Guía de Estándares de la Industria sobre Codificación Segura para Terceros

Versión 1.0

Fecha: 19/10/2016,

Área de Seguridad de BT

Contenido

1. Introducción

1.1. Antecedentes

Este documento orienta en las buenas prácticas generales para el desarrollo de software realizado por terceros que proveen aplicaciones o sistemas a BT.

Las vulnerabilidades en las capas de aplicaciones son extremadamente eficaces para que las personas consigan acceso no autorizado a datos o sistemas. Así es que se vuelve cada vez más importante que el modo de desarrollar software tenga las consideraciones de seguridad como eje central.

1.2. Objetivos

Este documento intenta proporcionar un conjunto de guías sobre cómo desarrollar software de manera que el código resultante sea confiable y seguro. El desarrollo de software incluye muchos aspectos y es un campo que cambia rápidamente; por eso, se ha delineado un conjunto de principios de alto nivel que se han de seguir en la medida de lo posible en las distintas etapas de un proyecto de software. El documento está dividido en tres secciones principales:

Diseño del sistema

Esta sección orienta en qué hacer en el comienzo de un proyecto de software y delinea varias áreas que deben considerarse antes de escribir un código. Si se siguen estos pasos, será más simple garantizar que se entrega un código seguro cuando el proyecto esté más avanzado.

Entrega del sistema

Esta sección cubre el desarrollo mismo del código. Abarca algunos principios generales (por ejemplo, gestión del código fuente), así como la manera de evitar algunos errores o problemas comunes que podemos encontrar.

Dotación de seguridad al sistema:

Esta sección incluye la implementación y la gestión del código que ha producido. Esto podría no ser pertinente según cada contrato en particular, pero usted debe garantizar que su aplicación o su sistema puede funcionar dentro de estos parámetros.

Además, hemos incluido varios apéndices con especificaciones de BT para el Control de Acceso y Criptografía. Es importante garantizar que su aplicación cumpla con estos controles.

1.3. Otros materiales y consejos

Según los enlaces en el documento.

2. Diseño del sistema

Ref.	Control	Razón
1	Los datos deben categorizarse según el Estándar de Clasificación de Datos, tal como está determinado en su organización. Incluye información de diseño y desarrollo (por ejemplo, diseños, problemas, requerimientos) así como datos sobre la aplicación. Los datos deben mantenerse de acuerdo con la política de retención de datos.	Si comprende el tipo de datos que contendrá su sistema, usted podrá instaurar las protecciones apropiadas. Es importante que la información de diseño y desarrollo esté segura (en particular las cuestiones sensibles), ya que el acceso no autorizado podría permitirle a alguien atacar su aplicación.
2	Deben identificarse todos los requerimientos legales y regulatorios.	Cuando se escriben aplicaciones que manejan algunos tipos de datos (por ejemplo, información de tarjetas de crédito), existen obligaciones legales y regulatorias específicas que van a afectar el tratamiento de esos datos.
3	El proceso de diseño debe considerar si existen requerimientos de seguridad aplicables a quienes trabajan en el proyecto. (por ejemplo, solo el Reino Unido o si se necesitan autorizaciones de seguridad)	Los proyectos que manejan información marcada por el gobierno o sensible podrían exigir que se recurra a cierto tipo de personal.
4	Use técnicas de análisis de riesgos para identificar y cuantificar riesgos detallados de seguridad. Documente los riesgos y las mitigaciones propuestas.	Cada aplicación tendrá un conjunto diferente de riesgos asociados (por ejemplo, en qué entorno se ubica, qué nivel de datos maneja). Es importante garantizar que los haya tenido en cuenta y haya determinado cómo protegerse de ellos. El análisis de los riesgos podría venir determinado por el entorno. Existen muchas

