

1. Detección y corrección de vulnerabilidades de aplicaciones web.

Caso práctico



[Pixabay](#) (Dominio público)

Julián ha sido asignado a un nuevo proyecto de transformación digital en una importante entidad financiera. El objetivo del proyecto es poder dotar a los usuarios de la entidad de la capacidad de gestionar sus datos y realizar operaciones bancarias a través de una nueva plataforma web.

Este proyecto es muy atractivo para Julián y está deseando comenzar a trabajar en él, pero por otra parte hay dos cosas que le preocupan, la gran exposición que tendrá el aplicativo ya que será web y estará accesible para todo el mundo en Internet y por otra parte el hecho de que el aplicativo trabajará con varias bases de datos que tienen información personal de los usuarios.

Sabe además que hay una fundación (OWASP) y su Top10 que podrá usar

para evaluar riesgos pero aún así es consciente de que aunque el software sea robusto y seguro necesitará ayuda y es posible que el proyecto requiera que se instalen Firewalls de aplicativo web (WAF - *Web Application Firewall*) para controlar el volumen y tipo de peticiones que se reciban así como sistemas Captchas y demás que garanticen que sólo usuarios reales acceden al aplicativo descartando redes de *bots* o software de automatización de ataques.

Durante las últimas dos década estamos siendo testigos de un proceso de transformación digital a varios niveles que está provocando un gran crecimiento en la cantidad de aplicaciones de todo tipo que surgen para habilitar esta transformación. Al mismo tiempo y como consecuencia de la exposición se está produciendo un aumento significativo de los ataques dirigidos a las aplicaciones, debido a su interacción con los datos. Todo eso está generando por un canal directo para que los atacantes exploten vulnerabilidades introducidas en el desarrollo de las aplicaciones.



[Ministerio de Educación y Formación Profesional](#) (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#)

1.1.- Entrada basada en formularios.

[Particulares](#) [Empresas](#) [← Volver](#)



Te damos la bienvenida a tu Banca Online

Documento
NIF

▼

Nº de documento


Clave de acceso


 

☐ **Recordar usuario**

Entrar

¿Problemas con tu clave de acceso?

 **Centro de ayuda**

 **Oficinas y cajeros**

- Alta en Banca Online
- Seguridad

- Demo

David Conde (Dominio público)

Los formularios no son mas que formas o vías de comunicación del aplicativo web que nos permiten interactuar con él proporcionándole información de entrada para que nos devuelva otro tipo de información. Suele ser la vía de comunicación estándar con un aplicativo web a nivel de usuario y se usa para muchos fines desde realizar consultas hasta poder autenticarnos en el aplicativo.

Un formulario está diseñado con el propósito de que el usuario introduzca datos estructurados (nombre, apellidos, teléfono, contraseña, etc.) en las zonas del aplicativo destinadas a ese fin, para ser registradas y procesadas posteriormente por el aplicativo web.

Alguno de los elementos más comunes que podemos encontrar en un formulario son:

- ✓ Cajas o **boxes** de texto
- ✓ Listas de selección
- ✓ **Checkbox** o casillas de verificación
- ✓ Botones
- ✓ Desplegables informativos

A nivel de seguridad es uno de los principales vectores de ataque que los atacantes usan, ya que si no está debidamente configurado y secularizado se puede manipular para realizar acciones no esperadas como obtener el contenido de la base de datos, saltarnos la autenticación o incluso introducir contenido malicioso en la base de datos o en el aplicativo. Este tipo de ataques se llaman ataques de inyección, donde inyectaremos o introduciremos nuestro código malicioso para hacer que la aplicación se comporta de una manera no esperada.

Para prevenir esto es importante que el aplicativo valide correctamente las entradas de datos, que realmente compruebe si su contenido se corresponde con lo esperado (por ejemplo sería raro que el campo "teléfono" tuviera comandos de ejecución de código), que higienice la entrada por si tuviera algún carácter extraño y también es recomendable que la interacción con la base de datos sea a través de variables o parámetros dinámicos.

Debes conocer

Cuando pensamos en ataques de inyección comúnmente pensamos en ataques que tienen como objetivo cambiar la manera en que el aplicativo web se comporta, un ejemplo clásico son ataques de tipo **SQL Injection**. Estos ataques se conocen como **Server Side injection**.

