Resolución Número 430
(Julio 17 de 2020)

"Por la cual se adopta el documento T-DT-008 Protocolo para asegurar los desarrollos de software de TRANSMILENIO S.A"

EL JEFE DE LA OFICINA ASESORA DE PLANEACIÓN DE LA EMPRESA DE TRANSPORTE DEL TERCER MILENIO "TRANSMILENIO S.A.", En uso de sus facultades conferidas mediante la Resolución 342 del 16 de junio de 2020, y

CONSIDERANDO:

Que de conformidad con lo señalado en el artículo segundo del Acuerdo 4 de 1999, corresponde a TRANSMILENIO S.A., la gestión, organización y planeación del servicio de transporte público masivo urbano de pasajeros en el Distrito Capital y su área de influencia, bajo la modalidad de transporte terrestre automotor.

Que cumpliendo con lo ordenado en el parágrafo único del artículo 1º de la Ley 87 de 1993, se adoptó el Manual de Procedimientos de TRANSMILENIO S.A.

Que, siendo TRANSMILENIO S.A., el ente gestor del Sistema Integrado de Transporte Público, considera necesario actualizar los Manuales de Procedimientos de las diferentes dependencias de la Entidad, con el objeto de ajustarlos a los nuevos parámetros documentales, necesidades y desarrollo del Sistema.

En mérito de lo expuesto,

RESUELVE:

ARTÍCULO 1°: Adoptar el siguiente documento con el nombre, código y versión que se registran a continuación:

CÓDIGO	VERSIÓN	NOMBRE
T-DT-008	0	Protocolo para asegurar los desarrollos de software de TRANSMILENIO S.A.

ARTÍCULO 2°: La presente Resolución rige a partir de su publicación en la Gaceta Distrital.

PUBLÍQUESE Y CÚMPLASE

Dada en Bogotá, D.C., a los diecisiete (17) días del mes de julio de dos mil veinte (2020).

SOFÍA ZARAMA VALENZUELA
Jefe de Oficina Asesora de Planeación

TABLA DE CONTENIDO

1. OBJETO 2
2. ALCANCE 2
3. RESPONSABLE 3
4. DOCUMENTOS DE REFERENCIA 3
5. DEFINICIONES 3
6. CONDICIONES GENERALES 7
7. PROTOCOLO PARA ASEGURAR LOS DESARROLLOS DE SOFTWARE DE TRANSMILENIO S.A. 7
7.1 Requisitos de Validación de Entrada 8
7.2 Codificación de Salida 9
7.3 Autenticación y gestión de contraseñas 10
7.4 Manejo de Sesión 13
7.5 Control de acceso 14
7.6 Criptografía 16
7.7 Manejo de errores y gestión de logs 17
7.8 Protección de datos 18
7.9 Seguridad en las comunicaciones 19
7.10 Configuración del sistema para desarrollo de software 20
7.11 Seguridad en base de datos 21
7.12 Gestión de Archivos 22
7.13 Manejo de Memoria 23
7.14 Políticas generales de desarrollo 24

1. OBJETO

Definir los requisitos mínimos de seguridad que deben aplicarse sobre los desarrollos de software que se realicen en TRANSMILENIO S.A., con el fin prevenir y reducir la existencia de vulnerabilidades en el código fuente e infraestructura.

2. ALCANCE

Este documento cubre los requisitos mínimos de seguridad relacionados con:

- ✓ Validación de Entrada
- ✓ Codificación de Salida
- ✓ Autenticación y gestión de contraseñas
- ✓ Manejo de Sesión
- ✓ Control de acceso
- ✓ Criptografía
- ✓ Manejo de errores y gestión de logs
- ✓ Protección de datos
- ✓ Seguridad en las comunicaciones
- ✓ Configuración del sistema para desarrollo de software
- ✓ Seguridad en base de datos
- ✓ Gestión de Archivos
- ✓ Manejo de Memoria
- ✓ Políticas generales de desarrollo

Este protocolo aplica a todos los usuarios internos de TRANSMILENIO S.A. ya sean funcionarios de planta, contratistas por honorarios, asesores, consultores, practicantes, y otros trabajadores, incluyendo las empresas que presten servicios a la Entidad y que lleven a cabo desarrollo de software.

3. RESPONSABLE

Los Profesionales Especializados Grado 06 de Seguridad Informática y Coordinador de Procesos Corporativos de la Dirección de TIC, son los responsables por la actualización y aplicación de este documento a su vez el Director de TIC dará su estricta implementación y cumplimiento.

La revisión y/o actualización de este protocolo debe realizarse cuando se considere pertinente por parte de los responsables de su aplicación y cumplimiento.

4. DOCUMENTOS DE REFERENCIA

- **ISO-IEC-27001/2013:** por la cual se estandariza la implementación del Sistema de Gestión de la Seguridad de la Información (SGSI).
- **OWASP Secure Coding Practices Quick Reference:** Donde se promueven las mejores prácticas de desarrollo seguro de software.
- **M-DT-001.** Manual de Políticas de Seguridad y privacidad de la Información de TRANSMILENIO S.A.

5. DEFINICIONES

Administración de archivos: conjunto de controles que cubren la interacción entre el código y otros archivos del sistema.

Administración de memoria: conjunto de controles que abordan el uso de memoria y búfer.

Administración de sesiones: conjunto de controles que ayudan a garantizar que las aplicaciones web manejen las sesiones HTTP de manera segura.

Agente de amenazas: cualquier entidad que pueda tener un impacto negativo en el sistema. Este puede ser un usuario malicioso que quiere comprometer los controles de seguridad del sistema; sin embargo, también podría ser un mal uso accidental del sistema o una amenaza más física como incendio o inundación.
Autenticación: conjunto de controles que se utilizan para verificar la identidad de un usuario u otra entidad que interactúa con el software.

