NORMA ISO 27001
Tecnología de la información
Técnicas de seguridad
Sistemas de Gestión de la Seguridad de la Información (SGSI)
Controles de seguridad de la información

GUÍA DE SEGURIDAD EN EL DESARROLLO Y DESPLIEGUES DE SISTEMAS

Última actualización: julio 05, 2022

SISTEMA DE GESTIÓN DE LA SEGURIDAD DE LA INFORMACIÓN
Matamoros SN, Zona Centro
Ciudad Victoria, Tamaulipas,
C.P. 87000

(834) 318-1800, ext. 2823
www.uat.edu.mx

CONTROL DEL DOCUMENTO

Datos del Documento	
Aplicado a	Direcciones del alcance del Sistema de Gestión de la Seguridad de la Información (SGSI)
Aprobado por	Directores del SGSI
Fecha de Aprobación	05/07/2022
Fecha de Revisión:	05/07/2022
Versión	5
Clasificación	Público
Cambios desde la última versión Se actualiza el nombre de la Dirección (DTI) Dirección de Tecnologías de la Información a (DIT) Dirección de Infraestructura Tecnológica Se modifica la tabla de control de versiones	

TABLA DE CONTENIDOS

SECCIÓN I
ASPECTOS GENERALES

1.1 Introducción

La seguridad de la información se ha situado como uno de los principales puntos de preocupación y riesgo dentro de las organizaciones, incluyendo a las universidades. El sector educativo se caracteriza por la alta implantación de nuevas tecnologías, tanto para su labor educativa como para la comunicación de centros, alumnos, profesores y demás empleados, haciendo un uso muy intensivo de software crítico, equipos informáticos y redes a la que se conectan cada vez más todo tipo de dispositivos. Esto hace que, como otros sectores, también esté expuesto a riesgos generales de seguridad y ataques informáticos.

Este documento tiene la finalidad de definir un conjunto de lineamientos que proporcionen una base de referencia de seguridad de las aplicaciones web, a las cuales todo equipo de desarrollo debe apegarse en la medida de lo posible. Este documento está organizado de la siguiente manera. La sección I aborda la introducción al tema, el objetivo principal y objetivos específicos. La sección II describe lineamientos organizados por tópicos relacionados con el desarrollo de software, mientras que la sección III explica el mecanismo de seguimiento y control que se debe realizar a los riesgos que se identifican en el desarrollo de software. La sección IV presenta la lista de referencias bibliográficas de este documento y en la sección V se describe el glosario. Al final se colocan los anexos los cuales soportan el cuerpo de esta política para dar detalle acerca de los riesgos más destacados de seguridad, controles y prácticas de verificación para entregar código seguro.

1.2 Objetivo general

Especificar buenas prácticas de desarrollo seguro que se deben seguir para diseñar, desarrollar y desplegar aplicaciones en recursos de la infraestructura tecnológica de la Universidad Autónoma de Tamaulipas.

1.3 Objetivos específicos

- Asegurar la integridad, confidencialidad, disponibilidad y desempeño de las prácticas de codificación web para el desarrollo de aplicaciones,
- Prevenir el acceso y divulgación no autorizada de datos confidenciales de bases de datos, servidores web y las comunicaciones.
- Asegurar que la institución está protegida contra daños en sus procesos de desarrollo de software y aplicaciones productivas.
- Asegurar que la configuración, actualización de los sistemas y aplicativos se mantienen consistentes con los principios de seguridad.

1.4 Alcance

Este documento aplica a todas las áreas institucionales o externas a la Universidad Autónoma de Tamaulipas que tengan la responsabilidad de entregar software desarrollado para la Universidad. Los usuarios principales de este documento son todos los empleados que trabajen en el desarrollo y/o mantenimiento del portal institucional, sistemas académicos y financiero.

Se considera fuera del alcance todo software tercerizado dado que todo el software se desarrolla propiamente en la Universidad Autónoma de Tamaulipas.

1.5 Definiciones, siglas y acrónimos

N/A

SECCIÓN II
LINEAMIENTOS

2.1 Generalidades

Este documento está alineado a la norma ISO 27001:2013 en el Anexo A.12.1.4, A.12.3.1, A.12.6.1, A.14.1.1, A.14.1.2, A.14.2.1, A.14.2.2, A.14.2.3, A.14.2.4, A.14.2.5, A.14.2.6, A.14.2.8.

2.2 Ambientes de Desarrollo

1. Se deben mantener entornos de desarrollo (DEV) y / o de prueba (TEST) separados del entorno de producción (PROD) de las aplicaciones.
2. El desarrollo y las modificaciones en la construcción del software solo se realiza en entornos de desarrollo (DEV).
3. La información de los entornos de producción (PROD) no se utiliza en entornos de prueba (TEST) o desarrollo (DEV).
4. Toda la publicación del código en entornos PROD, será realizada por la persona que tenga esta función.
5. Tener un registro de las actividades del administrador del sistema en una bitácora de logs de los diferentes entornos.
6. Todas las aplicaciones deben generar registros de las transacciones realizadas por los usuarios (bitácora de logs) en caso de posibles fallas o intrusiones.
7. Administrar un control de acceso basado en el perfil de los diferentes ambientes.
8. Las aplicaciones deben personalizar en la medida de lo posible los mensajes de error de los diferentes componentes.
9. Los sistemas universitarios deben implementar, en la medida de lo posible, el mecanismo de autenticación universitaria (ADFS) para controlar el acceso a los sistemas. En los casos donde no fuera posible el uso de dicho mecanismo se utilizará el mecanismo de autenticación que garantice la seguridad y confiabilidad en el acceso.
10. Validar el ingreso utilizando las direcciones IP, a fin de que sólo se acceda desde aquellas IP autorizadas, cuando el proceso al que soporta el sistema así lo requiera.
11. Cuando los sistemas se encuentren en etapas de desarrollo o de pruebas, se debe promover el uso de credenciales de acceso que sean diferentes para cada instancia del sistema.
12. Asegurar el uso de un repositorio y un mecanismo de control de cambios de código para manejar y registrar las modificaciones o nuevos desarrollos.
13. Procurar en la medida de lo posible, que todas las aplicaciones entregadas a usuario, que sean alojadas en infraestructura Universitaria, lleven a cabo la política de respaldo de información de la Universidad.

2.3 Ciclo de vida del desarrollo del sistema

1. Gestión de Requerimientos:
    a. Los responsables del análisis dentro de los equipos de desarrollo deben identificar cuáles son los requerimientos de seguridad desde el inicio del proyecto, durante el levantamiento de las historias de usuario/casos de uso/ en las etapas de análisis de la aplicación.

2. Diseño:
    a. Los componentes de la aplicación deben planificarse de manera coherente con los datos y la seguridad de la red.
    b. De ser posible, se deben crear y revisar que los diseños arquitecturales, conceptuales, etc. no contengan errores en la lógica; también se deben identificar sus principales amenazas que podrían afectar a la aplicación, a fin de generar acciones específicas de mitigación.
3. Desarrollo:
    a. Se deberá organizar el proceso de desarrollo en esquemas segmentados por programador. Asignando permisos correspondientes a los segmentos que le corresponden.
    b. El equipo de desarrolladores debe tener en cuenta todas las vulnerabilidades de las aplicaciones, pero principalmente las señaladas en el Anexo A, a fin de evitar prácticas débiles de codificación que introduzcan vulnerabilidades desde el punto de vista técnico.
    c. Las aplicaciones de software se desarrollarán según las mejores prácticas de código de la industria e incorporarán marcos de trabajo robustos para la seguridad de la información a lo largo del ciclo de vida del desarrollo del software. Algunos ejemplos se ilustran en el Anexo B.
    d. Para las aplicaciones que manejan datos confidenciales o que requieren máxima seguridad, el software y el código del programador se debe verificar con lineamientos OWASP del Anexo C. Si su aplicación está desarrollada en Microsoft .NET, el siguiente enlace sirve de apoyo para abordar controles de seguridad (Recursos de Seguridad OWASP para .NET). Si su aplicación está desarrollada en Java revisar en (Recursos de Seguridad OWASP para Java), si está desarrollada en otro lenguaje deberá examinar en la comunidad web siguiendo las especificaciones OWASP del siguiente enlace (Guía de Desarrollo OWASP).
    e. Utilizar consultas parametrizadas para la interacción con la base de datos en lugar de consultas generadas dinámicamente. Esto con la finalidad de controlar amenazas de inyección SQL.
4. Revisión de código:
    a. Buscando identificar problemas de seguridad (vulnerabilidades) se podrán realizar revisiones al código nuevo o modificado por parte de terceros dentro de los miembros del equipo de desarrollo.
    b. Específicamente, las vulnerabilidades que se consideran durante las revisiones y pruebas del código son
        i. Cross-site scripting (XSS);
        ii. Error de inyección, particularmente inyección SQL. También considere las fallas de inyección LDAP y XPath, así como otros defectos de inyección;
        iii. Ejecución de archivos maliciosos;
        iv. Referencia a objetos directos inseguros;

v. Falsificación de registros Cross-site;
vi. Fuga de información y manejo inadecuado de errores;
vii. Pérdida de autenticación y administración de sesiones;
viii. Almacenamiento criptográfico inseguro;
ix. Comunicaciones inseguras;
x. Fallas para restringir el acceso a URL.

5. Pruebas de Software QA:
    a. Además de las pruebas funcionales y de rendimiento, se deben probar todas las características de seguridad de la aplicación para detectar vulnerabilidades. Algunos puntos mínimos son:
        i. Validar todas las cajas de entrada de datos de la aplicación
        ii. Verificar que todas las salidas de tipo HTML por todas las aplicaciones web utilizan "HTML Output Encoding"
        iii. Verificar que implementan controles de seguridad basados en el navegador web para las aplicaciones a fin de ayudar a proteger tanto las aplicaciones como a sus usuarios. (Ver Anexo B)
6. Documentación:
    a. Todas las características de la aplicación y la documentación de despliegues deben incluir instrucciones sobre las configuraciones de seguridad adecuadas.
7. Despliegue:
    a. La publicación de la versión del sistema no debe comprometer los controles de seguridad existentes, ni introducir nuevas vulnerabilidades.
    b. El despliegue de la aplicación en un entorno de pruebas o de producción, debe ser de manera apropiada a los requerimientos o políticas de seguridad de acceso a entornos de pruebas o productivos.
    c. No se debe desplegar una versión de software si el personal de pruebas de software QA no da su validación para que sea desplegado en un entorno productivo.
8. Mantenimiento:
    a. Todo el mantenimiento en etapas de postproducción de la aplicación no debe comprometer los controles de seguridad existentes o introducir nuevas vulnerabilidades. Cualquier código nuevo será revisado y probado como se detalla en la sección de revisión de código.

