GML Software

Guía para el desarrollo de código seguro.

Manual de Procedimientos

Mayo de 2017

Este documento contiene una lista de prácticas de codificación que debe ser seguida en todos los desarrollos realizados en **GML Software** y está basada en OAWASP SCP Quick Reference Guide

Archivo:	**2017-04-17 - Principios y buenas prácticas de desarrollo seguro_v1.0.docx**
Páginas:	12
Fecha:	
Autor:	Pedro Arciniegas
Contacto:	
Versión	1

TABLA DE CONTENIDO

INTRODUCCIÓN

Este documento define un conjunto de prácticas de codificación para el desarrollo de software seguro que debe ser integrado al ciclo de vida de desarrollo de software en **GML Software** y que pretende mitigar las vulnerabilidades más comunes en el software.

Generalmente, es mucho menos costoso construir software seguro que corregir problemas una vez el software ha construido, eso, sin mencionar los costos no cuantificables que pueden estar asociados con una brecha de seguridad.

Esta guía no cubre los detalles de cómo implementar cada práctica, de manera que la responsabilidad de su implementación es del equipo de trabajo en general y del líder técnico (arquitecto) en particular.

Objetivo

El principal objetivo de la utilización de prácticas de codificación seguras, es lograr mayores niveles de calidad y seguridad en los productos de software desarrollados en **GML Software**.

Alcance

Este documento aplica a todos los desarrollos web independientemente del lenguaje de programación utilizado. No se incluyen referencias o prácticas relacionadas al uso de las bibliotecas de clases de .Net Framework, Java o cualquier otro lenguaje o plataforma.

Condiciones de uso de este documento

Una regla puede romperse sólo ante razones justificadas, discutidas, con previa autorización del responsable del producto, y en caso que no pueda aplicarse ninguna alternativa razonable. El autor de la excepción, obligatoriamente debe documentar el código explicando la causa de la violación de la regla.

Las preferencias personales no se consideran una razón justificada.

Fuentes de información consultadas

Basado en: OWASP SCP Quick Reference Guide - version 2

LISTADO DE PRÁCTICAS

1. Validación de entrada

1.1. Lleve a cabo todas las validaciones en un sistema de confianza (ej: El Servidor)
1.2. Identifique todos los orígenes de datos y clasifíquelos en “confiables” y “no confiables”. Valide todos los datos procedentes de orígenes “no confiables” (ej: bases de datos, flujos de archivos)
1.3. Asegúrese de tener una única rutina centralizada de validación de entradas para la aplicación.
1.4. Especifique conjuntos de caracteres adecuados, como UTF-8, para todas las fuentes de entrada.
1.5. Codifique los datos a un conjunto de caracteres común antes de validar.
1.6. Todas las fallas de validación deben resultar en rechazo de entrada.
1.7. Determine si el sistema admite conjuntos de caracteres extendidos UTF-8 y, de ser así, valide después de que se complete la descodificación UTF-8
1.8. Valide todos los datos proporcionados por el cliente antes del procesamiento, incluidos todos los parámetros, direcciones URL y contenido de encabezado HTTP (por ejemplo, nombres y valores de cookies).
1.9. Compruebe que los valores de encabezado en las solicitudes y respuestas contienen sólo caracteres ASCII.
1.10. Valide los datos de los re-direccionamientos (Un atacante puede enviar contenido malintencionado directamente al destino del re-direccionamiento, evitando así la lógica de la aplicación y cualquier validación realizada antes del re-direccionamiento).
1.11. Validar los tipos de datos esperados.
1.12. Valide los rangos de datos esperados.
1.13. Valide la longitud de los datos esperados.
1.14. Valide todas las entradas con una lista "blanca" de caracteres permitidos, siempre que sea posible.
1.15. Si se debe permitir la entrada de caracteres potencialmente peligrosos, asegúrese de implementar controles adicionales como la codificación de salida. Algunos ejemplos de caracteres peligrosos comunes son: < > " ' % ( ) & + \ \' \"
1.16. Si su rutina de validación estándar no puede validar las entradas siguientes, entonces deben ser verificadas independientemente.
    1.16.1. Compruebe si hay bytes nulos (%00)
    1.16.2. Compruebe si hay nuevos caracteres de línea (%0d, %0a, \r, \n)
    1.16.3. En los casos en que se admite la codificación de conjunto de caracteres extendida UTF-8, resuelva la representación alternativa como: %c0% ae%c0%ae/