Autenticación multi-factor: proceso de autenticación que requiere que el usuario produzca múltiples tipos distintos de credenciales. Por lo general, esto se basa en algo que tienen (por ejemplo, tarjeta inteligente), algo que saben (por ejemplo, un pin) o algo que son (por ejemplo, datos de un lector biométrico).

Autenticación secuencial: cuando los datos de autenticación se solicitan en páginas sucesivas en lugar de solicitarse todos a la vez en una sola página.

Canonicalizar: es elegir la mejor URL para mostrar un mismo contenido.

Caso de abuso: describe los usos indebidos intencionales y no intencionales del software. Los casos de abuso deberían cuestionar los supuestos del diseño del sistema.

Caracter peligroso: cualquier carácter o representación codificada de un carácter que pueda afectar el funcionamiento previsto de la aplicación o el sistema asociado si se interpreta que tiene un significado especial, fuera del uso previsto del carácter. Estos caracteres pueden usarse para:

- Alterar la estructura del código o declaraciones existentes.
- Insertar nuevo código no deseado
- Alterando caminos
- Causar resultados inesperados de las funciones o rutinas del programa.
- Causando condiciones de error
- Tener cualquiera de los efectos anteriores en aplicaciones o sistemas posteriores

Codificación de entidad HTML: el proceso de reemplazar ciertos caracteres ASCII con sus equivalentes de entidad HTML. Por ejemplo, la codificación reemplazaría el carácter menor que "<" con el equivalente HTML "& lt;". Las entidades HTML son "inertes" en la mayoría de los intérpretes, especialmente los navegadores, que pueden mitigar ciertos ataques del lado del cliente.

Codificación de salida: conjunto de controles que abordan el uso de la codificación para garantizar que la salida de datos de la aplicación sea segura.

Codificación de salida contextual: codificación de datos de salida en función de cómo será utilizada por la aplicación. Los métodos específicos varían según la forma en que se utilizan los datos de salida. Si los datos se van a incluir en la respuesta al cliente, tenga en cuenta escenarios de inclusión como: el cuerpo de un documento HTML, un atributo HTML, dentro de JavaScript, dentro de un CSS o en una URL. También debe tener en cuenta otros casos de uso como consultas.

Confidencialidad: para garantizar que la información se divulgue solo a las partes autorizadas.

Configuración del sistema: conjunto de controles que ayudan a garantizar que los componentes de infraestructura que admiten el software se implementen de forma segura.

Consultas (Queries) parametrizadas (declaraciones preparadas): Serie de líneas de código que permite consultar los datos dentro de la base de datos, mantiene la consulta y los datos separados mediante el uso de marcadores de posición. La estructura de consulta se define con marcadores de posición, la instrucción SQL se envía a la base de datos y se prepara, y luego la instrucción preparada se combina con los valores de los parámetros. El previene la consulta de ser alterado, porque los valores de los parámetros se combinan con la declaración compilada, no con una cadena SQL.

Control de acceso: conjunto de controles que otorgan o niegan a un usuario u otra entidad el acceso a un recurso del sistema. Esto generalmente se basa en roles jerárquicos y privilegios individuales dentro de un rol, pero también incluye interacciones de sistema a sistema.

Controles de seguridad: acción que mitiga una vulnerabilidad potencial y ayuda a garantizar que el software se comporte solo de la manera esperada.

Datos de estado: cuando la aplicación o el servidor utilizan datos o parámetros para emular una conexión persistente o rastrear el estado de un cliente a través de un proceso o transacción de solicitud múltiple.

Datos de eventos de registro de log: Información registrada como una traza de las actividades realizadas en los sistemas de información. Esta información debe incluir lo siguiente:

- Marca de tiempo de un componente de sistema confiable
- Calificación de gravedad para cada evento
- Etiquetado de eventos relevantes para la seguridad, si se mezclan con otras entradas de registro
- Identidad de la cuenta / usuario que causó el evento.
- Dirección IP de origen asociada con la solicitud
- Resultado del evento (éxito o fracaso)
- Descripción del evento.

Desinfectar datos: el proceso de hacer que los datos potencialmente dañinos sean seguros mediante el uso de la eliminación de datos, el reemplazo, la codificación o el escape de los caracteres.

Disponibilidad: una medida de accesibilidad y usabilidad de un sistema.

Explotar: aprovechar una vulnerabilidad. Por lo general, esta es una acción intencional diseñada para comprometer los controles de seguridad del software al aprovechar una vulnerabilidad.

Impacto: una medida del efecto negativo para el negocio que resulta de la ocurrencia de un evento no deseado; cuál sería el resultado de una vulnerabilidad siendo explotada.

Integridad: la garantía de que la información es precisa, completa y válida, y que no ha sido alterada por una acción no autorizada.

Límites de confianza: Normalmente, un límite de confianza constituye los componentes del sistema bajo su control directo. Todas las conexiones y datos de sistemas fuera de su control directo, incluidos todos los clientes y sistemas administrados por otras partes, deben considerarse no confiables y validados en el límite, antes de permitir una mayor interacción del sistema.

Mitigar: Pasos para reducir la gravedad de una vulnerabilidad. Estos pueden incluir eliminar una vulnerabilidad, hacer que sea más difícil de explotar o reducir el impacto negativo de una explotación exitosa.

Prácticas criptográficas: un conjunto de controles que garantizan que las operaciones criptográficas dentro de la aplicación se manejen de manera segura.

Prácticas generales de codificación: un conjunto de controles que cubren prácticas de codificación que no encajan fácilmente en otras categorías.

Protección de datos: un conjunto de controles que ayudan a garantizar que el software maneje el almacenamiento de información de manera segura.

Requisitos de seguridad: un conjunto de requisitos funcionales y de diseño que ayudan a garantizar que el software se construya e implemente de manera segura.

