GUIA PARA IMPLEMENTACION DE SEGURIDAD EN APLICACIONES WEB Y DE ESCRITORIO BASADA EN TECNOLOGIAS .NET

DAVID EDUARDO BARRIOS DEL RIO
COD. 0205032

FRANCISCO JAVIER ESGUERRA ESTARITA
COD. 0205010

SUPERVISOR
MOISES QUINTANA
ING. SISTEMAS

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR
PROGRAMA DE INGENIERÍA DE SISTEMAS
FACULTAD DE INGENIERÍAS
CARTAGENA
2006

GUIA PARA IMPLEMENTACION DE SEGURIDAD EN APLICACIONES WEB Y DE ESCRITORIO BASADA EN TECNOLOGIAS .NET

Presentado por:

DAVID E. BARRIOS DEL RIO
Estudiante de Ingeniería de Sistemas
Universidad Tecnológica de Bolívar

FRANCISCO J. ESGUERRA ESTARITA
Estudiante de Ingeniería de Sistemas
Universidad Tecnológica de Bolívar

Revisado por:

MOISES QUINTANA
Supervisor de Monografía

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

OBJETIVO GENERAL

Ilustrar de una manera dinámica e interactiva como implementar la seguridad en aplicaciones Escritorio (GUI) y aplicaciones Web.

OBJETIVOS ESPECIFICOS

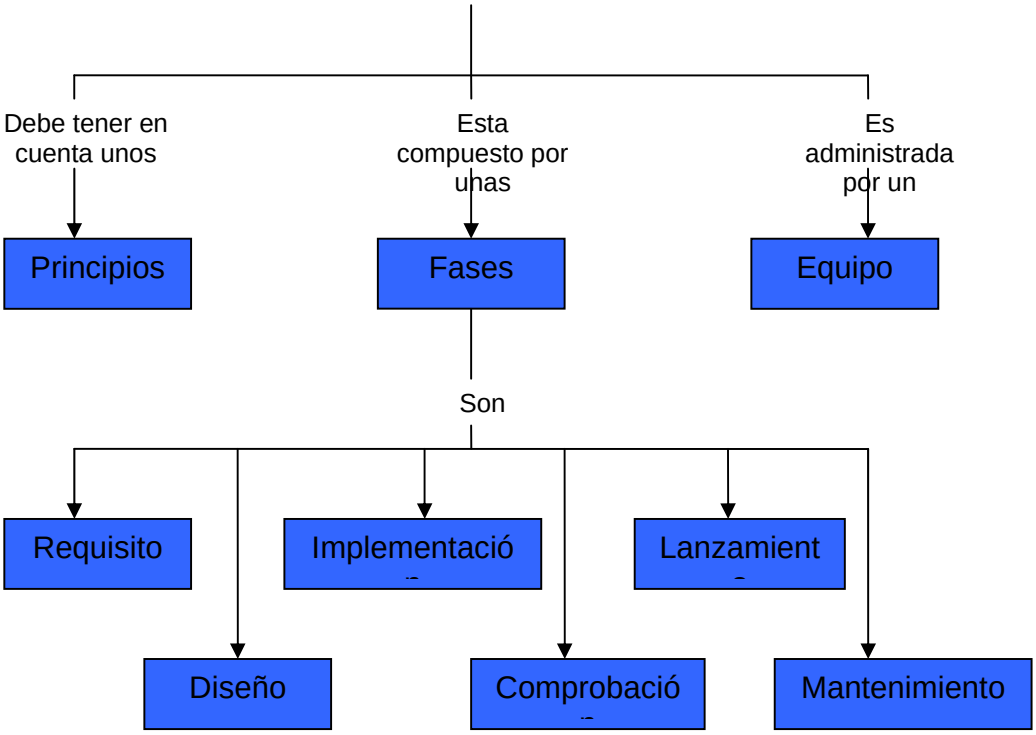
- Se focalizará en entrenar a desarrolladores y administradores de sistemas para pensar en la seguridad en todos los niveles del desarrollo y diseño de software.
- Expondremos el Ciclo de vida del desarrollo de sistemas seguros y confiables.
- Enseñaremos a evitar errores comunes de la programación.
- Expondremos los diferentes tipos de vulnerabilidades que se pueden presentar en las aplicaciones de escritorio y Web.
- Expondremos las fases de las que consta un ataque.
- Expondremos las mejores medidas y herramientas utilizadas en la actualidad para combatir las amenazas que se pueden presentar en cada una de las fases durante la creación de un software.
- Revisaremos los mecanismos y medidas de defensa.
- Presentaremos diversos ataques que los desarrolladores se enfrentan a diario.
- Definiremos todos los componentes que entran en acción cuando una página es llamada.
- Aprenderemos los diferentes tipos de amenazas.
- Mostraremos las técnicas para construir un modelado de amenazas.
- Crearemos mecanismos y medidas de defensas ante ataques.

INTRODUCCION

Todos los desarrolladores y administradores de sistemas deben tener en cuenta las amenazas de seguridad. La seguridad es un requisito básico para desarrolladores y administradores, obligados por las fuerzas del mercado, dada la necesidad de proteger los sistemas y crear y preservar la confiabilidad en la computación. Uno de los retos más importantes a los que se enfrentan los desarrolladores y administradores es crear cada vez mas un software más seguro que requiera menos actualizaciones y una administración de seguridad menos onerosa.

En el sector del software, la clave para cumplir la exigencia actual de una mayor seguridad está en implementar procesos reproducibles que proporcionen de manera confiable una mayor seguridad que se pueda medir. Por tanto, los desarrolladores y administradores deben adoptar un proceso de desarrollo más estricto que se centre, en mayor medida, en la seguridad. Este proceso debe diseñarse para minimizar el número de vulnerabilidades de seguridad presentes en el diseño, la programación y la documentación, así como para detectarlas y eliminarlas cuanto antes en el ciclo de vida de desarrollo. La necesidad de disponer de este proceso es mayor para el software profesional y doméstico que suele procesar información procedente de Internet, de la Intranet o procesar información personal.

CICLO DE VIDA DE DESARROLLO DE SEGURIDAD



CAPITULO I

EL CICLO DE VIDA DE DESARROLLO DE SEGURIDAD

Para lograr un software más seguro, hay que tener en cuenta tres aspectos: *proceso reproducible, conocimientos del desarrollador e indicadores y responsabilidad*. Este capitulo se centra en el proceso que propone el SDL (*SDL, Security Development Lifecycle*) de Trustworthy Computing (computación confiable), un proceso que Microsoft utiliza para desarrollar software que pueda resistir ataques malintencionados), aunque también aborda los conocimientos del desarrollador y ofrece algunos indicadores que muestran el impacto actual del SDL.

La adopción del SDL no debería suponer un costo adicional para el desarrollo de software. Según la experiencia obtenida por las casas desarrolladoras de software más importantes, las ventajas de ofrecer un software más seguro (por ejemplo, menor número de actualizaciones, mayor satisfacción de los clientes) compensan los costos.

El SDL implica cambiar los procesos de una organización de desarrollo de software mediante la integración de medidas que mejoren la seguridad de este mismo. Estas modificaciones no pretenden examinar el proceso de manera total, sino agregar puntos de control y materiales de seguridad bien definidos.

Este documento recomienda que haya un grupo central en la compañía (u organización de desarrollo de software) que controla el desarrollo y la evolución de las prácticas recomendadas de seguridad y las mejoras de los procesos, actúa como fuente de conocimientos para toda la organización y realiza una revisión (la revisión final de seguridad (*FSR, Final Security Review*)) antes del lanzamiento del software.

Según la experiencia, la existencia de tal organización es vital para implementar adecuadamente el SDL, así como para mejorar la seguridad del software. No obstante, algunas organizaciones pueden plantearse delegar en un consultor o asesor externo esta función de "equipo de seguridad central".
El objetivo de estas mejoras de procesos es reducir el número y la gravedad de las vulnerabilidades de seguridad del software utilizado por los clientes.

El equipo de seguridad debe estar disponible para recurrir a él con frecuencia durante el diseño y el desarrollo del software, y es preciso confiarle información técnica y empresarial confidencial. Por estos motivos, la mejor solución es constituir un equipo de seguridad dentro de la organización de desarrollo de software (aunque tal vez sea preciso contratar a consultores que participen en la creación y entrenamiento de los miembros del equipo).

PRINCIPIOS PARA LOGRAR SOFTWARE MAS SEGURO

La experiencia en seguridad del software real ha permitido establecer una serie de principios de alto nivel para lograr un software más seguro. Microsoft hace referencia a estos principios como SD3+C: Seguro por diseño, Seguro por definición, Seguro en distribución y Comunicaciones.

- Seguro por diseño

  En este principio la arquitectura, el diseño y la implementación del software se deben realizar de manera que proteja tanto el software como la información que procesa, además de poder resistir ataques.

- Seguro por definición

  En el mundo real, el software no es totalmente seguro, por lo que los diseñadores deben asumir que habrá errores de seguridad. Para minimizar los daños que se producirán cuando los atacantes descubran estos errores, el estado predeterminado del software debe elegir las opciones más seguras. Por ejemplo, el software debe ejecutarse con los mínimos privilegios necesarios (Least Privilege) y los servicios y las características que no sean necesarios de manera habitual deben deshabilitarse de manera predeterminada o establecer que sólo unos pocos usuarios puedan tener acceso a ellos.

- Seguro en distribución

  Durante este principio se debe incluir con el software, información y herramientas que ayuden a los administradores y a los usuarios a utilizar el software con seguridad. Además, la implementación de las actualizaciones debe ser sencilla.

- Comunicaciones

  Los programadores de software deben estar preparados para detectar las vulnerabilidades de seguridad del producto y deben comunicarse de manera abierta y responsable con los usuarios y los administradores para ayudarles a tomar las medidas de protección adecuadas

Aunque todos los elementos de SD3+C imponen ciertos requisitos durante el proceso de desarrollo, los dos primeros elementos, seguro por diseño y seguro por definición, son los que más favorecen la seguridad. Seguro por diseño obliga a utilizar procesos que tratan de evitar la inclusión de vulnerabilidades de seguridad desde el principio, mientras que Seguro por definición exige el desarrollo de ejecutarse con las opciones más seguras.

FASES PARA EL DESARROLLO DE SOFTWARE SEGURO

FASE DE REQUISITOS

La necesidad de considerar la seguridad "de abajo a arriba" es uno de los principios fundamentales del desarrollo de sistemas seguros. Teniendo en cuenta que muchos proyectos de desarrollo generan la siguiente versión a partir de la anterior, la fase de requisitos y el planeamiento inicial de una nueva versión o lanzamiento ofrece una oportunidad estupenda para crear software seguro.

Durante la fase de requisitos, el equipo de producto se pone en contacto con el equipo de seguridad central para solicitar la asignación de un asesor de seguridad (conocido como el encargado de la seguridad en la implementación del SDL) que actúa como punto de contacto, recurso y guía a través de los procedimientos de planeamiento.
El asesor de seguridad ayuda al equipo de producto a revisar los planes, aporta recomendaciones y asegura que el equipo de seguridad planee los recursos necesarios de acuerdo con el programa de fechas del equipo de producto. El asesor de seguridad actúa como contacto entre el equipo de producto y el equipo de seguridad desde el inicio del proyecto hasta la finalización de la revisión final de seguridad y el lanzamiento del software.

La fase de requisitos es la oportunidad ideal para que el equipo de producto se plantee cómo se integrará la seguridad en el proceso de desarrollo, identifique los objetivos de seguridad clave, y, por lo demás, maximice la seguridad del software procurando minimizar el impacto sobre los planes y los programas.
Como parte de este proceso, el equipo debe considerar cómo se integrarán las características de seguridad y las medidas de control que se utilizarán con el software que están desarrollando.