2. Codificación de salida

2.1. Realice toda la codificación en un sistema de confianza (por ejemplo, El servidor)
2.2. Utilice una rutina estándar probada para cada tipo de codificación saliente
2.3. De forma contextual, codifique todos los datos de salida devueltos al cliente que se originaron fuera del límite de confianza de la aplicación. Por ejemplo, puede usar codificación de entidad HTML, pero no funciona en todos los casos
2.4. Codifique todos los caracteres a menos que se sepa que son seguros para el receptor.
2.5. “Desinfecte” contextualmente toda la salida de datos no confiables a las consultas de SQL, XML y LDAP
2.6. “Desinfecte” toda la salida de datos no confiables a los comandos del sistema operativo.

3. Autenticación y gestión de contraseñas

3.1. Requiera autenticación para todas las páginas y recursos, excepto aquellos específicamente destinados a ser públicos.
3.2. Todos los controles de autenticación deben aplicarse en un sistema de confianza (por ejemplo, El servidor)

3.3. Establezca y utilice servicios de autenticación estándar, probados siempre que sea posible.
3.4. Utilice una implementación centralizada para todos los controles de autenticación, incluidas las bibliotecas que llaman a servicios de autenticación externa.
3.5. Desagregue la lógica de autenticación del recurso solicitado y utilice la redirección hacia y desde el control de autenticación centralizado.
3.6. Todos los controles de autenticación deben fallar de forma segura.
3.7. Todas las funciones administrativas y de administración de cuentas deben ser al menos tan seguras como el mecanismo de autenticación principal.
3.8. Si su aplicación gestiona un almacén de credenciales, debe asegurarse de que sólo se almacenen los hashes de contraseñas salientes unidireccionalmente criptográficos y que la tabla / archivo que almacene las contraseñas y claves sólo pueda ser escrita por la aplicación. (No utilice el algoritmo MD5 si puede evitarse).
3.9. El hash de contraseña debe implementarse en un sistema de confianza (por ejemplo, El servidor).
3.10. Valide los datos de autenticación sólo al completar todos los datos de entrada, especialmente para las implementaciones de autenticación secuencial.
3.11. Las respuestas de error de autenticación no deberían indicar qué parte de los datos de autenticación es incorrecta. Por ejemplo, en lugar de "Nombre de usuario no válido" o "Contraseña no válida", utilice "Nombre de usuario y / o contraseña no válidos" para ambos.
3.12. Las respuestas de error deben ser idénticas tanto en la pantalla como en el código fuente.
3.13. Utilice autenticación para conexiones a sistemas externos que impliquen información o funciones sensibles.
3.14. Las credenciales de autenticación para acceder a los servicios externos a la aplicación deben ser cifradas y almacenadas en una ubicación protegida en un sistema de confianza (por ejemplo, El servidor). El código fuente NO es una ubicación segura.
3.15. Utilice sólo solicitudes HTTP POST para transmitir credenciales de autenticación.
3.16. Sólo envíe contraseñas no temporales a través de una conexión cifrada o datos cifrados, como en un correo electrónico cifrado. Las contraseñas temporales asociadas con los restablecimientos de correo electrónico pueden ser una excepción.
3.17. Aplique los requisitos de complejidad de contraseñas establecidos por la política o la normativa. Las credenciales de autenticación deben ser suficientemente fuertes para resistir ataques típicos de las amenazas en el entorno desplegado. (Por ejemplo, que requieren el uso de caracteres alfabéticos, así como numéricos y / o especiales).
3.18. Aplique los requisitos de longitud de contraseña establecidos por la política o la regulación. Ocho caracteres es de uso común, pero 16 es mejor o considerar el uso de frases de paso de varias palabras.
3.19. La entrada de la contraseña debe estar oscurecida en la pantalla del usuario. (Por ejemplo, en formularios web, utilice el tipo de entrada "contraseña").
3.20. Imponga la desactivación de la cuenta después de un número establecido de intentos de inicio de sesión no válidos (por ejemplo, cinco intentos son comunes). La cuenta debe estar deshabilitada durante un período de tiempo suficiente para desalentar la adivinación de la fuerza bruta de las credenciales, pero no tanto como permitir que un ataque de denegación de servicio se realice.
3.21. Las operaciones de reinicio y cambio de contraseñas requieren el mismo nivel de controles que la creación y autenticación de cuentas.
3.22. Las preguntas de restablecimiento de contraseña deben tener respuestas suficientemente aleatorias. (Por ejemplo, "libro favorito" es una mala pregunta porque "La Biblia" es una respuesta muy común).
3.23. Si utiliza restablecimientos basados en correo electrónico, sólo envíe correo electrónico a una dirección pre-registrada con un enlace / contraseña temporal.
3.24. Las contraseñas temporales y los enlaces deben tener un corto tiempo de caducidad.
3.25. Imponga el cambio de contraseñas temporales en el siguiente uso.
3.26. Notifique a los usuarios cuando se produce un restablecimiento de contraseña.