Pero dentro de la inyección hay otro tipo de ataques tanto o más críticos que no tienen como objetivo el sistema o aplicativo sino al propio usuario. Buscan robar información sensible del usuario o incluso sus **cookies** para ser re usadas más tarde de forma maliciosa. Son lo que denominamos **Client Side Injection** y los ejemplos más claros serían los ataques de tipo XSS (Cross-Site-Scripting) y HTML Injection.

En el primero de los tipos de ataque, tanto los programadores como los ingenieros de seguridad dedican grandes esfuerzos en controlar la entrada de datos y su ejecución, pero en el segundo de los casos aunque el programador haya revisado todo su código y se considera seguro para el servidor puede seguir siendo un riesgo para el usuario.

Por ejemplo uno de los ataques mas importantes ante el que conviene estar concienciado son los ataques de robo de sesión o **Session Hijacking** que utilizan ataques de tipo inyección en el lado del cliente para robar las cookies del usuario y poder luego usar esa cookie (que normalmente garantiza una

sesión autenticada y autorizada contra un recurso) de forma maliciosa, por ejemplo para hacer transacciones de banco de forma no autorizada.

Para saber más

Los parámetros dinámicos (también llamados bind o parámetros bind) son una alternativa para presentar los datos a la base de datos. En lugar de poner los valores directamente dentro de la sentencia SQL (o similares) y que esta se ejecute en la base de datos, se puede usar un marcador de posición específico como `?`, `:name` o `@name` y proveer el valor actual de la variable. Esto provoca que el contenido introducido por el usuario solo pueda afectar de una manera limitada y controlada a la consulta o instrucción que se ejecutará en la base de datos.

1.2.- Estándares de autenticación y autorización.

Al publicar datos a través de un servicio web es necesario identificar quién puede acceder a dicho servicio, salvo casos específicos en que los servicios web sean de acceso público. La autenticación es el proceso de verificación del usuario o entidad, mientras que la autorización es el proceso de verificación de los permisos que tiene dicho usuario o entidad sobre un recurso específico.



[Unsplash](#) (Dominio público)

Siempre se debe realizar primero la autenticación y luego la autorización. Esto debe verificarse en cada petición que realice el usuario hacia el recurso o entidad.

Por otra parte para tener modularidad en los permisos y otorgar únicamente los permisos realmente necesarios para los usuarios o entidades. Esto se consigue mediante la asignación de **roles**.

Para ambos procesos a día de hoy existen los siguientes estándares

Autenticación Básica (Basic Auth)

La autenticación básica es el método más simple y utiliza un usuario y contraseña para identificar al usuario o entidad de origen.

Se recomienda usar este tipo de autenticación en los servicios web cuando los datos que se quiere intercambiar no tengan muchas restricciones de uso. Además, este tipo de autenticación permitirá identificar mínimamente el usuario o entidad de origen. Cabe destacar que este tipo de autenticación puede ser utilizado tanto con un tipo de servicio web de tipo REST como SOAP.

Autenticación con certificados

La autenticación por medio de certificados se realiza solicitando el certificado digital del cliente y verificando cada mensaje enviado contra dicho certificado, lo que garantiza que el cliente del servicio web es quien dice ser. Este mecanismo es tedioso si estamos autenticando personas/navegadores pero es especialmente útil para autenticar otro servicio web ya que se integra dentro del protocolo HTTPS y de esta forma asegura la identificación, la autenticación, la confidencialidad y la integridad. Esta modalidad se conoce como autenticación mutua TLS.

Autenticación mediante JWT

JWT es el medio principal de autenticación en servicios de tipo REST, es un medio compacto y autocontenido para enviar datos de manera segura entre las partes en formato JSON.

Autenticación OAuth (v1 y v2)

Existen dos versiones: OAuth 1.0a y OAuth2, siendo esta última la más utilizada.

OAuth2 es un protocolo de autorización y la manera de realizar la autenticación va más allá del alcance de la especificación, por lo que se puede implementar el proceso de autenticación que se desee. Una gran mayoría de las implementaciones del estándar ya ofrecen mecanismos de autenticación.

Para saber más

Dentro del proceso de transmisión de datos entre aplicaciones podemos optar por dos enfoques diferentes que condicionaran la manera en que nuestra aplicación se comporta y los requisitos. Estos métodos son REST y SOAP. Puedes consultar mas información sobre estos métodos en el siguiente enlace de [RedHat](#).