		herramientas que pueden ayudarlo. El OWASP (Open Web Application Security Project, proyecto abierto de seguridad de aplicaciones web) preparó una guía para identificar y evaluar los riesgos, con una sección útil sobre cómo calificarlos. Asimismo, Microsoft desarrolló una herramienta para detectar amenazas a su aplicación, y otra para comprender su superficie expuesta a ataques antes y después de que se instalen las aplicaciones nuevas. Si tiene preguntas o necesita asistencia con la metodología, póngase en contacto con el titular de la norma. Nota: La técnica que deba utilizar podría depender del proyecto. Por ejemplo, IS1 para trabajo acreditado por HMG.
5	Defina y documente un esquema de revisión de los riesgos.	Eso le asegurará que los tipos de riesgos siguen vigentes después de introducir cambios a la aplicación o el entorno. Se pueden sumar nuevos riesgos y plantear acciones de mitigación.
6	Documente la Arquitectura de Seguridad para atender cada riesgo identificado. Proporcione guías de seguridad para diseños o implementaciones de nivel bajo bibliotecas confiables cifrados aceptables algoritmos hash aceptables longitudes de clave mínimas procedimientos de gestión de claves encriptación o segregación de almacenamiento	Esto va a garantizarle que está aplicando las acciones de mitigación apropiadas para los riesgos que identificó, y de manera coherente. También lo ayudará en el diseño de las pruebas de aceptación.

	transmisión de o acceso a credenciales protocolos inaceptables	
7	Defina y documente un ciclo de desarrollo.	Tener definido el ciclo de desarrollo significa que cuenta con un enfoque repetible y uniforme para pasar de requerimientos o problemas a una resolución en producción. Sin eso, podría ser difícil e insumir mucho tiempo resolver las cuestiones de seguridad. Microsoft tiene un modelo que puede utilizarse como punto de partida.
8	Documente e implemente los siguientes procedimientos de diseño: Control de acceso (quiénes pueden acceder a qué porciones de la aplicación y cómo se aplica, mantiene y revoca ese acceso): si no se gestiona, es probable que las personas terminen teniendo un tipo de acceso inapropiado (por ejemplo, usuarios que pueden realizar tareas de admin. o que retienen el acceso después de cambiar de trabajo). Sincronización de los relojes: si los relojes no son precisos, es muy difícil usar los archivos de registro para identificar problemas, en especial donde necesita correlacionar con las actividades en otros sistemas. Transferencias de datos: los datos en tránsito deben contar con protecciones adecuadas para su clasificación. Formas de usar herramientas y	La seguridad de la aplicación depende mucho de cómo se la gestione durante todo su ciclo de vida. Al definir e implantar los procesos sugeridos, usted puede asegurar que el trabajo realizado en la etapa de diseño e implementación no se revierta durante la ejecución. Muchos de ellos solo una cuestión de sentido común y se seguirán de manera natural. Sin embargo, es importante explicitar los procesos para garantizar que se respeten durante todo el ciclo de vida de la aplicación (por ejemplo, si cambia el equipo de soporte).

	técnicas para atacar el sistema: esto le permitirá adelantarse a los intentos de ataque e implementar acciones de mitigación desde el primer día. Recuperación ante desastres / reserva: esto debe estar definido con claridad y probado, de manera que el sistema pueda reaccionar ante eventos inesperados. Documente e implemente los siguientes procedimientos de desarrollo: Incluya herramientas de prueba en el proceso de integración continua: un entorno de desarrollo automatizado permite que cada desarrollo sea probado automáticamente en busca de problemas comunes antes de lanzar la aplicación. Use herramientas de análisis de seguridad estático y dinámico como parte del proceso de desarrollo: esto ayuda a encontrar vulnerabilidades temprano en el proceso de desarrollo. Etiquete los problemas de seguridad en su repositorio fuente y en el rastreador de problemas: si etiqueta los defectos de seguridad, es posible evaluar la eficacia de las acciones de mitigación y las soluciones que se evalúan. También pueden devolverse al análisis de riesgos para el proyecto, y los problemas comunes se pueden compartir con otros desarrolladores para mejorar el conocimiento general. Use procesos de revisión de pares: la revisión del código por parte de sus pares	

	permite identificar problemas que no se detectan automáticamente y evitan las vulnerabilidades causadas por los errores de codificación (por ejemplo, errores lógicos). Instale desde un entorno limpio de desarrollo: esto reduce el riesgo de que el resultado se contamine con malware, bibliotecas antiguas o no deseadas, o componentes que podrían producir fallas de compromiso de la seguridad. Comparta las mejores prácticas y la experiencia: con esto, la comunidad puede evitar los problemas comunes que podrían llevar a vulnerabilidades. Documente e implemente los siguientes procedimientos para el ciclo de gestión de un desarrollo: Aplicación oportuna de parches de seguridad que incluyan los componentes de terceros: esto resolverá los problemas de seguridad que podrían llevar a comprometer la aplicación y sus datos. Mantenimiento de hardware y software: es importante planificar de qué manera va a mantener el soporte al hardware y el software para resolver problemas y recibir actualizaciones de seguridad. Cierre o inicio de la aplicación: el cierre incompleto de la aplicación podría llevar a conductas inesperadas o a que los artefactos no queden bien limpios. Aplicación y gestión de trabajos: debe	

