser alta y el impacto alto; por tanto, el riesgo sería crítico. Las vulnerabilidades se corrigen priorizando el riesgo, no simplemente en el orden en que se descubren.

3.9. ¿Qué es NIST?

NIST (National Institute of Standards and Technology) es una agencia del Departamento de Comercio de Estados Unidos y una referencia internacional en estándares, marcos y guías de ciberseguridad. Para DAW no es necesario estudiar sus documentos en profundidad; interesa comprender que la seguridad profesional sigue procesos sistemáticos.

Función NIST CSF 2.0	Pregunta simplificada
GOVERN	¿Cómo vamos a gestionar la seguridad?
IDENTIFY	¿Qué tenemos y qué riesgos existen?
PROTECT	¿Cómo lo protegemos?
DETECT	¿Cómo sabemos que está ocurriendo algo?
RESPOND	¿Qué hacemos ante un incidente?

RECOVER

¿Cómo recuperamos el servicio?

Aplicación en el módulo

IDENTIFICAR vulnerabilidad → VALORAR riesgo → PROTEGER mediante una corrección → COMPROBAR repitiendo la prueba.

4. Prácticas (3 horas)

La primera sesión práctica busca que el alumnado deje preparado su entorno de trabajo y conozca dos plataformas que se reutilizarán durante el módulo. Las pruebas deben realizarse exclusivamente contra laboratorios educativos autorizados.

P1. Instalar DOCKER

Duración aproximada: 30 minutos

- Puedes descargar DOCKER DESKTOP para windows con su modo grafico... Opción más pesada.
- Puedes instalar las WSL2 con Ubuntu que es lo que vamos a hacer.

1.- Abre POWER SHELL en modo Administrador

```
wsl --version
```

Para ver si está instalado

Caso A. SI LO TIENE INSTALADO

Abre **PowerShell** como **administrador**:

```
wsl -install -d ubuntu
```

Reinicia Windows.

Después abre **Ubuntu** desde Inicio. La primera vez pedirá:

```
Enter new UNIX username:  
New password:
```

Crear, por ejemplo:

```
usuario: alumno
```

Después, desde **PowerShell**, comprobar:

```
wsl -l -v
```

Debe aparecer:

NAME	STATE	VERSION
Ubuntu	Running	2

Si pone `VERSION 2`, ya está preparado para instalar Docker Engine.

Caso B — WSL/Ubuntu YA estaba instalado

Antes de hacer nada, abre **PowerShell como administrador** y ejecuta:

```
wsl --update
```

Después:

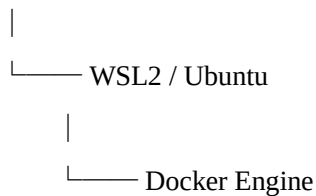
```
wsl --shutdown
```

Y comprueba:

```
wsl -l -v
```

INSTALAR DOCKER ENGINE EN UBUNTU

WINDOWS



Desde Ubuntu

```
wsl -l -v
```

Fíjate en el prompt

```
sudo apt update
```

```
sudo apt install -y ca-certificates curl
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
  
sudo apt update  
  
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
  
sudo usermod -aG docker $USER  
  
sudo docker run hello-world
```

P2. Instalación y exploración de OWASP Juice Shop

Duración aproximada: 30 minutos

OWASP Juice Shop es una aplicación web deliberadamente vulnerable creada con fines educativos. Simula una tienda online funcional e incorpora vulnerabilidades de distintas categorías.

Sitio oficial: <https://owasp.org/www-project-juice-shop/>

Instalación mediante Docker

Con Docker instalado y ejecutándose, cada alumno puede disponer de su propia instancia local:

```
docker pull bkimminich/juice-shop  
docker run -d --name juice-shop -p 3000:3000 bkimminich/juice-shop  
  
CUANDO TERMINES LO PARAS  
docker stop juice-shop  
  
OTRO DÍA PARA CONTINUAR  
docker start juice-shop
```

Abrir en el navegador:

```
http://localhost:3000
```

1. Realizar una captura en la que se vea Juice Shop ejecutándose localmente.
2. Explorar durante unos 10 minutos productos, búsqueda, registro, login, cesta, perfil, opiniones y funciones que requieran autenticación.
3. Crear una cuenta de prueba y realizar operaciones normales.

Observar HTTP con DevTools

Abrir F12 → Network, recargar Juice Shop y realizar distintas operaciones. Seleccionar tres peticiones y localizar URL, método, código de estado, Request Headers, Response Headers, Payload, Response y cookies cuando existan.

Petición	Método	Información enviada	Información recibida	¿Datos sensibles?
1				
2				
3				

Pregunta de reflexión

¿Qué ocurriría si pudiéramos modificar manualmente una de estas peticiones antes de que llegue al servidor? Esta idea prepara el trabajo posterior con Burp Suite.

P3. Score Board y primeras vulnerabilidades de Juice Shop

Duración aproximada: 60 minutos.

4. Encontrar el Score Board sin recibir inicialmente su URL. Se pueden utilizar navegación, menús, DevTools, rutas e investigación del frontend.

Pista 1. Abre F12 → Sources y observa el JavaScript que recibe el navegador.