La consideración general por parte del equipo de producto de los objetivos, los retos y los planes de seguridad debe reflejarse en los documentos de planeamiento generados durante la fase de requisitos. Aunque es posible que estos planes cambien a medida que el proyecto avanza, articularlos desde el principio garantiza que no se pase por alto ningún requisito ni surgen sorpresas de última hora.

Cada equipo de producto debe considerar los requisitos de características de seguridad como parte de esta fase de requisitos.

FASE DE DISEÑO

La fase de diseño identifica la estructura y los requisitos globales del software. Desde el punto de vista de la seguridad, los elementos clave de la fase de diseño que se debe seguir son:

- Definir la arquitectura de seguridad y las directrices de diseño

  Definir la estructura global del software desde el punto de vista de la seguridad e identificar los componentes cuyo correcto funcionamiento es esencial para la seguridad (base de computación confiable). La identificación de técnicas de diseño, como el uso de capas o lenguaje con tipos inflexibles, la aplicación de privilegios mínimos y la minimización de la superficie de ataque, que se aplican al software de manera global.

  El uso de capas se refiere a la organización del software en componentes bien definidos que se estructuran para evitar dependencias circulares entre componentes. Los componentes se organizan en capas y una capa superior puede depender de los servicios de capas inferiores, pero se prohíbe que las capas inferiores dependan de las capas superiores.

  Los detalles específicos de cada uno de los elementos de la arquitectura se indican en las especificaciones de diseño individuales, pero la arquitectura de seguridad corresponde a una perspectiva global sobre el diseño de seguridad.

- Documentar los elementos de la superficie de ataque del software

  Teniendo en cuenta que el software no logrará una seguridad perfecta, es importante que únicamente se expongan de manera predeterminada las características que utilicen la mayoría de los usuarios y que dichas características se instalen con el mínimo nivel de privilegios posible. La medición de los elementos de la superficie de ataque ofrece al equipo de producto un indicador continuo de la seguridad predeterminada y les permite detectar las instancias en las que el software es más susceptible de recibir ataques. Aunque algunas instancias con mayor superficie de ataque pueden estar justificadas por una mayor facilidad de uso o unas mejores funciones del

producto, es importante detectar y considerar cada una de estas instancias durante el diseño y la implementación para lanzar el software con la configuración predeterminada más segura posible.

- Realizar un modelado de las amenazas

  El equipo debe realizar un modelado de amenazas por componentes.

  El proceso de modelado identifica las amenazas que pueden dañar el software y la probabilidad de que se ocasione dicho daño (estimación del riesgo). Por tanto, el modelado de amenazas ayuda al equipo de producto a identificar las necesidades de características de seguridad y las áreas en las que es necesario revisar con especial minuciosidad el código y probar la seguridad. Este proceso se debe realizar con una herramienta capaz de capturar modelos de amenazas, guardar los resultados y actualizarlos cuando sea necesario, para tratar de conservar el software cada vez mas seguro.

- Definir los criterios de publicación adicionales

  Se debe definir criterios de seguridad básicos en la organización, para la publicación y lanzamientos de software, los cuales son preciso cumplir para poder lanzar el software. A demás, el proceso de desarrollo debe descubrir las vulnerabilidades de seguridad y solucionarlas antes de que se detecten, en vez de tener que solucionarlas después de su detección.

FASE DE IMPLEMENTACIÓN

Durante la fase de implementación, el equipo de producto prueba, programa e integra el software. Los pasos destinados a eliminar los errores de seguridad o a impedir que se incluyan desde el principio son de gran utilidad, ya que reducen considerablemente la probabilidad de que las vulnerabilidades de seguridad lleguen a la versión final del software que se lanzará a los clientes.

Los resultados del modelado de amenazas ofrecen una orientación especialmente importante durante la fase de implementación.

Elementos del SDL a aplicar en la fase de implementación:

- Aplicar estándares de codificación y de pruebas
  Los estándares de codificación evitan que los programadores incluyan errores que puedan producir vulnerabilidades de seguridad. Por ejemplo, el uso de construcciones de manipulación de búferes y de manejo de cadenas más seguras y coherentes ayuda a evitar la aparición de vulnerabilidades de seguridad de saturación del búfer. Los estándares de pruebas y las prácticas recomendadas permiten garantizar y detectar posibles vulnerabilidades de seguridad.
- Aplicar herramientas de confusión
  Estas herramientas ofrecen entradas estructuradas pero no válidas a las interfaces de programación de aplicaciones (API) de software y a las interfaces de red para maximizar la probabilidad de detectar errores que puedan ocasionar vulnerabilidades de seguridad del software.
- Aplicar herramientas de exploración del código de análisis estático
  Las herramientas pueden detectar algunos tipos de errores de codificación que producen vulnerabilidades de seguridad, incluidas saturaciones de búfer, desbordamientos con enteros y variables no inicializadas.

En la actualidad existen varias herramientas que ayudan con esta labor:

- Herramientas para código administrado o manejado: Code Analyzer o Code Analisys tool.
- Herramientas para código no administrado o código nativo: Prefast, Application Verrifier, Safe CRT Libraries, Buffer Security Check.

- Realizar revisiones del código

  Estas ayudan complementar las herramientas automatizadas, ya que aplican el esfuerzo de programadores expertos para examinar el código fuente y detectar y eliminar posibles vulnerabilidades de seguridad. Estas revisiones constituyen un paso fundamental para eliminar las vulnerabilidades de seguridad del software durante el proceso de desarrollo.

FASE DE COMPROBACIÓN

La fase de comprobación es el punto en el que software ya incorpora toda la funcionalidad y los usuarios pueden comenzar a probar la versión beta. Durante esta fase, mientras se prueba la versión, el equipo de producto debe realizar una campaña de seguridad que incluye revisiones del código de seguridad aparte de las realizadas en la fase de implementación.

Se recomienda realizar una campaña de seguridad durante esta fase, tanto para asegurar que el software final cumpla los requisitos, como para permitir una revisión en detalle de todo el código heredado de versiones anteriores del software.

FASE DE LANZAMIENTO

Durante la fase de lanzamiento, el software debe someterse a una Revisión Final De Seguridad (FSR Final Security Review). Esta FSR debe responder a la siguiente pregunta: "Desde el punto de vista de la seguridad, ¿está este software preparado para los clientes?" La FSR se debe realizar antes de la finalización del software. El software debe ser estable antes de la FSR y es de esperar que antes del lanzamiento sólo se realicen cambios mínimos y no relacionados con la seguridad.

La FSR es una revisión independiente del software que realiza el equipo de seguridad central de la organización. El asesor de seguridad aconseja al equipo de producto sobre el ámbito de la FSR que requiere el software y ofrece al equipo de producto una lista de los requisitos de recursos antes de la FSR.

El equipo de producto debe proporcionar al equipo de seguridad los recursos y la información necesaria para llevar a cabo la FSR. Al comienzo de la FSR, el equipo de producto debe rellenar un cuestionario y entrevistarse con un miembro del equipo de seguridad asignado a la FSR. En toda FSR se deben revisar los errores que se identificaron en un principio como errores de seguridad, pero que tras analizarlos se consideró que no afectaban a la seguridad, para asegurarse de que este análisis es correcto.

Una FSR también incluye una revisión de la capacidad del software para soportar vulnerabilidades de seguridad detectadas recientemente en un software similar.
Una FSR para una versión de software importante requerirá realizar pruebas de penetración y, posiblemente, recurrir a asesores de revisión de seguridad externos que ayuden al equipo de seguridad.

La FSR no es únicamente un examen que se puede aprobar o suspender ni tampoco pretende detectar todas las vulnerabilidades de seguridad que quedan en el software, lo que no sería factible, lo que hace es proporcionar al equipo de producto y a la administración superior de la organización una idea global del nivel de seguridad del software y de la probabilidad de que pueda resistir ataques una vez que se haya entregado a los clientes.

Si la FSR detecta patrones de vulnerabilidades de seguridad restantes, no bastará con solucionar las vulnerabilidades detectadas, sino que habrá que repetir la fase anterior y tomar las acciones necesarias para tratar los orígenes (por ejemplo, mejorar los conocimientos, mejorar las herramientas, etc.).

FASE DE SERVICIO TÉCNICO Y MANTENIMIENTO

A pesar de la aplicación del SDL durante el desarrollo, las prácticas de desarrollo más avanzadas no consideran que se pueda publicar software que no tenga ninguna vulnerabilidad de seguridad.

Incluso aunque el proceso de desarrollo pudiera eliminar todas las vulnerabilidades de seguridad del software que se va a publicar, se descubrirían nuevos ataques y el software considerado "seguro" pasaría a ser vulnerable. Por tanto, los equipos de producto deben prepararse para responder a nuevas vulnerabilidades en el software que se entrega a los clientes.

Parte del proceso de respuesta consiste en la preparación para evaluar los informes de vulnerabilidades y lanzar consejos y actualizaciones de seguridad siempre que sea necesario. El otro componente del proceso de respuesta consiste en realizar un estudio exhaustivo de las vulnerabilidades detectadas y tomar las medidas oportunas contra estas vulnerabilidades.

La medida más común y en muchos casos mas utilizada puede ser el lanzamiento de una actualización que resuelva un error aislado de seguridad.
También es conveniente en algunos casos concientizar a los usuarios finales de usar contraseñas difíciles de detectar, y de no dejarlas en lugares donde cualquier hacker o persona malintencionada las pueda encontrar.

El objetivo durante la fase de respuesta consiste en aprender de los errores y utilizar la información de los informes de vulnerabilidades para detectar y eliminar otras vulnerabilidades antes de que se descubran en la práctica y se utilicen para poner en peligro a los clientes.

EL EQUIPO DE SEGURIDAD

Es importante que en toda organización desarrolladora de software o en donde el departamento de Informática sea grande e importante exista un grupo de seguridad, que se encargue de cumplir con todo lo concerniente a la seguridad, a demás de los ítems anteriores, se deben seguir también:

- Educación obligatoria del personal de ingeniería.
- Implementación obligatoria del SDL.
- Indicadores de los equipos de producto.
- Ofrecer cursos y consultorías a todo el personal relacionado con informática.
- Asignación de los "asesores de seguridad" que guían a los equipos de producto a través del proceso.
- Servicio de expertos en la materia en una amplia gama de temas de seguridad.
- Realización de las revisiones finales de seguridad antes del lanzamiento del software.
- Investigación técnica de las vulnerabilidades detectadas en el software entregado a los clientes, para asegurarse de que se entienden las causas que las han originado y se tomen las medidas adecuadas.

RECOMENDACIÓNES

La adopción del SDL no debería suponer un costo adicional para el desarrollo de software. Más seria un valor agregado ya que las ventajas de ofrecer un software más seguro compensan los costos.

La implementación del SDL debería ser por un equipo de seguridad dentro de la organización, que se encargue de implementar y mejorar el SDL.