2.4 Administración de Sistemas de Bases de Datos

1. Los contenidos del servidor de base de datos deben protegerse contra acceso no autorizado a través del uso de técnicas de cifrado. Los discos duros de los servidores de base de datos se pueden cifrar de forma completa (full disk encryption).

2. Los servidores web y los servidores de base de datos deben estar separados física o virtualmente y en su funcionalidad dado que colocar las bases de datos usadas por las aplicaciones web en el mismo servidor físico que el servidor web, puede exponerlas a una mayor posibilidad de compromiso de un adversario.
3. La información comunicada entre el servidor de base de datos y el servidor web sobre el internet debe ser encriptada, ya que están susceptibles para ser capturados por un adversario. (Algunos ejemplos https, y el uso de puerto 443)
4. Se deben colocar los servidores de bases de datos en segmentos de red diferentes respecto de las estaciones de trabajo de la organización, a fin de permitir la comunicación con otros recursos de la red para reducir la posibilidad de compromiso por parte de un atacante.
5. Se deben colocar controles de acceso mediante credenciales o equipos autorizados (IPs) a la red para restringir las comunicaciones de los servidores de base de datos a recursos de red como: servidores web, aplicativos y redes de área de almacenamiento.
6. En caso de habilitar un servidor local de base de datos se debe restringir el acceso al mismo por medio de un firewall local u otros controles de red.
7. Los entornos de prueba (TEST) y desarrollo (DEV) no utilizan los mismos servidores de base de datos que los entornos productivos (PROD) a fin de reducir el riesgo de daños accidentales cuando se desarrollen actividades de prueba y desarrollo.
8. Toda la instalación de archivos y logs temporales se eliminan después de haber instalado el software DBMS, ya que la información de estos archivos podría incluir contraseñas o frases que podría tomarla un adversario.
9. El software DBMS se configura de acuerdo con la guía del proveedor, caso contrario una configuración deficiente podría proveer oportunidad a un atacante en accesos no autorizados a la BD.
10. Las características del software DBMS, los procedimientos almacenados, las cuentas y las bases de datos que no son necesarias se desactivan o eliminan; dado que el software DBMS usualmente se instala con la mayoría de las características habilitadas por defecto.
11. El software DBMS se configura para ejecutarse como una cuenta separada con privilegios mínimos requeridos para desempeñar sus funciones, ya que si el software DBMS opera como "local administrator" o "root account" podría ser comprometido por un adversario y esto podría representar un riesgo significativo en la seguridad para el sistema operativo subyacente.
12. Se debe deshabilitar la capacidad del DBMS para leer archivos locales o en su caso autenticar y legitimar de los archivos de los cuales se van a hacer uso
13. Las cuentas de administrador de la base de datos predeterminadas se desactivan, se les cambia el nombre o se modifican sus frases de contraseña, en caso de que el software DBMS venga preconfigurado con cuentas de administrador de base de datos y frases de contraseña predeterminadas.
14. Las cuentas de administrador de la base de datos no deben ser compartidas entre diferentes bases de datos, ya que puede dificultar las investigaciones relacionadas con un intento, o éxito, de una intrusión cibernética dirigida.
15. El acceso del administrador de la base de datos está restringido a roles definidos en lugar de cuentas con permisos administrativos predeterminados o todos los permisos. Usualmente al crear nuevas cuentas de administrador de BD se asignan todos los

privilegios disponibles, cuando normalmente los administradores solo necesitarán un subconjunto de todos los privilegios disponibles.

16. Se debe mantener un inventario preciso de todas las bases de datos y sus contenidos; el cual se debe de auditar periódicamente para proteger adecuadamente sus activos.
17. Se deberán implementar controles de acceso basados en logs o bitácoras de la base de datos a fin de proteger la realización de respaldos no autorizados.
18. Las contraseñas almacenadas en las bases de datos deben transformarse con un algoritmo de "hashing" fuerte de criptografía, ya que al almacenarlas en texto sin formato representa un riesgo de seguridad. Existen usuarios que reutilizan nombre y contraseña para muchos accesos.
19. Se debe acotar la capacidad de los usuarios de base de datos en función de sus tareas laborales a fin de reducir la probabilidad de modificaciones no autorizadas. En caso de que se necesite saber más información de la base de datos, se deben otorgar vistas de bases de datos, o en su caso, nuevos roles con privilegios mínimos y/o temporales.
20. Las bases de datos de producción no se utilizan en bases de datos de prueba o desarrollo a menos que los entornos de prueba o desarrollo estén protegidos al mismo nivel que el entorno de producción.
21. Todas las consultas (queries) a bases de datos desde aplicaciones web se deben filtrar por sintaxis correcta y legítima para proteger contra ataques de inyección SQL.
22. Utilizar consultas parametrizadas para la interacción con la base de datos en lugar de consultas generadas dinámicamente. Esto con la finalidad de controlar amenazas de inyección SQL.
23. Las aplicaciones web deben diseñar un mecanismo de manejo de fallas, de modo que, al momento de un error presentado al usuario, no se muestre información detallada sobre el esquema de la base de datos.

2.5 Servidor Web IIS

Instalación y Configuración del Servicio.

1. Se debe contar con una versión de IIS con soporte técnico vigente de Microsoft.
2. Se debe instalar/habilitar solamente los módulos necesarios en el IIS para evitar que el administrador omita configuraciones de seguridad en módulos del IIS que no se están utilizando en los despliegues de los sistemas de información. Para mayor información: https://docs.microsoft.com/es-es/iis/get-started/introduction-to-iis/iis-modules-overview
3. Habilitar las últimas versiones de protocolos de seguridad y deshabilitar los que se encuentran activos por default, se debe configurar acorde al (Anexo D) dependiendo del sistema o plataforma que administre el IIS.
4. Realizar la configuración necesaria en los encabezados para evitar ataques sobre el código, se debe configurar acorde al (Anexo B).
5. Deshabilitar archivos o directorios donde se exponga la versión de plataforma o extensión que se utiliza, el bloqueo se debe realizar con la configuración detallada en el (Anexo D)

6. Todos los sitios que se configuren en el IIS y se encuentren en producción, deben tener la versión del lenguaje de programación utilizado desactivado como se recomienda en el (Anexo D).
7. La carpeta física donde se aloja los archivos del sistema o plataforma deben estar en otra unidad diferente del sistema operativo.
8. El IIS no debe estar instalado en un controlador de dominio en ninguno de los casos.
9. Habilitar los registros de actividad (logs) y configurar la ruta donde se mantendrán alojados.
10. Se recomienda utilizar la Autentitación de Windows para el manejo de credenciales.
11. Para la actualización de los sistemas de hosteo evitar el uso de carpetas compartidas y en caso de ser necesario desactivar las versiones del protocolo SMB anteriores (Anexo D).

Monitoreo y Mantenimiento.

1.- Todos los sitios deben tener un registro de usuarios permitidos.

2.- Se debe realizar un respaldo de la configuración del IIS y tener un registro de la fecha de respaldo.

3.- Estar pendiente de la última versión del IIS siguiendo los siguientes pasos:

- Acceder desde el servidor a Windows Update.
- Comprobar si existe alguna actualización con referencia a IIS o protocolos de seguridad.
    - Si se encuentra alguna actualización publicada:
        - Realizar el proceso de respaldo de la configuración e información basándose en la guía de Respaldos.
        - Aplicar la actualización siguiendo el proceso recomendado por la aplicación de Windows Update
        - Validar el correcto funcionamiento de la actualización.
        - Si la actualización afecta el funcionamiento del sistema o aplicación iniciar el proceso de restauración basándose en la guía de respaldos.
- La revisión debe ser semanal y si marca alguna actualización se debe aplicar fuera del horario de operaciones, en coordinación con las áreas informáticas de la DIT.

4.- Hacer análisis de vulnerabilidades a la aplicación en desarrollo cada seis meses.

2.6 Servicios de Escritorio Remoto de Windows

Los Servicios de Escritorio Remoto dentro de Windows nos permite administrar de manera remota los servidores y estaciones de trabajo basados en la plataforma de Microsoft. Este servicio fue diseñado para operar sobre un canal seguro (encriptado) evitando que la información transmitida pueda ser capturada o escuchada, durante la conexión.

RDP (Remote Desktop Protocol) es el protocolo utilizado por los servicios de escritorio remoto, para su conexión y funcionamiento. Aunque se considera un canal seguro, se tiene que tomar ciertas precauciones en su configuración para asegurar su mejor operación y seguridad de la información. Algunas consideradas como fundamentales son:

1. **Conexiones Seguras**

Se debe permitir solo conexiones de equipos que ejecuten Escritorio Remoto con Autenticación a nivel de red, las versiones incluidas en Windows 10 o superior; en el caso de OS X, usar la aplicación de RDClient. En caso de no hacerlo así, se corre el riesgo de que un adversario pueda leer la información que se transmite para realizar un exploit. (ejemplo: man in the middle).

2. **Usuarios**

Permitir acceso solo a los usuarios que usarán y tendrán acceso al servicio; o bien, crear un usuario especial para el acceso y la administración del servidor, aplicación o sistema y nadie más. Esto con la finalidad de controlar el acceso a los activos relacionados con el sistema.

3. **Uso de Certificados**

Usar certificados con nivel de encriptación 2048 o superior para cifrar la comunicación de la conexión de forma más segura. Este tipo de certificados solo deben ser otorgados por la DIT dado que se verifican pruebas de aceptación con entidades certificadoras reconocidas al nivel mundial para el uso de estos certificados en la institución.

4. **Puerto de Conexión**

Se debe cambiar el puerto de acceso por defecto al escritorio remoto, dado que es conocido que se usa el puerto 3389 comúnmente en la conectividad de escritorio remoto. Se debe cambiar este puerto por cualquier otro puerto para no facilitarle a un adversario este conocimiento y pueda realizar ataques a este puerto.