	definirse la manera en que la aplicación gestiona los trabajos, en especial con cargas inusualmente pesadas. Monitoreo y alarmas: debe definirse e implementarse una manera de monitorear y responder a los registros y otros resultados. Archivo y limpieza de datos: la gestión de datos debe existir para que se eliminen o archiven los datos que ya no se necesitan. Es posible que esto sea exigencia de las leyes de protección de datos. Migración: permite la prestación continua del servicio después de una actualización; por ejemplo, para resolver problemas de seguridad. Control de cambios: es importante que los cambios a la aplicación estén controlados para poder comprender si son legítimos, y saber qué causó los problemas cuando surgen.	
9	La arquitectura de seguridad debe revisarse y actualizarse con cada ciclo o iteración. Los cambios a la aplicación planificados deben revisarse para garantizar que no comprometen los requerimientos de seguridad.	Esto es para garantizar que tiene incorporados los cambios más recientes a los requerimientos del proyectos o la implementación.
10	Use kits de herramientas estándar de la industria que tengan mantenimiento continuo, en lugar de diseñar los propios.	Los kits de herramientas de uso más común están bien probados, son menos propensos a contener fallas severas y serán emparchados si se encuentran problemas. ¡También le ahorrarán tiempo! Por ejemplo: OpenSSL, OpenSSH, Solaris KCF,

		la API de criptografía de Windows, Active Directory, OpenLDAP.
11	Solo use los componentes y servicios que necesite para la entrega de las funcionalidades del producto. No exponga las interfaces en otro lugar que no sea donde debe hacerlo.	Al minimizar las funcionalidades innecesarias, reduce la exposición de la aplicación y el impacto de los defectos que se encuentren.
12	Solo use componentes o bibliotecas de terceros que tengan mantenimiento y soporte constantes. Para el software con soporte comercial, esto significa que debe contar con un contrato de servicios de soporte con el proveedor durante el ciclo de vida de la aplicación.	Esto significará que, si surgen problemas de seguridad, usted podrá actualizar los componentes que está usando su aplicación. Esto puede resultar difícil de valorar para el software de código abierto, ya que el mantenimiento depende de la participación de la comunidad. Ante la duda, solo use el componente si usted puede asumir las tareas de mantenimiento.
13	Las funcionalidades de cumplimiento de los requerimientos de seguridad deben estar definidas en un conjunto mínimo de ubicaciones bien determinadas y documentadas en el código. Documente la ubicación de las funcionalidades de cumplimiento de los requerimientos de seguridad.	El impacto de los cambios puede evaluarse con mayor facilidad y es más simple reforzar los puntos débiles de la seguridad. Al mantener las funcionalidades de seguridad en una cantidad pequeña de ubicaciones, son más sencillas de comprender y probar que si están dispersas en varios lugares de un gran código base.
14	Garantice la utilización y el mantenimiento de versiones vigentes y probadas de los componentes de terceros.	Las versiones de software anteriores pueden contener defectos o puntos débiles en la seguridad, que podrían llevar a errores o acceso no autorizado a la aplicación.
15	El acceso al sistema debe seguir los controles de la sección Control y Gestión de Acceso.	Eso va a garantizar que solamente las personas autorizadas tendrán acceso a los datos de la aplicación.
16	La implementación predeterminada del sistema	Eso reduce el riesgo de que este tipo de