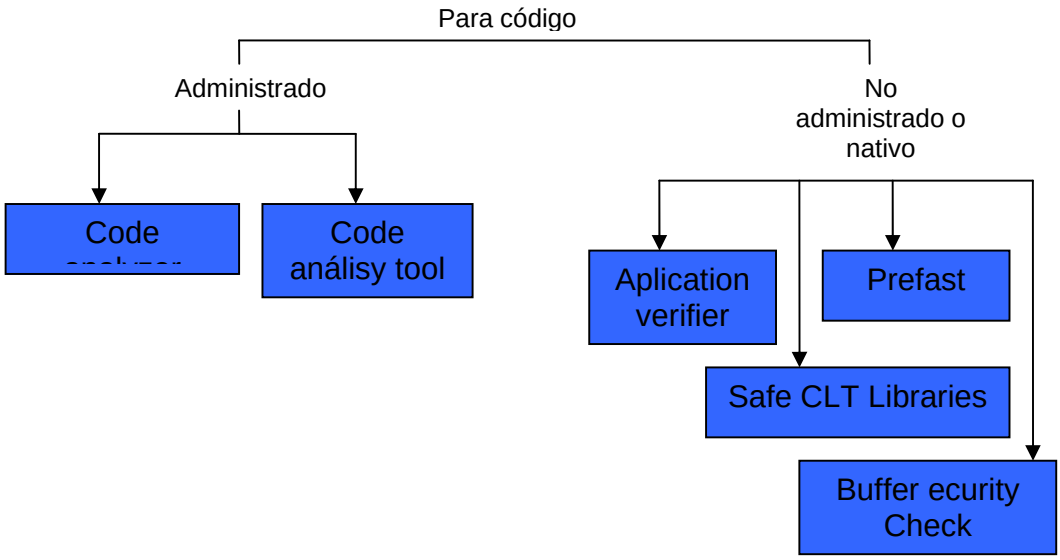
Cumplir con los principios SD3+C ayuda a lograr que un software sea mas seguro.

Se recomienda realizar una campaña de seguridad durante la fase de comprobación.

Investigar, desarrollar, y mejorar el SDL, ya que los hacker nunca paran de trabajar y cada vez se hacen más fuertes.

El director o administrador del proyecto, debe inspeccionar que los grupos de producto y de seguridad estén trabajando en conjunto para implementar todas y cada una de las políticas descritas en el capitulo I.

HERRAMIENTAS DE SEGURIDAD



CAPITULO II

HERRAMIENTAS DE SEGURIDAD

MODULO I

HERRAMIENTAS QUE AYUDAN A CREAR SOFTWARE SEGURO

Existen unas herramientas que ayudan de forma simple, económica y eficaz a hacer diversas verificaciones alrededor del tema de seguridad.

Las herramientas de Análisis de código proporcionan a los desarrolladores información sobre posibles defectos en su código fuente.

Estas herramientas representan las comprobaciones que se realiza durante el análisis; como advertencias, estandarización para la declaración de variables, métodos, funciones y/o propiedades, entre otras. Los mensajes de advertencia identifican cualquier problema pertinente de programación y diseño y, cuando es posible, proporciona información sobre cómo corregir el problema.

Estas herramientas de análisis de código que están enfocadas para código manejado (código .NET) y código no manejado o código no administrado (C/C++). Utilizar estas herramientas sobre aplicaciones, ayuda al desarrollador a tener en cuenta la seguridad, antes, durante y después de la aplicación.

Para el código administrado, estas herramientas analizan y muestran información sobre los ensamblados administrados, las infracciones de las reglas de programación y diseño estipuladas en las Instrucciones de diseño de Microsoft .NET Framework.

Estas herramientas estáticas de análisis de código, no son una autoridad en seguridad, ya que tienen un objetivo limitado y no sustituyen las verificaciones manuales de código.

Herramientas para código administrado

- Code analyzer o code analisys tool.

Herramientas para código no administrado (nativo).

- Prefast
- Aplication Verifier
- Safe CLT Libranes
- Buffer Security Check.

1. **CODE ANALYZER:** Esta herramienta sirve para cualquier lenguaje. NET (especialmente Visual C# y Visual Basic).

   Code analyzer es una evaluación del FXCOP, la cual es una herramienta que esta disponible para las versiones anteriores del FrameWork (versión 1.0 y 1.1). FXCOP era una herramienta aparte del FrameWork, la cual se podía utilizar para controlar, comprobar y construir códigos fuentes seguros.

   Code Analyser puede ser llamado durante compilación o ejecución y actúa sobre los assemblies.
   Las ventajas del Code Analyzer son:

   - Seguridad
   - Garantizar que el código siga una cierta convención para:
     - Nombres de variables
     - Formatos de funciones
     - Formatos de métodos
     - Formatos de propiedades
   - Verificar las consultas y procedimientos almacenados en Sql Server contra vulnerabilidades de seguridad.
   - Punteros que no deben ser visibles
   - No atrapar errores sin excepción de tipo genérico.
   - Los Ensamblados deben tener un nombre fuerte que tenga varias implicaciones de seguridad.
   - Las reglas Reability y User

Activar todas las opciones de Seguridad del Code Analyzer, pueden conllevar en algunas alarmas que sean falsas, lo que pueden inducir a los desarrolladores a estar extraños a algunas advertencias y hacer dudar que el Code Analyzer, realmente sea efectivo.

Seleccionar todas las opciones de seguridad de Code Analyzer no es recomendable, es importante realizar esta labor con buenos conocimientos tanto de seguridad como de desarrollo, para así establecer cualidades para estos ajustes quitando lo que sea considerado innecesario.

Algunas razones por las cuales no activar el Code Analyzer:

- La Nomenclatura que se utiliza podría no coincidir con los estándares del desarrollador o los de una compañía.
- Difícilmente el desarrollador quiere solucionarlo todo.
- Firmar criptográficamente las aplicaciones que están afectando la manera en que se distribuyen las mismas.

Es posible configurarlo de manera tal que se verifiquen las políticas del Code Analyzer, cada vez que se este protegiendo el código, esto es cuando se esta haciendo un CHECKING, que es lo que en realidad se recomienda hacer.

Code Analyzer es una herramienta que es extensible, es decir, el desarrollador puede crear sus propias reglas, dependiendo de los requerimientos de cada aplicación.

2. **PREFAST:** Es una herramienta de análisis estático para C.

   Prefast analiza el código fuente C o C++ y emite un informe de errores o de prácticas de programación que aunque sean validas son cuestionables o peligrosas, como por ejemplo:

   La utilización de funciones inseguras de manipulación de cadenas de caracteres, entre otras.

3. **Application Verifier:** Es una herramienta que se utiliza en tiempo de ejecución para controlar el código no administrado.

   Application Verifier ayuda a depurar el uso indebido de la memoria, ya que este es un gran problema de seguridad del cual se valen la mayoría de los hacker para volcar aplicaciones, a demás encuentra problemas de programación sutiles que pueden ser difíciles de encontrar y ejecuta la aplicación en un ambiente controlado, monitorizando la integración de la aplicación con el sistema operativo, esto incluye:

   - Utilización de ventanas
   - Acceso al registro de Windows
   - Acceso al sistema de archivos
   - Segmento de memoria y bloqueos de memoria y mucho mas

   Una buena practica en algunas ocasiones es operar bajo una cuenta de usuario limitada (también conocida como una “cuenta de usuario menos privilegiado”, o LUA) en vez de una cuenta administrativa; los derechos limitados de usuario se aplican a cualquier programa malicioso que intente infiltrar su sistema y así se minimiza el daño que pueden hacer.

   Application Verifier puede verificar como se comporta una aplicación bajo un usuario que tiene pocos privilegios (LUA).

4. **Safe CRT Libraries:** Posee una librería en tiempo de ejecución para C y C++, tomando en cuenta la seguridad.

   En C y C++ existen muchas funciones que con llevan a vulnerabilidades de seguridad, por ejemplo:

   STRCPY: función de C y C++ obsoleta. En el .NET funciones nuevas y mas seguras se han creado para cubrir las obsoletas e inseguras.

   En .NET el manejo de caracteres en su conjunto fue reformulado.

   Las funciones nuevas verifican los valores que son pasados para ellas y generan mensajes de error en caso de que reciban parámetros equivocados, aun cuando estos parámetros no impidan el funcionamiento.

El tratamiento de los errores también fue mejorado, dado que este es un problema común de la biblioteca antigua, lo cual lleva a varias fallas de seguridad.

5. **Buffer Security Check:** Nueva opción del compilador de C++, esta instrumenta el código de manera que se pueda detectar un tipo nocivo de Buffer over array o saturación del buffer, esta vulnerabilidad se da a nivel de pila, lo cual hay que prestarle mucha atención ya que es muy importante, ya que aquí están contenidas las variables locales, como las direcciones de retorno.

   Un Buffer over array permite en la pila que un atacante ejecute código arbitrario.

   Esta herramienta no pretende detectar todos los buffer over array que se puedan presentar en una aplicación pero interrumpe la ejecución cuando es detectado un salteo de Buffer.

6. **Otras Herramientas:** Test Unitarios y test de carga son otras herramientas de análisis estático, las cuales hacen que se pueda evaluar si el código es seguro y confiable.

   Estas herramientas aunque no hayan sido diseñadas específicamente para combatir funciones de seguridad, pueden ayudar a solucionar y detectar este problema.

MODULO II

CODE ACCESS SEGURITY Y CLICK ONCE

La herramienta de la seguridad del acceso del código permite a usuarios y a administradores modificar la política de la seguridad para el nivel de: máquina, usuario, empresa.

El sistema de los permisos que un assembly recibe es determinado por la intersección de los sistemas del permiso permitidos por estos niveles de tres políticas. Cada nivel de la política es representado por una estructura jerárquica de los grupos del código.

En este modulo veremos como el CAS pretende solucionar el problema de las Listas de Control de Acceso (ACL).

Las ACLs usan valores más sólidos por defecto, son las que definen los criterios que usa un sistema operativo para proteger los recursos de red. Por ejemplo, crear el nuevo System Root ACL y configurarlo por defecto significa que los usuarios ya no pueden escribir archivos en la raíz del sistema, lo cual previene ciertos ataques engañosos.

Trabajar con ACLs todavía tiene sus limitaciones:

- Esta incompleto para Internet
- Un usuario (Principal) tiene privilegios razonablemente inmutables en el sistema.

En un entorno de trabajo los escenarios para descargar programas de Internet:

- Acceso del 100%: Para programas del sitio de trabajo
- Acceso limitado: Para programas de un compañero del sitio de trabajo
- Ningún acceso: Para programas de sitios desconocidos

Code Access Segurity expande el mecanismo de Listas de Control de Acceso, es decir el CAS en ningún momento violara la seguridad de los ACLs: por ejemplo si un Principal coloco una carpeta de solo lectura nada permitirá que esta carpeta sea modificada.

El CAS permite un control más fino basado en pruebas suministradas por los programas, estas pruebas evidentes están divididas en dos criterios: criterios Geográficos y criterios Criptográficos.

CRITERIOS GEOGRAFICOS: relacionados a los orígenes del cual el programa vino, los cuales pueden ser:

- Su propia carpeta, donde el aplicativo fue instalado.
- Global Assembly Cache (GAB)
- Nombre del sitio
- Zona de Internet Explorer, bien sea Internet, intranet o local.

CRITERIOS CRIPTOGRAFICOS:

- Hash del Assembly cifrado por medio del MD5 o SHA1
- Certificado de publicación de dicho software a través de una de las entidades certificadoras internacionales
- Strong Name – nombre de la firma criptográfica que brinda .NET

CLICK ONCE

ClickOnce es una nueva tecnología de implementación de Windows Forms para el framework 2.0. Esta tecnología facilita el proceso de instalación de aplicaciones, así como el de actualización.

Esta tecnología reúne las ventajas de las Interfaces Gráfica de Usuario (GUI) con las facilidades de distribución y actualización que posee la web, ClickOnce incluye:

- Aplicativo GUI
- .EXE y DLLs distribuidas a través de protocolo HTTP
- Se comunican a través de Web Service, aunque esta no es estrictamente necesaria, también pueden utilizar el protocolo HTTP.
- Luego descargadas las aplicaciones, son descargadas en el lado del cliente y se ejecutan incluso en modo desconectado.
- Existe una Interfaz de Programa de Aplicación (API) para controlar el proceso de actualizaciones y controlar la conectividad (on-line u off-line) con el origen que se instalo en el aplicativo.

Las aplicaciones ClickOnce poseen en su interior un archivo especial llamado Manifiesto, el cual lista todos los ensamblados o assemblies que componen ese proyecto, para permitir el control de las actualizaciones.
Adicionalmente el manifiesto contiene información de todos los permisos que el proyecto necesita para ejecutarse, construyendo un ambiente con permisos estrictamente necesarios. El manifiesto esta firmado criptográficamente para impedir que pueda ser alterado en forma maliciosa.