Autoevaluación

Identifica si las siguientes frases son verdaderas o falsas

La entrada de datos mediante formularios no es un punto crítico de las aplicaciones web.

☐ Verdadero ☐ Falso

Falso

Todo proceso de introducción de datos debe de ser revisado, securizado y monitorizado.

El riesgo de un aplicativo web viene condicionado por su exposición

☐ Verdadero ☐ Falso

Verdadero

A mayor exposición mayor volumen de peticiones y mayor probabilidad de ser atacado por la exposición.

OWASP no evalua el software sino que proporciona guías e identifica riesgos definiendo tambien controles de seguridad.

☐ Verdadero ☐ Falso

Verdadero

El trabajo de OWASP se centra en ayudar a entender riesgos y sus impactos.

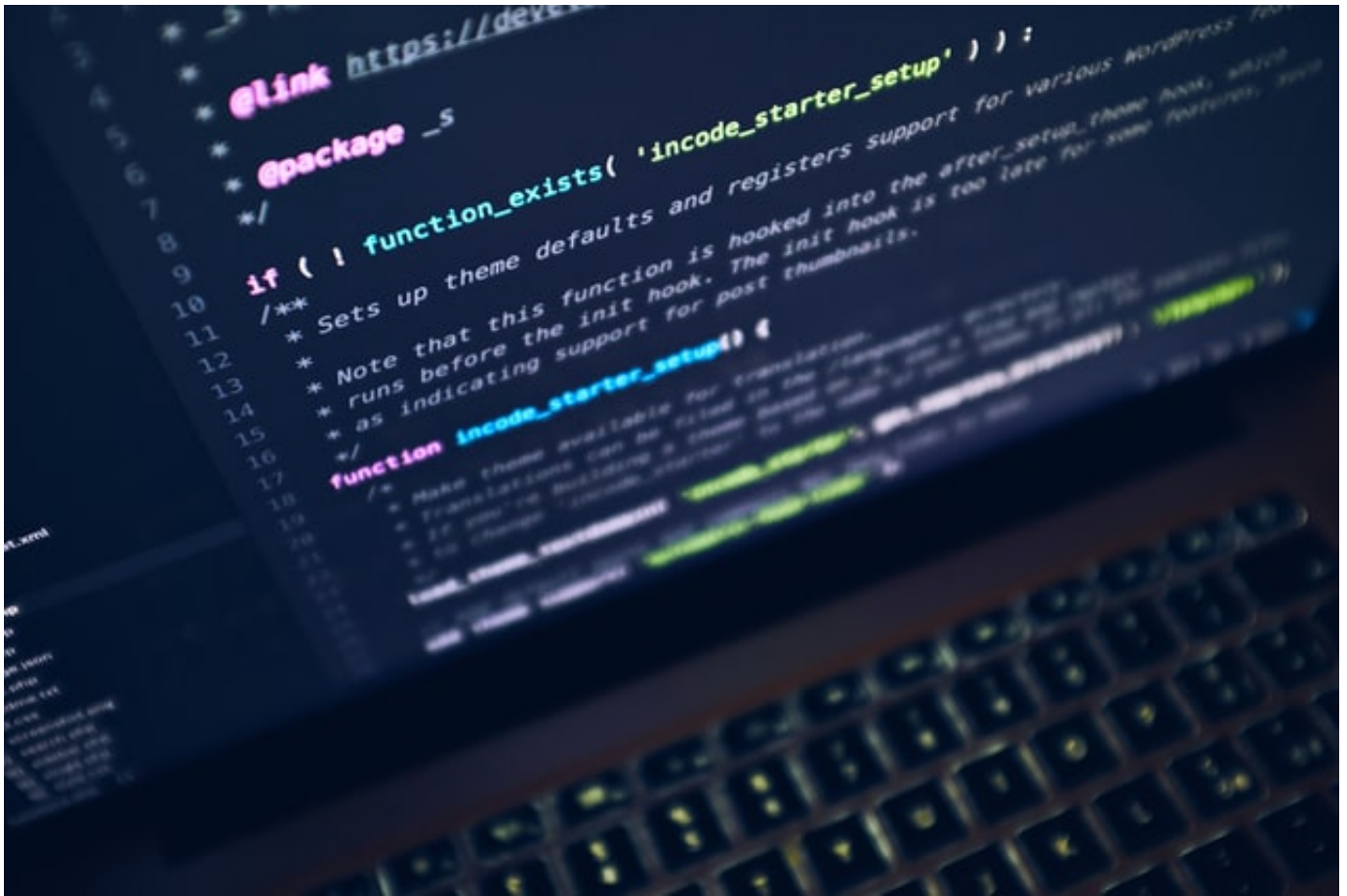
Un cuadro de texto NO es un elemento de un formulario

☐ Verdadero ☐ Falso

Falso

Es uno de los principales elementos y además suelen ser blanco de ataques como XSS.

