



Fundamentos del Diseño Seguro



Ciclo de vida del desarrollo seguro

Reducción de la Superficie de Ataque

- **Superficie de ataque:**
Cualquier parte de una aplicación que es accesible a un humano u otro programa
 - Cada una de estas puede ser potencialmente explotada por un usuario malicioso
- **Reducción de la superficie de ataque:**
Minimizar el número de puntos expuestos en la superficie de ataque que un atacante podría descubrir y explotar

Ejemplo de Superficie de Ataque



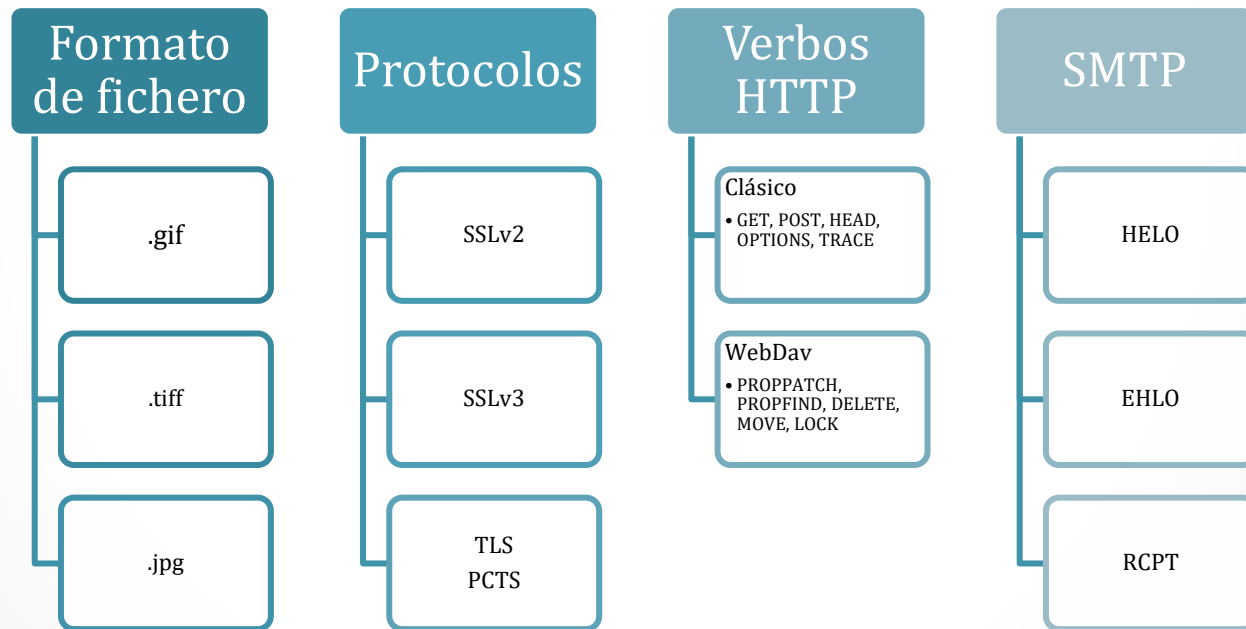
Análisis de la Superficie de Ataque



- Entradas/salidas de red
 - Entradas/salidas de ficheros
 - Etc.
- Autenticado o anónimo
 - Acceso administrativo o de nivel de usuario
 - En red o local
 - UDP o TCP
 - Etc.

Análisis de la Superficie de Ataque

- Proceso iterativo, para toda funcionalidad hay que analizar las subfuncionalidades
- Restringir el acceso a cada funcionalidad al máximo posible