A demás, ofrece permisos opcionales analizados por el programa en tiempo de ejecución para que se ejecuten en una maquina mas restringida, aun cuando uno u otro permiso no este disponible. La aplicación podrá tomar distintos comportamientos de acuerdo al entorno de seguridad o permisos de la cual dispongan en cada momento.

El desarrollador debe editar los permisos del ClickOnce, para crear un conjunto con el mínimo posible de privilegios posibles (Least Privilege). Visual Studio puede calcular los permisos basándose en la lista de ensamblados que la aplicación tenga como referencia.

Asignar los permisos es una tarea importante que debe tomar el desarrollador, ya que la aplicación podría entonces asignar permisos innecesarios.

RECOMENDACIÓNES

Seleccionar y configurar adecuadamente las herramientas estáticas para el desarrollo de código seguro a trabajar, dependiendo, de la tecnología que se este utilizando en la aplicación, ya que estas son unas herramientas probadas y certificadas que están disponibles desde años.

Proporcionar una infraestructura y un conjunto de reglas, que faciliten el trabajo para estas herramientas estáticas, por ejemplo el Code Analyzer.

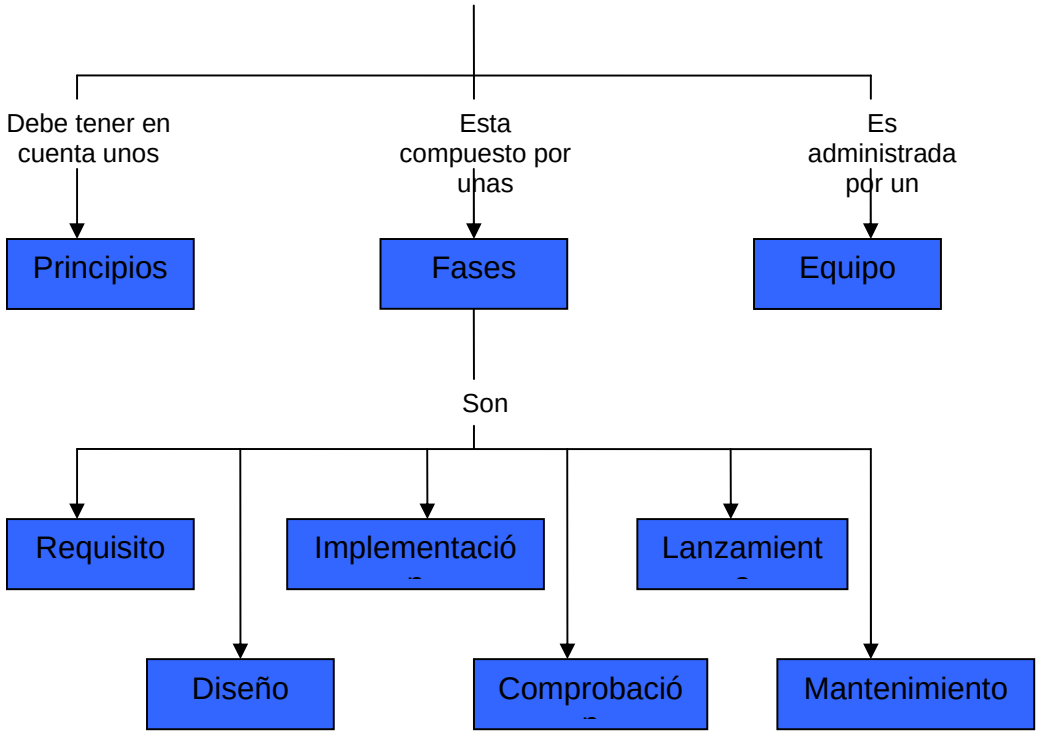
Crear políticas de seguridad del acceso del código.

Seguir los tres tipos de escenarios, al momento de descargar un programa de la Zona de Internet Explorer.

Crear un mínimo posible de privilegio en todas las aplicaciones.

Editar los permisos del archivo de manifiesto, según las políticas de seguridad preestablecidas, construyendo un ambiente con permisos estrictamente necesarios.

SEGURIDAD EN APLICACIONES WEB



CAPITULO III

SEGURIDAD EN APLICACIONES WEB

La seguridad es un aspecto crítico de las aplicaciones Web. Las aplicaciones Web, por definición, permiten el acceso de usuarios a recursos centrales, al servidor Web y, a través de éste, a otros como los servidores de base de datos. Con los conocimientos y la implementación correcta de medidas de seguridad, puede proteger sus propios recursos así como proporcionar un entorno seguro donde los usuarios trabajen cómodos con su aplicación.

Este modulo ofrece información general acerca de la seguridad en aplicaciones Web, y describe los tipos de cuestiones que se deben plantear cuando se crean aplicaciones.

El tema de crear aplicaciones Web seguras es un tanto complejo, ya que requiere realizar un estudio para comprender los puntos vulnerables de la seguridad. También es necesario familiarizarse con las posibilidades de seguridad que ofrecen Windows, .NET Framework y ASP.NET. Finalmente, resulta vital entender cómo utilizar estas funciones de seguridad para contrarrestar las amenazas.

La seguridad de aplicaciones web esta basada básicamente en dos componentes:

- Un componente administrativo
- Un componente de desarrollo de software

Algunas de las recomendaciones de seguridad en cuanto al componente administrativo, son:

- Mantener el Internet Information Server siempre actualizado con los últimos Service Pack y actualizaciones de seguridad, así como ajustar adecuadamente los routers.
- Configurar apropiadamente los firewall
- Realizar copias de seguridad con asiduidad y guárdelas en un lugar seguro.
- Mantener el equipo del servidor en un lugar físico seguro, de forma que los usuarios no autorizados no puedan tener acceso a él, apagarlo, llevárselo, etc.

- Proteger el equipo del servidor Web y todos los demás equipos de la misma red con contraseñas rigurosas.
- Cerrar los puertos que no se utilicen y desactive los servicios no usados.
- Ejecutar un programa antivirus que supervise el tráfico.
- Establecer y hacer respetar una política que prohíba a los usuarios tener sus contraseñas escritas en una ubicación fácil de localizar.
- Manténerse informado sobre las últimas revisiones de seguridad.
- Entre otros procedimientos

El software seguro es un proyecto bien pensado y deliberado, el problema es que la gran mayoría de los programadores no saben como escribir código seguro y escriben código inseguro sin darse cuenta.

INTERNET INFORMATION SERVER

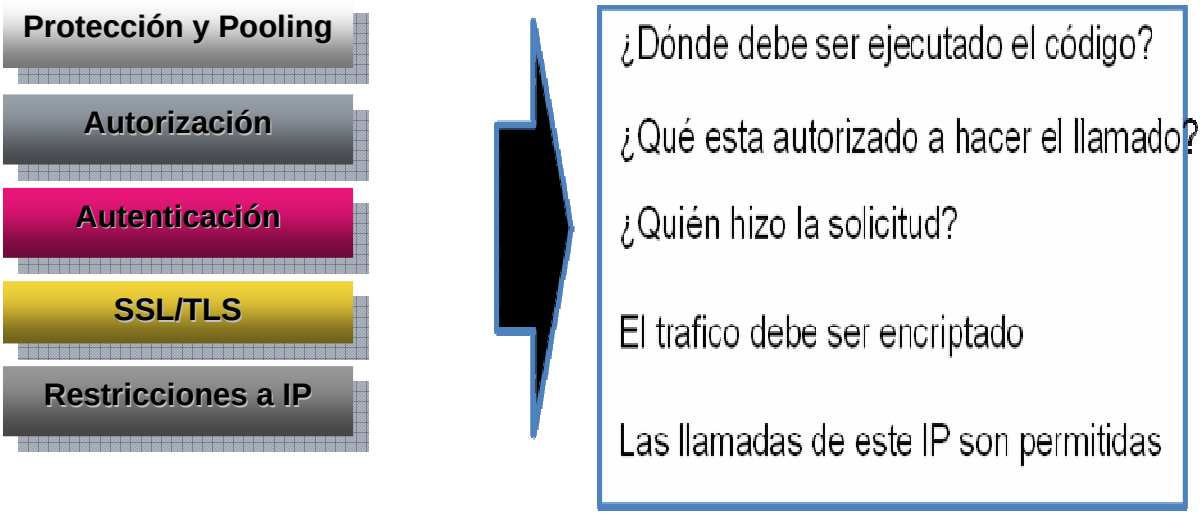


Ilustración 1

Internet Information Server da soporte para hacer pooling y protección del aplicativo, lo cual proporciona un margen de manera más amplio.

La autorización consiste en un conjunto de reglas que definen quien puede acceder a cuales recursos dentro de un sitio web.

La autenticación y la autorización van lado a lado, porque si no se sabe quien esta realizando la llamada a la pagina, no se puede aplicar las reglas de acceso basadas en la identidad de dicho cliente.

La autenticación es un mecanismo para identificar al cliente, IIS soporta diferentes mecanismos de autenticación, cada uno con sus ventajas y desventajas:

Tabla 1

	Basic	Digset	Integrada Windows	Certificado X509	Passport
Necesita cuentas Windows	S	S	S	S	N
Soporta delegación	S	N	N	S	N
Credenciales texto plano	S	N	N	N	N
Soporta navegador diferente Internet Explorer	S	S	N	N	S
Pasa por el firewall	S	S	N	N	S
Experiencia de usuario simple	N	N	S	S	S

AUTENTICACIÓN BÁSICA: Esta es la mas simple y menos segura, en este tipo de autenticación el navegador pide al usuario login y contraseña, y estas las transmite al IIS, este utiliza estas credenciales que necesariamente deben representar un usuario valido de Windows para iniciar sesión de autenticación, la autentica básica no encripta las credenciales, estas se transmiten en formato de texto simple.

AUTENTICACIÓN DIGSET: Tiene cierto parecido a la autenticación básica, en el sentido de que el navegador es quien pide credenciales al usuario.

La autenticación Digset es mas segura que la básica porque no transmite la información en forma de texto simple, encripta la información.

AUTENTICACION INTEGRADA DE WINDOWS: Permite una experiencia de usuario mas normalizada de lo que ofrece la autenticación Digset y la básica, porque no pide al usuario que suministre las credenciales, aprovecha que el usuario ya esta identificado localmente.

El trafico de red basado en NTLM o interbios funciona con deficiencias a través del firewall y por este motivo este tipo de autenticación es utilizado básicamente en ambientes de intranet.

CERTIFICADOS X509: Similar a la autenticación integrada de Windows en el aspecto en que tampoco arroja cuadros de dialogo pidiendo autenticación. Permite atravesar los firewall.

Puede ser configurada para realizar mapeos explícitos, esto es cada uno de los certificados apuntando a una cuenta de usuario, pero en realidad lo mas común es hacer mapeo de nombres de usuario a cuentas que utilicen ActiveX layetory.

AUTENTICACION PASSPORT: Centralizada en un servidor externo y esta permite un marco de seguridad e identificación que admita ser utilizado por diversos servidores al mismo tiempo.