3.27. Evite la reutilización de la contraseña.
3.28. Las contraseñas deben tener al menos un día de antigüedad antes de que puedan cambiarse, para evitar ataques a la reutilización de contraseñas.
3.29. Imponga cambios de contraseñas basados en los requisitos establecidos en la política o la regulación. Los sistemas críticos pueden requerir cambios más frecuentes. El tiempo entre restablecimientos debe ser controlado administrativamente.
3.30. Desactive la funcionalidad "Recuérdeme" para los campos de contraseña.
3.31. El último uso (exitoso o no exitoso) de una cuenta de usuario debe ser reportado al usuario en su siguiente inicio de sesión correcto.
3.32. Implemente monitoreo para identificar ataques contra múltiples cuentas de usuario, utilizando la misma contraseña. Este patrón de ataque se utiliza para evitar los bloqueos estándar, cuando los ID de usuario pueden ser cosechados o adivinados.
3.33. Fuerce volver a autenticar a los usuarios antes de realizar operaciones críticas.
3.34. Utilice autenticación multi-factor para cuentas transaccionales altamente sensibles o de alto valor.
3.35. Si utiliza código de terceros para la autenticación, inspeccione el código con cuidado para asegurarse de que no se vea afectado por ningún código malicioso.

4. Gestión de la sesión

4.1. Utilice los controles de administración de sesiones del servidor o del marco de trabajo. La aplicación sólo debe reconocer estos identificadores de sesión como válidos.
4.2. La creación del identificador de sesión siempre debe realizarse en un sistema de confianza (por ejemplo, El servidor).
4.3. Los controles de gestión de sesiones deben utilizar algoritmos bien analizados que aseguren los identificadores de sesión lo suficientemente aleatorios.
4.4. Establezca el dominio y la ruta de acceso para las cookies que contengan identificadores de sesión autenticados con un valor debidamente restringido para el sitio.
4.5. La funcionalidad de cierre de sesión debe finalizar completamente la sesión o la conexión asociada.
4.6. La funcionalidad de cierre de sesión debe estar disponible en todas las páginas protegidas por autorización.
4.7. Establezca un tiempo de inactividad de la sesión que sea lo más corto posible, basado en el equilibrio entre los riesgos y los requisitos funcionales del negocio. En la mayoría de los casos no debe ser más de varias horas.
4.8. Prohibir los inicios de sesión persistentes y hacer cumplir las terminaciones periódicas de la sesión, incluso cuando la sesión está activa. Especialmente para aplicaciones que soportan conexiones enriquecidas de red o que se conectan a sistemas críticos. Los tiempos de terminación deben respaldar los requisitos del negocio y el usuario debe recibir notificación suficiente para mitigar los impactos negativos.
4.9. Si se estableció una sesión antes de iniciar sesión, cierre esa sesión y establezca una nueva sesión después de un inicio de sesión correcto.
4.10. Genere un nuevo identificador de sesión en cualquier re-autenticación.
4.11. No permita inicios de sesión simultáneos con el mismo ID de usuario.
4.12. No exponga identificadores de sesión en URL, mensajes de error o registros. Los identificadores de sesión sólo deben estar ubicados en el encabezado de la cookie HTTP. Por ejemplo, no pase identificadores de sesión como parámetros GET.
4.13. Proteja los datos de sesión del servidor de acceso no autorizado, por otros usuarios del servidor, mediante la implementación de controles de acceso adecuados en el servidor.
4.14. Genere un nuevo identificador de sesión y desactive periódicamente el antiguo. (Esto puede mitigar ciertos escenarios de secuestro de sesión donde el identificador original estaba comprometido).
4.15. Genere un nuevo identificador de sesión si la seguridad de la conexión cambia de HTTP a HTTPS, como puede ocurrir durante la autenticación. Dentro de una aplicación, se recomienda utilizar consistentemente HTTPS en lugar de cambiar entre HTTP a HTTPS.