No consiste únicamente en apagar cosas

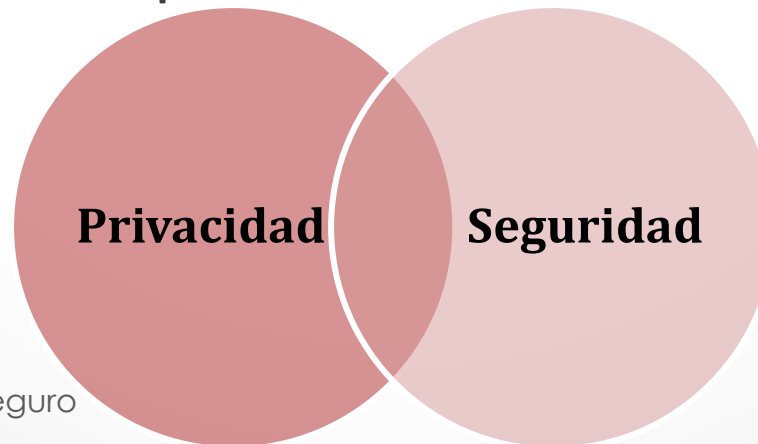
Mayor superficie de ataque	Menor superficie de ataque
Encendido por defecto	Apagado por defecto
Socket abierto	Socket cerrado
UDP	TCP
Acceso anónimo	Acceso autenticado
Constantemente activo	Activo bajo demanda
Acceso administrativo	Acceso a nivel de usuario
Accesible por Internet	Accesible por red local
Ejecución como SYSTEM (kernel)	Ejecución como usuario, servicio de red o servicio local (user space)
Valores por defecto generales	Valores por defectos definidos por el usuario
Código extenso	Código reducido
Controles de acceso débiles	Controles de acceso fuertes

Ejemplos de Reducción de SA

Producto	Reducción de superficie de ataque
Windows	<ul style="list-style-type: none">• Llamada de procedimiento remoto (RPC) autenticada• Firewall activo por defecto
Internet Information Services 6.0 y 7.0	<ul style="list-style-type: none">• Desactivado por defecto• Ejecución como servicio de red por defecto• Ficheros estáticos por defecto
SQL Server 2005 y 2008	<ul style="list-style-type: none">• Procedimiento xp_cmdshell deshabilitado por defecto• CLR y COM deshabilitados por defecto• Conexiones remotas deshabilitadas por defecto
Visual Studio 2005 y 2008	<ul style="list-style-type: none">• Servidor Web en localhost únicamente• SQL Server Express en localhost únicamente

Privacidad Básica

- Privacidad vs Seguridad
 - **Privacidad:** permitir a los usuarios controlar el uso, recolección y distribución de su información personal
 - **Seguridad:** establecer medidas de protección para garantizar la integridad, disponibilidad y confidencialidad de la información
- La privacidad y seguridad son factores clave en la construcción de aplicaciones de confianza



La seguridad no garantiza la privacidad

Es posible tener un sistema seguro donde NO se preserve la privacidad de los usuarios