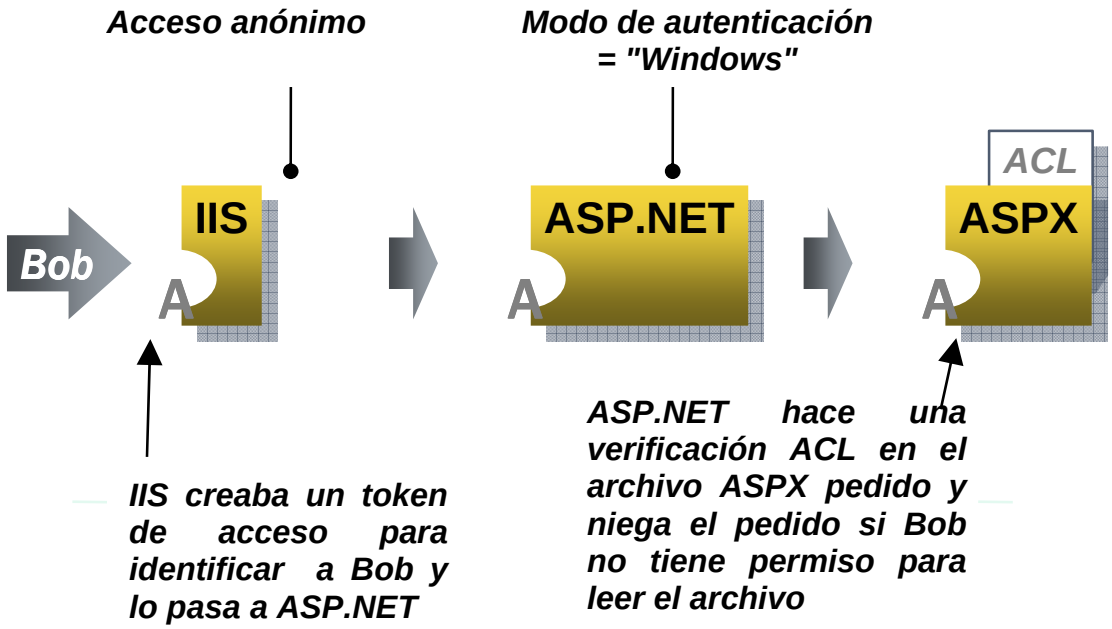
Secure Socket Layer (SSl) para encriptar las comunicaciones entre los navegadores y el servidor web. Encriptar la comunicación es importante y se debe hacer si ese tráfico contiene datos confidenciales o peligrosos tales como:

- Contraseñas
- Números de tarjetas de crédito
- Cookies de autenticación, etc…

Las restricciones a IP permiten que los administradores concedan acceso a los navegadores que vienen de determinadas direcciones IP o nombres de dominio.

TIPOS DE AUTORIZACION EN ASP.NET

AUTORIZACION POR ACL esta se basa en impedir que un determinado usuario visualice una página utilizando los permisos propios del sistema operativo para negarle la capacidad de leer el archivo ASPX relativo a la página.



AUTENTICACION URL basada en la entrada del archivo de configuración WEB.CONFIG. En el cual se pasa la cadena de conexión y propiedades adicionales para realizar conexión a bases de datos.

LA COMBINACION PERFECTA

En los escenarios de Intranet, o cuando los usuarios posean cuentas Windows, s recomienda autenticación Windows y autenticación basada en ACL, por lo tanto se pueden utilizar login SQL o autenticación Windows. Acá en este tipo de escenario la mayoría de los accesos ocurren dentro del firewall.

Escenarios de Internet, aquí en este tipo de escenario se recomienda autenticación de formulario y autorización basada en URL, ya que el token pasado de IIS a ASP.NET puede ser un usuario anónimo, y seria tedioso tener usuario SQL o Windows por cada usuario que acceda al aplicativo. Contrario a la anterior la mayoría de acceso esta fuera del firewall.

RECOMENDACIÓNES

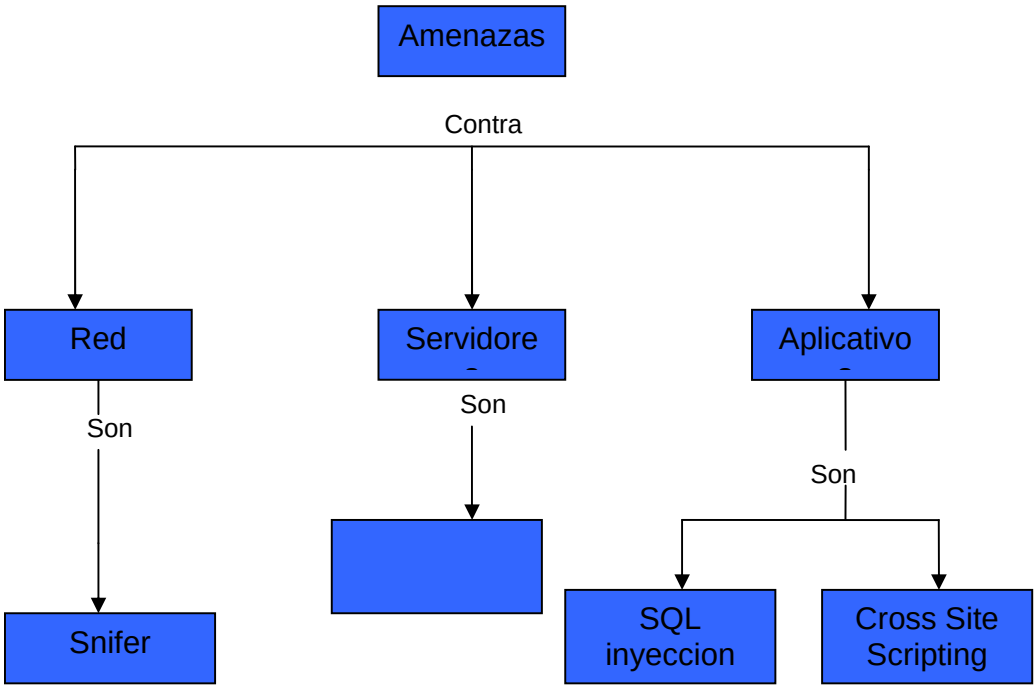
Mantener el Internet Information Server siempre actualizado con los últimos Service Pack, ajustar adecuadamente los routers, configurar apropiadamente los firewall.

Realizar copias de seguridad con asiduidad, conservar el servidor en un lugar físico seguro, de forma que los usuarios no autorizados no puedan tener acceso.

Tener en cuenta los diferentes tipos de escenarios para escoger la combinación adecuada entre la autenticación y la autorización.

Encriptar la comunicación entre los servidores y la aplicación.

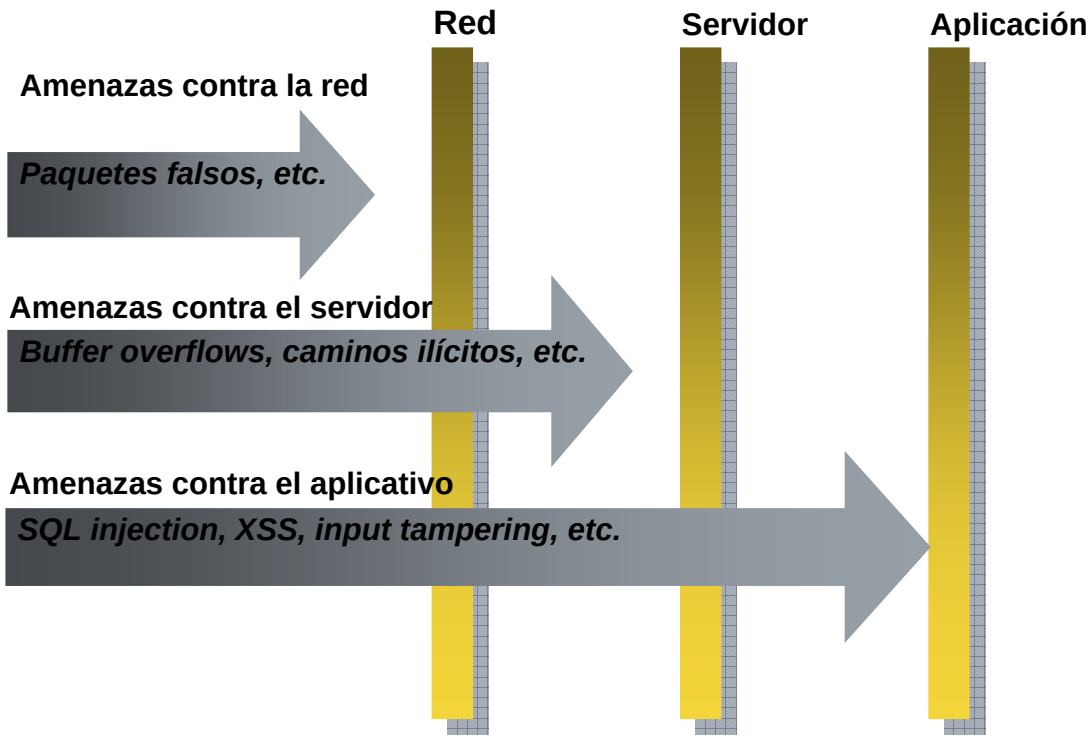
AMENAZAS Y MODELOS DE AMENAZAS



CAPITULO IV

AMENAZAS Y MODELOS DE AMENAZAS

Modelar la etapa de desarrollo de una aplicación para resistir amenazas y construir código seguro, es lo recomendado y es el camino que debe seguir todo desarrollador.



En general las defensas contra amenazas de red y de servidor son de tipo administrativa y contra aplicaciones consisten en escribir código seguro.

AMENAZA CONTRA LA RED

La verificación de puertos es muy importante, porque los puertos que han sido dejado abiertos en los firewall exponen los servidores que se encuentran por detrás de ellos. Por ejemplo, el fallo en el bloqueo de la puerta (TSP 1433) y la puerta (UDP 1434) en el firewall posibilito que atacantes se conectaran remotamente con una vulnerabilidad de Buffer, la cual se puede diseminar como un gusano.

El espionaje en redes dentro del hilo de conexión, o sea el cable, requiere que el atacante tenga acceso físico a la red. En cambio el espionaje en redes inalámbricas no requiere esto, lo único que hace falta es un sabueso de tipo wireless, motivo por el cual es aconsejable utilizar *Wireless seriamblant privacy,* para encriptar el tráfico de las redes inalámbricas o algún otro que garantice la privacidad de la información.
Las amenazas contra la red de denegación de servicio, se deben a que se envían paquetes mas formados generalmente buscando vulnerabilidades en las pilas TCP/IP, por ejemplo: sobrecarga de buffer TCMP o los también conocidos TIN de la muerte. Paquetes con dirección de retorno falsa pueden ser utilizados para anular restricciones IP colocadas en un servidor de Base de datos.

Existen notorias herramientas dentro del mundo UNIX que exploran las vulnerabilidades en las pilas TCP/IP y falsifican inclusive las direcciones de origen, que es uno de los mecanismos de realizar este tipo de ataque.

AMENAZA CONTRA SERVIDORES

La sobrecarga de buffer permite que un atacante inyecte código malintencionado y lo ejecute en un servidor Web.

Varios ataques de denegación de permisos en listados por pantalla hacen que el IIS detenga su ejecución por completo y por lo tanto no contiene respondiendo a los requerimientos.

AMENAZA CONTRA APLICATIVOS

INYECCION SQL: Saca provecho de los aplicativos que concatenan entrada de datos extrema con comandos de base de datos. Los datos pueden provenir:

- Entrada desde campos <Forms>
- Entrada desde Query String

LA TECNICA

- Localizar el campo <form> o parámetros del Query String usado para generar comando SQL
- Enviar entrada que modifica los comandos

La técnica consiste en enviar la cadena de caracteres que cuando son concatenados a los comando SQL de la propia aplicación logran un resultado útil para el atacante como alterar o destruir datos.

La inyección SQL puede permitir la exposición de datos que debieron permanecer ocultos, corromper o inclusive destruir información.

Un uso para infringir esto es a través de formularios que utilizan login para la autenticación.