5. **Solo Red Local**

Al momento de proporcionar accesos de escritorio remoto en la red de trabajo, los accesos deben darse únicamente por direcciones IP para cualquier tipo de equipo de cómputo. Adicionalmente, este acceso no debe ser configurado de modo que esté publicado hacia internet a fin de no dar visibilidad en la extranet a un adversario que quiera atacar el servicio.

SECCIÓN III
SEGUIMIENTO Y CONTROL

3.1 Cumplimiento

3.1.1 Medida de Cumplimiento

Subcomité de Seguridad de la Información verificará el cumplimiento de esta guía a través de varios métodos, que incluyen, entre otros, visitas periódicas, informes de herramientas de negocios, auditorías internas y externas, y comentarios al propietario de este documento.

3.1.2 Excepciones

Cualquier excepción a la guía debe ser aprobada por el con anticipación.

Matamoros SN, Zona Centro
Ciudad Victoria, Tamaulipas,
C.P. 87000
(834) 318-1800, ext. 2823
www.uat.edu.mx

SECCIÓN IV
NORMAS, POLÍTICAS Y REFERENCIAS RELACIONADAS

4.1 Normas

- ISO 27001:2013
- ISO/IEC TR 19759:2015

4.2 Políticas

- Sistemas de Gestión de Tecnologías y Seguridad de la Información (SIGETSI)

4.3 Referencias

- Open Web Application Security Project (OWASP) https://www.owasp.org
- Australian Cyber-Security Center https://acsc.gov.au

SECCIÓN V
GLOSARIO DE TÉRMINOS

Amenaza: Cualquier entidad que puede poseer un impacto negativo en el sistema. Puede ser desde un usuario malicioso que desea comprometer los controles de seguridad del sistema;

sin embargo, también puede referirse al mal uso accidental del sistema o a una amenaza física como fuego o inundación.

Aplicación: cualquier programa o grupo de programas que estén diseñados para el usuario final. El software de aplicación (también llamadas aplicaciones de usuario final) incluye manejadores de bases de datos, procesadores de texto, navegadores web y hojas de cálculo, entre otros.

Autenticación: Conjunto de Controles utilizados para verificar la identidad de un usuario o entidad que interactúa con el software

Canonicalización: Convertir distintas codificaciones y representaciones de datos a una forma estándar predefinida.

Caracteres considerados peligrosos: Cualquier carácter o representación codificada de un carácter que puede afectar la operación intencionada de la aplicación o sistema asociado por ser interpretado de una manera especial, fuera del uso intencionado del carácter. Estos caracteres pueden ser utilizados para:

- Alterar la estructura de las sentencias o código existente
- Insertar código inintencionado
- Alterar caminos
- Causar salidas inesperadas de rutinas o funciones
- Causar condiciones de error
- Obtener cualquiera de los efectos anteriores sobre el flujo de aplicaciones o sistemas

Ciclo de vida del desarrollo del sistema (SDLC): proceso utilizado por la industria del software para diseñar, desarrollar y probar software de alta calidad. También se le llama como Proceso de Desarrollo de Software. SDLC es un marco que define las tareas realizadas en cada paso del proceso de desarrollo de software.

Clickjacking. Es cuando un atacante usa varias capas transparentes u opacas para engañar a un usuario para que haga clic en un botón o enlace en otra página cuando intentaba hacer clic en la página de nivel superior. Por lo tanto, el atacante está "secuestrando" los clics destinados a su página y enrutándolos a otra página, probablemente propiedad de otra aplicación, dominio o ambos.

Código: Conjunto de instrucciones que forman un programa de computadora ejecutado por una computadora. Este código fuente es trasladado a un código máquina por medio de un compilador o interprete de modo que la computadora pueda ejecutarlo para realizar sus tareas.

Codificación de Entidades HTML: Proceso por el cual se reemplazan ciertos caracteres ASCII por sus entidades equivalentes en HTML. Por ejemplo: este proceso reemplazaría el carácter de menor "<" con su equivalente en HTML "<". Las entidades HTML son 'inertes' en la mayoría de los intérpretes, especialmente en los navegadores, pudiendo mitigar ciertos tipos de ataque en los clientes.

Codificación de Salida: Conjunto de controles que apuntan al uso de una codificación para asegurar que los datos producidos por la aplicación son seguros.

Confidencialidad: Propiedad de la información por la que se garantiza que está accesible únicamente a entidades autorizadas.

Configuración del sistema: Conjunto de controles que ayuda a asegurar que los componentes de infraestructura que brindan soporte al software fueron desplegados de manera segura.

Consultas parametrizadas (prepared statements): Mantiene la consulta y los datos separados a través del uso de marcadores. La estructura de la consulta es definida utilizando marcadores, la consulta SQL es enviada a la base de datos y preparada, para luego ser combinada con los valores de los parámetros. Esto previene a las consultas de ser alteradas debido a que los valores de los parámetros son combinados con la consulta compilada y con el string de SQL.
Control de Acceso: Un conjunto de controles que permiten o niegan el acceso a un recurso de un usuario o entidad dado.
Cross-Site Scripting (XSS). una brecha de seguridad que aprovecha las páginas web generadas dinámicamente.
Criptografía: el proceso de convertir información ordinaria (llamada texto plano) en texto ininteligible (llamado texto cifrado). El descifrado es lo contrario: en otras palabras, pasar del texto cifrado ininteligible a texto plano.
Datos de estado: Cuando datos o parámetros son utilizados, ya sea por la aplicación o el servidor, emulando una conexión persiste o realizando el seguimiento del estado de un cliente a través de un proceso multi-pedido o transacción.
Datos del registro de log: Debe incluir lo siguiente:

- Time stamp obtenida de un componente confiable del sistema
- Nivel de severidad para cada evento
- Marcado de eventos relevantes a la seguridad, si se encuentran mezclados con otras entradas de la bitácora
- Identidad de la cuenta/usuario que ha causado el evento
- Dirección IP del origen asociado con el pedido
- Resultado del evento (suceso o falla)
- Descripción del evento

Disponibilidad: Medida de Accesibilidad y Usabilidad del sistema.
Error de inyección: Es una clase de vulnerabilidad de seguridad que permite a un usuario "salir" del contexto de la aplicación web. Si su aplicación web toma la captura del usuario y la inserta esa captura del usuario en una base de datos del back-end, instrucción shell o llamada del sistema operativo; su aplicación puede ser susceptible a una falla de inyección.
Exploit: Forma de tomar ventaja de una vulnerabilidad. Típicamente se trata de una acción intencional diseñada para comprometer los controles de seguridad del software utilizando una vulnerabilidad.
Falsificación de petición en sitios cruzados (CSRF): Una aplicación externa o sitio web fuerza a un cliente a realizar un pedido a otra aplicación en la que el cliente posee una sesión activa. Las Aplicaciones son vulnerables cuando utilizan parámetros o URLs predecibles o conocidas y cuando el navegador transmite automáticamente toda la información de sesión con cada pedido a la aplicación vulnerable. (Este ataque es discutido específicamente en este documento por ser extremadamente común y poco comprendido).
Frontera de Confianza: Una frontera de confianza típicamente constituye los componentes del sistema bajo control directo. Todas las conexiones y datos prevenientes de sistemas fuera del control directo, incluyendo todos los clientes y sistemas gestionados por terceros, deben ser considerados no confiables y ser validados en la frontera, antes de permitir cualquier futura interacción con el sistema.

Gestión de Archivos: Conjunto de controles que cubren la interacción entre el código y otro sistema de archivos.
Gestión de memoria: Conjunto de controles de direccionamiento de memoria y uso de buffers.
Gestión de sesión: Conjunto de controles que ayudan a asegurar que la aplicación web maneja las sesiones HTTP de forma segura.
Integridad: La seguridad de que la información es precisa, completa y válida, y no ha sido alterada por una acción no autorizada.
Inyección de SQL: una técnica de inyección de código, utilizada para atacar aplicaciones controladas por datos, en la que se insertan las sentencias de SQL maliciosas en un campo de captura para su ejecución (por ejemplo, para vaciar el contenido de la base de datos por parte del atacante).
Lightweight Directory Access Protocol (LDAP): es un protocolo de software que permite a cualquier persona ubicar organizaciones, personas y otros recursos, como archivos y dispositivos en una red, ya sea en la Internet pública o en una intranet corporativa
Manejo de Errores y Registro en bitácora: Conjunto de prácticas que aseguran que las operaciones de manejo de errores y registro en bitácora se manejan correctamente.
Mitigar: Pasos tomados para reducir la severidad de una vulnerabilidad. Estos pueden incluir remover una vulnerabilidad, hacer una vulnerabilidad más difícil de explotar, o reducir el impacto negativo de una explotación exitosa.
Referencia a objetos directos inseguros. Se produce cuando una aplicación proporciona acceso directo a objetos según la captura proporcionada por el usuario. Como resultado de esta vulnerabilidad, los atacantes pueden omitir la autorización y acceder a los recursos en el sistema directamente, por ejemplo, registros o archivos de bases de datos.
Seguridad de Comunicaciones: Conjunto de controles que ayudan a asegurar que el software maneja de forma segura el envío y la recepción de datos.
Sistema: Término genérico que cubre sistemas operativos, servidores web, frameworks de aplicaciones e infraestructura relacionada.
Validación de entrada: Conjunto de controles que verifican que las propiedades de los datos ingresados coinciden con las esperadas por la aplicación, incluyendo tipos, largos, rangos, conjuntos de caracteres aceptados excluyendo caracteres peligrosos conocidos.
Vulnerabilidad: Debilidad en un sistema que lo hace susceptible a un ataque o daño.

ANEXO A
LOS 10 RIESGOS MÁS DESTACADOS EN SEGURIDAD DE APLICACIONES (EXTRAÍDO DE OWASP TOP TEN 2017)

A1: 2017 Inyección. Las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.
A2: 2017 Pérdida de Información. Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente).
A3: 2017 Exposición de Datos Sensibles. Muchas aplicaciones web y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o Información Personalmente Identificable (PII). Los atacantes pueden robar o modificar estos datos protegidos inadecuadamente para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito
A4:2017 Entidades Externas XML. Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).
A5: 2017 Pérdida de Control de Acceso. Las restricciones sobre lo que los usuarios autenticados pueden hacer no se aplican correctamente. Los atacantes pueden explotar estos defectos para acceder, de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos, etc.
A6: 2017 Configuración de Seguridad Incorrecta. La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, ad hoc o por omisión (o directamente por la falta de configuración). Son ejemplos: S3 buckets abiertos, cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, frameworks, dependencias y componentes desactualizados, etc.
A7: 2017 Secuencia de Comandos en Sitios Cruzados. Los XSS ocurren cuando una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación

apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta JavaScript en el navegador. Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (defacement) los sitios web, o redireccionar al usuario hacia un sitio malicioso

A8: 2017 Deserialización Insegura. Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor

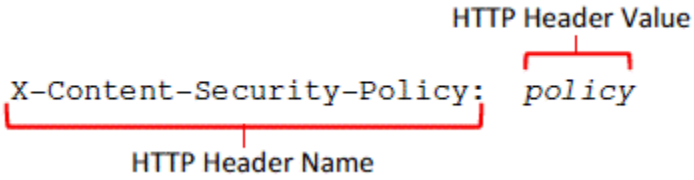
A9: 2017 Componentes con Vulnerabilidades Conocidas. Los componentes como bibliotecas, frameworks y otros módulos se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor. Las aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.