1.3.- Vulnerabilidades web.



[unsplash](#) (Dominio público)

Entendemos por vulnerabilidad un fallo o debilidad en el código que permite ser explotado para poder realizar acciones distintas al fin del aplicativo, en este caso de forma maliciosa.

Cómo vimos anteriormente la fundación OWASP identifica los diez riesgos principales, que podemos relacionar con ataques y vulnerabilidades. Vamos a analizar en detalle el TOP5

- 1.- Pérdida del control de acceso (Broken Access Control)
- 2.- Fallos criptográficos (Cryptographic Failures)
- 3.- Inyección (Injection)
- 4.- Diseño inseguro (Insecure Desing)
- 5.- Configuración de seguridad defectuosa (Security Misconfiguration)

Si bien OWASP en su lista incluye mas este Top5 engloba la mayoría de los ataques que se producen diariamente en los aplicativos web, si conseguimos proteger nuestro aplicativo con las suficientes contramedidas tanto a nivel de software seguro como tecnología adicional podremos dotar de gran robustez a nuestro aplicativo.

Debes conocer

El Instituto Nacional de Ciber Seguridad (INCIBE) dispone de un artículo muy interesante sobre el Top10 de OWASP. Puedes consultarlo [aquí](#).

1.4.- Almacenamiento seguro de contraseñas.



[Pixabay](#) (Dominio público)

Los aplicativos web deberían de trabajar y sobre todo almacenar los menores datos posibles, es decir mediante el principio del menor acceso posible. Pero a veces por diversas cuestiones necesitan almacenar contraseñas o datos sensibles.

La mejor manera de protegerlos es mediante funciones de cifrado y funciones de hash. También los protocolos usados (incluidos los de autenticación y autorización) usan sus propios mecanismos de protección así como sistemas de cifrado y funciones hash.

Cifrado

Permite cifrar la información mediante sistemas de cifrado de clave simétrica o asimétrica. Este último es el más seguro pero requiere de mas carga computacional.

Funciones Hash

Permite en vez de guardar las credenciales en texto plano guardar una imagen *hash* de la contraseña. Este tipo de funciones hace que pequeños cambios en la contraseña se reflejen en grandes cambios en su hash, resultando por tanto muy complicado hacer el análisis

criptográfico y obtener la contraseña del usuario. Hoy en día los algoritmos hash mas usados son SHA1, SHA2, SHA3. Antiguamente se usaba el algoritmo MD5 pero actualmente no se considera seguro por su grado de colisiones. Es importante remarcar que las funciones hash hoy en día se usan con **salt** que no es mas que bits aleatorios que se usan para generar imágenes hash aún mas aleatorias y evitar su cripto análisis.

Protocolos

Muchos de los protocolos que se usan hoy en día en programación y comunicación de aplicativos ya llevan embebidas funciones hash y sistemas de cifrado. Por ejemplo openssl gestiona la comunicación segura de las comunicaciones cliente-servidor.

RECOMENDACIÓN

La mejor recomendación de seguridad no es el tipo de sistema de cifrado o función hash a usar sino evitar almacenar o gestionar cualquier tipo de información sensible a menos de que sea estrictamente necesario.

Debes conocer

Seguro que encuentras interesante este [link](#) de discusión sobre el uso de MD5.

1.5.- Contramedidas.



[Pixabay](#) (Dominio público)

Si bien proteger nuestro software mediante una revisión exhaustiva que evite vulnerabilidades y un diseño seguro es una de las mejores recomendaciones a veces no es suficiente. Muchas veces no podemos contemplar todas las variables y aunque lo hagamos podemos sufrir ataques de programas maliciosos que son capaces de generar miles de peticiones e intentos de explotar nuestro software por segundo que hace que sea complicado de prevenir.