COMO FUNCIONA LA INYECCION SQL

Modelo de consulta

```
SELECT COUNT (*) FROM
Users
WHERE UserName='Jeff'
AND Password='imbatman'
```

Consulta mal-intencionada

```
SELECT COUNT (*) FROM Users
WHERE UserName='' or 1=1--
AND Password=''
```

"--" comenta el Resto de la consulta

"or 1=1" corresponde a todos los registros en la tabla

CROSS SITE SCRIPTING: Explora aplicativos que emiten salidas crudas, sin filtrar para páginas web:

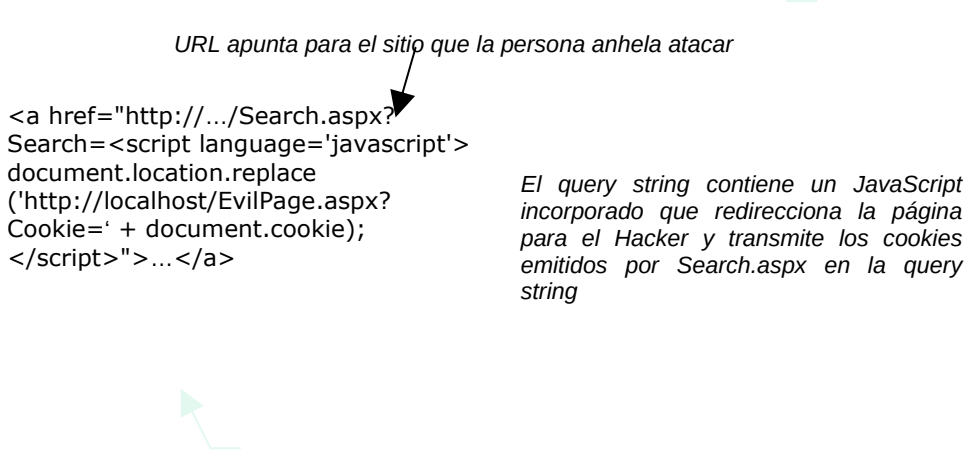
- Entradas desde campos <form>
- Entradas desde Query String

LA TECNICA

- Localizar un campo <form> o parámetros de Query String cuyo valor este siendo mostrado en la pagina web
- Insertar un script mal intencionado y hacer que un usuario desprevenido navegue hasta la pagina afectada. El script mal intencionado puede enviar la información redirigida a otra persona o navegar directamente hacia otra pagina que es comúnmente idéntica a la original pero en el servidor del atacante.

De esta forma el objetivo de cualquiera de los dos mecanismos es robar cookies y contraseñas, desfigurar y desactivar sitios, dado que en algunos casos la ejecución del script permite el acceso directo a la información especifica del usuario, seria factible que el script contenga toda la información contenida en por ejemplo todos los cookies de la maquina local.

COMO FUNCIONA EL CSS



Document.location.replace: transmite los cookies para el sitio del atacante ósea las redirige a la pagina maligna (Evilpage.aspx) dentro de un queryString.

Replace: analiza todo lo que viene en el queryString para obtener todos los cookies.

ALTERACION DE CAMPOS OCULTOS

HTTP es un protocolo sin estado

- No ofrece ninguna manera embutida de persistir datos entre consultas.

Personas son entidades con estado

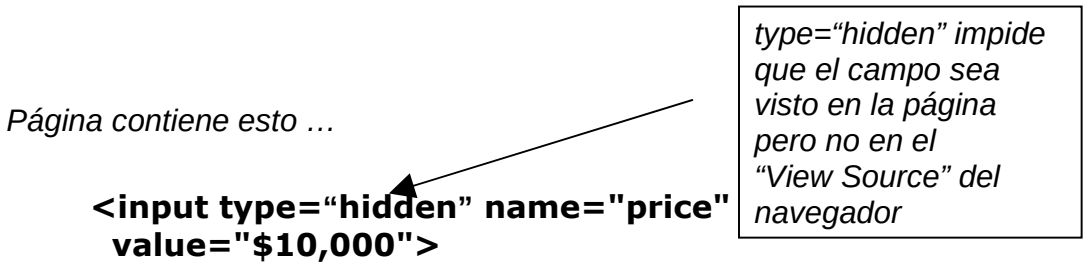
- Anhelan datos persistidos con estado
- Carrito de compra, preferencia de los usuarios, etc.

Los desarrolladores web a veces usan campos ocultos para persistir datos, y hay que tener en cuenta que los campos ocultos, no son realmente ocultos.

HTTP por ser un protocolo sin estado no ofrece ninguna forma de persistir datos entre diferentes llamadas de páginas. Sin embargo muchas veces las aplicaciones necesitan mantener este estado entre paginas navegadas, ejemplo necesita procesar carritos de compra, preferencias de usuario.

En muchas ocasiones los programadores web utilizan campos ocultos para hacer llamados a páginas. El problema es que estos campos ocultos o hider no son realmente ocultos, pueden ser fácilmente visualizados por el lado del cliente simplemente ejecutando un comando de hacer clic en el botón derecho sobre la página y acceder a la opción ver código fuente o ver source.

COMO FUNCIONA LOS CAMPOS OCULTOS



Datos de retorno debía contener esto …

price="$10,000"

En vez de eso, contienen esto …

price="$1"

Este es un ataque de violación de entrada, en el cual, el hacker puede alterar los precios de una compañía de a través de los campos ocultos, probando puede seleccionar el billete y alterar el precio de a un dollar (US $ 1).
Las cookies se utilizan para conocer la sesión específica de un usuario desde el lado del servidor, la cual se le denomina cookie de sesión.

Sin embargo estas cookies de sesión, pueden ser robadas por medio de participar en la escucha de las conexiones HTTP o de las conexiones de la versión CSS.
Cuando el estado de sesión ASPNET se configura de modo cookieles ASPNET incorpora el ID de la sesión dentro de la URL. Si alguien escucha esa URL en un mensaje de correo y la envía a otra persona, esta segunda persona podría compartir sesión de aquel que le envió el correo, siempre y cuando la sesión este activa.
Existe también la posibilidad de que los indicadores de sesión sean impredecibles, esto no ocurre generalmente en ASPNET porque utilizan una clase de tipo *RMG Cripto Server Provider* del NET framework para generar los ID de sesión de 120 bits que son altamente aleatorios.

Sin embargo las empresas pueden y a veces lo hacen sustituir esos ID de sesión por los suyos propios. De realizarse esa tarea hay que asegurarse que al menos los ID de sesión sean tan complejos como los que se generan en la clase *RMG Cripto Server Provider* para evitar la sustitución.

Teóricamente un atacante podría conectarse remotamente al servicio de estado de ASPNET a través de la puerta TCP/IP 42424 siempre y cuando esta puerta hubiese sido dejado abierta en el firewall, el atacante podría entonces llegar detrás del firewall, analizar el flujo de esta comunicación con el servicio de estado y obtener información de las diferentes sesiones. La conexión a la base de datos de estado de ASPNET, si es el caso de un repositorio de estado de Sql Server podría llevarse a cabo a través de la puerta TCP/IP 1433 y/o 1434.

SECUESTRO DE SESION

- Los Aplicativos Web usan sesiones para almacenar estado
- Las Sesiones son para uso exclusivo de usuarios individuales
- Las Sesiones pueden ser comprometidas

Tabla 2

Amenaza	Factor de Riesgo
Robo y reutilización (replay) de cookie identificador de sesión	**Alto***
Vínculos a sitios que usan estados de sesión sin cookies (cookieless = true)	**Medio***
Identificadores de sesión previsibles	**Bajo***
Escucha de conexiones con servidor de estado	**Medio**

ROBO DE IDENTIDAD

- La Seguridad depende de la autenticación
- Si la autenticación puede ser comprometida, la seguridad se perderá
- La Autenticación puede ser comprometida

Tabla 3

Amenaza	Factor de Riesgo
Robo de credenciales de autenticación Windows	**Alto**
Robo de credenciales de forms authentication	**Alto**
Robo y reutilización (replay) de cookies de autenticación	**Medio***
Ataques de diccionario y acertijo de señas	**Alto**

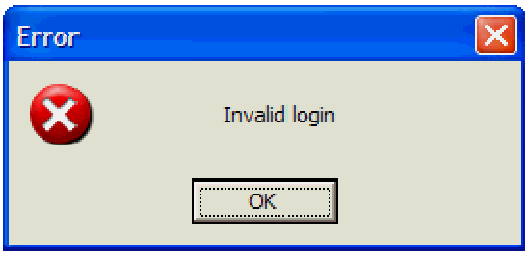
El robo de credenciales de autenticación Windows es un consumado por medio de espionaje en conexiones HTTP no criptografadas por ejemplo cuando se utiliza la autenticación Basic la cual pasa nombres de usuario y contraseñas en textos simples.

El robo de credenciales de Forms Autenticación puede ser logrado por medio de escucha en formularios de login que pueden ser accesibles a través de conexiones HTTP no criptografadas, por ejemplo las que no utilizan el protocolo HTTPS de transmisión segura.

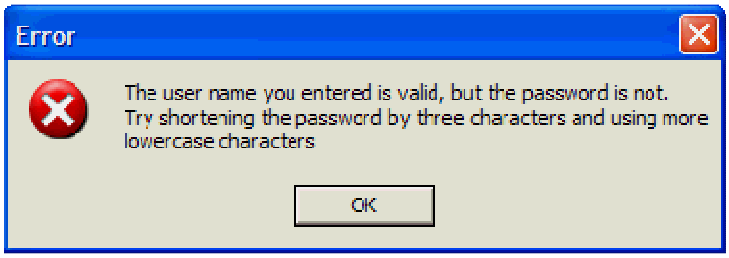
Es posible robar cookies de Forms Autenticación interceptando el tráfico HTTP no criptografado o con cross-sitio scripting. Limitar la vida útil de los cookies de forms authentication ayuda a reducir su utilidad a los atacantes.
La mejor defensa contra ataques de diccionario es reforzar políticas de contraseña altamente seguras, que impidan el uso de palabras poco comunes o familiares en las contraseñas.

REVELAR INFORMACION

¿Cual es el mejor mensaje de error?



El punto es que los mensajes de error con respecto a la seguridad deben ser lo mas genérico posibles. En la instalación de ASPNET y después de esta, hay que configurarlo adecuadamente, ya que ASPNET por defecto en los mensajes de error revela mucha información o aun cuando ocurra una excepción no tratada.



Este cuadro de dialogo revela información que no debería mostrar al confirmar que el usuario es valido y muestra información acerca de la forma en que se esta ingresando las contraseñas y dice que el usuario existe pero que la password es incorrecta, lo cual significa que el agresor va a continuar intentando respectando que la contraseña debe ser de 3 caracteres y que tiene que tiene que usar mayor cantidad de caracteres en minúscula.

MODELADO DE AMENAZAS

Es una forma de abordar estructuradamente la identificación, cuantificación y enfoque de las amenazas. Ayuda a colocar prioridad en el uso de los recursos de manera de lograr resultados mas seguros de acuerdo con lo que se esta estableciendo.

El modelado de amenazas es parte esencial del modelo de desarrollo así como la especificación, el proyecto, la codificación y las pruebas. La siguiente técnica es la más importante porque se conoce por su versatibilidad y capacidad de controlar agresiones dentro del desarrollo.

PROCESO DEL MODELADO DE AMENAZAS

1. Identificación de recursos

En la identificación de recursos es donde hacemos referencia a aquellas cosas que deseamos proteger, o son de uso vital para la aplicación.

¿Qué usted anhela proteger?

- Datos confidenciales (ej.: lista de clientes)
- Datos propietarios (ej.: propiedad intelectual)
- Datos pasibles de ataques (ej.: tarjeta de crédito, llaves de criptografía)

Éstos también cuentan como “recursos“

- Integridad de la base de datos
- Integridad de páginas Web (sin afeamiento)
- Integridad de otras máquinas en la red
- Disponibilidad del aplicativo

2. Documentar arquitectura

Realizar diagramas del aplicativo prestando mayor atención a cada subsistema, establecer los limites entre cada uno de ellos y los flujos de datos a establecer.