A10: 2017 Registro y Monitoreo Insuficientes. El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos

ANEXO B
CONTROLES DE SEGURIDAD BASADOS EN EL NAVEGADOR (EXTRAÍDO DEL CENTRO DE CIBERSEGURIDAD AUSTRALIANA)

Seguridad de Contenidos

Una política de seguridad de contenido (por sus siglas en inglés, CSP) proporciona controles de seguridad que pueden mitigar ataques de tipo "Cross-Site Scripting" (XSS); así como otros ataques basados en la inyección de contenido malicioso no deseado en una aplicación web. Esto se logra al especificar una lista blanca de orígenes de contenido que un navegador web aplica sobre una aplicación web. Se puede controlar una variedad de contenidos: secuencias de comandos, imágenes, audio y videos.
Para implementar este tipo de políticas en una aplicación web implica configurar el servidor web para incluir el encabezado HTTP de CSP en todas las respuestas HTTP, como se ilustra a continuación:



Los siguientes ejemplos a continuación ilustran implementaciones de CSP en una aplicación web.

1. Permitir solo dominio propio
   X-Content-Security-Policy: default-src 'self'
   El navegador web solo obtendrá contenido a nivel de www.govtenders.gov.au. Las medidas de seguridad de CSP JavaScript predeterminadas se aplican ya que no hay una directiva de exclusión voluntaria en esta política.
2. Permitir subdominios
   X-Content-Security-Policy: default-src *.govtenders.gov.au
   El navegador web generará contenido de govtenders.gov.au y todos los subdominios. Se aplican las mitigaciones de seguridad de JavaScript predeterminada
3. Restringir a sí mismo, permitir JavaScript en línea
   X-Content-Security-Policy: default-src 'self' 'unsafe-inline'
   El navegador web solo obtendrá contenido de www.govtenders.gov.au y permitirá la ejecución de JavaScript en línea del dominio GovTenders.
4. Permitir todo desde cualquier lugar pero bloquear scripts de terceros.
   X-Content-Security-Policy: default-src *; script-src 'self'
   El navegador web cargará cualquier contenido de cualquier dominio, pero solo cargará JavaScript de www.govtenders.gov.au. Inline JavaScript no se ejecuta.
5. Permitirse a sí mismo y autorizar fuentes externas.
   X-Content-Security-Policy: default-src 'self'; img-src *.cdn.example.com; media-src *.youtube.com; script-src *.jquery.com
   El navegador web cargará cualquier cosa de www.govtenders.gov.au; imágenes de cdn.example.com y cualquier subdominio; contenido de audio y video de youtube.com y cualquier subdominio; y scripts de jquery.com y cualquier subdominio

6. Forzar todas las solicitudes a través de HTTPS
   X-Content-Security-Policy: default-src https://*:443
   El navegador web cargará cualquier cosa desde cualquier lugar, pero todas las solicitudes utilizarán HTTPS
7. Política de informe de violación
   X-Content-Security-Policy: default-src 'self'; report-uri /cspreport
   El navegador web solo obtendrá contenido de www.govtenders.gov.au. Si algo infringe esta política, el navegador web enviará un informe de infracción a www.govtenders.gov.au/cspreport.

Actualmente hay utilerías que facilitan la implementación política CSP por ejemplo: bookmarklet, Web Scarab, SafeBrowsing, etc. En caso de querer desarrollar CSP propias hay más información en: https://w3c.github.io/webappsec-csp/ , https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

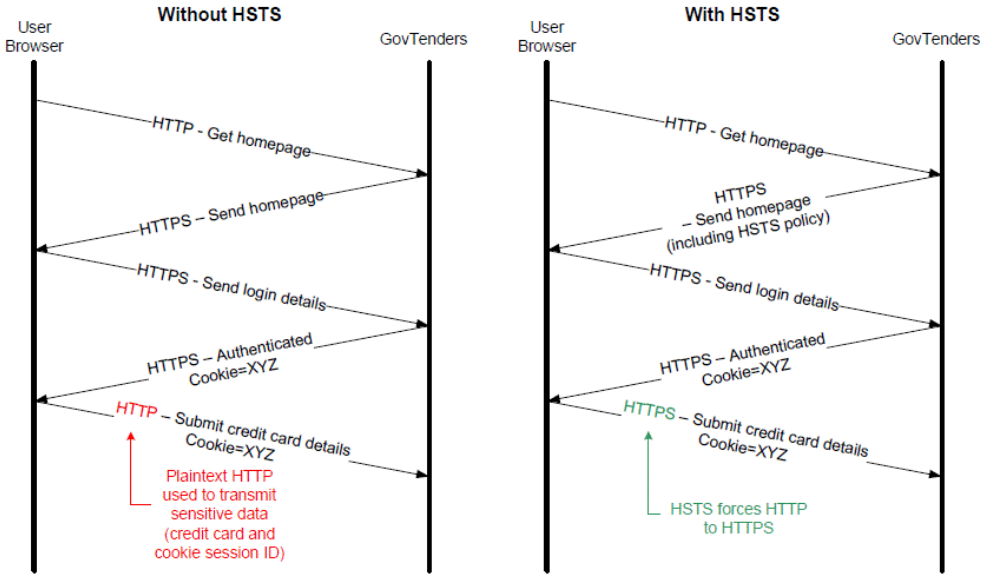
Seguridad de Transporte Estricto HTTP

HTTP Strict Transport Security (HSTS) mitiga la amenaza de espionaje ilegal y divulgación de información. HSTS hace esto al indicar a los navegadores web compatibles que solo usen conexiones seguras (es decir, HTTPS).

Al manejar información confidencial, es importante que una aplicación web use conexiones seguras para todas las comunicaciones. Si bien esto puede ser desafiante, especialmente para aplicaciones web más grandes o complejas, los riesgos de seguridad asociados con no usar comunicaciones integrales seguras incluyen:

- HTTPS solo se usa para enviar un nombre de usuario y contraseña, después de lo cual una aplicación web recurre a HTTP, exponiendo así información confidencial. Esto puede incluir sesiones en las cuales un adversario puede usar para hacerse pasar por un usuario legítimo. (Recordemos que la muy publicitada herramienta Firesheep explotó la falta de HTTPS de Facebook y Twitter para todas las comunicaciones https://bit.ly/2TqwFyy).
- Permitir conexiones no seguras para asegurar el contenido. El enfoque predeterminado de una aplicación web puede ser usar HTTPS para contenido seguro, sin embargo, todavía puede permitir conexiones no seguras a este contenido. Por ejemplo, un usuario visita una aplicación web mediante HTTP y no es dirigido a la versión HTTPS.

El siguiente ejemplo muestra una secuencia de transacción típica (aunque abreviada) cuando un usuario visita una aplicación web (GovTenders), luego de ello inicia sesión y proporciona algunos datos confidenciales. Desafortunadamente, GovTenders se desarrolló de forma deficiente y permitió que se pasaran datos confidenciales utilizando HTTP de texto sin formato.



Similar a la Política de seguridad de contenido (CSP), la implementación de HSTS para una aplicación web implica configurar el servidor web asociado para incluir el encabezado HSTS en todas las respuestas de HTTPS. Una directiva HSTS puede tomar dos formas diferentes:

- Seguridad de transporte estricta: edad máxima = segundos
- Seguridad de transporte estricta: edad máxima = segundos; includeSubDomains

Hay una serie de consideraciones al implementar una directiva HSTS. Éstos incluyen:

- Los encabezados de HSTS deben enviarse en respuestas HTTPS solo porque los navegadores web compatibles no aplicarán la política de HSTS enviada en texto sin formato HTTP.
- La respuesta HTTPS, que incluye un encabezado HSTS, no debe tener ningún error o advertencia de transporte seguro. Esto incluye el uso de certificados autofirmados, discrepancias de nombre de dominio y certificados caducados. Si se produce un error o advertencia de transporte seguro, un navegador web puede terminar la conexión y no presentarle al usuario una opción para continuar.
- Si una aplicación web no implementa un uso generalizado de HTTPS, puede haber un aumento en la sobrecarga de procesamiento debido a la implementación de HSTS. La cantidad de gastos generales depende de una serie de factores que incluyen el contenido, la duración de la sesión, el comportamiento de almacenamiento en caché y otros. La evaluación comparativa del rendimiento de una

aplicación web con y sin HSTS revelará cuánto procesamiento de sobrecarga se desempeña.

Con estas consideraciones en mente, se debe establecer un encabezado HSTS en todas las respuestas de HTTPS. Un navegador web compatible actualizará la caducidad de HSTS para un host cada vez que reciba una directiva HSTS válida de dicho host. El período de tiempo especificado en max-age dependerá de cuánto tiempo una organización está dispuesta a comprometerse con HTTPS solamente. Es preferible un tiempo de caducidad más largo, ya que cuanto menos tenga que conectarse un usuario a través de HTTP de texto sin formato, menos vulnerables serán los datos intercambiados con una aplicación web.

Para más información ver https://tools.ietf.org/html/rfc6797

Opciones "FRAME"

Las Opciones de marco impiden el uso de sitios web legítimos como parte de un ataque de *clickjacking* . Opciones "Frame" logra esto al definir si el contenido del sitio se puede incluir en un HTML <frame> o <iframe>,lo cuales luego son impuestos en los navegadores compatibles.