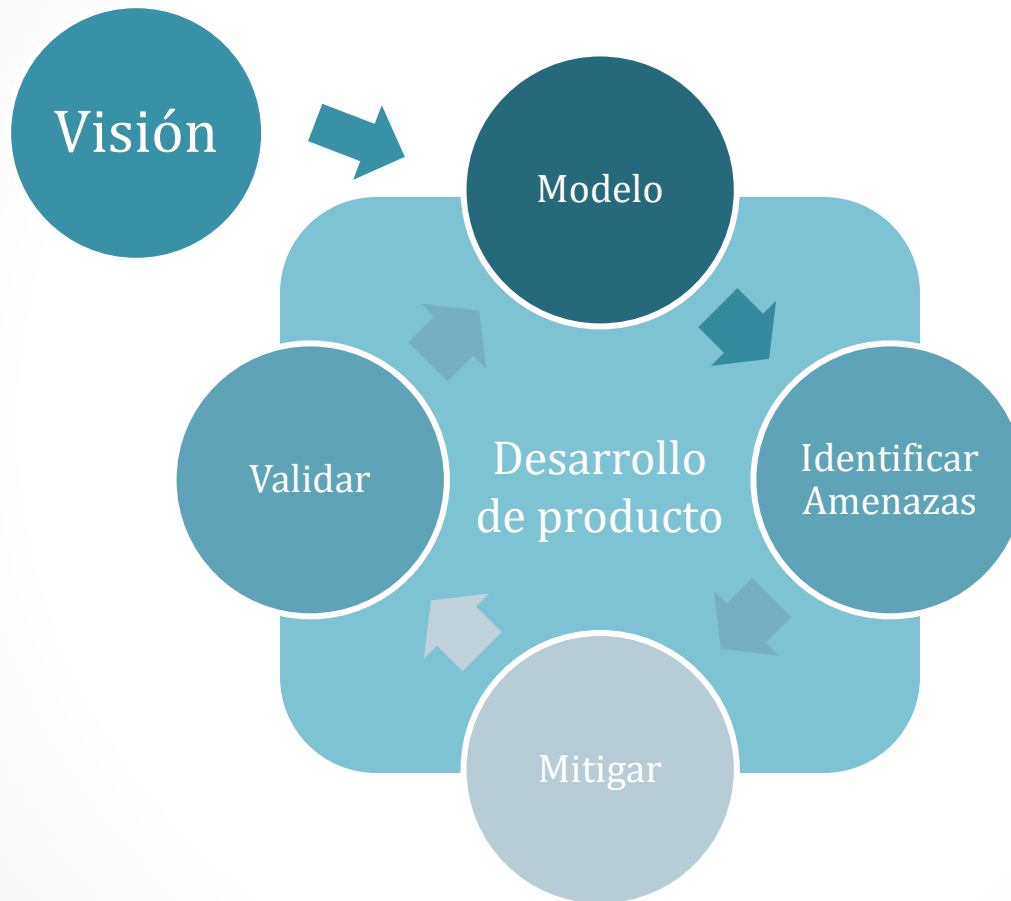
Comportamientos y Controles

Comportamiento	Control
Se dirige a menores	Children Online Privacy Protection Act (COPPA)
Transfiere información personal sensible	Gramm-Leach-Bliley Act (GLBA), Health Insurance Portability and Accountability Act (HIPAA)
Transfiere información personal no sensible	Unión Europea (UE) o Federal Trade Commission (FTC)
Modifica el sistema	Computer Fraud and Abuse Act (CFAA)
Monitoreo continuo	Legislación Anti-Spyware, Bloqueo de instalación
Transferencia anónima	Bloqueo de instalación

Modelado de Amenazas

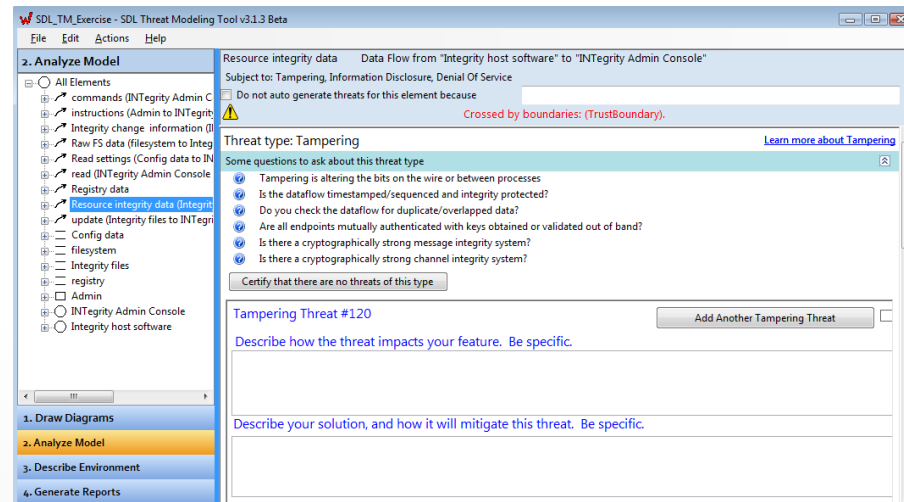
- **Modelado de amenazas:**
Un proceso para entender las amenazas que puede sufrir una aplicación
- Amenazas y vulnerabilidades no son lo mismo:
 - **Amenazas:**
Lo que un usuario malicioso puede intentar para comprometer un sistema
 - **Vulnerabilidades:**
Una forma concreta en la que una amenaza es explotable, como un error de programación por ej.

Modelado de Amenazas



Herramienta de Microsoft

- Microsoft ha publicado la herramienta de modelado de amenazas que usa internamente
- Se puede encontrar [aquí](#).