Seguridad de la base de datos: un conjunto de controles que garantizan que el software interactúa con una base de datos de manera segura y que la base de datos está configurada de forma segura.
Manejo y registro de errores: un conjunto de prácticas que garantizan que la aplicación maneje los errores de manera segura y realice un registro de eventos adecuado.

Seguridad de comunicación: un conjunto de controles que ayudan a garantizar que el software maneje el envío y la recepción de información de manera segura.

Sistema: un término genérico que abarca los sistemas operativos, el servidor web, los frameworks de aplicaciones y la infraestructura relacionada.

Validación de entrada: un conjunto de controles que verifican que las propiedades de todos los datos de entrada coincidan con lo que la aplicación espera, incluidos los tipos, longitudes, rangos, conjuntos de caracteres aceptables y no incluye caracteres peligrosos conocidos.

Vulnerabilidad: una debilidad que hace que el sistema sea susceptible a ataques o daños.

6. CONDICIONES GENERALES

- La Dirección de Tecnologías de la Información y Comunicaciones definirá las mejores prácticas de desarrollo seguro de software que la entidad deberá cumplir y aplicar en los proyectos de desarrollo de software. Estas prácticas definidas se convierten en políticas y requisitos exigidos por esta dependencia para cualquier proyecto de desarrollo de software que se lleve a cabo en TRANSMILENIO S.A.

- Las políticas descritas en el presente documento deberán aplicarse a todos los proyectos de desarrollo de software de TRANSMILENIO S.A. De igual forma, si existen aplicaciones que hayan sido desarrolladas previamente a la oficialización del presente documento, estas también deberán adoptar las practicas descritas según apliquen.

- Las prácticas de desarrollo mencionadas en este protocolo, deberán aplicarse a los proyectos de desarrollo de software durante su ciclo de desarrollo, desde la concepción del proyecto, como parte de los requerimientos no funcionales y pruebas de seguridad antes su producción. Por lo tanto, la aplicación de los requisitos de desarrollo seguro de software deberá revisarse en el producto antes de que se realice el paso a producción.

- Todos los proyectos de desarrollo de software que se lleven a cabo dentro de TRANSMILENIO S.A. deberán documentar las políticas y controles de seguridad que se aplicaron en el producto de software. Dicha documentación deberá justificar la aplicabilidad o no de las prácticas mencionadas en el presente documento.

- El presente protocolo debe aplicarse como verificación para todos los desarrollos de la Entidad antes de ser liberados en ambiente de producción.

7. PROTOCOLO PARA ASEGURAR LOS DESARROLLOS DE SOFTWARE DE TRANSMILENIO S.A.

A continuación, se describen las directrices de verificación de desarrollo seguro de software para TRANSMILENIO S.A. Todo desarrollador deberá verificar que se apliquen las directrices durante el ciclo de desarrollo de cualquier producto de software de la entidad.

Nota: El presente documento es una traducción al español adaptada a la entidad de la guía **OWASP Secure Coding Practices Quick Reference.**

7.1 Requisitos de Validación de Entrada

Se deberán cumplir con los siguientes requisitos de validación de entrada en formularios, campos o cualquier punto de inserción de datos de entrada en las aplicaciones:

- Toda validación de datos se deberá realizar del lado del servidor o punto confiable, nunca de forma local o scripts en el navegador.

- Todas las fuentes de datos deberán identificarse y clasificarse en confiables y no confiables. De igual forma, cuando se usen fuentes de datos que no sean confiables, entonces se deberá realizar su respectiva validación antes de ser procesados (por ejemplo, bases de datos, secuencias de archivos, etc.)

- En el caso que sea posible se deberá construir una rutina de validación de entrada centralizada para la aplicación.

- Se deben especificar conjuntos de caracteres adecuados, como UTF-8, para todas las fuentes de entrada.

- Se deberán codificar los datos en un conjunto de caracteres común antes de validarlos (Canonicalización)

- Todas las fallas de validación de los datos de entrada deberán resultar en el rechazo de la entrada.

- Si el sistema admite juegos de caracteres extendidos UTF-8 estos se deberán validar después de que se complete la decodificación UTF-8.

- Se deben validar todos los datos proporcionados por el cliente antes de procesarlos, esto incluye todos los parámetros, las URL y el contenido del encabezado HTTP (por ejemplo, nombres y valores de cookies). Se deben incluir en esta práctica también las respuestas automáticas de JavaScript, Flash u otro código embebido.

- Se deberá verificar que los valores de los encabezados en las solicitudes (Request) y respuestas (response) contengan solo caracteres ASCII.

- Los datos de las redirecciones deben validarse (un atacante puede enviar contenido malicioso directamente al objetivo de la redirección, evitando así la lógica de la aplicación y cualquier validación realizada antes de la redirección)

- Siempre se deberá validar:
  - Los tipos de datos esperados por la aplicación.
  - Rango de datos
  - Longitud de datos

- Las entradas deberán ser validadas a través de una lista "blanca" de caracteres permitidos, siempre que sea posible.

- Si se deben permitir caracteres de entrada potencialmente peligrosos, se deben implementar controles adicionales como codificación de salida (Output encoding), APIs seguras para tareas específicas y medición de la utilización de esos datos en toda la aplicación. Ejemplos de caracteres peligrosos comunes incluyen: <> "'% () & + \ \' \".

- Si la rutina de validación estándar no puede abordar los siguientes tipos de entradas, entonces deberán verificarse manualmente.
  - Bytes nulos (% 00)
  - Caracteres de nueva línea (% 0d, % 0a, \ r, \ n)
  - Caracteres de alteración de ruta "dot-dot-slash" (../ o .. \). En los casos en que se soporte la codificación de juego de caracteres extendido UTF-8, se debe adoptar una representación alternativa como: % c0% ae% c0% ae / (Utilice la canonicalización para adoptar la doble codificación u otras formas de ataques de ofuscación)

7.2 Codificación de Salida

Se deberán cumplir con los siguientes requisitos de codificación de salida de datos del producto de software:

- Siempre se deberá realizar toda la codificación del lado del servidor.

