1.- Protege tu marca. Tus clientes Confian.

La publicación especial de Harvard Business Review, "Ideas Breakthrough para 2008," b enumerado "Ciberdelincuencia Servicio Economía" como uno de los 20 principales transformaciones del mundo de los negocios. De Scott Berinato, editor ejecutivo de la revista CSO, que contribuyó a la publicación, afirma que la nueva generación de hackers apenas no causar interrupciones en un negocio, pero amenazan con socavar la confianza comercial y la confianza del cliente. Su conclusión es notable; en el caso de los delitos cibernéticos, las víctimas buscarán

alguien que se hace responsable, y no serán los hackers, pero las marcas que las víctimas confiaron para protegerlos.

La seguridad es un desafío que nunca termina. A medida que los cibercriminales evolucionan, así es necesario que los defensores. Son los defensores y sus organizaciones que necesitan para mantenerse un paso por delante de los delincuentes o de lo contrario serán considerados responsables de las violaciones de seguridad. Las infracciones que llevan a situaciones críticas, como la divulgación de la información del cliente, denegación de servicio, y las amenazas a la continuidad de las operaciones comerciales

puede tener consecuencias financieras desastrosas. Sin embargo, el costo real de la organización será la pérdida de confianza del cliente y la confianza en la marca de la organización. Tal pérdida puede ser irreparable e imposible de cuantificar en meros términos monetarios. Grabado en la vanguardia de la mente de cualquier SSLP debe ser la necesidad de proteger la marca, los clientes de confianza. Fundamentalmente, el reconocimiento de que la organización tiene la obligación de proteger a los clientes de gran alcance debería motivar a la organización en la creación de software más seguro.

2.- Conozca su negocio y apóyelo con soluciones seguras.

La mayoría de los profesionales de seguridad calificados de acuerdo en que, junto con una sólida formación en tecnología, un conocimiento profundo de la empresa es de suma importancia cuando se trata de la creación de soluciones seguras para ese negocio. Aunque algunos tecnólogos seguridad puristas pueden encontrar difícil de aceptar, no es menos cierto que la seguridad está allí para el negocio y no al revés. Existe seguridad para que el negocio, no sea un impedimento. La respuesta a la pregunta: "¿Por qué fueron los frenos

? inventado "podría ser respondida en dos formas: para evitar que el vehículo de un accidente, o para permitir que el vehículo vaya más rápido. Seguridad es similar; se puede evitar que el negocio de un accidente, o permitir que el negocio vaya más rápido.

Para una SSLP, entender el negocio puede ayudar en la identificación de los requisitos regulatorios y de cumplimiento, riesgo aplicable, arquitecturas que se utilizará, controles técnicos que se incorporen, y los usuarios para ser entrenados o educados. Por ejemplo, una organización de banca por Internet tendrá que hacer frente a los requisitos normativos tales como la privacidad financiera, salvaguardias y normas c pretextos como parte del cumplimiento de la Ley Gramm

Leach Bliley (GLBA). La organización tendrá que abordar el riesgo de divulgación de información de identificación personal y financiera, la necesidad de tener una arquitectura multi-factor de autenticación, encriptación, autenticación, autorización y controles de auditoría, así como la EED para educar a los empleados en la ingeniería social y phishing .

Un comerciante minorista puede necesitar para cumplir con la Payment Card Industry Data Security Standard (PCI DSS), dependiendo de cómo manejan las transacciones con tarjetas de crédito.

3.- Entienda la tecnología del software

No sólo es fundamental para conocer el negocio, pero uno debe tener una sólida formación en tecnología para ser eficaz en la construcción o la compra de software seguro. La falta de comprensión de la tecnología utilizada para construir o comprar software puede llevar a la inseguridad implementaciones de software.

Cuando se trata de construir el software de la casa, un conocimiento profundo de los componentes de infraestructura existentes, como la segregación de la red, los hosts endurecidos, y la infraestructura de clave pública, es necesario asegurarse de que el despliegue del software, primero, ser operativamente funcional y, segundo, no debilitar la seguridad del entorno existente. En otras palabras, la comprensión de la interacción de sus componentes tecnológicos actuales con el software a construir y / o implementar ayudará a determinar el impacto en la seguridad global. Además, la comprensión de la tecnología utilizada en la construcción de software puede ayudar a tomar decisiones que favorezcan la seguridad. A modo de ejemplo, sabiendo que logró código (.Net y Java) tienen menos probabilidad de corrupción de memoria y por lo tanto son menos susceptibles a desbordarse ataques que el código no administrado (C / C ++), ayudaría en la elección más reciente generación de código administrado como parte

de la norma de codificación para desarrollar software.