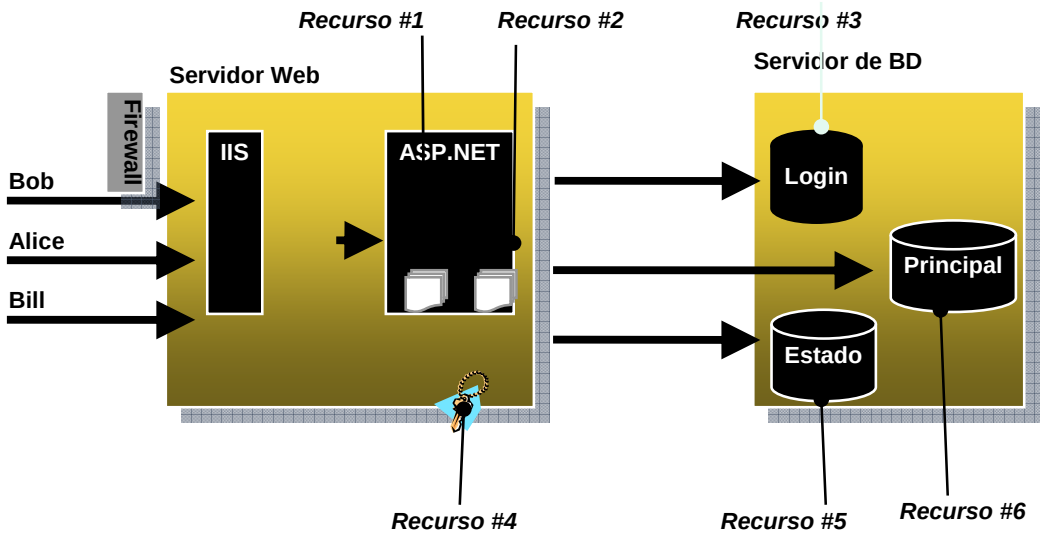
Definir lo que el aplicativo hace y como es usado:

- Usuarios ven páginas. con apartados de catálogo
- Usuarios hacen búsquedas en apartados de catálogo y agregan apartados a carrito de compra

Diagramar el aplicativo

- Exhibir subsistemas y flujo de datos
- Enlistar recursos

EJEMPLO



Recurso #1: Recurso de pagina publica, donde el acceso anónimo esta permitido.

Recurso #2: Páginas privadas que requieren autenticación de parte de los visitantes.

Recurso #3: Base de datos de identificación de usuario, que contiene el nombre de usuario y contraseña.

Recurso #4: Llaves de criptografía para encriptar la información.

Recurso #5: Base de datos de sesión de ASPNET

Recurso #6: Base de datos del aplicativo en si mismo

3. Descomponer aplicativo

Crear un perfil de seguridad para ayudar a identificar las vulnerabilidades del aplicativo.

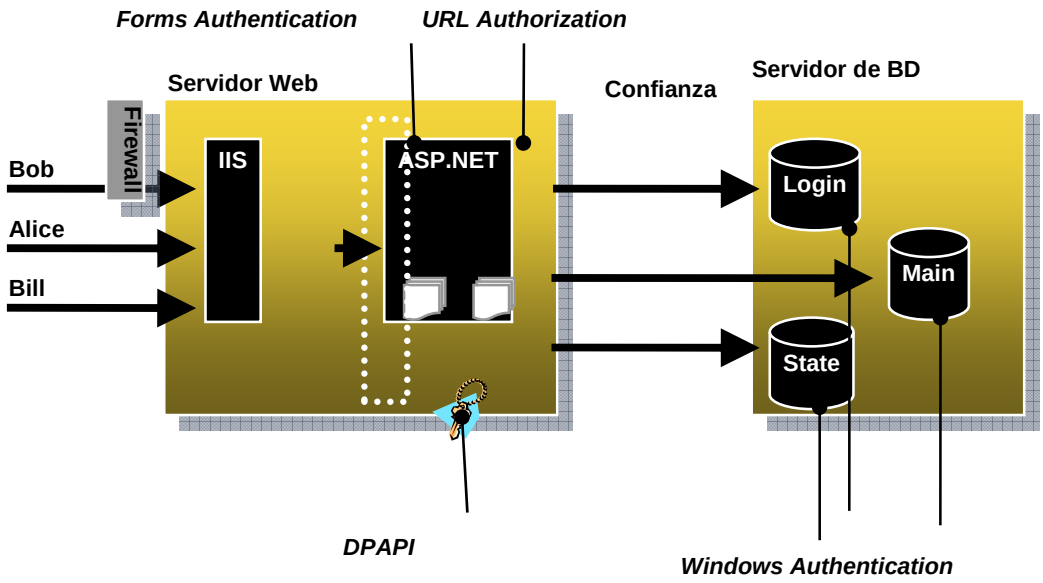
Refinar el diagrama de la arquitectura

- Exhibir mecanismos de autenticación
- Exhibir mecanismos de autorización
- Exhibir tecnologías (ej.: DPAPI)
- Diagramar límites de confianza
- Identificar puntos de entrada

Empezar a pensar como un atacante

- ¿Dónde están mis vulnerabilidades?
- ¿Qué voy a hacer respeto de ellas?

EJEMPLO



En el ejemplo anterior se utiliza autenticación basada en formulario y autorización basada en URL para definir cuales son las reglas de acceso a la aplicación. Utilizara autenticación integrada de Windows para identificarse ante la base de datos. A demás se utilizara la API DPAPI para proteger las claves criptográficas.

El límite de confianza abarca ASPNET como el servidor de base de datos, porque el servidor de base de datos confía plenamente en el proceso de ASPNET, ya que este será el responsable de autenticar al cliente.

4. Identificar amenazas

Método #1: Lista de amenazas

- Comience con una lista de posibles amenazas
- Identifique las amenazas pertinentes a su aplicativo

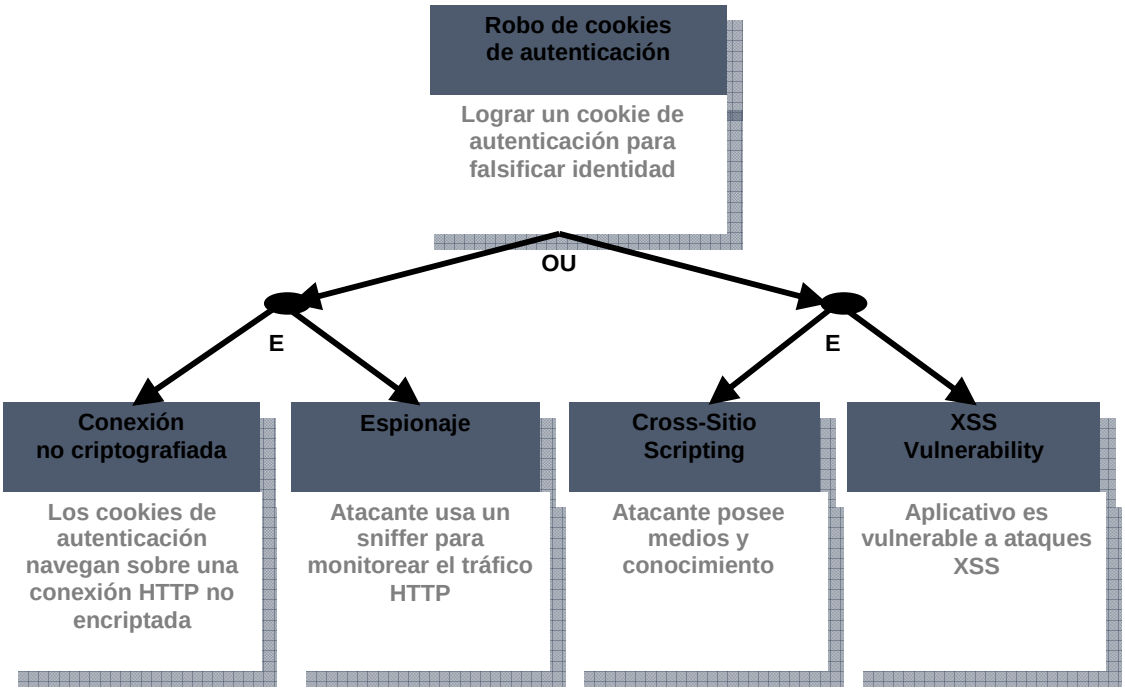
Método #2: STRIDE

- Categorice la lista de los tipos de amenazas (red, servidor o aplicación)
- Identifique las amenazas por tipo/categoría

Opcionalmente, dibuje árboles de amenazas

- Nudo raíz representa las metas del atacante
- Árboles ayudan a identificar condiciones de amenaza, ayudan a recorrer cada una de sus ramas en forma descendente, lo cual nos permitirá identificar en que puntos aun no hemos realizado nada para contener la posibilidad de un ataque.

Árboles de amenazas



Este es un ejemplo de un árbol de amenaza simple, en la vida real existe un sin fin de numero de árboles de amenaza y a veces mas complejos que este, sin embargo es importante seguir, cada uno de los mecanismos de detener el ataque en cada uno de los nodos de cruce.

Stride

Este método se denomina así basándose en las iniciales en ingles de las distintas acciones que hay que realizar.

Spoofing (Falseamiento)
¿Un atacante puede lograr acceso con una falsa identidad?

Tampering (Violación)
¿Un atacante puede modificar datos durante el trayecto de éstos en el aplicativo?

Repudiation (Repudio)
¿Si el atacante niega una acción, puede usted probar qué él está engañando?

Information disclosure (Revelación)
¿Un atacante puede tener acceso a datos confidenciales o peligrosos?

Denial of service (Negación de servicio)
¿Un atacante puede interrumpir o reducir la disponibilidad del sistema?

Elevation of privilege (Elevación de privilegios)
¿Un atacante puede asumir la identidad de un usuario privilegiado?

5. Documentar amenazas

Este método utiliza cuadros para definir defensas de una correcta documentación de las posibles amenazas.

Tabla 4

Robo de Cookies de Autenticación por escucha de la conexión	
Blanco amenaza	Conexiones entre navegadores y servidor Web
Riesgo	
Técnicas ataque	Atacante usa husmeador (sniffer) para monitorizar tráfico
Medidas defensivas	Usar SSL/TLS para encriptar el tráfico

Tabla 5

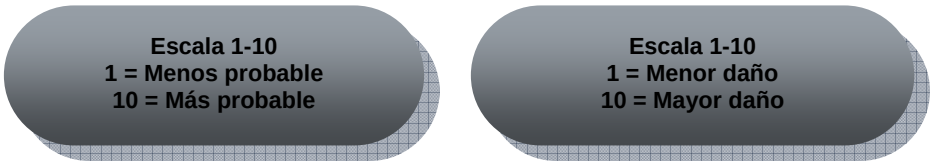
Robo de Cookies de Autenticación vía a Cross-Site Scripting	
Blanco de la amenaza	Código de aplicativo vulnerable
Riesgo	
Técnicas de ataque	Atacante envía a los usuarios e-mail con links maliciosos
Medidas defensivas	Validar entrada; usar HTML Encode antes de presentarla como salida

6. Clasificar las amenazas

¿Cuáles de estas amenazas tienen un potencial mayor para causar un mayor daño? Este método permitirá establecer mecanismo de protección los más temprano posibles en el desarrollo del aplicativo.

MODELO SIMPLE

Vulnerabilidad = Probabilidad * Potencial de daño



Modelo DREAD

- Mayor granularidad del potencial de amenaza
- Clasifica (prioriza) cada amenaza en una escala de 1 a 15

DREAD

Este método se denomina así basándose en las iniciales en ingles de las distintas acciones que hay que realizar.