- Se deberá utilizar una rutina estándar probada para cada tipo de codificación saliente

- Se deben codificar de forma contextual la salida de todos los datos que se retornan al cliente y que se originaron fuera (del límite de confianza) de la aplicación. La codificación “HTML Entity” es un ejemplo, pero no funciona en todos los casos.

- Todos los caracteres deberán codificarse a menos que se sepa que son seguros para el intérprete utilizado.

- Se deben sanitizar de forma contextual todas las salidas de datos que no sean confiables y que se envíen como consultas de SQL, XML y LDAP.

- Sanitizar todas las salidas de datos que no sean confiables y que vayan a utilizar comandos del sistema operativo

7.3 Autenticación y gestión de contraseñas

Se deberán cumplir con los siguientes requisitos de autenticación y de gestión de contraseñas de las aplicaciones:

- Se debe requerir autenticación para todas las páginas y recursos, excepto aquellos específicamente destinados a ser públicos.

- Todos los controles de autenticación deben aplicarse en un sistema confiable (por ejemplo, del lado del servidor).

- Se deben establecer y utilizar servicios de autenticación estándar y probados, siempre que sea posible.

- Se deberán implementar los controles de autenticación de forma centralizada. Esto incluye las librerías que llaman a servicios de autenticación externos.

- Siempre se debe separar la lógica de la autenticación del recurso solicitado. En este caso también se debe usar la redirección hacia y desde el control de autenticación centralizado.

- Todos los controles de autenticación deben fallar de forma segura.

- Todas las funciones administrativas y de administración de cuentas deben ser al menos tan seguras como el mecanismo de autenticación principal.

- Si la aplicación administra el almacenamiento de credenciales (contraseñas), solo se deben almacenar los hashes de dichas contraseñas de forma unidireccional y criptográficamente fuerte. De igual forma la tabla o el archivo que almacene las contraseñas solo podrá ser modificado por la aplicación. (Los algoritmos de hash utilizados solo podrán contemplar SHA256 en adelante)

- El cálculo del hash de contraseña debe implementarse del lado del servidor.

- Los datos de autenticación deberán validarse solo al completar toda la entrada de datos, especialmente para implementaciones de autenticación secuencial.

- Las respuestas de falla de autenticación no deben indicar qué parte de los datos de autenticación es la incorrecta. Por ejemplo, en lugar de "Nombre de usuario no válido" o "Contraseña no válida",

simplemente use "Nombre de usuario y / o contraseña no válidos" para ambos. Las respuestas de error deben ser realmente idénticas tanto en la pantalla como en el código fuente.

- Se debe utilizar la autenticación para conexiones a sistemas externos que involucren información o funciones sensibles. Por ejemplo, todos los servicios web (web services) deberán implementarse con autenticación ya sea con certificado o con usuario y contraseña, utilizando suites criptográficas fuertes.

- Las credenciales de autenticación para acceder a servicios externos a la aplicación deben cifrarse y almacenarse en una ubicación protegida del lado del servidor. No se pueden almacenar credenciales en el código fuente.

- Se deben utilizar solo solicitudes HTTP POST para transmitir credenciales de autenticación.

- Solo se deben enviar contraseñas fijas a través de una conexión cifrada o como datos cifrados, como en un correo electrónico cifrado. Las contraseñas que son temporales y que están asociadas al restablecimiento de correo electrónico pueden ser una excepción.

- Se deberán cumplir los requisitos de complejidad de contraseña establecidos por la política de TRANSMILENIO S.A. Las credenciales de autenticación deberán ser suficientes para resistir los ataques que son típicos de las amenazas en el entorno implementado. (p. ej., requerir el uso de caracteres alfabéticos y numéricos y / o especiales)

- Se deben cumplir los requisitos de longitud de contraseña establecidos por la política de seguridad de TRANSMILENIO S.A.

- Al ingresar la contraseña en un formulario, esta debe estar oculta en la pantalla del usuario, ya sea con asteriscos o puntos. (por ejemplo, en los formularios web use el tipo de entrada "contraseña").

- Se debe aplicar la desactivación de la cuenta después de cinco intentos de inicio de sesión fallidos. La cuenta debe deshabilitarse durante un período de tiempo de 1 minuto para desalentar la suposición de credenciales por fuerza bruta, pero no tanto como para permitir que se realice un ataque de denegación de servicio.

- Las operaciones de restablecimiento y cambio de contraseña requieren el mismo nivel de controles que la creación y autenticación de la cuenta.

- Las preguntas de restablecimiento de contraseña deben admitir respuestas suficientemente aleatorias. (por ejemplo, "libro favorito" es una mala pregunta porque "La Biblia" es una respuesta muy común)

- Si se utilizan restablecimientos basados en correo electrónico, solo se debe enviar un correo electrónico a una dirección registrada previamente y con un enlace o contraseña temporal.

- Las contraseñas y enlaces temporales deben tener un tiempo de vencimiento de máximo 12 horas.

- Se debe aplicar el cambio de contraseñas temporales en el siguiente uso.

- Se debe notificar a los usuarios cuando se restablezca la contraseña, esto puede ser por correo electrónico o mensaje de texto según aplique.

- Se debe evitar la reutilización de la contraseña.

- Las contraseñas deben tener al menos un día de antigüedad antes de que puedan cambiarse, para evitar ataques a la reutilización de contraseñas.