Defensa en profundidad

- Asumir que el software y el hardware fallarán en algún momento
 - Aplicaciones de confianza: características y mecanismos de seguridad y privacidad
- La mayoría de las aplicaciones actuales son comprometidas al romper una, habitualmente única, capa de seguridad (firewall)
- **Defensa en profundidad:**
Si una capa de defensa es rota, ¿qué otras capas (si existen) proporcionan seguridad adicional a la aplicación?

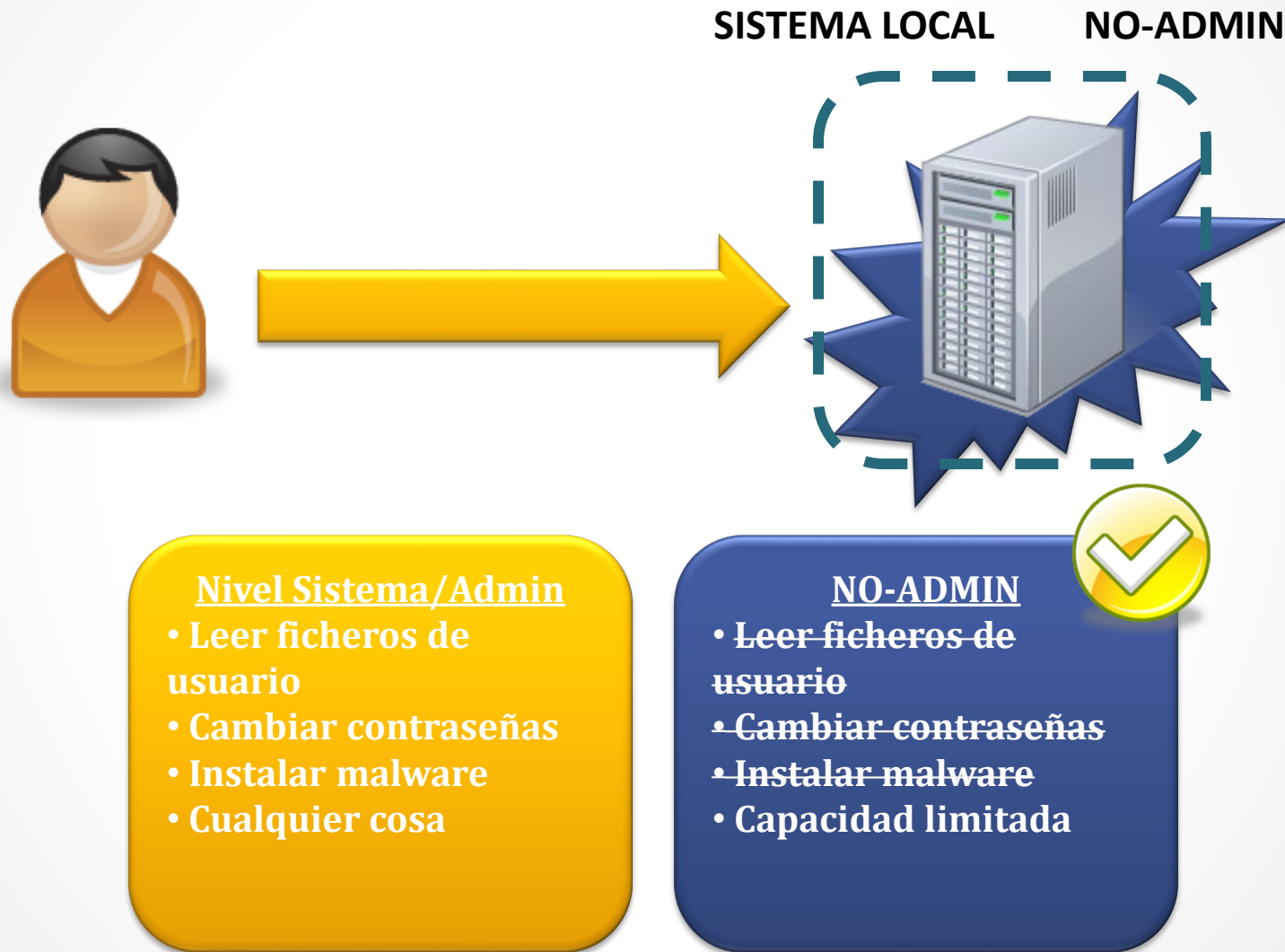
Ejemplo de defensa en profundidad



Minimización de privilegio

- Asumir que todas las aplicaciones pueden ser y serán comprometidas
- **Minimización del privilegio:** si una aplicación es comprometida, el daño potencial que puede causar el atacante es contenido y minimizado convenientemente
- Evaluar la aplicación y pensar de forma minimalista
- ¿Cuál es el nivel de acceso mínimo que requiere la aplicación para realizar sus funciones?
- Elevar los privilegios sólo cuando sea necesario, para posteriormente liberar dichos privilegios elevados cuando se han satisfecho los requisitos

Ejemplo de minimización de privilegio



Valores por defecto seguros

- **Valores seguros:**
Entrega de aplicaciones con configuraciones más seguras por defecto
- Ayuda a que los clientes tengan una experiencia más segura al instalar la aplicación, no tras una configuración extensa
- Se deja al usuario la posibilidad de reducir la seguridad o los niveles de privacidad

Ejemplos de valores por defecto seguros

Componente	Configuración por defecto
Firewall	Firewall activo
Socket SSL	Exigir última versión de SSL (v3, TLS, etc.)
Usuario puede acceder a la aplicación tanto de forma anónima como autenticada	Sesiones de usuario autenticadas
Se puede forzar una complejidad determinada de contraseñas	Complejidad de contraseña forzada
Almacenar contraseñas como hashes o en claro	Almacenaje de hashes de contraseñas