Habilitar las Opciones de marco para una aplicación web implica configurarla de modo que se pueda incluir el encabezado HTTP de Opciones de marco, en todas las respuestas HTTP o al menos en las páginas web con las que un usuario debe interactuar de forma segura. Una directiva de Opciones de marco puede ser uno de tres valores:

- X-Frame-Options: DENY - La directiva de denegación impide que el contenido protegido se incluya en cualquier marco.
- X-Frame-Options: SAMEORIGIN- La directiva sameorigin solo permite que el contenido protegido se incluya en un marco si el sitio de marcos se sirve desde el mismo origen.
- X-Frame-Options: ALLOW-FROM origin(s)- La directiva allow-from permite que uno o más orígenes enmarquen el contenido protegido. Los comodines no están autorizados para especificar múltiples dominios.

Antes de implementar las Opciones de marco, las organizaciones deben determinar si el contenido de una aplicación web que se quiere proteger, está legítimamente incluida en cualquier "Frame". Esto puede ser difícil de determinar para aplicaciones web fuera de una organización. Revisar el encabezado de "HTTP Referer" en las peticiones, puede revelar desde dónde está vinculado un usuario, pero esto también incluirá enlaces legítimos en lugar de solo el "framing". Además, el encabezado del "Referer" no siempre es confiable debido a la forma en que los navegadores web y otros intermediarios manejan el encabezado.

Si se usa un "framing" legítimo, esta debe determinar si se usa una directiva de uso interno o de uso externo. Adicionalmente, dado que los comodines no están permitidos en una directiva de autorizaciones, cada origen individual debe identificarse e incluirse.

Para más información de Opciones de Marco: https://tools.ietf.org/html/rfc7034

Mejoras en la seguridad de las Cookies.

Las cookies son vitales para las aplicaciones web, ya que proporcionan un medio para almacenar información sobre los clientes. Esta información puede incluir preferencias e información de seguimiento, pero lo más importante es las sesiones de usuario. Debido a la naturaleza sin estado de HTTP, los identificadores de sesión son el único medio práctico de seguimiento del estado, como el estado de autenticación.

Los ataques como "XSS" y "persona en el medio", aprovechan la confianza de las aplicaciones web en los identificadores de sesión para hacerse pasar por un usuario legítimo y secuestrar su sesión. Por lo tanto, la protección de cookies que contienen ID de sesión, es vital para la seguridad de las aplicaciones web. Hay dos controles de seguridad disponibles, a nivel de cookies, para ayudar a proteger las cookies; las opciones Secure y HttpOnly

Secure cookies

Como se identificó anteriormente en las discusiones de HSTS, una aplicación web puede usar HTTPS en algunas páginas web, pero no puede implementar o forzar su uso en otras. Debido a esto, las cookies que contienen ID de sesión pueden enviarse de forma clara, lo que permite el secuestro de una sesión. Para protegerse contra esto, la opción de Seguridad debe configurarse al emitir cookies que contienen datos confidenciales. La opción segura obliga a los navegadores web compatibles a enviar solo cookies a través de conexiones seguras, lo que evita que las cookies se envíen a través de HTTP de texto sin formato.

Para implementar cookies seguras, la opción segura se agrega al valor de la cookie cuando el servidor establece una cookie, por ejemplo Set-Cookie: PHPSESSID=1a9vnsk3haqpi29kamrnrul06c5; path=/; Secure

HttpOnly cookies

Si bien la opción Segura ayuda a garantizar que las cookies no se filtren a través de comunicaciones inseguras, no protege contra los ataques XSS . Un uso potencial de un ataque XSS es robar la cookie de un usuario y enviarla a un adversario. El adversario puede usar la cookie robada para hacerse pasar por el usuario, omitiendo así los controles de autenticación de una aplicación web. Para protegerse contra esto, la opción HttpOnly debe configurarse para evitar el acceso de JavaScript a las cookies.

Con el conjunto de opciones HttpOnly, los navegadores web compatibles solo recuperarán una cookie cuando se envíe como parte de una solicitud HTTP al origen emisor.

Al igual que la opción Segura, la opción HttpOnly se agrega al valor de la cookie cuando se establece una cookie, por ejemplo, Set-Cookie: PHPSESSID=1a9vnsk3haqpi29kamrnrul06c5; path=/; HttpOnly

Las opciones Secure y HttpOnly se pueden configurar al mismo tiempo.

Consideraciones de implementación

Al implementar las opciones de Secure y HttpOnly, las organizaciones primero deben revisar las aplicaciones web para determinar si estas mejoras de seguridad de cookies causarán algún problema. Por ejemplo, un componente clave de una aplicación web que no use HTTPS pero que se queda en espera de recibir una cookie. Dado que se ha configurado la opción Secure, la cookie no se envía al servidor a través de HTTP de texto sin formato, lo que puede causar problemas. Esto es indicativo de un problema más amplio de no usar HTTPS para proteger una aplicación web, en lugar de una incompatibilidad con la opción segura.

ANEXO C
LISTAS DE VERIFICACIÓN DE PRÁCTICAS SEGURAS DE CODIFICACIÓN (EXTRAÍDO Y TRADUCIDO DEL PROYECTO OWASP)

Validación de entradas

1. Realizar todas las validaciones de datos en un sistema confiable (por ejemplo: el servidor).
2. Identificar todas las fuentes de datos y clasificarlos como confiables o no confiables. Validar todos los datos provenientes de fuentes no confiables (por ejemplo: bases de datos, secuencias de archivo, etc.).
3. Debería existir una rutina de validación de datos de entrada centralizada para la aplicación.
4. Especificar sets de caracteres apropiados, tales como UTF-8, para todas las fuentes de entrada.
5. Codificar los datos a un set de caracteres común antes de validar (Canonicalización).
6. Todas las fallas en la validación deben terminar en el rechazo del dato de entrada.
7. Determinar si el sistema soportará sets de caracteres UTF-8 extendidos y de ser así, validarlos luego de terminada la decodificación del UTF-8.
8. Validar todos los datos brindados por el cliente antes de procesarlos, incluyendo todos los parámetros, URLs y contenidos de cabeceras HTTP (por ejemplo: nombres de Cookies y valores). Asegurarse de incluir post backs automáticos desde JavaScript, Flash u otro código embebido.
9. Verificar que los valores de la cabecera tanto en solicitudes como en respuestas contengan solo caracteres ASCII.
10. Validar datos redireccionados (un atacante puede enviar contenido malicioso directamente en el destino de la redirección, eludiendo entonces la lógica de la aplicación y cualquier otra validación realizada antes de la redirección).
11. Validar tipos de datos no esperados.
12. Validar rangos de datos.
13. Validar largos de datos.
14. Validar toda entrada con una lista "blanca" que contenga una lista de caracteres aceptados, siempre que sea posible.
15. Si es necesario, permitir el ingreso de algún carácter considerado peligroso, asegúrese de implementar controles adicionales tales como la codificación de la salida, API de seguridad y el registro del uso de tales datos a lo largo de la aplicación. Entre los ejemplos de caracteres peligrosos podemos encontrar: < > " ' % ( ) & + \ / \' \".
16. Si su rutina estándar de validación no contempla el ingreso de los ejemplos de datos anteriormente ejemplificados, entonces deberán verificarse de forma puntual.
17. Compruebe si hay bytes nulos (%00).
18. Compruebe si hay caracteres de nueva línea (%0d, %0a, \r, \n).
19. Compruebe si hay caracteres de alteraciones de ruta "punto, punto, barra (../ o ..\). En los casos en que se soportan sets de caracteres UTF-8 extendidos, implemente representaciones alternativas tales como: %c0%ae%c0%ae/ (utilice la canonicalización como forma de implementar la doble codificación u otras formas de ofuscación de ataques).

Codificación de salidas

20. Realice toda la codificación en un ambiente seguro (por ejemplo: el servidor).
21. Utilice una rutina probada y estándar para cada tipo de codificación de salida.
22. Contextualice la codificación de salida de todos los datos devueltos por el cliente que se originen desde fuera de la frontera de confianza de la aplicación. La codificación de entidades HTML es un ejemplo, aunque no sea suficiente en todos los casos.
23. Codifique todos los caracteres salvo que sean reconocidos como seguros por el interpretador al que están destinados.
24. Sanitice contextualmente todos las salidas de datos no confiables hacia consultas SQL, XML y LDAP.
25. Sanitice todas las salidas de datos no confiables hacia un comando del sistema operativo.

Administración de autenticación y contraseñas

26. Requerir autenticación para todos los recursos y páginas excepto aquellas específicamente clasificadas como públicas.
27. Todos los controles de autenticación deben ser efectuados en un sistema en el cual se confíe. (por ejemplo: el servidor).
28. Establecer y utilizar servicios de autenticación estándares y probados cuando sea posible.
29. Utilizar una implementación centralizada para todos los controles de autenticación, incluyendo librerías que llamen a servicios externos de autenticación.
30. Segregar la lógica de la autenticación del recurso solicitado y utilizar redirección desde y hacia el control centralizado de autenticación.
31. Todos los controles de autenticación deben fallar de una forma segura.
32. Todas las funciones administrativas y de administración de cuentas deben ser al menos tan seguras como el mecanismo primario de autenticación.
33. Si la aplicación administra un almacenamiento de credenciales, se debe asegurar que únicamente se almacena el "salty hash" de las contraseñas y que el archivo/tabla que guarda las contraseñas y claves solo puede ser escrito por la aplicación (si es posible, no utilizar el algoritmo de hash MD5).
34. El hash de las contraseñas debe ser implementado en un sistema en el cual se confíe. (por ejemplo: el servidor).
35. Validar los datos de autenticación únicamente luego haber completado todos los datos de entrada, especialmente en implementaciones de autenticación secuencial.
36. Las respuestas a los fallos en la autenticación no deben indicar cual parte de la autenticación fue incorrecta. A modo de ejemplo, en lugar de "usuario invalido" o "contraseña invalida", utilizar "usuario y/o contraseña inválidos" en ambos casos. Las repuestas a los errores deben ser idénticas tanto a nivel de lo desplegado como a nivel de código fuente.
37. Utilizar autenticación para conexiones a sistemas externos que involucren información o funciones sensibles.
38. Las credenciales de autenticación para acceder a servicios externos a la aplicación deben ser encriptados y almacenados en ubicaciones protegidas en un sistema en el cual se confíe (ejemplo: el servidor). El código fuente NO es una ubicación segura.
39. Utilizar únicamente pedidos del tipo HTTP POST para la transmisión de credenciales de autenticación.