	debe producir un número mínimo de privilegios.	implementaciones genere accesos innecesarios y de alto privilegio. Si esto fuera necesario, debe estar explícitamente permitido para evitar que se olvide durante la implementación.
17	El software debe ejecutarse solo con los privilegios que se necesitan para realizar la tarea requerida. En los casos en que se necesiten mayores privilegios, se deben obtener de maneras reconocidas y como está documentado en la arquitectura de seguridad, y retirados luego lo antes posible.	Gracias a que el software se ejecuta con privilegios mínimos, el impacto de los defectos o puntos débiles en la seguridad puede reducirse, ya que si se los explota, solo se logrará un acceso mínimo.
18	Las tareas que exigen privilegios elevados constantemente deben estar separadas del software de la aplicación principal y sujetas a un escrutinio máximo durante la revisión de sus pares. Preste especial atención a las tareas que requieren software no confiable.	Si lo hace, evitará que los defectos o puntos débiles en la seguridad de la aplicación principal permitan el acceso de alto privilegio. Al reducir la cantidad de código que se ejecuta con privilegios elevados, se reduce el riesgo de que un defecto o un punto débil en la seguridad habilite el acceso de alto privilegio.
19	Cree un plan de prueba de su aplicación que incluya pruebas sobre los controles de seguridad o las acciones de mitigación que haya implementado.	
20	Defina límites de seguridad que le permitan controlar: los derechos de acceso a los datos la autenticidad e integridad de los mensajes la validación de los datos de entrada y salida	Eso hace que los controles se apliquen en el lugar correcto para proteger los datos de la aplicación. No implementar controles en el lugar correcto podría llevar a modificación de los datos o a acceso no autorizado a ellos.
21	Los sistemas deben garantizar que se mantiene y se preserva la integridad de los datos.	Un control deficiente de la integridad de los datos podría llevar a fraude, incumplimiento de los requerimientos regulatorios (ley Sarbanes-Oxley) o las normas de la industria (PCI DSS).

3. Entrega del sistema

Ref.	Política	Razón
22	Guarde el código fuente en un lugar con acceso restringido de manera que solo las personas autorizadas puedan realizar cambios.	Eso reducirá el riesgo de que se realicen cambios no autorizados o maliciosos al código que pudieran comprometer la seguridad de la aplicación. La manera más simple de materializarlo es utilizando un sistema de control de revisiones, tales como Git o SVN, que tenga el acceso limitado a un grupo de individuos designados.
23	Asegúrese de que todos los cambios al código estén relacionados con un individuo designado. Cada cambio debe ser auditable de manera que se pueda identificar quién hizo qué, cuándo y cómo.	Eso significa que cualquier cambio al código (por ejemplo, que introduzca funcionalidades no deseadas) puede relacionarse con un individuo. La manera más simple de materializarlo es utilizando un sistema de control de revisiones, tales como Git o SVN, que tenga el acceso limitado a un grupo de individuos designados.
24	Siga las guías de mejores prácticas que sean pertinentes a su tipo de aplicación o entorno de desarrollo. Asegúrese de saber cuáles son los problemas comunes de la plataforma o el lenguaje que está usando.	Por lo general, las publican los proveedores de herramientas u otros grupos de terceros. Comprenderlas y seguirlas lo ayudarán a evitar los problemas comunes. Esta es una selección de las guías más usadas: Aplicaciones web: la lista de las 10 vulnerabilidades más importantes del OWASP detalla las vulnerabilidades comunes de las aplicaciones web. Aplicaciones Android: Estándar de codificación segura para Android de CERT: Windows / teléfonos Windows: iOS / Mac SO X: Guía de codificación segura de

		Apple: Linux: Para más información o asesoramiento, póngase en contacto con el titular del estándar.
25	Use un estilo de codificación documentado y uniforme.	Si los estilos de codificación no son uniformes en el código base de la aplicación, se puede dificultar comprender la lógica de la aplicación y detectar los problemas que podrían llevar a conductas no deseadas. La mayoría de los proveedores ofrecen una guía de estilo para su entorno o lenguaje. En ausencia de todo otro requerimiento, se recomienda utilizar esto. Algunas guías de estilo sugeridas: Java: Estándar de codificación de SEI CERT Oracle para Java C/C++: guías de codificación segura de CERT o las guías MISRA-C C#: Python: PEP8 php: guía de estilo de CodeIgniter
26	Use cadenas de herramientas confiables y con soporte.	Se entiende por "herramientas con soporte" aquellas que están disponibles y con mantenimiento activo (ya sea por un proveedor único, por ejemplo Microsoft Visual Studio, o por una comunidad, por ejemplo GNU Compilers Collection). Herramientas confiables son aquellas provistas (y firmadas) por un tercero confiable (por ejemplo, Microsoft, Ubuntu) o están disponibles en forma masiva y son verificables con hashes conocidos (por ejemplo, las descargas de fuentes GCC). Al usar herramientas con soporte, recibirá las actualizaciones para corregir defectos y otros problemas de seguridad. Cuando verifica la confiabilidad de las