Con la adquisición de software, es vital reconocer que las reclamaciones de los proveedores con respecto a las características de "seguridad" deben ser examinadas y verificadas de viabilidad aplicación dentro de su organización.

La mera presencia de características de seguridad en un producto no significa necesariamente que el producto es seguro. La correcta y adecuada ejecución de las funciones de seguridad es lo que ayuda a hacer un producto seguro.

4.- Velar por el cumplimiento de la gobernabilidad, reglas y privacidad.

En este día y edad, una industria que no está regulada es más la excepción que la norma a diferencia de hace apenas unos años, cuando la industria que fue regulada fue la excepción.

El aumento de los requisitos reglamentarios y de privacidad impone una pesada carga para las organizaciones. Gobierno, Riesgo y Cumplimiento (GRC) no es sólo una frase industria rumor, sino

una realidad y un medio hacia el cumplimiento de los requisitos reglamentarios y de privacidad. Como SSLP, hay que entender las políticas internas y externas que rigen el negocio, su asignación a los controles de seguridad necesarios, el riesgo residual de post-implementación de controles de seguridad en el software, y los aspectos de hoja perenne de la conformidad a las reglas y los requisitos de privacidad.

5.- Conocer los principios básicos de la seguridad de software

Cuando se trata de conseguir el software, hay algunos principios con los que el SSLP debe estar familiarizado. Estos principios básicos son: protección contra la divulgación (confidencialidad); la protección de la alteración (integridad); la protección de la destrucción (disponibilidad); que está haciendo la solicitud (autenticación); qué derechos y privilegios no el solicitante tiene (autorización); la capacidad de construir evidencia histórica (auditoría); y la gestión de configuración, sesiones y excepciones. El conocimiento de estos principios básicos, y cómo pueden ser implementados en software, es de vital importancia para el SSLP.

Algunos mecanismos que se pueden implementar para apoyar estos principios se describen a continuación. Encryption, por ejemplo, puede servir como un control de la confidencialidad, mientras que hashing puede servir tanto como un control de la confidencialidad y la integridad de control. Encryption utiliza algoritmos para convertir la información humanamente legible (texto plano) en forma críptica no legible (texto cifrado). El descifrado es el proceso de convertir el texto cifrado de nuevo en texto claro.

Cifrado y descifrado requieren una clave que se celebró en secreto para realizar sus peraciones.

Los algoritmos de cifrado puede ser simétrica o asimétrica. Los algoritmos simétricos SE de la misma clave para cifrar y descifrar y si bien esto puede ser rápido, el número de eys necesarios para gestionar la comunicación entre las partes pueden llegar a ser engorroso ery rápidamente y hacer la gestión de claves de un desafío. Los algoritmos asimétricos usan diferentes claves para el cifrado y descifrado, también conocido como un par de claves pública-privada. Los certificados se utilizan para compartir la clave pública y la existencia de una infraestructura de clave pública es necesaria. Esto es mucho más lento que los algoritmos simétricos pero clave distribución y gestión se simplifica.

Hashing por otra parte utiliza funciones matemáticas que son de una sola vía. La diferencia entre el cifrado y de hashing es que con el cifrado, el valor original se puede volver a factorizado; con hashing, esto no es posible. Cualquier valor pasado es un refrito con la misma función y luego se compara su validez. Hashing es más una medida de la integridad para detectar la manipulación de datos o archivos, pero puede ser utilizado para proteger la información sensible como contraseñas cuando se almacena y se utiliza para la autenticación.

Balanceo de carga adecuada y la funcionalidad de carga de monitoreo en el software pueden proteger contra la destrucción o la denegación de servicio, y asegurar la disponibilidad. Contraseña, token o biométrica significa para identificar y validar las credenciales de uno son

ejemplos de mecanismos de autenticación, mientras que el control de acceso basado en roles que el software

cheques es un ejemplo de control de autorización. Software seguro también debe registrar y grabar todas las transacciones comerciales administrativos, críticos y claves para que una pista de auditoría histórica se puede construir cuando sea necesario. La información de configuración debe ser tratada como

información y protegido sensible. Las sesiones deben ser únicos para evitar ataques de repetición o de secuestro, y mensajes de excepción genéricos y lacónicos deben ser explícitamente especificado para evitar la divulgación de demasiada información.