40. Utilizar únicamente conexiones encriptadas o datos encriptados para el envío de contraseñas que no sean temporales (por ejemplo: correo encriptado). Contraseñas temporales como aquellas asociadas con reseteos por correo electrónico, pueden ser una excepción.
41. Hacer cumplir por medio de una política o regulación los requerimientos de complejidad de la contraseña. Las credenciales de autenticación deben ser suficientes como para resistir aquellos ataques típicos de las amenazas en el entorno del sistema. Ejemplo: obligar el uso de combinaciones de caracteres numéricos/alfanuméricos y/o caracteres especiales.
42. Hacer cumplir por medio de una política o regulación los requerimientos de longitud de la contraseña. Comúnmente se utilizan ocho caracteres, pero dieciséis es mejor, adicionalmente considerar el uso de frases de varias palabras.
43. No se debe desplegar en la pantalla la contraseña ingresada. A modo de ejemplo, en formularios web, utilizar el tipo de entrada "password" (input type "password").
44. Deshabilitar las cuentas luego de un número establecido de intentos inválidos de ingreso al sistema. A modo de ejemplo, cinco intentos fallidos es lo común. La deshabilitación de la cuenta debe ser por un periodo de tiempo suficiente como para desalentar una inferencia de credenciales por fuerza bruta, pero no tan alto como para permitir un ataque de negación de servicio.
45. El cambio y reseteo de contraseñas requieren los mismos niveles de control como aquellos asociados a la creación y autenticación de cuentas.
46. Las preguntas para el reseteo de contraseñas deben contemplar un amplio rango de respuestas aleatorias. A modo de ejemplo: "libro favorito?" es una mala pregunta dado que "la biblia" es una respuesta muy común.
47. Si se utiliza un reseteo por correo electrónico, únicamente enviar un link o contraseñas temporales a casillas previamente registradas en el sistema.
48. Las contraseñas y links temporales deben tener un corto periodo de validez.
49. Forzar el cambio de contraseñas temporales luego de su utilización.
50. Notificar a los usuarios cada vez que se produce un reseteo de contraseña.
51. Prevenir la reutilización de contraseñas.
52. Las contraseñas deben tener al menos un día de antigüedad antes de poder ser cambiadas, de forma de evitar ataques de reutilización de contraseñas.
53. Hacer cumplir por medio de una política o regulación los requerimientos de cambio de contraseña. Los sistemas críticos pueden requerir cambios más frecuentes que otros sistemas. El tiempo entre cada reseteo debe ser controlado administrativamente.
54. Deshabilitar la funcionalidad de "recordar" campos de contraseñas.
55. El último acceso (fallido o exitoso) debe ser reportado al usuario en su siguiente acceso exitoso.
56. Implementar un monitoreo para identificar ataques a múltiples cuentas utilizando la misma contraseña. Este patrón de ataque es utilizado parar superar bloqueos estándar cuando los nombres de usuario pueden ser obtenidos o adivinados de alguna forma.
57. Cambiar todos los usuarios y contraseñas por defecto o deshabilitar las cuentas asociadas.
58. Re autenticar usuarios antes de la realización de operaciones críticas.
59. Utilizar autenticación multi-factor para las cuentas más sensibles o de mayor valor.

60. Si se utiliza un código de terceros para la autenticación, inspeccionarlo minuciosamente para asegurar que no se encuentre afectado por cualquier código malicioso.

Administración de sesiones

61. Utilizar los controles del servidor o del framework para la administración de sesiones. La aplicación solo debe reconocer estos identificadores como válidos.
62. La creación de identificadores de sesión solo debe ser realizada en un sistema en cual se confíe (por ejemplo: el servidor).
63. Los controles de administración de sesiones deben utilizar algoritmos que generen identificadores suficientemente aleatorios.
64. Definir el dominio y ruta para las cookies que contienen identificadores de sesión autenticados con un valor apropiadamente estricto para el sitio.
65. La función de logout debe terminar completamente con la sesión o conexión asociada.
66. La función de logout debe estar disponible en todas las páginas protegidas por autenticación.
67. Establecer un tiempo de vida de la sesión lo más corto posible, balanceando los riesgos con los requerimientos del negocio. En la mayoría de los casos, nunca debería ser superior a varias horas.
68. Deshabilitar logeos persistentes y efectuar finalizaciones periódicas de sesiones, incluso cuando la sesión se encuentra activa.
69. Si una sesión fue establecida antes del login, cerrar dicha sesión y establecer una nueva luego de un login exitoso.
70. Generar un nuevo identificador de sesión luego de cada reautenticación.
71. No permitir logeos concurrentes con el mismo usuario.
72. No exponer identificadores de sesión en URLs, mensajes de error ni logs. Los identificadores de sesión solo deben ser ubicados en la cabecera de la cookie HTTP. A modo de ejemplo, no transmitir el identificador de sesión como un parámetro GET.
73. Proteger la información sobre las sesiones del lado del servidor implementando los controles de acceso apropiados.
74. Generar un nuevo identificador de sesión y desactivar el anterior de forma periódica. De esta forma se pueden mitigar algunos escenarios de robo de sesiones donde el identificador se compromete.
75. Generar un nuevo identificador de sesión si la seguridad cambia de HTTP a HTTPS, como puede suceder durante la autenticación. Dentro de la aplicación es recomendable usar siempre HTTPS en lugar de cambiar entre HTTP y HTTPS.
76. Manejo de sesión complementario para operaciones sensible del lado del servidor, como pueden ser: gestión de cuentas o utilizando tokens o parámetros por sesión. Este método puede ser utilizado para prevenir ataques de Cross Site Request Forgery Attacks (Falsificación de petición en sitios cruzados, *conocido por sus siglas CSRF).*
77. Manejo de sesión complementario para operaciones sensibles o críticas utilizando tokens o parámetros por pedido (per request) en lugar de por sesión.
78. Configurar el atributo "seguro" para las cookies transmitidas sobre una conexión TLS.
79. Configurar las cookies con el atributo HttpOnly, salvo que se requiera específicamente scripts del lado del cliente en la aplicación, para leer o configurar una cookie.

Control de Acceso

80. Utilizar únicamente objetos confiables del sistema. (por ejemplo: objetos de sesión del servidor para la toma de decisiones de autorización).
81. Utilizar un único componente para el chequeo de autorizaciones para todo el sitio. Esto incluye librerías que llamen a servicios de autorización externos.
82. Los controles de acceso en caso de falla, deben actuar en forma segura.
83. Denegar todos los accesos en caso de que la aplicación no pueda acceder a la información de configuración de seguridad.
84. Requerir controles de autorización en cada solicitud o pedido, incluyendo aquellos creados por scripts en el servidor, "includes" y pedidos desde AJAX o Flash desde el lado del cliente.
85. Separar lógica privilegiada de otro código de la aplicación.
86. Restringir acceso a ficheros u otros recursos, incluyendo aquellos fuera del control directo de la aplicación, únicamente a usuarios autorizados.
87. Restringir el acceso a URLs protegidas, solo a usuarios autorizados.
88. Restringir el acceso a funciones protegidas, solo a usuarios autorizados.
89. Restringir las referencias directas a objetos, solo a usuarios autorizados.
90. Restringir el acceso a servicios, solo a usuarios autorizados.
91. Restringir el acceso a información de la aplicación, solo a usuarios autorizados.
92. Restringir el acceso a usuario, atributos y política de información utilizada por los controles de acceso.
93. Restringir el acceso a información relevante de la configuración, solo a usuarios autorizados.
94. Las reglas de control de acceso implementadas del lado del servidor y en la capa de presentación, deben coincidir.
95. Si existen datos de estado que deben ser guardados en el cliente, use cifrado y chequeos de integridad del lado del servidor para poder evaluar el estado.
96. Requerir flujos de aplicación que cumplan con las reglas del negocio.
97. Limitar el número de transacciones que un usuario común o un dispositivo puede desarrollar en un cierto período de tiempo.
98. Utilizar el header "referer" solo como un chequeo complementario. Nunca debe ser utilizado como chequeo de autorización, ya que es posible modificarlo.
99. Si las sesiones de autenticación extensas son permitidas, periódicamente hay que revalidar la autorización de los usuarios y asegurar que sus privilegios no han sido modificados.
100. Implementar auditorías de cuentas y requerir que se deshabiliten las contraseñas de las cuentas.
101. La aplicación debe permitir deshabilitar y terminar cuentas una vez que se termina la autorización (Cambio de rol, estatus de empleo, etc).
102. Cuentas de servicio o cuentas soportando conectividad deben tener el mínimo privilegio.
103. Crear una política de control de acceso para documentar las reglas del negocio de la aplicación, los tipos de datos, criterios para autorización de acceso y los controles asociados para otorgarlos y controlarlos. Esto incluye la identificación de accesos requeridos tanto para los datos como para los sistemas.

Prácticas Critpográficas

104. Todas las funciones de criptografía de la aplicación deben ser implementadas en entornos confiables (por ejemplo: el servidor).
105. Proteger secretos maestros (master secrets) de acceso no autorizado.
106. Los módulos de criptografía deberían en caso de falla, fallar en forma segura.
107. Todos los números aleatorios, nombres aleatorios, GUIDs, y frases aleatorias, deberían generarse utilizando módulos aprobados para su generación.
108. Los módulos criptográficos del sistema deben cumplir con FIPS 140-2 o con su estándar equivalente. (Ver: http://csrc.nist.gov/groups/STM/cmvp/validation.html).
109. Establecer y utilizar una política y un proceso de cómo manejar las claves criptográficas.