		herramientas que usa, está asegurando que no van a introducir funcionalidades no deseadas.
27	Asegure el mantenimiento de las cadenas de herramientas.	Asegúrese de que las herramientas cuentan con todas las actualizaciones de seguridad antes del uso. Eso es para resolver los problemas que pudiera tener la cadena de herramientas y que podrían introducir conductas no deseadas en los binarios de salida.
28	Use funcionalidades de cadena de herramientas y lenguaje que lo ayuden a identificar o mitigar los problemas simples.	Al permitir las advertencias de tiempo de ejecución o del compilador, puede detectar con anticipación los problemas que podrían llevar a puntos débiles o defectos de seguridad (por ejemplo, fstack-protector de GCC le avisará si aparecen condiciones potenciales de desbordamiento del búfer; o /GS o /SafeSEH en VisualC++ dificultan explotar las saturaciones del búfer de pila).
29	Use las funcionalidades del sistema operativo que mitigan la acción de los vectores de ataques comunes.	Esto depende del sistema operativo, pero usted debe habilitar funciones de seguridad que sumen protección a su aplicación (por ejemplo, ASLR o DEP). Existe soporte para los sistemas operativos basados en Microsoft. Para Linux, ASLR y DEP están habilitados en forma predeterminada (para kernels posteriores a 2.6.12). SELinux es una manera de hacer cumplir las políticas de control de acceso y puede utilizarse para reducir el impacto de un ataque. Redhat tiene una buena guía sobre desarrollo de políticas.
30	Logre la sanitización de todos los datos de entrada según la arquitectura antes de su procesamiento. Convierta los datos de entrada en forma canónica antes de la corrección.	Si se usan datos de entrada de los usuarios como base para los comandos o las operaciones de la base de datos, entonces esos podrían usarse para generar una operación no deseada, por ejemplo, inyección de código SQL. Los datos de entrada pueden tomar formas

		inusuales (por ejemplo, objetos Java serializados) así que es importante que usted considere todos los casos en los que los usuarios e intermediarios podrían modificar o inyectar contenido. Los datos de entrada que no se conviertan pueden exceptuar la sanitización con grupos de caracteres que el mismo proceso no puede manejar porque no fue diseñado para ellos.
31	Logre la sanitización de todos los datos de salida a otros sistemas según la arquitectura de seguridad.	Cuando se usan los datos de entrada del usuario como datos de salida (ya sea a otra aplicación o de regreso al usuario), los datos de entrada pueden ser manipulados para provocar conductas no deseadas en lo que sea que reciba los datos de salida (por ejemplo, scripts entre sitios).
32	Implante criptografía con los controles que detallamos en la sección sobre criptografía.	La criptografía puede resultar compleja y difícil de comprender bien. Si se respetan los controles apropiados, usted podrá estar seguro de que el nivel de criptografía que usa es el adecuado para la información que está protegiendo.
33	Implemente mecanismos para evitar operaciones con memoria que no sean seguras. Ver OWASP	Algunas operaciones con memoria pueden permitir que un usuario sin los privilegios correspondientes pueda manipular los contenidos de memoria. En algunos contextos, eso puede llevar a la manipulación del flujo de programas (por ejemplo, exceptuando los controles) o la ejecución de código arbitrario.
34	Use herramientas que impidan alteraciones donde así lo indique la revisión de riesgos.	Permitir la modificación del firmware o el código de la aplicación puede ignorar las funciones de seguridad. Eso puede aplicarse a los dispositivos móviles, donde usted desea detener la ejecución de su aplicación en dispositivos con *rooting* o *jailbreak*. Si hace esto, será más difícil para los usuarios realizar análisis de tiempo de ejecución o depuraciones para aplicar ingeniería inversa a las funcionalidades de la aplicación.