4.16. Complemente la gestión de sesiones estándar para operaciones sensibles del lado del servidor, como la administración de cuentas mediante la utilización de tokens o parámetros aleatorios fuertes por sesión. Este método se puede utilizar para prevenir ataques Cross Site Request Forgery.
4.17. Complemente la gestión de sesión estándar para operaciones altamente sensibles o críticas mediante la utilización de tokens o parámetros aleatorios fuertes por petición, en contraposición por sesión.
4.18. Establezca el atributo "seguro" para las cookies transmitidas a través de una conexión TLS.
4.19. Establezca las cookies con el atributo HttpOnly, a menos que requiera específicamente scripts de cliente en su aplicación para leer o establecer el valor de una cookie.

5. Control de acceso

5.1. Utilice únicamente objetos del sistema de confianza, por ejemplo: objetos de sesión de servidor, para tomar decisiones de autorización de acceso.
5.2. Utilice un único componente de todo el sitio para verificar la autorización de acceso. Esto incluye bibliotecas que llaman a servicios de autorización externos.
5.3. Los controles de acceso deben fallar de forma segura.
5.4. Niegue todos los accesos si la aplicación no puede acceder a su información de configuración de seguridad.
5.5. Haga cumplir los controles de autorización en cada solicitud, incluidos los realizados por secuencias de comandos de servidor, "include" y las solicitudes de las “tecnologías ricas” del cliente, como AJAX y Flash.
5.6. Segregue la lógica privilegiada de otro código de aplicación.
5.7. Restrinja el acceso a archivos u otros recursos, incluidos los que están fuera del control directo de la aplicación, solo a usuarios autorizados.
5.8. Restrinja el acceso a las URL protegidas únicamente a los usuarios autorizados.
5.9. Restrinja el acceso a funciones protegidas únicamente a usuarios autorizados.
5.10. Restrinja las referencias directas a los usuarios autorizados.
5.11. Restrinja el acceso a servicios sólo a usuarios autorizados.
5.12. Restrinja el acceso a los datos de la aplicación únicamente a los usuarios autorizados.
5.13. Restrinja el acceso a los atributos de datos y de usuario y a la información de políticas utilizados por los controles de acceso.
5.14. Restrinja la información de configuración relevante de seguridad de acceso a sólo usuarios autorizados.
5.15. Las representaciones de las reglas de control de acceso en la implementación y presentación del servidor deben coincidir.
5.16. Si los datos de estado deben almacenarse en el cliente, utilice el cifrado y la comprobación de integridad en el lado del servidor para detectar la manipulación del estado.
5.17. Aplique flujos lógicos de aplicación para cumplir con las reglas de negocio.
5.18. Limite el número de transacciones que un solo usuario o dispositivo puede realizar en un período de tiempo dado. Las transacciones / tiempo deben estar por encima de los requerimientos reales del negocio, pero lo suficientemente bajos como para disuadir a los ataques automatizados.
5.19. Utilice el encabezado "referer" como un chequeo suplementario únicamente, nunca debe ser la única verificación de autorización, ya que puede ser falsificada.
5.20. Si se permiten sesiones largas autenticadas, periódicamente vuelva a validar la autorización de un usuario para asegurarse de que sus privilegios no han cambiado y, de haberlo hecho, obligué volver a autenticar.
5.21. Implemente auditoría de cuentas e imponga la desactivación de cuentas no utilizadas (p. Ej., Después de no más de 30 días a partir de la caducidad de la contraseña de una cuenta).
5.22. La aplicación debe soportar la inhabilitación de las cuentas y la finalización de las sesiones cuando cesa la autorización (por ejemplo, Cambios en el rol, la situación laboral, el proceso empresarial, etc.).