- Se deben cumplir con los cambios de contraseña de acuerdo con la política de seguridad de TRANSMILENIO S.A. Los sistemas críticos pueden requerir cambios más frecuentes. El tiempo entre reinicios de contraseña debe controlarse administrativamente.

- Se debe desactivar la funcionalidad "recordarme" para los campos de contraseña.

- El último uso (exitoso o no exitoso) de una cuenta de usuario debe informarse al usuario en su próximo inicio de sesión exitoso. Esto incluye mostrar el mensaje de fecha y hora de ultimo inicio de sesión (exitoso o no exitoso).

- Se deben implementar actividades monitoreo para identificar ataques contra múltiples cuentas de usuario utilizando la misma contraseña. Este patrón de ataque se usa para omitir los bloqueos estándar, cuando los nombres de usuario se pueden recolectar o adivinar.

- Se deben cambiar todas las contraseñas predeterminadas y los nombres de usuario proporcionados por el proveedor. En su defecto se deben desactivar las cuentas asociadas.

- Si el software incluye operaciones críticas, entonces se debe volver a autenticar a los usuarios antes de realizar dichas operaciones.

- Se debe usar la autenticación multi-factor para cuentas transaccionales altamente sensibles o de alto valor.

- Si se usa un código de terceros para autenticación o manejo de sesión, inspeccione el código cuidadosamente para asegurarse de que no se vea afectado por ningún código malicioso. De igual forma, valide que las porciones de código o scripts reutilizados no tengan vulnerabilidades públicamente conocidas.

- Todas las aplicaciones que sean desarrolladas por cualquier área de TRANSMILENIO S.A.deberán integrarse con el directorio activo de la entidad para lo referente a autenticación. En caso de que se utilicen usuarios locales, estos deberán documentarse apropiadamente.

7.4 Manejo de Sesión

- Se deben utilizar los controles de administración de sesión del servidor. La aplicación solo debe reconocer estos identificadores de sesión como válidos.

- La creación del identificador de sesión siempre debe hacerse del lado del servidor.

- Los controles de gestión de sesión deben usar algoritmos revisados y evaluados públicamente, que garanticen identificadores de sesión suficientemente aleatorios.

- El dominio y la ruta de las cookies que contienen identificadores de sesión autenticados deben establecerse de tal forma que tengan un valor controlado/restringido únicamente para el sitio.

- La funcionalidad de cierre de sesión debe finalizar por completo la sesión o conexión asociada del cliente.

- La funcionalidad de cierre de sesión debe estar disponible en todas las páginas protegidas por autorización.

- Se debe establecer un tiempo de espera de inactividad de la sesión que sea lo más breve posible, basado en el riesgo y los requisitos funcionales del negocio. En la mayoría de los casos no debe durar más 30 minutos.

- No se deben permitir inicios de sesión persistentes. Se deben aplicar terminaciones de sesión periódicas, incluso cuando la sesión esté activa. Especialmente para aplicaciones que admiten conexiones de red que se conectan a sistemas críticos. Los tiempos de terminación de sesión deben cumplir con los requisitos del negocio y el usuario debe recibir notificaciones de la actividad que se ha realizado como parte del cierre de sesión.

- Si se ha establecido una sesión y se presenta el inicio de una nueva sesión, se debe cerrar la sesión anterior y establecer una nueva.

- Se debe generar un nuevo identificador de sesión en cualquier nuevo proceso de autenticación.

- No se deben permitir inicios de sesión concurrentes con el misma ID de usuario.

- No se deben exponer identificadores de sesión en los URL, mensajes de error o en los registros. Los identificadores de sesión solo deben ubicarse en el encabezado de las cookies HTTP. Por ejemplo, no pase identificadores de sesión como parámetros del método GET.

- Se deben proteger los datos de la sesión que se almacenan del lado del servidor contra acceso no autorizado por otros usuarios.

- Se debe generar un nuevo identificador de sesión y desactivar el antiguo de forma periódica. (Esto puede mitigar ciertos escenarios de secuestro de sesión donde el identificador original se haya comprometido)

- Se debe generar 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 debe utilizar constantemente HTTPS en lugar de cambiar de HTTP a HTTPS.

- Se debe complementar el manejo de sesión, donde se involucren operaciones confidenciales, con tokens o parámetros aleatorios fuertes para cada sesión. Este método se puede usar para prevenir ataques de XSS Cross Site Scripting.

- Se debe complementar la administración de sesión donde se involucren operaciones altamente sensibles o críticas con tokens o parámetros aleatorios fuertes para cada solicitud, en vez de utilizar sesión.

- Se debe establecer el atributo "secure" para las cookies transmitidas a través de una conexión TLS (En su última versión).

- Se deben establecer las cookies con el atributo HttpOnly, a menos que se requieran específicamente scripts del lado del cliente para leer o establecer el valor de una cookie.

7.5 Control de acceso

- Se deben usar solo objetos confiables del sistema. Por ejemplo, objetos de sesión del lado del servidor para tomar decisiones de autorización de acceso.

- Se debe usar un solo componente para verificar la autorización de acceso de todo el sitio. Esto incluye librerías que llaman a servicios de autorización externos.

