Desarrollo web Seguro

Los motivos para atacar una aplicación web van desde un beneficio económico hasta el robo de datos de los usuarios, provocando la denegación de servicio, afectando la imagen de las corporaciones o simplemente por ocio.

No es suficiente con que el código sea funcional. Se deben incluir prácticas de seguridad de aplicaciones web desde las fases de diseño y codificación.

Fase de Diseño

THREAT MODELING - MODELADO DE AMENAZAS

Documentar, localizar, abordar y validar son los cuatro pasos para el Modelado de Amenazas. Usando éste junto con un equipo de profesionales en seguridad evaluarán si el diseño del producto es compatible y seguro. La intención es discutir todas las posibles vulnerabilidades para así disminuir riesgos.

Fase de Desarrollo

CONFIGURACIÓN DEL SISTEMA

El software obsoleto es una fuente importante de vulnerabilidades y brechas de seguridad. Las actualizaciones de software periódicas son una de las prácticas de codificación más vitales y seguras.

Los desarrolladores deben estar capacitados en OWASP Top 10 del Open Web Application Security Project y en la lista de verificación de seguridad de aplicaciones web SANS del Instituto SANS.

Código seguro

Validar entradas

Se deben validar los campos de entrada tanto en el lado del servidor como en el del cliente. Los procesos maliciosos pueden evitarlo fácilmente en el lado del cliente. Para los casos en los que un usuario malintencionado ha pasado por alto la validación del

lado del cliente, el lado del servidor lo manejará. Realiza siempre verificaciones de los límites para evitar problemas de desbordamiento del búfer, ya que puede abrir el código a muchos tipos de riesgos, como la denegación de servicio y la inyección remota de código.

Usa listas blancas en lugar de listas negras para verificar los campos de entrada. El enfoque de listas negras puede dificultar la restricción de entradas maliciosas o puede dejar la puerta abierta a los actores de amenazas. El uso de listas blancas para permitir solo el tipo de caracteres requerido ayudará a prevenir muchos tipos de riesgos de validación de entrada.

Command Injection

Debes asegurarte de que el código evite ejecutar comandos directamente desde el valor de entrada recibido. Si lo haces, puedes encontrar aperturas como la inyección de comandos del sistema operativo. En este caso, el ciberdelicuente podría ejecutar comandos del sistema operativo en el servidor inyectándolos en los campos de entrada que no se han desinfectado adecuadamente. Incluso en los casos en los que sea necesario ejecutar comandos, se debe hacer siempre con el mínimo de privilegios requerido.

SQL Injection

La inyección de lenguaje de consulta estructurado (SQL) es uno de los principales riesgos que puedes encontrar. En este tipo de ataque, una declaración SQL ingresa a los campos de entrada, lo que da como resultado la ejecución de estas declaraciones en la base de datos revelando el contenido de la base de datos y permite la inserción de valores maliciosos en la base de datos.

Other Web Application Security Best Practices

Se han definido muchos encabezados de seguridad para evitar problemas, como secuencias de comandos entre sitios (XSS), clickjacking y otros. El uso de encabezados es una manera fácil de proporcionar un nivel mínimo de seguridad para tales problemas y proporcionar una barrera de defensa en profundidad contra esos riesgos.

Algunos tipos comunes de encabezados de seguridad son seguridad de transporte estricta HTTP (HSTS), protección X-XSS, opciones de tipo de contenido X, opciones de marco X y política de seguridad de contenido.

En los casos en los que se proporciona al usuario una opción de carga de archivos, restrinja el tipo de archivo que se carga solo al tipo esperado. Asegúrate de exigir que se verifique la extensión del archivo y el contenido del archivo que se está cargando. Además, se recomienda un escaneo en el archivo cargado para verificar si hay o no hay contenido malicioso.

Evita tener un localizador de recursos (URL) uniforme o un campo de entrada de ruta. El uso de la entrada de ruta directamente en el código puede generar riesgos como la inclusión de archivos locales, la inclusión de archivos remotos, la falsificación de solicitudes del lado del servidor y la redirección y reenvío no validados. Si se requiere tener rutas y URL en el valor de entrada, usa la lista blanca adecuada para evitar cualquier acceso indebido.

Cifrado

El cifrado es uno de los aspectos más importantes para proteger su trabajo. Asegúrese de que esté en su lugar para los datos en tránsito y en reposo, teniendo especial cuidado cuando los datos incluye información confidencial. Utiliza siempre HTTPS y nunca permitas el acceso a través de HTTP.

Es importante utilizar técnicas de cifrado conocidas en lugar de intentar implementar las propias. Junto con el cifrado, comprueba que los datos estén seguros mediante el uso de técnicas como el hash. Al utilizar el cifrado, se deben evitar los algoritmos, cifrados o versiones débiles conocidos. Cuando se almacenan datos confidenciales en archivos de registro o bases de datos, los datos deben también deben estar cifrados.

Contraseñas, accesos e inicios de sesión

• Administración de contraseñas
Almacenar solo hashes criptográficos de contraseñas y nunca almacenar contraseñas de texto sin formato. Hacer cumplir los requisitos de complejidad y longitud de las contraseñas. Desactive la entrada de contraseña después de varios intentos fallidos de inicio de sesión.

• Control de acceso
Adopte un enfoque de "denegación predeterminada" para los datos sensibles. Limite los privilegios y restrinja el acceso a los datos seguros solo a los usuarios que los necesiten. Deniegue el acceso a cualquier usuario que no pueda demostrar su

autorización. Asegúrese de que las solicitudes de información confidencial estén marcadas para verificar que el usuario esté autorizado a acceder a ella.

Asegúrese de que las contraseñas que elijan sus usuarios sean complejas. La contraseña debe tener un mínimo de ocho caracteres (cuanto más larga, mejor) y contener una combinación de caracteres superiores, inferiores y especiales. Esto hará que los ataques de diccionario y de fuerza bruta sean más difíciles de ejecutar. Para mejorar aún más este paso, utilice la autenticación de dos factores.

• Inicios de sesión
Además, debe implementar un bloqueo de cuenta cuando el sistema detecta el número máximo de intentos de contraseña. Para asegurarse de que solo los usuarios autorizados ingresen, use el acceso escalonado y basado en privilegios con el mínimo privilegio para el rol predeterminado. Utilice el acceso basado en roles a los recursos para garantizar que el acceso a recursos específicos solo se otorgue a los usuarios con el privilegio requerido.

Herramientras de seguridad [DevSecOps, SAST, DAST, Penetration Testing]

Para garantizar aún más las prácticas de seguridad puedes agregar algunos métodos de prueba de seguridad.

DevSecOps
Para ir escaneando el código conforme se va desarrollando.

SAST
Sirve para escanear el código fuente. Se debe ser cuidadoso con la herramienta que se usa para este método ya que podría dar falsos positivos, por lo que deberás analizar muy cuidadosamente y usar filtros para poder interpretar los resultados obtenidos.

DAST
Dynamic application security testing (DAST). Estas herramientas envían muchas solicitudes con paquetes deformados al código, con la finalidad de encontrar agujeros.

Penetration Testing

Es de los métodos más avanzados para pruebas, usa una técnica de escaneos y con técnicas de explotación para encontrar vulneabilidades. Con este se busca minimizar el riesgo de acceso y manipulación de datos.

Manejo y registro de errores

Lleva un registro para documentar los errores para que los desarrolladores puedan diagnosticar y mitigar su causa. La documentación y el registro de todas las fallas, excepciones y errores deben implementarse en un sistema confiable para cumplir con los estándares de codificación segura.