5.23. Las cuentas de servicio o las cuentas que soportan conexiones hacia o desde sistemas externos deben tener el menor privilegio posible.
5.24. Cree una directiva de control de acceso para documentar las reglas empresariales de la aplicación, los tipos de datos y los criterios y/o procesos de autorización de acceso para que el acceso pueda ser provisto y controlado adecuadamente. Esto incluye la identificación de los requisitos de acceso para los datos y los recursos del sistema.

6. Prácticas Criptográficas

6.1. Todas las funciones criptográficas utilizadas para proteger los secretos del usuario de la aplicación deben implementarse en un sistema de confianza (por ejemplo, El servidor).
6.2. Proteja los secretos maestros del acceso no autorizado.
6.3. Los módulos criptográficos deben fallar de forma segura.
6.4. Todos los números aleatorios, nombres de archivos aleatorios, GUID aleatorios y cadenas aleatorias deben generarse usando el generador de números aleatorios aprobado del módulo criptográfico cuando estos valores aleatorios están destinados a ser no adivinables.
6.5. Los módulos criptográficos utilizados por la aplicación deben ser compatibles con FIPS 140-2 o un estándar equivalente. (Ver http://csrc.nist.gov/groups/STM/cmvp/validation.html)
6.6. Establezca y utilice una política y un proceso para la gestión de claves criptográficas.

7. Gestión y registro de errores

7.1. No divulgue información confidencial en respuestas de error, incluyendo detalles del sistema, identificadores de sesión o información de cuenta.
7.2. Utilice controladores de error que no muestran información de depuración o de seguimiento de pila.
7.3. Implemente mensajes de error genéricos y utilice páginas de error personalizadas.
7.4. La aplicación debe manejar errores de aplicación y no depender de la configuración del servidor.
7.5. Libere correctamente la memoria asignada cuando se producen condiciones de error.
7.6. La lógica de manejo de errores asociada con los controles de seguridad debe denegar el acceso por defecto.
7.7. Todos los controles de registro deben implementarse en un sistema de confianza (por ejemplo, El servidor).
7.8. Los controles de registro deben admitir el éxito y el error de los eventos de seguridad especificados.
7.9. Asegúrese de que los registros contengan datos importantes de eventos de registro.
7.10. Asegúrese de que las entradas de registro que incluyan datos no confiables no se ejecuten como código en la interfaz o el software de visualización de registro.
7.11. Restrinja el acceso a registros sólo a personas autorizadas.
7.12. Utilice una rutina maestra para todas las operaciones de registro.
7.13. No almacene información confidencial en registros, incluidos detalles innecesarios del sistema, identificadores de sesión o contraseñas.
7.14. Asegurar que existe un mecanismo para llevar a cabo el análisis del registro.
7.15. Registe todas las fallas de validación de entrada.
7.16. Registre todos los intentos de autenticación, especialmente los errores.
7.17. Registre todas las fallas de control de acceso.
7.18. Registre todos los eventos de manipulación aparente, incluidos cambios inesperados en los datos de estado.
7.19. Registre intentos de conexión con tokens de sesión no válidos o caducados.
7.20. Registre todas las excepciones del sistema.
7.21. Registre todas las funciones administrativas, incluidos los cambios en la configuración de seguridad.
7.22. Registrar todos los fallos de conexión TLS del backend.
7.23. Registre todos los fallos en el módulo criptográfico.