35	Use técnicas que impidan la ingeniería inversa donde así lo indique la revisión de riesgos.	La ingeniería inversa podría revelar las funciones de seguridad. Sin embargo, debe recordar que la mayoría de las técnicas que buscan evitar la ingeniería reversa puede incrementar el tiempo necesario para ejecutar el proceso en lugar de evitarlo por completo, por lo cual no debe ser la única forma de defensa de la seguridad.
36	Evite tener archivos temporarios. No use la función tmpnam() ni otras similares para generar los nombres de archivos.	Usar archivos temporarios puede provocar que los datos se distribuyan en forma masiva en un disco que puede ser recuperable si se gana acceso no autorizado al sistema. La función tmpnam() debe evitarse ya que entre que se genera el nombre y se abre el archivo, es posible que otro proceso haya creado un archivo con el mismo nombre usando tmpnam. Esto podría desencadenar que el otro proceso afectara la integridad de los datos almacenados o leyera los datos a los que no debería acceder.
37	No use contraseñas predefinidas (codificadas de forma rígida).	Las contraseñas que vienen predefinidas en la aplicación, por lo general, se conocen y comparten entre los usuarios. Esto puede brindar acceso a usuarios no autorizados o dificultar saber quién accedió, ya que no están asignadas a un solo usuario. Tampoco es posible actualizar y gestionar una contraseña codificada en forma rígida de acuerdo con las especificaciones de BT para la gestión de cuentas.
38	Implemente mecanismos que eviten el acceso no autorizado a los datos de la memoria.	Según su plataforma y la revisión de riesgos, tal vez debe realizar ciertas acciones para impedir que alguien con acceso a la máquina que está ejecutando la aplicación recupere datos sensibles de la memoria. Esto puede suceder de varias maneras: • sobrescribiendo porciones de la memoria que ya no se necesitan

		• asignando memorias • usando construcciones de marco diseñadas para proteger la información; por ejemplo, la clase SecureString en .Net
39	Los reportes de error deben contener la información suficiente para identificar la causa de los problemas. No obstante, cuando se muestran errores a los usuarios, no se debe revelar información interna de las aplicaciones. Por ejemplo, no use tipos de excepciones genéricas para condiciones de error específicas.	Si los reportes de error están incompletos, se pueden frustrar investigaciones de problemas y ocultar actividades no autorizadas. Un ataque siempre va a generar errores en sus primeras etapas cuando se trata de acceder, y, si se registran apropiadamente, pueden ayudar a la investigación de un problema. Es difícil comprender lo que causó el error si se registra solo una excepción genérica. Si la condición de error fue causada por un ataque, es más complicado deducir qué operaciones se llevaron a cabo.
40	Todo el software debe someterse a pruebas antes de su migración a un entorno en vivo. Represente el plan de pruebas que creó como parte del diseño de la aplicación.	Un software sin probar podría: Introducir riesgos inaceptables para la seguridad No funcionar según los requerimientos, en especial los de seguridad detallados en el Registro de Seguridad Tener efectos adversos en las operaciones existentes
41	Todos los controles de seguridad deben probarse específicamente para garantizar que no pueden evadirse.	Las vulnerabilidades de los controles de seguridad que no pueden identificarse podrían explotarse en el entorno en vivo.
42	El propietario de los datos de las pruebas debe eliminarlos tras un periodo determinado.	La divulgación de los datos de las pruebas podría brindar información estratégica al sistema donante.

4. Dotación de seguridad al sistema

Ref.	Control	Razón
43	Todas las vulnerabilidades de la seguridad identificadas en el código o cualquiera de los componentes que usa el código serán categorizados usando los criterios de evaluación de vulnerabilidades de BT. Las vulnerabilidades se deben resolver de acuerdo con el tiempo asignado para cada categoría. Esto debe ajustarse según la postura de riesgo de la plataforma en la que está la aplicación.	Las vulnerabilidades no resueltas pueden usarse como parte de un ataque a la aplicación o el sistema.
44	Rastree y etiquete los cambios realizados para resolver los problemas de seguridad explícitamente en el control de cambios.	Al hacer el seguimiento de la resolución de los problemas de seguridad, podrá auditar rápidamente su aplicación o entorno y probar que han sido aplicados.
45	Instale las versiones de la aplicación que hayan sido diseñadas desde la fuente usando integración continua.	La implementación desde un entorno limpio garantiza que la aplicación comienza su vida operativa desde un código conocido, y que los problemas pueden rastrearse hasta el código fuente que los introdujo.
46	Asegúrese de que se estén usando la versión más reciente de la aplicación y los componentes de terceros sobre los que se sustenta.	Si se actualizan los componentes de terceros, disminuye el riesgo de que se explote un problema de seguridad de un componente.
47	Aplique parches al entorno que aloja la aplicación de acuerdo con la política	Es importante aplicar parches de seguridad regularmente porque las versiones de software más antiguas se

	correspondiente.	convierten en objetivo de ataque. Un problema de seguridad en el entorno de alojamiento podría poner en riesgo los datos de la aplicación.
48	Las siguientes áreas de proceso deben definirse y documentarse: Aplicación de parches de seguridad Control de acceso Inicio y cierre Gestión de trabajos, aplicación y sincronización del reloj Transferencias de datos Monitoreo y alarmas Gestión de problemas y derivaciones Respaldo y recuperación Archivo Migración Control de cambios Recuperación ante desastres y reserva Mantenimiento de hardware & software Registro, revisión y acción sobre eventos inusuales	Si no se documentan los procedimientos operativos, es común que se omitan procesos importantes, o que el equipo quede sin saber cómo llevarlos a cabo.