Análisis de Código

...

Ciclo de vida del desarrollo seguro



Intro: Análisis de código

- **Herramientas de análisis de código:**
Herramientas software que analizan la implementación de la aplicación para comprobar que siguen las prácticas recomendadas
- **Dos clases:**
 1. Análisis estático de código fuente
 2. Análisis binario
- Estas herramientas no son panaceas para identificar problemas
- Pueden reducir en gran medida los costes de ingeniería

Análisis estático vs binario




COMPILADOR
Y
ENLAZADOR




Código fuente

Fichero binario



```
void function(char * str)
{
    char buffer[32];
    strcpy(buffer, str);
}

void main(int argc, char ** argv)
{
    function(argv[0]);
    printf(argv[0]);
}
```



```
mov eax, DWORD PTR _str$[ebp]
push eax
lea ecx, DWORD PTR _buffer$[ebp]
push ecx
call _strcpy
add esp, 8

...

mov edx, DWORD PTR _argv$[ebp]
mov     eax, DWORD PTR [edx]
push    eax
call    _printf
```

Análisis estático de código fuente

- **Herramientas de análisis estático de código fuente:**
Herramientas software que analizan las implementaciones en código fuente en busca de posibles mejoras
 - **Entradas:** código fuente legible, como ficheros C (*.c), C++ (*.cpp, *.cc) o C# (*.cs)
- Algunas ventajas clave:
 - Facilidad de diagnóstico de problemas
 - Tecnología más madura

Análisis de código binario

- **Herramientas de análisis de código binario:**
Herramientas software que analizan código compilado o una versión en binario del código fuente en busca de posibles mejoras
 - **Entradas:** código máquina o ficheros binarios, como ficheros ejecutables (*.exe) y librerías (*.dll)
- **Ventaja:**
 - La herramienta de análisis de código binario puede ver el código binario en sí mismo (resultado final)

Análisis de código: pros y contras

Pros

- Ayuda a escalar el proceso de revisión de código
- Ayuda a cumplir políticas de programación segura