7.24. Utilice una función de hash criptográfico para validar la integridad de la entrada de registro

8. Protección de datos

8.1. Implemente el privilegio mínimo, restrinja los usuarios únicamente a la funcionalidad, los datos y la información del sistema que se requieren para realizar sus tareas.
8.2. Proteja todas las copias en caché o temporales de los datos confidenciales almacenados en el servidor de acceso no autorizado y purgue los archivos de trabajo temporales una vez que ya no son necesarios.
8.3. Cifre la información almacenada altamente sensible, como los datos de verificación de autenticación, incluso en el lado del servidor. Siempre use algoritmos bien revisados, vea "Prácticas Criptográficas" para orientación adicional.
8.4. Proteja el código fuente del servidor de ser descargado por un usuario.
8.5. No guarde las contraseñas, cadenas de conexión u otra información sensible en texto claro o en cualquier forma no criptográficamente segura en el lado del cliente. Esto incluye incrustar en formatos inseguros como: MS viewstate, Adobe flash o código compilado.
8.6. Elimine los comentarios en el código de producción accesible al usuario que puedan revelar el sistema backend u otra información sensible.
8.7. Elimine la documentación innecesaria de la aplicación y del sistema, ya que esto puede revelar información útil a los atacantes.
8.8. No incluya información confidencial en los parámetros de solicitud HTTP GET.
8.9. Deshabilite las funciones de completar automáticas en los formularios que se espera contengan información confidencial, incluida la autenticación.
8.10. Deshabilite la caché del lado del cliente en las páginas que contienen información confidencial. puede utilizarse Cache-Control: no-store, junto con el control de encabezado HTTP "Pragma: no-cache", que es menos efectivo, pero es compatible con HTTP / 1.0.
8.11. La aplicación debe admitir la eliminación de datos confidenciales cuando ya no se necesiten. (Por ejemplo, información personal o ciertos datos financieros).
8.12. Implemente controles de acceso adecuados para los datos confidenciales almacenados en el servidor. Esto incluye datos almacenados en caché, archivos temporales y datos que sólo deben ser accesibles por usuarios específicos del sistema

9. Comunicación segura

9.1. Implemente el cifrado para la transmisión de toda la información sensible. Esto debería incluir TLS para proteger la conexión y puede complementarse con encriptación discreta de archivos confidenciales o conexiones no basadas en HTTP.
9.2. Los certificados TLS deben ser válidos y tener el nombre de dominio correcto, no estar expirados, e instalarse con certificados intermedios cuando sea necesario.
9.3. Las conexiones TLS fallidas no deben recurrir a una conexión insegura.
9.4. Utilice conexiones TLS para todo el contenido que requiera acceso autenticado y para cualquier otra información confidencial.
9.5. Utilice TLS para conexiones a sistemas externos que impliquen información o funciones sensibles.
9.6. Utilice una implementación TLS estándar única configurada correctamente.
9.7. Especifique codificaciones de caracteres para todas las conexiones.

10. Configuración del sistema

10.1. Asegúrese de que los servidores, los marcos de trabajo (frameworks) y los componentes del sistema estén ejecutando la versión aprobada más reciente.
10.2. Asegúrese de que los servidores, los marcos de trabajo (frameworks) y los componentes del sistema tengan todos los parches emitidos para la versión en uso
10.3. Desactive las listas de directorios.