5. Apéndices:

5.1.Control de acceso

Ref.	Política	Razón
49	Defina los derechos de acceso al sistema para los usuarios y otros sistemas que interactúan con él. Los derechos de acceso deben basarse en los siguientes elementos: Operaciones realizadas por el sistema Interacción entre sistemas Acceso del Usuario a la política sobre sistemas e información Los usuarios deben tener los privilegios mínimos que les permitan hacer sus trabajos.	Si los roles no están bien definidos, podrían aparecer operaciones accidentales o malintencionadas sobre el sistema.
50	Deben definirse y gestionarse las cuentas con Privilegios Compartidos*.	Sin controles formales, es difícil hacer el seguimiento de los responsables de los cambios que impactan en la seguridad.
51	Diseñe los procesos de la aplicación de manera que se ejecuten con los derechos de acceso mínimos requeridos para la operación correcta. Operaciones realizadas por el sistema Interacción entre sistemas No use cuentas privilegiadas para los procesos de la aplicación.	Los procesos de la aplicación que se ejecutan con privilegios excesivos podrían ser utilizados para ganar acceso a los privilegios de los datos o el sistema.

	Documente todos los accesos privilegiados. La derivación de privilegios debe realizarse solo con API conocidas. La derivación de privilegios debe ser solo por el tiempo mínimo requerido.	
52	Las sesiones de usuario deben terminarse a los 30 minutos de inactividad como máximo. Cuando se llega al límite del tiempo de espera, debe desaparecer toda la información de la pantalla.	El tiempo de espera limita la oportunidad de acceder sin autorización si el sistema queda desatendido.
53	Una sesión de usuario no debe exceder las 12 horas.	Para garantizar que los usuarios inician sesión y reautentican la identidad al menos cada 12 horas.
54	En todos los puertos de gestión debe haber controles de acceso basado en roles y privilegios, ya sean esos locales (terminal de consola o independiente, o servidor de la terminal) o remoto (vía EMS). Deben aplicarse listas de control de acceso (ACL) a todas las interfaces de gestión (IP), con lo cual se restringe el puerto y la dirección de la fuente, y el protocolo que se invoca; en particular, la posibilidad de limitar el acceso a ICMP, HTTP, SNMP, FTP, TFTP, SSH, Telnet y protocolos avanzados de enrutamiento como OSPF.	Las interfaces de gestión expuestas pueden explotarse para obtener acceso no autorizado.

55	Deben usarse solamente protocolos de gestión autenticados con métodos seguros; por ejemplo, SNMP v3 (AuthnNoPriv como mínimo) SSHv2 - ver la sección sobre criptografía HTTPS - ver la sección sobre criptografía	Los protocolos de gestión que no son seguros pueden explotarse para obtener acceso no autorizado.

5.2 Criptografía

Ref.	Política	Razón
56	Deben usarse las librerías criptográficas existentes.	Las librerías criptográficas deben actualizarse regularmente. Además de actualizar paquetes de software en sintonía con lo estipulado por el proveedor, los paquetes deben revisarse y actualizarse con regularidad.
57	En el proceso de encriptación, se podrán usar solamente suites de cifrado aprobadas y estándar en la industria. Por ejemplo, para TLS SSLv2. Tome nota de la advertencia anterior. Ante la duda, consulte a ITSAC.	Los cifrados no aprobados podrían introducir vulnerabilidades.
58	Debe utilizarse la versión más reciente de TLS en las implementaciones nuevas. SSL V1, 2 y 3 ya no podrán utilizarse.	Las versiones anteriores, hasta TLS1.0 incluida, ya no se consideran seguras.