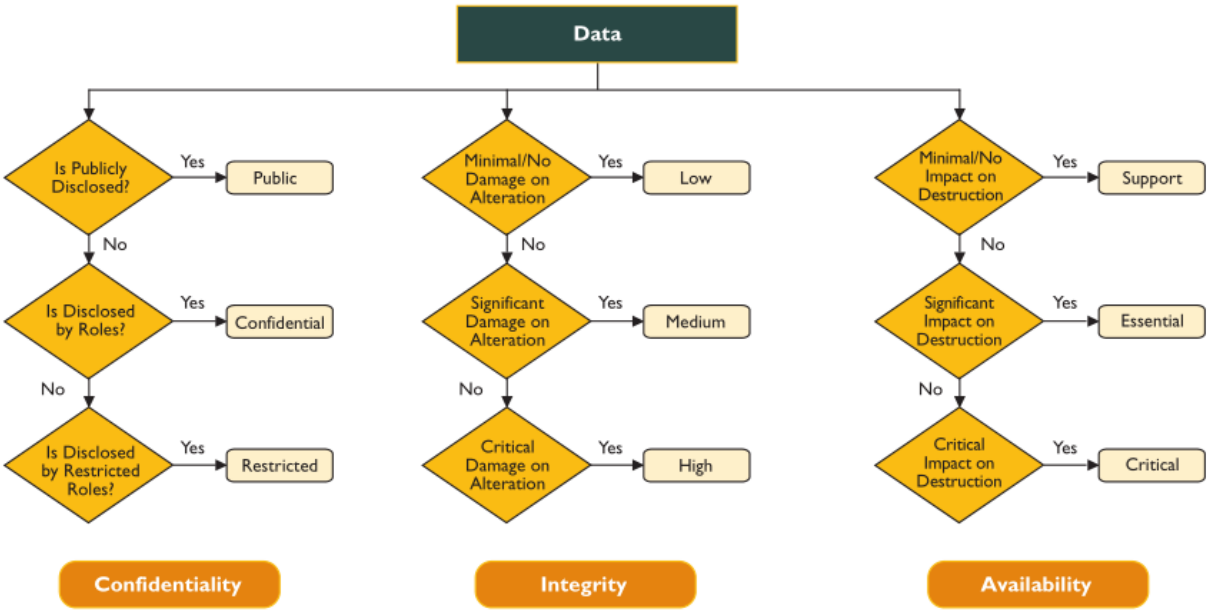
6.- Garantizar la protección de información confidencial.

Además de asegurar que la marca a sus clientes la confianza está protegido, es esencial que cualquier información sensible ser protegido también. La información sensible se refiere a cualquier información en la que la organización pone un valor medible. Implícitamente, se trata de información que no está en el dominio público y se traduciría en pérdidas, daños, o incluso el colapso del negocio se debe perder la información, robado, dañado, o de alguna manera comprometida. La información confidencial puede ser personal, salud, financiero, o cualquier otra información que puede afectar a la competitividad de su organización.

Si bien es fácil identificar la sensibilidad de determinados elementos de datos tales como, registros de salud o información de tarjetas de crédito, otros no puede ser que evidente. La determinación de lo que es sensible y lo que no se puede llevar a cabo mediante la realización de un ejercicio de clasificación de datos, con los accionistas de la empresa implicados en este proceso. Software que, o bien transportes, procesos o almacena información sensible debe construir en los controles de seguridad necesarias para proteger esta información.

Un SSLP debe estar familiarizado con los mecanismos de clasificación y protección de datos contra la divulgación. La clasificación de datos es la decisión consciente de asignar un nivel de sensibilidad a los datos a medida que se está creando, en su versión modificada, almacena, transmite o mejoradas. La clasificación de los datos debe entonces determinar el grado en el que los datos necesita ser controlado / garantizado. La Figura 2 representa un ejemplo de un diagrama de flujo que puede ser utilizado para clasificar la información.

