UNIDAD 7: SEGURIDAD EN EL DESPLIEGUE DE APLICACIONES WEB

Introducción

Hasta ahora habéis trabajado en local o en entornos controlados.
En el momento en que una aplicación entra en producción, "**ya no es vuestra: es de Internet**".

- En local no hay atacantes
- En producción hay:
  - bots
  - escáneres automáticos
  - personas buscando errores
  - scripts probando vulnerabilidades **las 24 horas**

La seguridad no es una fase final, es una tarea permanente.

Qué es la seguridad en una web : Modelo CIA

Confidencialidad

Solo accede quien debe acceder.
Ejemplos:

- Un endpoint devuelve datos de otros usuarios
- Una base de datos accesible desde fuera
- Un backup sin cifrar en la nube
- API /users devuelve emails, roles o contraseñas hasheadas

Integridad

Los datos no se alteran sin permiso.
Ejemplos:

- Cambiar el precio de un producto desde el frontend
- Modificar pedidos manipulando una request
- Subir archivos que no deberían subirse
- Cambiar role=user por role=admin en una petición mal validada

Disponibilidad

La aplicación debe seguir funcionando.
Una web caída **también es un fallo de seguridad**

Ejemplos:

- Ataque DDoS
- Saturación por bots
- Borrado accidental de la base de datos
- Error tras una actualización mal hecha

Amenazas reales en producción

Ataques automáticos

La mayoría de ataques **no son personales**.

- Bots que:
  - prueban /admin, /login, /wp-admin
  - lanzan fuerza bruta
  - buscan versiones vulnerables

Errores humanos (eslabón más débil)

La mayoría de incidentes vienen de aquí.
Errores comunes:

- Puertos abiertos “por probar”
- Credenciales en el código
- Permisos excesivos
- Backups manuales que “luego se hacen”

Ejemplo:
Un desarrollador abre SSH temporalmente.Ese temporal se queda meses.

Fallos técnicos

- Discos que fallan
- VPS que muere
- Cortes de red
- Proveedores que caen

Software desactualizado

Supone:

- Vulnerabilidades públicas
- Exploits automatizados
- Ataques que no requieren talento

Principio clave: seguridad por capas

No existe la seguridad absoluta, existe una buena defensa en profundidad.

Capas típicas:

1. Firewall
2. Autenticación
3. Validaciones backend
4. Backups
5. Monitorización
6. Actualizaciones

Si una falla, otra debe cubrirla.

PILAR 1: Firewalls y control de tráfico

Qué es un firewall

Un firewall **decide qué entra y qué sale** del servidor.
No protege la app por sí solo, pero:

- reduce superficie de ataque
- bloquea accesos innecesarios
- frena ataques básicos

Regla del Mínimo privilegio

Solo se abre **lo imprescindible**.
Ejemplo de servidor web:

- 80 / 443: web
- 22: SSH (idealmente restringido)

Todo lo demás: cerrado

Deny by default

- Todo bloqueado
- Solo permites lo que necesitas

Si no necesitas FTP, ¿por qué está abierto?

Firewall de red vs WAF

Firewall de red

- Puertos
- IPs
- Protocolos

WAF (Web Application Firewall)

- Analiza HTTP
- Detecta SQL Injection, XSS, bots
- Protege la **aplicación**, no solo el servidor

Ejemplo:

- El firewall deja pasar 443
- El WAF bloquea una request maliciosa dentro de ese 443

Herramientas

- **UFW**: firewall sencillo en Linux
- **iptables / nftables**: control avanzado
- **Fail2Ban**: bloquea IPs por fuerza bruta
- **ModSecurity**: WAF para Apache/Nginx

PILAR 2: Copias de seguridad (backups)

Qué no es un backup

- Copiar archivos a mano
- Descargar la BD “de vez en cuando”
- Tener una única copia en el mismo servidor

Qué si es un backup profesional

Un backup debe ser:

- automático
- periódico
- verificable
- restaurable
- seguro (cifrado si hay datos personales)

Regla 3-2-1

- 3 copias
- 2 soportes distintos
- 1 fuera del servidor

Ejemplo:

- copia local
- copia en otro servidor
- copia en la nube

Restaurar

Un backup que no se ha probado **no existe**.

Herramientas típicas

- rsync / scp
- cron
- BorgBackup / Duplicity
- almacenamiento cloud (S3, GCS, etc.)

PILAR 3: Actualizaciones y mantenimiento

Qué hay que actualizar

- Sistema operativo
- Servidor web
- Lenguaje runtime
- Frameworks
- Dependencias
- CMS

Actualizar puede:

- romper cosas
- introducir errores

Pero **no actualizar**:

- deja vulnerabilidades abiertas

La seguridad no es solo técnica.

Incluye:

- contraseñas seguras
- uso de claves SSH
- MFA
- revisión de logs
- documentación
- formación continua