59	Se deberá habilitar la confidencialidad directa total.	Los algoritmos de confidencialidad directa total evitan que se descifren los mensajes capturados incluso si la clave de autenticación privada se viera comprometida en el futuro.
60	No deben utilizarse certificados autofirmados.	Los certificados autofirmados privan de la ventaja de la autenticación del punto de conexión y reducen significativamente la posibilidad de que un individuo detecte un ataque de intermediario.
61	Las contraseñas deben protegerse usando una función matemática unidireccional irreversible (algoritmo hash) con un factor de aleatoriedad único (sal) por cada contraseña.	Los archivos de contraseñas guardados pueden extraerse y, por lo tanto, deben protegerse todas las entradas para evitar la recuperación de contraseñas no cifradas.
62	Las contraseñas protegidas deben almacenarse separadas de los archivos de configuración del sistema; se debe tener algún sistema de control de acceso de manera que solo los usuarios con los privilegios correspondientes puedan leer o copiar el contenido.	Deberá ser imposible que se puedan recuperar contraseñas protegidas por mecanismos como *directory traversal*, SNMPwalk, el volcado de la configuración etc., ya que podrían intentar causar grietas offline.

5.2.1 Controles criptográficos para la web

Ref.	Política	Razón
63	Las cookies que almacenan o se usan para acceder a información Confidencial	Las cookies pueden robarse por distintos medios, como ser XSS y *sniffing*.

	o de nivel superior deben transportarse en forma segura.	
64	No debe mezclarse contenido seguro y no seguro en la misma página.	El contenido no seguro podría robar información segura del contenido.
65	Debe usarse TLS para todas las páginas de inicio de sesión y todas las páginas autenticadas. El acceso a la página de inicio de sesión y todas las páginas autenticadas subsiguientes debe ser exclusivamente por TLS. El acceso a la página de inicio de sesión y todas las páginas autenticadas subsiguientes debe ser exclusivamente por TLS.	Si no se usa TLS para la página de inicio de sesión, podría ser que un atacante modifique la acción del formulario de inicio de sesión, lo cual provocaría que las credenciales del usuario se publiquen en un lugar arbitrario. Si no se usa TLS para páginas autenticadas después del inicio de sesión, es posible que un atacante pudiera ver la identificación de la sesión sin cifrar y comprometer la sesión autenticada del usuario. Si no se usa TLS, podría haber un ataque de intermediario.
66	Se debe indicar a los clientes que no copien los datos sensibles en cache.	El protocolo TLS brinda confidencialidad solo para los datos en tránsito, pero no ayuda en caso de filtraciones de datos del lado del cliente.
67	Use técnicas estándar de firma digital aceptados nacional e internacionalmente.	El uso de técnicas de firma digital no estándar podría llevar a que se confiara en la firma incorrecta; la normativa nacional e internacional ejerce los controles sobre lo digital reuniendo los controles de importación, exportación y encriptación local en todo el mundo.
68	No deben usarse certificados comodines en ninguna circunstancia.	Los certificados comodines son un objetivo más atractivo. Pueden certificar a un servidor no deseado y volver vulnerable un domino de encriptación. Si se ve comprometido un servicio o un subdominio, todos los subdominios podrían verse comprometidos.

		Si se deben revocar los certificados comodines, todos los subdominios necesitarán certificado nuevo.

5.2.2 Gestión de claves generales

Ref.	Política	Razón
69	Las claves criptográficas para todas las implementaciones deben cumplir con los estándares mínimos o excederlos.	Las claves cortas o débiles pueden verse comprometidas con facilidad durante el ciclo de vida de los datos.
70	Las Claves Simétricas y Asimétricas deben generarse usando herramientas y bibliotecas aprobadas.	Las herramientas y bibliotecas no aprobadas podrían generar claves débiles.
71	Las contraseñas que controlan el uso de claves criptográficas deben brindar protección que sea equivalente a la misma clave.	Las contraseñas débiles le quitan fuerza a la clave criptográfica.
72	Un gerente senior de BT debe ser responsable de la seguridad del material clave.	Deberá tener la antigüedad, capacidad y confiabilidad proporcional a la seguridad de los datos que protejan las claves.
73	El Gerente de Claves Senior debe garantizar que se asignen las responsabilidades en la columna de detalle y que esas se lleven a cabo por personal idóneo y capacitado:	Responsabilidades Política de gestión de claves Generación o adquisición de claves Diseño de distribución de claves, periodos de vigencia, revocación y estructura de responsabilidad Creación del procedimiento de gestión de claves

		Operación de la gestión de claves Protección de las claves privadas y el material relacionado Procedimientos de emergencia, tal como la revocación Auditoría de las operaciones clave Recuperación de claves Destrucción de claves

Control de documentación

Guía de Estándares de la Industria sobre Codificación Segura para Terceros

Redactado por: Área de Seguridad de BT

Versión 1, publicado en octubre de 2016