- Los controles de acceso deben fallar de forma segura
- Se debe denegar todo acceso si la aplicación no puede acceder a su información de configuración de seguridad. Por ejemplo, privilegios o restricciones de acceso.
- Se deben aplicar controles de autorización en cada solicitud, incluidas las realizadas por scripts del lado del servidor, solicitudes de tecnologías del lado del cliente como AJAX y Flash
- Se debe segregar la lógica privilegiada de otro tipo de código de la aplicación.
- Se debe restringir el acceso a archivos u otros recursos, incluidos los que están fuera del control directo de la aplicación y solo a usuarios autorizados.
- Se debe restringir el acceso a URLs protegidas y solo a usuarios autorizados.
- Se debe restringir el acceso a funciones protegidas y solo a usuarios autorizados.
- Se deben restringir las referencias directas de objetos y solo a usuarios autorizados.
- Se debe restringir el acceso a los servicios y solo a usuarios autorizados.
- Se debe restringir el acceso a los datos de la aplicación y solo a usuarios autorizados.
- Se debe restringir el acceso a los atributos de usuario, datos y a la información de política utilizada por los controles de acceso.
- Se debe restringir la información de configuración de acceso de seguridad y solo a usuarios autorizados.
- Si los “datos de estado” deben almacenarse en el cliente, entonces se debe utilizar cifrado y verificación de integridad del lado del servidor para detectar la manipulación de dicha información.
- Se debe hacer cumplir los flujos lógicos de la aplicación para cumplir con las reglas negocio.
- Se debe limitar el número de transacciones que un solo usuario o dispositivo puede realizar en un período de tiempo determinado. Las transacciones/tiempo deben estar por encima del requisito de negocio, pero lo suficientemente bajo como para disuadir ataques automáticos.
- Se debe utilizar el encabezado "Referer" solo como verificación complementaria, nunca debe ser la única verificación de autorización, ya que puede ser falsificada.

- Si se permiten sesiones autenticadas largas, se debe volver a validar la autorización del usuario para asegurarse de que sus privilegios no hayan cambiado y, si lo han hecho, se debe cerrar la sesión del usuario y obligarlo a volver a autenticarse.

- Se debe implementar la auditoría de cuentas y aplicar la desactivación de las cuentas no utilizadas después de no más de 15 días desde el vencimiento de la contraseña de una cuenta.

- La aplicación debe admitir la desactivación de cuentas y la finalización de sesiones cuando cesa la autorización (Ej., Cambios de función, situación laboral, proceso de negocio, etc.)

- Las cuentas de servicio o las cuentas que admiten conexiones hacia o desde sistemas externos deben tener el menor privilegio posible.

- Se debe crear una Política de control de acceso para documentar las reglas de negocio, los tipos de datos y los criterios y / o procesos de autorización de acceso de una aplicación para que el acceso pueda ser aprovisionado y controlado adecuadamente. Esto incluye la identificación de los requisitos de acceso tanto para los datos como para los recursos del sistema.

- Se deberá documentar una matriz de roles y permisos establecidos en la aplicación así como la justificación de cada rol.

7.6 Criptografía

- Todas las funciones criptográficas utilizadas para proteger la información del usuario de la aplicación deben implementarse del lado del servidor.

- Se deben proteger las contraseñas maestras de acceso no autorizado.

- Los módulos criptográficos deben fallar de forma segura.

- Todos los números aleatorios, nombres de archivos aleatorios, GUID aleatorios y cadenas aleatorias deben generarse utilizando el generador de números aleatorios probado públicamente.

- Los módulos criptográficos utilizados por la aplicación deben cumplir con FIPS 140-2 o un estándar equivalente. (Ver http://csrc.nist.gov/groups/STM/cmvp/validation.html)

- Se debe definir y establecer una política y un proceso sobre cómo se gestionarán las llaves criptográficas de los desarrollos/aplicaciones.

7.7 Manejo de errores y gestión de logs

- No se debe divulgar información confidencial en respuestas (pantallas) de error, incluidos detalles del sistema, identificadores de sesión o información de la cuenta.
- Se deben utilizar controladores de errores que no muestren información de depuración o de seguimiento de la pila.
- Se deben implementar mensajes de error genéricos, así como usar páginas de error personalizadas.
- La aplicación debe manejar sus errores, no depender de la configuración del servidor.
- Debe asignarse y disponerse de memoria libre para cuando ocurran condiciones de error.
- La lógica de manejo de errores asociada con los controles de seguridad debería denegar el acceso por defecto.
- Todos los controles de registro de logs deben implementarse del lado del servidor.
- Los logs deben registrar eventos de seguridad éxitos y no exitosos.
- Los logs deben contener datos importantes del evento registrado.
- Los logs que contengan datos no confiables no deben ejecutarse en el archivo de registro o en la interfaz de visualización de los mismos.
- Se debe restringir el acceso a los registros de log a solo las personas autorizadas.
- Se debe utilizar una rutina maestra para todas las operaciones de registro de logs.
- No se debe almacenar información confidencial en registros de log, incluidos detalles innecesarios del sistema, identificadores de sesión o contraseñas.
- Debe existir un mecanismo para realizar el análisis de registros de logs.
- Se deben registrar todas las fallas de validación de entrada.
- Se deben registrar todos los intentos de autenticación, especialmente los fallos de inicio de sesión.
- Se deben registrar todas las fallas de control de acceso.

- Se deben registrar todos los eventos sospechosos de manipulación, incluidos los cambios inesperados en los “datos de estado”.

- Se deben registrar los intentos de conexión con tokens de sesión no válidos o caducados.

- Se deben registrar todas las excepciones del sistema.

- Se deben registrar todas las funciones administrativas, incluidos los cambios en la configuración de seguridad.

- Se deben registrar todos los errores de conexión de TLS (En su última versión) de back-end.

- Se deben registrar las fallas de los módulos criptográficos utilizados.

- Se debe usar una función hash criptográfica para validar la integridad de la entrada del registro de log. Esta debe ser por lo menos un SHA256.

7.8 Protección de datos

- Se debe implementar el menor privilegio. Por lo tanto, se debe restringir el acceso de los usuarios solo a la funcionalidad, datos e información del sistema que se requiere para realizar sus tareas.

- Se deben proteger del acceso no autorizado todas las copias en caché o ubicaciones temporales donde se almacenen datos confidenciales del lado del servidor. Estos datos se deben purgar tan pronto como ya no sean necesarios.