Contras

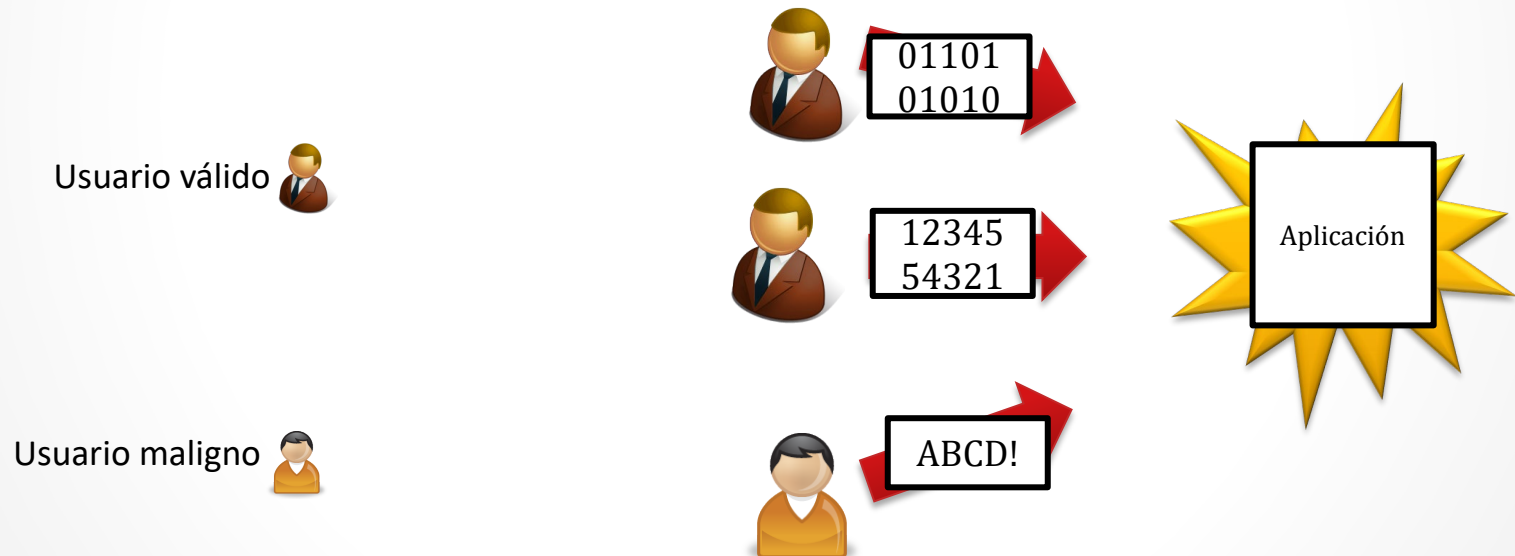
- Falsos positivos
- Falsos negativos
- Dependiente del lenguaje
- Problemas únicamente a nivel de código

Fuzz Testing

...

Introducción a Fuzz testing

- Consiste en introducir datos malformados y/o erróneos en la aplicación comprobando la reacción de la misma
- Si la aplicación falla es que se ha encontrado una vulnerabilidad potencial



Fuzz testing

Ventajas

- Las vulnerabilidades identificadas suelen ser de naturaleza severa
- Se puede automatizar con facilidad

Desventajas

- No puede identificar vulnerabilidades que no causan una excepción:
 - filtración de información
 - fallo en la criptografía
 - etc.

No debe utilizarse como remplazo de otras técnicas de evaluación y no debe ser considerada una panacea, es un complemento

Tipos de fuzz testing

Aleatorio

- Se introducen datos en la aplicación que cambian de forma puramente aleatoria

Kevin, 123, Microsoft, Internet,
SQLServer, k3v1n11, AAAAAAAAAAAAA,
00000_, !@#%^

Inteligente

- Se modifican valores específicos en función de la experiencia previa y/o el comportamiento esperado

(AAA) 123-4567, (123) 123*1234, (000
123-4567, (<empty>)123-4567, (123)
456---5678, ((123)4567890,(AAAAAAAAA)
123-4567, (<U+07C0>23) 123-4579

Realización de Fuzz tests

1. Determinar los puntos de entrada a la aplicación
2. Ordenar dichos puntos de entrada en función del privilegio y la accesibilidad
3. Crear e introducir datos malformados en la aplicación para evaluar los puntos de entrada que están en riesgo
 - Intentar automatizar el proceso en la medida de lo posible
4. Analizar cualquier fallo, excepción o error inesperado o no gestionado debidamente para identificar vulnerabilidades severas
5. Reparar y volver a comprobar