Manejo de errores y Logs

110. No difundir información sensible en respuestas de error, incluyendo detalles del sistema, identificadores de sesión o información de la cuenta.
111. Utilizar manejadores de errores que no muestren información de debugging o de memoria.
112. Implementar mensajes de error genéricos y utilizar páginas de error adaptadas.
113. La aplicación debería manejar los errores de la aplicación y basarse en la configuración del servidor.
114. Liberar espacio de memoria en cuanto una condición de error ocurra.
115. La lógica para la gestión de errores debe estar asociada a los controles de seguridad que no permitirán acceso por defecto.
116. Todos los controles de logging deben estar implementados en sistemas confiables.
117. El logging de controles de acceso debe incluir tanto los casos de éxito como de falla.
118. Asegurar que los datos del registro de log contengan información importante.
119. Asegurar que los logs de entrada que incluyen información no segura, no serán ejecutados en interfaces o aplicativos.
120. Restringir el acceso a los logs, solo a personal autorizado.
121. Utilizar una rutina centralizada para todas las operaciones de logging.
122. No guardar información sensible en logs, incluyendo detalles innecesarios del sistema.
123. Asegurar que existen mecanismos para conducir un análisis de los logs.
124. Registrar en un log todas las fallas de validación.
125. Registrar en un log todos los intentos de autenticación, en particular los fallidos.
126. Registrar en un log todas las fallas en los controles de acceso.
127. Registrar en un log todos los eventos de intento de evasión de controles, incluyendo cambios en el estado de la información no esperados.
128. Registrar en un log todos los intentos de conexión con tokens inválidos o vencidos.
129. Registrar en un log todas las excepciones del sistema.
130. Registrar en un log todas las funciones administrativas, incluyendo cambios en la configuración de seguridad.
131. Registrar en un log, todas las fallas de conexión de TLS (Transport Layer Security).
132. Registrar en un log las fallas de los módulos criptográficos.
133. Utilizar una función de hash para validar la integridad de los logs.

Protección de datos

134. Implementar el mínimo privilegio, restringir el acceso de los usuarios solamente a la funcionalidad, datos y sistemas de información que son necesarios para realizar sus tareas.
135. Proteger todos los almacenamientos temporales (en memoria o archivo -"cached") de los datos sensibles guardados en el servidor de los accesos no autorizados y eliminar todos los archivos y memoria de trabajo temporal tan pronto como no sean requeridos.
136. Encriptar toda la información altamente sensible almacenada, como por ejemplo: datos para la verificación de la autenticación, inclusive también en el servidor. Siempre utilice algoritmos de encriptación que hayan sido validados y con buenos antecedentes. Vea "Prácticas Criptográficas" para obtener información adicional.
137. Proteja el código fuente del servidor de forma de que no pueda ser descargado por el usuario.
138. No almacene contraseñas, cadenas de conexión u otra información sensible en texto claro o de forma que no sea criptográficamente segura del lado del cliente. Esto incluye almacenarlas en formatos inseguros tales como: MS Viewstate, Adobe Flash o código compilado.
139. Remueva los comentarios en el código de producción que sea accesible por el usuario que puedan revelar información sobre los servidores o información sensible.
140. Remueva cualquier aplicación que no sea necesaria y la documentación de los sistemas que pueda revelar información útil para los atacantes.
141. No incluya información sensible en los parámetros del HTTP GET.
142. Deshabilite las funcionalidades de completar automáticamente en aquellos formularios que contienen información sensible, incluyendo la autenticación.
143. Deshabilite el almacenamiento temporal del lado del cliente de páginas que contienen información sensible. "Cache-Control: no-store" debería ser utilizado en conjunto con el control en el cabezal de HTTP "Pragma: no-cache", que es menos efectivo, pero mantiene compatibilidad con HTTP/1.0.
144. La aplicación debe de soportar la eliminación de datos sensibles cuando los datos ya no son requeridos (por ejemplo: información personal o ciertos datos financieros).
145. Implementar controles de accesos apropiados para los datos sensibles almacenados en el servidor. Esto incluye memoria temporal, archivos temporales y datos que solamente puedan ser accedidos por usuarios específicos del sistema

Seguridad en las comunicaciones

146. Implemente encriptación para todas las transmisiones de información sensible. Esto debería incluir TLS (Transport Security Layer) para proteger la conexión y se puede combinar con una encriptación discreta de archivos sensibles en conexiones que no estén basadas en HTTP.
147. Los certificados TLS deben de ser válidos y contener el nombre de dominio correcto, no deben estar expirados y deberán ser instalados con los certificados intermedios si son requeridos.
148. Las conexiones TLS que fallen no deben de transformarse en una conexión insegura.
149. Utilizar conexiones TLS para todo el contenido que requiera acceso autenticado y para otro tipo de información sensible.
150. Utilizar TLS para las conexiones a sistemas externos que involucren funciones o información sensible.

151. Utilizar una única implementación estándar de TLS que se encuentre configurada correctamente.
152. Especificar los caracteres de codificación para todas las conexiones.
153. Filtrar los parámetros que contengan información sensible de los referer HTTP, cuando existen vínculos a sitios externos.

Configuración de los sistemas

154. Asegurarse de que los servidores, los frameworks y los componentes del sistema están corriendo la última versión aprobada.
155. Asegurar que los servidores, los frameworks y los componentes del sistema están actualizados con todos los parches emitidos para las versiones en uso.
156. Deshabilitar el listado de directorio.
157. Restringir el servidor web, los procesos y las cuentas de servicios con el mínimo privilegio posible.
158. Cuando ocurra una excepción, se debe de fallar de forma segura.
159. Remover todas las funcionalidades y archivos que no sean necesarios.
160. Remover código de testeo o cualquier funcionalidad que no sea tenida en cuenta en producción, previo a realizar la puesta en producción.
161. Prevenir la revelación de la estructura de directorios en el archivo robots.txt colocando directorios que estén disponibles para el índice público en un directorio raíz aislado. Luego "Deshabilitar" el directorio raíz en el archivo robots.txt en vez de "Deshabilitar" cada directorio individual.
162. Definir cuáles de los métodos HTTP, GET o POST, la aplicación va a soportar y si deben de ser manejados de forma diferente en las distintas páginas de la aplicación.
163. Deshabilitar métodos HTTP innecesarios como extensiones WebDAV. Si una extensión del método HTTP que soporte manejo de archivos es necesaria, utilice un mecanismo de autenticación bien comprobado.
164. Si el servidor web maneja HTTP 1.0 y HTTP 1.1, asegurarse que ambos están configurados de manera similar, o bien, asegurarse de entender bien las diferencias que puedan existir (por ejemplo: el manejo de métodos extendidos de HTTP).
165. Remover información innecesaria en los cabezales de HTTP de respuesta referidas al SO, versión del servidor web y frameworks de aplicación.
166. La configuración de seguridad de donde se encuentra almacenada la aplicación debe de ser listada en un formato legible para su auditoría.
167. Implementar un Sistema de Manejo de Activos y registrar los componentes del sistema en él.
168. Aislar los ambientes de desarrollo de los ambientes de producción y permitir el acceso solamente a los grupos de desarrollo y testeo específicamente autorizados. Los ambientes de desarrollo a menudo son configurados de forma menos segura que los ambientes de producción y los atacantes pueden utilizar estas diferencias para descubrir vulnerabilidades o como un camino para poder explotarlas.
169. Implementar un Sistema de Control de Cambios del Software para manejar y registrar los cambios al código tanto para los ambientes de desarrollo como para los de producción.

Seguridad de Base de Datos

170. Utilice consultas parametrizadas con tipos de datos fuertemente tipados.
171. Utilice validación de las entradas y codificación de las salidas y asegúrese de manejar los metacaracteres. Si esto falla, no ejecute el comando de la base de datos.

172. Asegúrese de que todas las variables tengan tipos de datos asociados.
173. La aplicación debe de utilizar el mínimo nivel de privilegios cuando accede a la base de datos.
174. Utilice credenciales seguras para acceder a la base de datos.
175. Las cadenas de conexión a la base de datos no deben de estar incluidas en el código de la aplicación. Las cadenas de conexión a la base de datos deben de estar en un archivo de configuración separado en un sistema confiable y debería de estar encriptado.
176. Utilice procedimientos almacenados para abstraer el acceso a los datos y elimine los permisos de las tablas en la base de datos.
177. Cierre la conexión a la base de datos tan pronto como sea posible.
178. Remueva o cambie todas las contraseñas administrativas por defecto. Utilice contraseñas fuertes o implemente autenticación de múltiples factores.
179. Deshabilite todas las funcionalidades innecesarias de la base de datos (por ejemplo: procedimientos almacenados innecesarios, servicios no utilizados, paquetes de utilerías, instale solo el conjunto mínimo de funcionalidades y opciones requeridas (reduzca el área de ataque).
180. Eliminar el contenido innecesario incluido por el proveedor (por ejemplo: esquemas de ejemplo).
181. Deshabilitar las cuentas por omisión que no son necesarias para la operación del negocio.
182. La aplicación debería conectarse a la base de datos con credenciales diferentes para cada nivel de confianza (por ejemplo: usuarios, usuarios solo lectura, invitados, administrador).

Manejo de Archivos

183. No utilizar directamente información provista por el usuario en ninguna operación dinámica.
184. Exigir autenticación antes de permitir la transferencia de un archivo al servidor.
185. Permita transferir al servidor únicamente los tipos de archivo (por ejemplo: pdf, doc, etc) requeridos por la operación del negocio.
186. Validar los tipos de archivo transferidos verificando la estructura de los encabezados. La validación del tipo de archivo únicamente por la extensión no es suficiente.
187. No guardar los archivos transferidos en el mismo contexto que la aplicación web. Los archivos deben almacenarse en un repositorio específico o en una base de datos.
188. Evite o restrinja la transferencia de archivos que puedan ser interpretados por el servidor web (por ejemplo: asp, php, jsp, etc).
189. Eliminar los permisos de ejecución a los archivos transferidos.
190. Implemente una transferencia de archivos segura en UNIX mediante el uso de discos lógicos y el uso de las rutas (path) correspondientes o mediante la utilización de un entorno sh root.
191. Cuando se referencie a un archivo existente en el servidor, utilizar una lista blanca de nombres y extensiones válidas. Validar el contenido del parámetro pasado contra esta lista blanca y si encuentra una coincidencia, denegar la operación o transferir un archivo predefinido.
192. No utilizar información provista por el usuario para la generación de redirecciones dinámicas. Si se debe proveer esta funcionalidad, la redirección debe aceptar

únicamente caminos relativos dentro de la URL previamente establecidos (lista blanca de directorios relativos).

193. No incluir en parámetros nombres de directorios o rutas de archivos, en su lugar utilizar índices que internamente se asocien a directorios o rutas predefinidas.
194. Nunca envíe la ruta absoluta de un archivo al cliente.
195. Asegúrese que los archivos y recursos de la aplicación sean de solo lectura.
196. Revise los archivos transferidos por los clientes en busca de virus y malware.