10.4. Restrinja el servidor web, el proceso y las cuentas de servicio a los privilegios mínimos posibles.
10.5. Cuando se producen excepciones, falla de forma segura.
10.6. Elimine toda la funcionalidad y archivos innecesarios.
10.7. Elimine el código de prueba o cualquier funcionalidad no destinada a la producción, antes de la implementación.
10.8. Evite la divulgación de su estructura de directorios en el archivo robots.txt colocando directorios no destinados a la indexación pública en un directorio padre aislado. Deshabilite ese directorio padre completo en el archivo robots.txt en lugar de deshabilitar cada directorio individual.
10.9. Defina qué métodos HTTP, Get o Post, la aplicación soportará y si se tratará de forma diferente en diferentes páginas de la aplicación.
10.10. Deshabilite los métodos HTTP innecesarios, como las extensiones WebDAV. Si se requiere un método HTTP extendido que admita el manejo de archivos, utilice un mecanismo de autenticación bien revisado.
10.11. Si el servidor web maneja tanto HTTP 1.0 como 1.1, asegúrese de que ambos estén configurados de forma similar o asegúrese de comprender cualquier diferencia que pueda existir (por ejemplo, el manejo de métodos HTTP extendidos).
10.12. Elimine información innecesaria de los encabezados de respuesta HTTP relacionados con el SO, la versión de servidor web y los marcos de aplicaciones.
10.13. El almacén de configuración de seguridad para la aplicación debe poder imprimirse en forma legible para soportar la auditoría.
10.14. Implemente un sistema de gestión de activos y registre los componentes del sistema en él.
10.15. Aísle los entornos de desarrollo de la red de producción y proporcione acceso sólo a grupos de desarrollo y prueba autorizados. Los entornos de desarrollo a menudo se configuran de forma menos segura que los entornos de producción y los atacantes pueden utilizar esta diferencia para descubrir las debilidades compartidas o como una vía para la explotación.
10.16. Implemente un sistema de control de cambio de software para gestionar y registrar cambios en el código tanto en desarrollo como en producción

11. Seguridad de base de datos

11.1. Utilice consultas parametrizadas “fuertemente tipadas”.
11.2. Utilice la validación de entrada y la codificación de salida y asegúrese de tratar los caracteres meta. Si estos fallan, no ejecute el comando de base de datos
11.3. Asegúrese de que las variables están “fuertemente tipadas”
11.4. La aplicación debe utilizar el nivel de privilegio más bajo posible al acceder a la base de datos.
11.5. Use credenciales seguras para acceso a bases de datos.
11.6. Las cadenas de conexión no deben codificarse en la aplicación. Las cadenas de conexión deben almacenarse en un archivo de configuración independiente en un sistema de confianza y deben ser cifradas.
11.7. Cierre la conexión tan pronto como sea posible.
11.8. Elimine o cambiar todas las contraseñas administrativas de base de datos predeterminadas. Utilice contraseñas fuertes o implemente autenticación de múltiples factores
11.9. Desactive toda la funcionalidad de base de datos innecesaria (por ejemplo, procedimientos o servicios almacenados innecesarios, paquetes de servicios públicos, sólo instale el conjunto mínimo de características y opciones requeridas (reducción de área de superficie)).
11.10. Elimine contenido de proveedor predeterminado innecesario (por ejemplo, esquemas de ejemplo).
11.11. Deshabilite las cuentas predeterminadas que no sean necesarias para dar soporte a los requisitos empresariales