Figure 2. Example of a data classification flowchart



7.- Diseñar software con características seguras.

El artículo de MSDN sobre "Lecciones Aprendidas de cinco años de construcción de software más seguro," d bajo el título "No es sólo el código", pone de relieve que muchas vulnerabilidades de seguridad del software no están codificando cuestiones a todos, pero de diseño cuestiones. Cuando uno se centra exclusivamente en la búsqueda de problemas de seguridad en el código, la persona corre el riesgo de

perdiendo clases enteras de vulnerabilidades. Las cuestiones de seguridad en el diseño y defectos semánticas (los que no son sintáctica o código relacionado), tales como defectos de lógica de negocios, no se pueden detectar en el código y deben ser inspeccionados mediante la realización de modelos de amenazas y casos de abuso de modelado durante la etapa de diseño del SDLC.

El modelado de amenazas es una técnica iterativa estructurado utilizado para identificar las amenazas que el software se está construyendo. Se inicia mediante la identificación de los objetivos de seguridad del software y perfiles de ella.

Se rompe el software en construcciones físicas y lógicas que generan el contexto de software que incluye diagramas de flujo de datos y escenarios de implementación de extremo a extremo, la entrada y la identificación de los puntos de salida, protocolos, componentes, identidades y servicios.

Análisis de la superficie de ataque es un subconjunto de modelado de amenazas y se puede realizar cuando se genera el contexto de software en el que las secciones del software expuesto a los usuarios no sean de confianza se analiza por cuestiones de seguridad. Una vez se genera el contexto software, amenazas y vulnerabilidades pertinentes pueden ser identificados.

Modelado de Amenazas se lleva a cabo durante la etapa de diseño para que los controles de seguridad necesarios (salvaguardias) pueden ser desarrollados durante la fase de desarrollo del software.

Table 1. Adapted from the Saltzer & Schroeder Protection of Information in Computer Systems

Design Principle	What is it?	Example
Economy of mechanism	Keeping the design simple and less complex	Modular code, Shared objects, and Centralized services
Fail-safe defaults	Access denied by default and granted explicitly	Denied transaction
Complete mediation	Checking permission each time subject requests access to objects	Credentials not cached
Open design	Design is not a secret, implementation of safeguard is	Cryptographic algorithms
Separation of privilege	More than one condition is required to complete a task	Split keys, Compartmentalized functions
Least privilege	Rights are minimum and users granted access explicitly	Non-administrative accounts, Need to know
Least common mechanisms	Common mechanisms to more than one user/ process/role is not shared	Role based dynamic libraries and functions
Psychological acceptability	Security protection mechanism unbeknownst to the end user for ease of use and acceptance	Help dialogs, Visually appealing icons

8.- Desarrollar software con características seguras.

Diseñar para la seguridad en el software es inútil a menos que planees para actuar en el diseño e incorporar controles de seguridad necesarias durante la etapa de desarrollo de su ciclo de vida del desarrollo de software. Es imperativo que las funciones de seguridad no se ignoran cuando los artefactos de diseño se convierten en construcciones sintácticas que un compilador o intérprete puedan entender. Escribiendo código seguro no es diferente de la escritura de código que se puede usar, confiable, o escalable.

Los controles, que se ocupan de los principios básicos de la seguridad del software, deben ser validados a través de revisiones de código de seguridad y pruebas de seguridad. Se recomienda llevar a cabo la revisión de código de seguridad, mientras que el código es revisado para la funcionalidad, y antes de que se libera el software para la prueba. Las revisiones de código de seguridad pueden ser manual o automático. Herramientas de revisión Código no son una panacea y pueden ayudar sólo hasta cierto grado para identificar las secciones del código que necesitan

atención. Para mantener la tasa negativa falsos positivos y falsos al mínimo, herramientas suelen perder ciertas vulnerabilidades. Por lo tanto,

revisión de código automatizado nunca debe llevarse a cabo en lugar de humanos (manual) opiniones. Definición del alcance de lo que está siendo revisado, el alcance de la revisión, las normas, requisitos de codificación segura, proceso de revisión de código con funciones y responsabilidades, y mecanismos de aplicación de codificación, debe ser pre-definido para una revisión del código de seguridad para ser eficaz.