- Se debe cifrar la información altamente confidencial que es almacenada, como datos de verificación de autenticación, incluso en el lado del servidor. Se deben utilizar siempre algoritmos probados y verificados públicamente,

- Se debe proteger el código fuente del lado del servidor para que no sea descargado por los usuarios.

- No se deben almacenar contraseñas, cadenas de conexión u otra información confidencial en texto claro o de forma no criptográfica segura en el lado del cliente o del servidor.

- Se deben eliminar los comentarios en el código fuente en ambiente de producción que pueda ser accesible para el usuario y que pueda revelar la información del sistema u otra información confidencial.

- Se debe eliminar la documentación innecesaria del sistema o de la aplicación, ya que esto puede revelar información útil a los atacantes.

- No se debe incluir información confidencial en los parámetros de solicitud (Request) HTTP GET.

- Se deben deshabilitar las funciones de autocompletar en los formularios en que se espera que contengan información confidencial, incluida la autenticación.

- Se debe deshabilitar el almacenamiento en caché del lado del cliente en páginas que contienen información confidencial. Cache-Control: no-store, se puede usar junto con el control de encabezado HTTP "Pragma: no-cache", que es menos efectivo, pero es compatible con versiones anteriores de HTTP / 1.0

- La aplicación debe admitir la eliminación de datos confidenciales cuando esos datos ya no sean necesarios. (por ejemplo, información personal o ciertos datos financieros)

- Se deben implementar controles de acceso adecuados para utilizar los datos confidenciales almacenados en el servidor. Esto incluye datos en caché, archivos temporales y datos a los que solo deberían poder acceder usuarios específicos del sistema.

7.9 Seguridad en las comunicaciones

- Se deben implementar cifrado para la transmisión de toda la información sensible. Esto debe incluir el protocolo TLS (En su última versión) para proteger la conexión entre cliente y servidor.

- Los certificados TLS (En su última versión) deben ser válidos y tener el nombre de dominio correcto, no caducar e instalarse con certificados intermedios cuando sea necesario.

- Las conexiones TLS (En su última versión) fallidas no deben recurrir a una conexión insegura.

- Se deben utilizar conexiones TLS (En su última versión) para todo el contenido que requiera acceso autenticado y para acceder a toda otra información confidencial.

- Se debe utilizar TLS (En su última versión) para conexiones a sistemas externos que involucren información o funciones sensibles.

- Se debe utilizar una única implementación estándar de TLS (En su última versión) la cual deberá estar configurada adecuadamente.

- Se deben especificar codificaciones de caracteres para todas las conexiones.

- Se deben filtrar los parámetros que contienen información confidencial del “referer” HTTP, cuando se vincula a sitios externos.

7.10 Configuración del sistema para desarrollo de software

- Los servidores, frameworks y componentes del sistema deben ejecutarse en la última versión aprobada.
- Los servidores, frameworks y componentes del sistema deben tener todos los parches emitidos para la versión en uso.
- Se deben desactivar los listados de directorios.
- Se deben restringir las cuentas de servidor web, proceso y servicio a los menores privilegios posibles.
- Cuando se producen excepciones, el sistema debe fallar de forma segura.
- Se deben eliminar todas las funcionalidades y archivos innecesarios.
- Se debe eliminar el código de prueba o cualquier funcionalidad no prevista en el ambiente de producción, antes de la implementación.
- No se debe divulgar la estructura de los directorios en el archivo robots.txt colocando directorios no destinados a la indexación pública en un directorio padre aislado. Posterior a esto se debe configurar el, "No permitir" ese directorio padre completo en el archivo robots.txt en lugar de No permitir cada directorio individual.
- Se debe definir qué métodos HTTP, Get o Post, admitirá la aplicación y si se manejará de manera diferente en diferentes páginas de la aplicación.
- Se deben deshabilitar todos los métodos HTTP innecesarios, como extensiones WebDAV. Si se requiere un método HTTP extendido que admita el manejo de archivos, se debe utilizar un mecanismo de autenticación previamente probado.
- Si el servidor web maneja HTTP 1.0 y 1.1, entonces ambos deben estar configurados de manera similar o se debe identificar cualquier diferencia que pueda existir (por ejemplo, manejo de métodos HTTP extendidos).
- Se debe eliminar la información innecesaria de los encabezados de respuesta HTTP relacionados con el Sistema Operativo, la versión del servidor web y los frameworks de aplicación.

- El almacenamiento de configuración de seguridad para la aplicación debe mantenerse en forma legible para efectos de auditoría.

- Se debe implementar un sistema de gestión de activos y registre los componentes y el software del sistema.

- Se deben aislar los entornos de desarrollo de los entornos de producción y proporcionar acceso solo a los grupos autorizados de desarrollo y prueba. Los entornos de desarrollo a menudo se configuran de manera menos segura que los entornos de producción y los atacantes pueden usar esta diferencia para descubrir debilidades compartidas o como una vía de explotación.

- Se debe implementar el procedimiento de control de cambios de TRANSMILENIO S.A para administrar y registrar cambios en el código tanto en desarrollo como en producción.

7.11 Seguridad en base de datos

- Se deben utilizar queries parametrizados.

- Se debe utilizar la validación de entrada y la codificación de salida con metacaracteres. Si estos fallan, no se debe ejecutar dicho comando en la base de datos.

- Las variables se deben es fuertemente definidas.

- La aplicación debe usar el nivel de privilegio más bajo posible al acceder a la base de datos.

- Se deben usar credenciales seguras para acceder a la base de datos.

- Las cadenas de conexión no deben estar codificadas dentro de la aplicación. Las cadenas de conexión deben almacenarse en un archivo de configuración separado, en un sistema confiable y deben cifrarse.

- Se deben usar procedimientos almacenados para abstraer el acceso a datos y permitir la eliminación de permisos a las tablas base en la base de datos.

- Se debe cerrar la conexión a la base de datos lo antes posible.