12. Gestión de archivos

12.1. No pase los datos suministrados por el usuario directamente a ninguna función de inclusión dinámica.
12.2. Requiera autenticación antes de permitir que se cargue un archivo.
12.3. Limite el tipo de archivos que se pueden cargar sólo a los tipos que se necesitan para fines comerciales.
12.4. Valide que los archivos cargados son el tipo esperado verificando los encabezados de los archivos. Comprobar el tipo de archivo por extensión solo no es suficiente.
12.5. No guarde archivos en el mismo contexto web que la aplicación. Los archivos deben ir al servidor de contenido o a la base de datos.
12.6. Impida o restrinja la carga de cualquier archivo que pueda ser interpretado por el servidor web.
12.7. Desactive los privilegios de ejecución en directorios de subida de archivos.
12.8. Al hacer referencia a los archivos existentes, utilice una lista blanca de nombres y tipos de archivos permitidos. Valide el valor del parámetro que se está pasando y si no coincide con uno de los valores esperados, rechácelo o utilice un valor de codificación de archivo predeterminado para el contenido.
12.9. No pase los datos suministrados por el usuario en un re-direccionamiento dinámico. Si esto debe ser permitido, entonces el re-direccionamiento debe aceptar sólo las URL de ruta relativa validadas.
12.10. No pase directorios o rutas de archivo, utilice valores de índice asignados a una lista predefinida de rutas.
12.11. Nunca envíe la ruta absoluta del archivo al cliente.
12.12. Asegúrese de que los archivos y recursos de la aplicación sean de sólo lectura.
12.13. Analizar los archivos cargados por el usuario en busca de virus y malware.

13. Gestión de la memoria

13.1. Utilice control de entrada y salida para datos no confiables.
13.2. Compruebe que el búfer sea tan grande como se haya especificado.
13.3. Compruebe los límites del búfer si llama una función de copiado de memoria en un bucle y asegúrese de que no hay peligro de escribir más allá del espacio asignado.
13.4. Trunque todas las cadenas de entrada a una longitud razonable antes de pasarlas a las funciones de copia y concatenación.
13.5. Libere explícitamente todos los recursos que no dependen del recolector de basura (garbage collector). (Por ejemplo, objetos de conexión, manejadores de archivos, etc.)
13.6. Utilice pilas no ejecutables cuando estén disponibles.
13.7. Evite el uso de funciones vulnerables conocidas (por ejemplo, printf, strcat, strcpy, etc.)
13.8. Libere correctamente la memoria asignada al completar las funciones y en todos los puntos de salida.

14. Practicas generales de codificación

14.1. Utilice código administrado probado y aprobado en lugar de crear código nuevo no administrado para tareas comunes.
14.2. Utilice APIs incorporadas específicas para realizar tareas del sistema operativo. No permita que la aplicación emita comandos directamente al sistema operativo, especialmente mediante el uso de shells de comandos iniciados por la aplicación.
14.3. Utilice sumas de comprobación o hashes para verificar la integridad del código interpretado, bibliotecas, archivos ejecutables y archivos de configuración.
14.4. Utilice el bloqueo para evitar múltiples solicitudes simultáneas o utilizar un mecanismo de sincronización para evitar condiciones de carrera.
14.5. Proteja las variables y recursos compartidos de acceso concurrente inadecuado.

14.6. Explícitamente inicialice todas sus variables y otros almacenes de datos, ya sea durante la declaración o justo antes de la primera utilización.
14.7. En los casos en que la aplicación debe ejecutarse con privilegios elevados, aumentar los privilegios lo más tarde posible, y soltarlos tan pronto como sea posible.
14.8. Evite errores de cálculo al comprender la representación subyacente de su lenguaje de programación y cómo interactúa con el cálculo numérico. Preste mucha atención a las discrepancias de tamaño de bytes, precisión, distinciones firmadas / no firmadas, truncamiento, conversión y conversión entre tipos, cálculos de "no un número" y cómo su idioma maneja números que son demasiado grandes o demasiado pequeños para su representación subyacente
14.9. No pase los datos suministrados por el usuario a ninguna función de ejecución dinámica
14.10. Restrinja a los usuarios de generar código nuevo o alterar el código existente
14.11. Revise todas las aplicaciones secundarias, código de terceros y bibliotecas para determinar la necesidad del negocio y validar la funcionalidad segura, ya que pueden introducir nuevas vulnerabilidades.
14.12. Implemente una actualización segura. Si la aplicación utiliza actualizaciones automáticas, utilice firmas criptográficas para su código y asegúrese de que los clientes de descarga verifiquen esas firmas. Utilice canales cifrados para transferir el código desde el servidor host.