Pista 2. Usa la búsqueda global Ctrl + Shift + F y busca:

score

Pista 3. No te limites a main.js: en Network → JS puedes ver todos los bundles/chunks JavaScript cargados.

Pista 4. Si el código está minificado, usa Pretty Print { } para hacerlo más legible.

Pista 5. Observa el patrón de las rutas de Juice Shop:

/#/...

Pista 6. Busca alguna referencia que contenga score y que pueda ser el nombre de una ruta.

5. Realizar una captura del Score Board.
6. Localizar desafíos de varias categorías y completar la tabla.(Usa internet)

Categoría	¿Qué crees que significa?	Ejemplo de riesgo
Injection		
XSS		
Broken Access Control		
Authentication		
Security Misconfiguration		

7. (Voluntario) Resolver un challenge sencillo (una estrella) o tutorial utilizando, si es necesario, el sistema de pistas.

Elemento	Respuesta
Nombre del challenge	
Categoría	
¿Qué vulnerabilidad existe?	
¿Dónde está el problema?	
¿Qué podría conseguir un atacante?	
Severidad estimada	Baja / Media / Alta
¿Cómo podría evitarlo un desarrollador?	

P3. PortSwigger Web Security Academy + SQL Injection

Duración aproximada: 60 minutos.

PortSwigger Web Security Academy es una plataforma gratuita de formación especializada en seguridad de aplicaciones web. Contiene explicaciones, rutas de aprendizaje, laboratorios interactivos y seguimiento del progreso.

Acceso: <https://portswigger.net/web-security>

OWASP Juice Shop	Web Security Academy
Aplicación completa que instalamos y exploramos.	Laboratorios específicos para practicar una vulnerabilidad concreta.
Permite descubrir múltiples fallos en un mismo sistema.	Cada laboratorio tiene un objetivo formativo delimitado.

Exploración inicial

Localizar las secciones SQL Injection, Cross-Site Scripting, Access Control, Authentication, File Upload, JWT y API Testing. En el módulo se comenzará fundamentalmente con laboratorios de nivel Apprentice.

Primer laboratorio: SQL Injection

Acceder a Web Security Academy → SQL Injection → Labs y seleccionar el laboratorio Apprentice indicado por el profesor. Leer la introducción antes de intentar resolverlo.

Metodología de trabajo

LEER → OBSERVAR → IDENTIFICAR ENTRADAS → FORMULAR HIPÓTESIS → PROBAR → ANALIZAR
RESULTADO → DOCUMENTAR → PROPONER SOLUCIÓN

Cuando el laboratorio indique LAB SOLVED, realizar una captura y responder:

- ¿Qué es SQL Injection? Explícalo con palabras propias.
- ¿Qué dato/datos de la aplicación podía controlar el usuario?
- ¿Qué comportamiento hacía sospechar que existía una vulnerabilidad?
- ¿Qué consiguió modificar la prueba realizada?

12. ¿Qué consecuencias podría tener en una aplicación real? Relaciónalo con confidencialidad, integridad y disponibilidad.
13. ¿Qué severidad asignarías (baja, media o alta)? Justifica la respuesta.
14. ¿Cómo debería solucionarlo un desarrollador?

5. Entrega de prácticas

El alumno entregará un único PDF breve. Se valorará más la comprensión y la explicación de la vulnerabilidad que la mera resolución de un challenge.

- Captura de Juice Shop funcionando localmente.
- Tabla de tres peticiones HTTP analizadas.
- Captura del Score Board.
- Ficha del challenge de Juice Shop.
- Captura de LAB SOLVED en Web Security Academy.
- Respuestas razonadas sobre SQL Injection.

6. Evaluación de las prácticas

Aspecto	Peso
Instalación y manejo de Juice Shop	15 %
Identificación de superficie de ataque y análisis HTTP	20 %
Challenge de Juice Shop	20 %
Laboratorio de Web Security Academy	20 %
Análisis de la vulnerabilidad y severidad	15 %
Propuesta de solución desde el punto de vista del desarrollador	10 %
TOTAL	100 %

Criterio didáctico

No se valora únicamente «resolver» el laboratorio. El objetivo es explicar qué vulnerabilidad existe, por qué existe, qué impacto puede tener, qué gravedad presenta y cómo debería corregirla un desarrollador.

7. Resultado final esperado

Al finalizar la unidad deberías ser capaz de observar una aplicación web y plantearse de forma sistemática:

¿Qué activos maneja? → ¿Qué datos controla el usuario? → ¿Cuál es la superficie de ataque? → ¿Existe alguna vulnerabilidad? → ¿Qué podría conseguir un atacante? → ¿Qué gravedad tendría? → ¿Cómo debería solucionarlo el desarrollador?

Además, quedarán preparados los entornos que se reutilizarán durante el resto del curso: OWASP Juice Shop, PortSwigger Web Security Academy y DevTools del navegador.

8. Recursos de referencia

OWASP Juice Shop: <https://owasp.org/www-project-juice-shop/>

PortSwigger Web Security Academy: <https://portswigger.net/web-security>

NIST: <https://www.nist.gov/>

NIST Cybersecurity Framework: <https://www.nist.gov/cyberframework>