Pruebas de seguridad debe complementar las pruebas de funcionalidad ya existente. Como mínimo, las pruebas de vulnerabilidades de software comunes, como el desbordamiento y errores de inyección, y probar el comportamiento del software para formatos de entrada inesperados y aleatorios (pruebas fuzz) deben llevarse a cabo en entornos de prueba que emulan la configuración del entorno de producción. Otras pruebas de estrés y la necesidad de actuación para llevar a cabo, así, ya que se relacionan directamente con el principio de "disponibilidad" de la seguridad. Un SSLP no sólo debe garantizar que el código escrito es segura, pero también saben cómo escribir código seguro, conducción, realizar y orquestar las revisiones de código y pruebas de seguridad.

9.- Implementar software con características seguras.

La mayoría de los equipos de desarrollo de software de acuerdo en que, a menudo, un software que funciona sin problemas en entornos de desarrollo y pruebas se iniciará el hipo que experimentan cuando se despliega / liberado en un entorno de producción más endurecido. Puesto

mortem analiza en la mayoría de estos casos revelan que los entornos de desarrollo y pruebas no simulan adecuadamente el entorno de producción. Fundamentalmente, se trata de un problema de gestión de la configuración. Los cambios realizados en el entorno de producción deben ser adaptados a los entornos de desarrollo y pruebas a través de procesos adecuados de gestión del cambio.

Otra cuestión relacionada es la gestión de versión del software que debe incluir el control de código fuente adecuada y control de versiones. Un fenómeno que se podría referir como "bichos regenerativas" se observa a menudo cuando se trata de procesos de gestión de liberación inapropiadas. Bichos regenerativos son fijos defectos de software que reaparecen en versiones posteriores del software. Esto sucede cuando la codificación de defecto (bug) software se detecta en el entorno de pruebas (tal como pruebas de usuario-aceptación) y la corrección se hace en ese entorno de prueba y ascendido a la producción, sin reequipamiento en el entorno de desarrollo. La última versión del entorno de desarrollo no tiene la solución y el problema vuelve a aparecer en las versiones posteriores del software.

Para implementar software seguro, se recomienda también que el software se someten a evaluación de la vulnerabilidad y pruebas de penetración en un entorno de pre-producción y, si es

necesario, en el entorno de producción con un estricto control. Esto ayudará a hacer evidentes los problemas potenciales que pueden ser descubiertos por un atacante, y le da al equipo una visión de desarrollo de software en las áreas débiles del software.

Despliegue seguro garantiza que el software es funcionalmente operativa y segura en el mismo tiempo. Esto significa que el software se implementa con la defensa en profundidad, y el área de superficie de ataque no se incrementa por la liberación inadecuada, cambiar o gestión de la configuración. También significa que la evaluación desde el punto de vista de un atacante se lleva a cabo antes o inmediatamente después de la implementación. Diseño seguro y el desarrollo de software deben ser aumentados con el despliegue seguro.

10.- Edúcate a ti mismo y a otros sobre como construir software seguro.

La necesidad de diseñar, desarrollar y desplegar software más seguro es evidente a partir de los incidentes de seguridad que prevalecen en la industria, y la gran cantidad de regulaciones y requisitos de privacidad que uno necesita para cumplir. El modus operandi de software de hoy en día es el ciclo de liberación-y-parche infame.

Para combatir este círculo vicioso de la liberación y de parches, hay una necesidad de un cambio - para crear una cultura que los factores de seguridad de software desde el principio de forma predeterminada. Creación de una cultura de la seguridad se puede lograr mediante la educación. El Instituto Nacional de Estándares y Tecnología (NIST) establece que la educación debe provocar un cambio de actitud, que a su vez va a cambiar la cultura de la organización. En esencia, este cambio cultural es la constatación de que la seguridad es fundamental, porque un fallo de seguridad tiene consecuencias potencialmente adversas para todos y, por lo tanto, la seguridad es tarea de todos. f Incluso las medidas de seguridad más caras pueden ser frustrados por la gente, y educar a la gente acerca de la seguridad del software es de suma importancia.

No sólo debe uno ser educados, pero también debe estar dispuesto a compartir sus conocimientos. Como Charles Dickens escribió una vez: "Cambiar engendra el cambio." Cuando uno que se educa a su vez educa a los demás, habrá un efecto compuesto en la creación de la cultura de la seguridad que tanto necesita.