Manejo de Memoria

197. Utilice controles en la entrada y la salida de información no confiable.
198. Revise dos veces que el largo de los buffers sean los requeridos y especificados.
199. Cuando utilice primitivas que requieran el número de bytes a copiar, ejemplo: strncpy(), sea consciente que si el tamaño del buffer destino es igual al tamaño del buffer origen, el destino podría quedar en NULL-byte requerido al final.
200. Verifique los límites de los buffers si se llama a las funciones dentro de un loop y asegúrese de no correr riesgo de escribir fuera del espacio reservado.
201. Trunque el largo de todos los strings de entrada a un tamaño razonable antes de pasarlos a una función de copia o concatenación.
202. Específicamente libere los recursos, no confíe en el garbage collector (por ejemplo: objetos de conexión, handlers de archivos, etc.)
203. Utilice stacks no ejecutables cuando sea posible (NX bit).
204. Evite el uso de primitivas con vulnerabilidades conocidas (por ejemplo: printf, strcat, strcpy, etc).
205. Libere adecuadamente la memoria previa a la salida de una función y de todos los puntos de finalización de la aplicación.

Practicas Generales para la Codificación

206. Para las tareas habituales, utilice código probado y verificado en lugar de crear códigos específicos.
207. Utilice las APIs previstas para el acceso a funciones específicas del sistema operativo. No permita que la aplicación ejecute comandos directamente en el sistema operativo, menos aún mediante la invocación de una shell.
208. Utilizar funciones de control de integridad (hash, CRC o checksum) para verificar la integridad del código interpretado, bibliotecas, ejecutables y archivos de configuración previo a su utilización.
209. Utilice locks para evitar múltiples accesos simultáneos a los recursos o mecanismos de sincronización (por ejemplo: semáforos) para evitar condiciones de borde (race conditions).
210. Proteja las variables y recursos compartidos de accesos concurrentes inadecuados.
211. Explícitamente inicialice todas las variables y mecanismos de almacenamiento de información (por ejemplo: buffers), durante su declaración o antes de usarlos por primera vez.
212. Las aplicaciones que requieran privilegios especiales deberán elevar los privilegios solo cuando sea necesario y devolverlos (bajar privilegios) lo antes posible. Mantener los privilegios especiales únicamente cuando sea estrictamente necesario.
213. Evitar errores de cálculo comprendiendo la forma en que el lenguaje de programación maneja las operaciones matemáticas y las representaciones numéricas. Prestar especial atención a las discrepancias en la cantidad de bytes utilizados para la representación, la precisión, diferencias entre valores con y sin signo, truncamiento,

conversiones y casting entre tipos de variables, cálculos "no-numéricos" y cómo el lenguaje maneja los números demasiado grandes o demasiado pequeños para su representación.

214. No utilizar datos provistos por el usuario para ninguna función dinámica.
215. Evite que los usuarios introduzcan o modifiquen código de la aplicación.
216. Revisar todas las aplicaciones secundarias, código provisto por terceros y bibliotecas para determinar su utilización en la necesidad del negocio, y validar el funcionamiento seguro, ya que estos pueden introducir nuevas vulnerabilidades.
217. Implementar mecanismos seguros para las actualizaciones. Si la aplicación realiza actualizaciones automáticas, utilizar firmas criptográficas para el código y asegurarse que el cliente que descarga la aplicación verifique dichas firmas. Utilizar canales encriptados para las transferencias de código desde el servidor de actualización.

ANEXO D
CONTROLES DE SEGURIDAD SOBRE LA CONFIGURACIÓN DEL IIS

Protocolos de seguridad sobre IIS (TLS / SSL) información basada en el sitio : http://sites.ieee.org/ocs-cssig/?page_id=658

TLS: Transport Layer Security

La Seguridad de la capa de transporte (TLS) es la sucesora de la Capa de sockets seguros (SSL). Los protocolos definen los mecanismos para garantizar la transmisión segura de datos a través de Internet. Los estándares están controlados por el Grupo de trabajo de ingeniería de Internet (IETF®).

El estándar define el *protocolo de* negociación que define el cifrado y el protocolo que se utilizará para transmitir *registros de* datos. La naturaleza variada de las implementaciones del estándar proporciona un lugar para la interrupción, es decir, es posible promover ataques exitosos por parte de piratas informáticos de CyberSecurity.

SSL1, SSL2, SSL3 y TLS1.0 tienen fallas conocidas. SSL fue desarrollado por Netscape en los años 90 y fue un medio preliminar y exploratorio para definir la comunicación segura. El 'primero en el bloque' permitió el inicio y desarrollo del concepto y la implementación del ideal, para promover el transporte seguro de datos. TLS1.0 corrigió algunas de las fallas en SSL pero, en un intento por ajustarse a las restricciones del gobierno de los EE. UU. A la exportación de material criptográfico, había conocido un cifrado débil con protocolos hackeables. TLS1.2 es el estándar actual, y TLS1.3 es el próximo estándar en revisión

Para desactivar los protocolos no seguros se debe modificar la configuración de web con los siguientes parámetros:

A través del Administrador de PowerShell:

#Disable PCT 1.0

```
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\"  -
Name "PCT
1.0" -Value "DefaultValue" -Force
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PC
T 1.0\"
-Name "Server" -Value "DefaultValue" -Force
New-ItemProperty
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\
Protocols\PCT 1.0\Server\" -Name Enabled -Value 0 -PropertyType "DWord" -Force
#Disable SSL 2.0
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\"  -
Name "SSL
2.0" -Value "DefaultValue" -Force
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SS
L 2.0\"
-Name "Server" -Value "DefaultValue" -Force
New-ItemProperty
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\
Protocols\SSL 2.0\Server\" -Name Enabled -Value 0 -PropertyType "DWord" -Force
#Disable SSL 3.0
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\"  -
Name "SSL
3.0" -Value "DefaultValue" -Force
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SS
L 3.0\"
-Name "Server" -Value "DefaultValue" -Force
New-ItemProperty
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\
Protocols\SSL 3.0\Server\" -Name Enabled -Value 0 -PropertyType "DWord" -Force
#Disable TLS 1.0
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\"  -
Name "TLS
1.0" -Value "DefaultValue" -Force
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TL
S 1.0\"
-Name "Server" -Value "DefaultValue" -Force
New-ItemProperty
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\
Protocols\TLS 1.0\Server\" -Name Enabled -Value 0 -PropertyType "DWord" -Force
```

#Disable TLS 1.1
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\" -Name "TLS
1.1" -Value "DefaultValue" -Force
ni
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.1\"
-Name "Server" -Value "DefaultValue" -Force
New-ItemProperty
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.1\Server\" -Name Enabled -Value 0 -PropertyType "DWord" - Force

A través de modificación de registro (como Administrador).

- HKey_Local_Machine\System\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols
- Si las claves que vemos en la imagen no existen se han de crear. Recordad que el Registro de Windows distinge entre Mayusculas y Minusculas

- Dentro de Client y Server de las claves TLS 1.1 y TLS 1.2 se han de añadir dos valores DWORD de 32bits
- Dentro de Client y Server de las claves SSL 2 se han de añadir dos valores DWORD de 32 bits, Activando el Disabled y forzando que no se active.

BLOQUEAR ARCHIVOS O DIRECTORIOS QUE MUESTREN LA VERSION DE LOS SERVICIOS WEB.

Para cambiar los permisos de acceso a un archivo o directorio expuesto en el sistema Web, una de las opciones es editar directamente el archivo de configuración del sitio. En el folder ISAPI el archivo Web.Config, agregar los siguientes parámetros.

```
<location path="_vti_inf.html">
    <system.web>
        <authorization>
            <deny users= "?" />
            <allow users = "*" />
        </authorization>
    </system.web>
</location>
```

Los usuarios autentificados se representan con el signo "*" los usuarios anónimos con el signo " ?". Para mayor información consulte: https://thuansoldier.net/4267/

DESACTIVAR LA VERSION ASP.NET.

Para desactivar que se muestre la versión de ASP.NET en los mensajes de error o cabeceras del sistema se debe agregar un parámetro en el Web.config

<System.Web> <httpRuntime enableVersionHeader="false" /> </System.Web>

Para mayor información consultar : https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/version-disclosure-aspnet/

ACTIVANDO O DESACTIVANDO CIPHER SUITE:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\

A través del PowerShell

```
([Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey([Microsoft.Win32.RegistryHive]::LocalMachine,$env:COMPUTERNAME)).CreateSubKey('SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 128/128')
    New-ItemProperty -
path 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 128/128' -name 'Enabled' -value '0' -PropertyType 'DWord' -Force | Out-Null

    ([Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey([Microsoft.Win32.RegistryHive]::LocalMachine,$env:COMPUTERNAME)).CreateSubKey('SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 40/128')
    New-ItemProperty -
path 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 40/128' -name 'Enabled' -value '0' -PropertyType 'DWord' -Force| Out-Null

    ([Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey([Microsoft.Win32.RegistryHive]::LocalMachine,$env:COMPUTERNAME)).CreateSubKey('SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 56/128')
    New-ItemProperty -
path 'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\RC4 56/128' -name 'Enabled' -value '0' -PropertyType 'DWord' -Force| Out-Null
```

Para Mayor información de los protocoles de seguridad sobre IIS consultar: https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/manage-ssl-protocols-in-ad-fs

Protocolo de SMB para carpetas compartidas.
El protocolo Server Message Block (SMB) es un protocolo de intercambio de archivos de red que permite que las aplicaciones en una computadora lean y escriban archivos y soliciten servicios de programas de servidor en una red de computadoras.
https://docs.microsoft.com/en-us/windows-server/storage/file-server/file-server-smb-overview.

Para el uso de carpetas compartidas se debe asegurar que se tiene la versión actualizada de los protocolos de comunicación como es en el caso de SMB, actualmente es la versión 3.

Para detectar, desactivar o activar SMB v1 SMBv2 y SMBv3 se hace desde la consola de PowerShell o desde el resgistro de Windows.
Detectar: `Get-WindowsOptionalFeature -Online -FeatureName smb1protocol`
Desactivar: `Disable-WindowsOptionalFeature -Online -FeatureName smb1protocol`
Habilitar: `Set-SmbServerConfiguration -EnableSMB3Protocol $true.`
Para mayor información dependiendo del sistema operativo consultar la siguiente página: https://docs.microsoft.com/en-us/windows-server/storage/file-server/troubleshoot/detect-enable-and-disable-smbv1-v2-v3

Matamoros SN, Zona Centro
Ciudad Victoria, Tamaulipas,
C.P. 87000
(834) 318-1800, ext. 2823
www.uat.edu.mx