Para ello se diseñaron medidas adicionales de seguridad como:

HSTS (*HTTP Strict Transport Security* o HTTP con Seguridad de Transporte Estricta)

Son una serie de mecanismos que soportan los principales servidores y navegadores web para evitar el robo de sesiones y datos confidenciales de los usuarios. Surgió en su momento para evitar que un atacante interceptará una comunicación entre cliente-servidor y robe o espíase su contenido o degradase la seguridad.

CSP (*Content Security Policy*)

Es un estándar que define un conjunto de orígenes pre-aprobados desde los que un navegador puede cargar recursos cuando visita un usuario visita un sitio web. También indica desde dónde podemos ejecutar scripts, imágenes y demás. Surgió y es efectivo para dar respuesta a la gran cantidad de ataques de tipo **Cross Site Scripting (XSS)**. Requiere la modificación del encabezado para prevenir la ejecución o redirección maliciosa del XSS.

CAPTCHA

Son las siglas de Completely Automated Public Turing test to tell Computers and Humans Apart. Son pruebas de tipo desafío-respuesta que permite a los aplicativos web saber cuando hay un usuario real de una máquina o software. Este tipo de control permite reducir el uso de ataques de tipo **botnet** o software automatizado contra nuestro aplicativo.

Autoevaluación

El software seguro es suficiente para que nuestro aplicativo web esté protegido.

☐ Verdadero ☐ Falso

Falso

Un software seguro no es suficiente, necesitamos controles y mecanismos adicionales tanto a nivel de software como de tecnología.

Un captcha ayuda a prevenir software malicioso atacando nuestro aplicativo web.

☐ Verdadero ☐ Falso

Verdadero

Un captcha nos ayuda a separar el tráfico legítimo de usuarios de actividad automática maliciosa.

CSP ayuda a evitar los ataques de tipo XSS del lado cliente.

☐ Verdadero ☐ Falso

Verdadero

Aporta una capa más de protección provocando que no se pueda ejecutar código aunque exista la vulnerabilidad.

HSTS ayuda a prevenir los ataques de tipo Man-In-The-Middle

☐ Verdadero ☐ Falso

Verdadero

El principal cometido de HSTS es impedir que un atacante convierta una conexión HTTPS en HTTP

1.6.- Seguridad de portales y aplicativos web.



[Pixabay](#) (Dominio público)

Si bien desde el punto de vista del programador tenemos el objetivo de garantizar que el código es seguro y se comporta de manera controlada y esperada, es posible que con todo esto no podamos aún así estar seguros, ya que muchas veces intervienen otros componentes como bases de datos que se escapan del control del programador o el aplicativo recibe miles de peticiones por segundos que hace muy difícil garantizar su comportamiento esperado.

Para ayudar en este aspecto surgen los cortafuegos de aplicación Web, o WAF (**Web Application Firewalls**). Podríamos pensar que son un cortafuegos tradicional con alguna tipo de capacidad adicional, pero nada mas lejos de la realidad. Los WAFs son dispositivos complejos, que requieren de unos conocimientos elevados para administrar y que proporcionan mucha capacidad de detección y bloqueo de ataques.

La diferencia principal de los WAFs de un cortafuego tradicional es que los WAFs saben como funciona un aplicativo web, cómo interacciona con la base de datos y que tipo de datos suelen entrar y salir del aplicativo, detectando ataques de manera mas sencilla. ¿Cómo lo consiguen? De una manera sencilla, cuando instalamos un WAF se dedica a estar durante

meses aprendiendo como funciona la aplicación, que tipo de datos maneja, cuando se conectan los usuarios, que datos devuelve y demás. Esto provoca que puedan detectar cualquier tipo de anomalía, ataque o parámetro malicioso bloqueando el ataque.

Además son capaces de detectar cuando no se trata de usuarios reales sino redes de **bots** (programas o máquinas actuando a la vez orquestados por un usuario) bloqueando estas conexiones y reduciendo el impacto de los ataques de denegación de servicio.

Al final disponer de un WAF no es más que otra medida más que ayuda a mejorar la seguridad de nuestro aplicativo, de forma externa al software en este caso.

Para saber más

Hay muchas maneras de desplegar un WAF y muchos modos de funcionamiento, por ejemplo puede estar únicamente detectando ataques o puede estar detectando y bloqueando. Por otra parte tenemos WAFs físicos en forma de **appliance** o máquina física y modelos en nube que podremos desplegar en cuestión de minutos mediante algunos cambios (principalmente en los registros DNS y en algunos certificados)

En este [vídeo](#) puedes ver las principales características de un WAF.