- Se deben eliminar y cambiar todas las contraseñas administrativas de la base de datos que estén predeterminadas. Se deben utilizar contraseñas / frases seguras o implementar la autenticación de múltiple factor.

- Se deben desactivar todas las funciones innecesarias de la base de datos (p. Ej., Procedimientos o servicios almacenados innecesarios, paquetes de servicios, se debe instalar solo el conjunto mínimo de características y opciones requeridas)

- Se debe eliminar el contenido predeterminado innecesario del fabricante (por ejemplo, esquemas de muestra)

- Se deben deshabilitar las cuentas predeterminadas que no sean necesarias para cumplir con los requisitos de negocio.

- La aplicación debe conectarse a la base de datos con diferentes credenciales para cada distinción de confianza (por ejemplo, usuario, usuario de solo lectura, invitado, administradores)

7.12 Gestión de Archivos

- No se deben pasar los datos proporcionados por el usuario directamente a ninguna función de inclusión dinámica.

- Se debe requerir autenticación antes de permitir que se cargue un archivo.

- Se debe limitar el tipo de archivos que se pueden cargar solo a los tipos necesarios para fines de la entidad.

- Se debe validar que los archivos cargados son del tipo esperado al verificar sus encabezados. Comprobar el tipo de archivo solo por extensión no es suficiente.

- No se deben almacenar los archivos en el mismo contexto web que la aplicación. Los archivos deben ir al servidor de contenido o a la base de datos.

- Se debe restringir o evitar la carga de cualquier archivo que pueda ser interpretado por el servidor web.

- Se deben desactivar los privilegios de ejecución en los directorios de carga de archivos.

- Se debe implementar la carga segura en UNIX montando el directorio de archivos de destino como una unidad lógica y utilizando la ruta asociada o el entorno con permisos de root (chrooted).

- Cuando se haga referencia a archivos existentes, se debe usar una lista blanca de nombres y tipos de archivos permitidos. Se debe validar el valor de los parámetros que se están pasando y, si no coincide

con uno de los valores esperados, se debe rechazar o usar un valor de archivo predeterminado codificado para el contenido.

- No se deben pasar los datos proporcionados por el usuario a una redirección dinámica. Si esto debe permitirse, la redirección debe aceptar solo el URL de ruta relativa validada.

- No se deben pasar rutas de directorio o archivo, se deben usar valores de índice asignados a una lista predefinida de rutas.

- Nunca se debe enviar la ruta absoluta del archivo al cliente.

- Los archivos y recursos de la aplicación deben ser de solo lectura.

- Se deben analizar los archivos cargados por el usuario en busca de virus y malware.

7.13 Manejo de Memoria

- Se debe utilizar el control de entrada y salida para datos no confiables.

- Se debe verificar que el búfer sea tan grande como se especifica.

- Cuando se utilicen funciones que acepten una cantidad de bytes para copiar, como strncpy (), se debe tener en cuenta que si el tamaño del búfer de destino es igual al tamaño del búfer de origen, no se puede terminar la cadena con NULL.

- Se deben verificar los límites del búfer si se llama a la función en un bucle y se debe asegurar de que no haya peligro de escribir más allá del espacio asignado.

- Se deben truncar todas las cadenas de entrada a una longitud razonable antes de pasarlas a las funciones de copia y concatenación.

- Se deben utilizar pilas no ejecutables cuando estén disponibles.

- Se debe evitar el uso de funciones vulnerables conocidas (por ejemplo, printf, strcat, strcpy, etc.)

- Se debe liberar la memoria asignada correctamente al finalizar las funciones y en todos los puntos de salida.

7.14 Políticas generales de desarrollo

- Se debe usar código administrado probado y aprobado en lugar de crear código nuevo no administrado para tareas comunes.

- Se deben utilizar las API integradas específicas de la tarea para realizar tareas del sistema operativo. No se debe permitir que la aplicación emita comandos directamente al sistema operativo, especialmente mediante el uso de shells de comandos iniciados por la aplicación.

- Se deben usar sumas de comprobación (CRC) o hashes para verificar la integridad del código interpretado, las librerías, los ejecutables y los archivos de configuración.

- Se debe utilizar el bloqueo para evitar múltiples solicitudes simultáneas o usar un mecanismo de sincronización para evitar “Race conditions”

- Se deben proteger las variables y recursos compartidos del acceso concurrente inapropiado.

- Se deben inicializar explícitamente todas las variables y otros almacenes de datos, ya sea durante la declaración o justo antes del primer uso.

- En los casos en que la aplicación debe ejecutarse con privilegios elevados, se deben aumentar los privilegios lo más tarde posible y liberarlos lo antes posible.

- Se deben evitar los errores de cálculo al comprender la representación subyacente del lenguaje de programación usado y cómo interactúa con el cálculo numérico. Se debe prestar mucha atención a las discrepancias en el tamaño de los bytes, la precisión, las distinciones con signo / sin signo, el truncamiento, la conversión y la conversión entre tipos, los cálculos "sin número" y la forma en que el idioma utilizado maneja los números que son demasiado grandes o demasiado pequeños para su representación subyacente.

- No se deben pasar los datos proporcionados por el usuario a ninguna función de ejecución dinámica.

- Se debe restringir a los usuarios de generar código nuevo o alterar el código existente.

- Se deben revisar todas las aplicaciones secundarias, el código de terceros y las librerías para determinar la necesidad de la entidad, así como validar la funcionalidad segura, ya que pueden introducir nuevas vulnerabilidades.
- Se deben implementar actualizaciones seguras. Si la aplicación utilizará actualizaciones automáticas, entonces se deben usar firmas criptográficas para el código y asegurarse de que los clientes de descarga verifiquen esas firmas. Se deben usar canales cifrados para transferir el código desde el servidor.