D **Damage potential (Potencial de daño)**
¿Cuáles son las consecuencias de una exploración exitosa?

R **Reproducibility (Capacidad de reproducción)**
¿Una exploración puede funcionar siempre o sólo bajo ciertas circunstancias?

E **Exploitability (Capacidad de exploración)**
Lo experimentado que necesita ser el atacante para explorar la vulnerabilidad

A **Affected users (Usuarios afectados)**
¿Cuántos usuarios serían afectados por una exploración bien devenida?

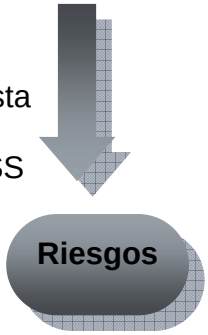
D **Discoverability (Capacidad de ser descubierto)**
¿Cuál es la probabilidad de un atacante saber de la existencia de la vulnerabilidad?

Dentro del modelo DREAD cada una de las amenazas recibe una clasificación de 1 a 3 según la categoría, cuanto mayor sea el numero mayor será la amenaza. El número final resultante de la sumatoria de todos estos valores es lo que establece el nivel de riesgo de dicha amenaza.

EJEMPLO

Tabla 6

Amenaza	D	R	E	A	D	Suma
Robo de cookie de autent. (escucha)	3	2	3	2	3	13
Robo de cookie de autent. (XSS)	3	2	2	2	3	12



D. Potencial de daño es alto (Identidades falsificadas, etc.)
R. Cookie puede ser robado a cualquier hora, pero solo será útil hasta expirar
E. Quien quiera puede ejecutar un packete sniffer; ataques XSS requieren experiencia comedida
A. Todos los usuarios podrán ser afectados, pero en la realidad la mayoría no entrará en vínculos maliciosos
D. Fácil de descubrir: apenas digite <script> block en un campo de entrada

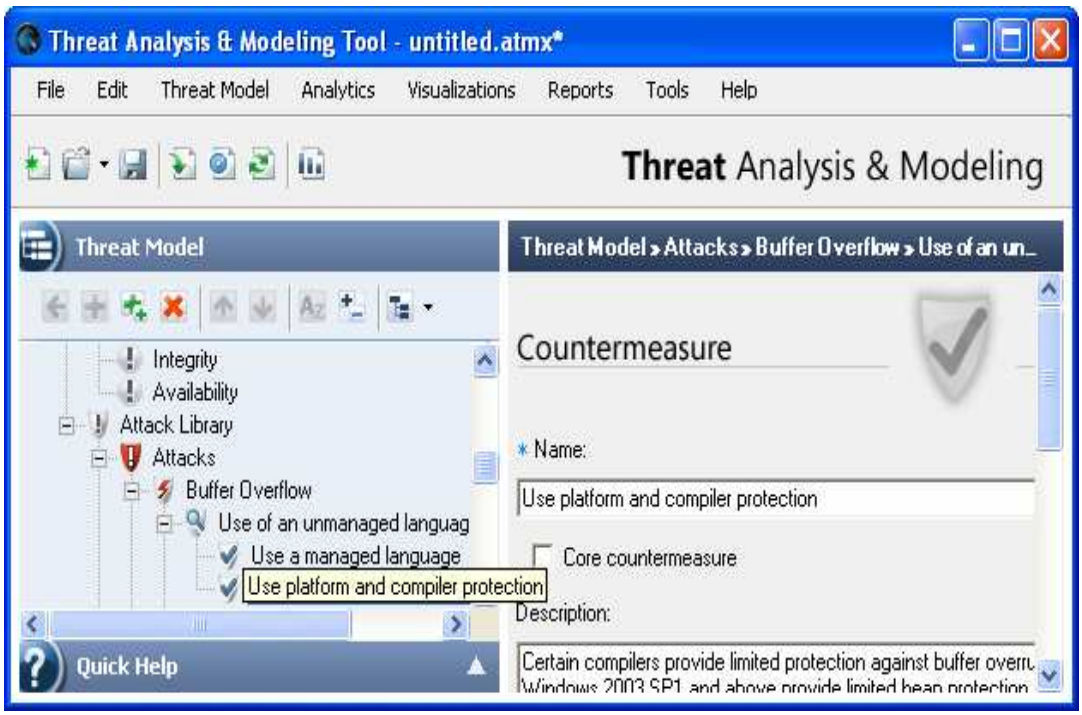
Tabla 7

	Alto (3)	Medio (2)	Bajo (1)
Potencial de daño	Atacante puede recobrar datos altamente confidenciales, comprometer servidores detrás del Firewall o corromper o destruir datos	Atacante puede recobrar datos confidenciales pero no puede hacer muchas otras cosas	Atacante solo puede recobrar datos que tengan poco o ninguno potencial de daño
Capacidad de reproducción	Funciona siempre; no requiere una ventana de tiempo	Dependiente de tiempo; funciona solamente en una ventana de tiempo especificada	Raramente funciona
Capacidad de exploración	Hasta Bart Simpson podría hacer eso	Atacante necesita tener conocimiento y experiencia	Atacante necesita de MUCHO conocimiento y experiencia
Usuarios afectados	Mayoría o todos los usuarios	Algunos usuarios	Pocos, si algún usuario
Capacidad de hallazgo	Atacante puede fácilmente descubrir la vulnerabilidad	Atacante puede descubrir la vulnerabilidad	Atacante necesitará “ahondar” para descubrir la vulnerabilidad

HERRAMIENTA PARA MODELADO DE AMENAZA

Threat Analisys & Modeling

Microsoft publico una herramienta para el modelado de amenazas, que es muy útil para poder evaluar la situación de una aplicación y ayudar a utilizar e implementar manteniendo ACLs de seguridad dentro de todo el proceso de desarrollo.
Es interesante e importante tener todas las herramientas posibles para poder llegar a un buen fin en el desarrollo de aplicaciones seguras.



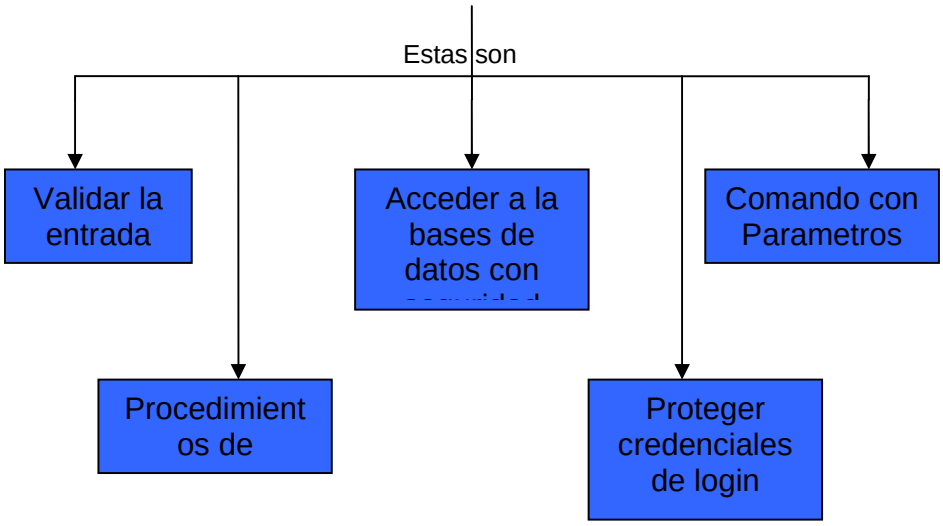
RECOMENDACIÓNES

Modelar la etapa de desarrollo para resistir amenazas y construir código seguro. Defender correctamente ante los tres tipos de ataques: contra la red, el servidor y la aplicación.

Editar los mensajes de error con respecto a la seguridad, estos deben ser lo mas genérico posibles.

Realizar un modelado de amenazas ya que esta es una forma estructurada de identificación, cuantificación y enfoque de los ataques.

MECANISMOS Y MEDIDAS DE DEFENSA



CAPITULO V

MECANISMO Y MEDIDAS DE DEFENSA

Abordar integralmente la seguridad siempre debe considerar proteger la red, servidor, y los aplicativos, ocuparse de uno solo de los puntos, generara que siempre tengamos un tipo de problema de seguridad en cualquiera de los otros dos.

PROTEGER LA RED

Para tener la red segura debemos tener en cuanta los siguientes elementos:

- Fortalecer firewalls
- Estar actualizado con parches y actualizaciones
- Bloquear puertas y protocolos no utilizados
- Usar una filtro para evitar solicitaciones ilícitas, este funciona en varios niveles: filtros de paquetes, filtros aplicativos, de servidor, etc…
- Fortalecer routers y switches
- Usar filtros de entrada/salida para detectar paquetes falsos
- Detectar tráfico ICMP desde la red interna
- Detectar solicitaciones de broadcast dirigidas desde la red interna, esto hace que sea mas difícil para los atacantes enumerar servidores en la red
- Evitar consultas de trace routing

Encriptar comunicaciones confidenciales.

También es buena practica proteger las interfaces administrativas en el hardware de la red, ejemplo de esto incluye la alteración de calidad que son a veces exploradas por los atacantes, uso de contraseñas complejas ósea contraseñas fuertes de ataques de diccionario y desactivación de interfaces de administración en puertas que terminaran siendo destinadas hacia Internet.

PROTEGIENDO EL SERVIDOR

Mantener el servidor seguro implica que debemos tener en cuanta los siguientes elementos:

- Estar actualizado con service packs y actualizaciones
- Fortalecer el IIS con IISLockdown, URLScan y PortReporter
- Fortalecer la pila TCP/IP del Servidor Web
- Ejecutar ASP.NET usando el principio de privilegio reducido, es decir con confianza parcial y no como system.
- Usar ACLs para limitar el acceso a recursos importantes, lo cual utiliza permisos NTFS para restringir el acceso a directorios ASPNET y a directorios del sistema.
- Desactivar recursos compartidos y servicios no utilizados, ayuda a reducir la superficie de ataque, por ejemplo desactivar los servicios SMTP impide que los atacantes utilicen paquetes mal formados para desactivar un servidor web.

Es importante tener en cuenta que en ocasiones agresiones de segundo nivel se presentan, porque se deja para un segundo tiempo la instalación de las últimas actualizaciones y esto favorece al ataque de los hackers.

IISLOCKDOWN

Herramienta pasiva, es decir es ejecutada una vez, y no se ejecuta todo el tiempo, realiza las siguientes tareas:

- Desactivar servidores FTP, SMTP y NNTP
- Eliminar mapeo de ciertos tipos de documentos, como .idq, .htr y .printer
- Eliminar IISSamples, IISHelp, Scripts y otros directorios virtuales
- Proteger con listas de acceso herramientas del sistema y directorios de contenido de la Web para limitar el acceso
- Instalar el utilitario URLScan

Si tiene instalada la versión 5 o inferior de Internet Information Server, no olvidar utilizar esta herramienta, ya que será de gran ayuda para detectar y prevenir futuros ataques o vulnerabilidades en el aplicativo. Por el contrario si posee la versión 6 de

Internet Information Server, no instale esta herramienta, ya que no será necesaria, ya que la versión 6 de Internet Information Server ya viene con la mayoría de servicios y opciones desactivada en la instalación.

URLSCAN

Herramienta activa, es decir es ejecutada todo el tiempo, como si fuera un servicio, realiza las siguientes tareas:

- Registra las solicitudes con fallo
- Limita los tamaños de las solicitudes para minimizar ataques de negación de servicio (DoS)
- Enmascara las cabeceras del contenido que revelan el tipo y versión del IIS
- Bloquea solicitudes que contiene caracteres potencialmente peligrosos
- Desactiva verbos específicos (por ejemplo, "DEBUG")

Instalar y configurar esta herramienta es importante, ya que la gran mayoría de las opciones que esta protege no vienen desactivadas en ninguna de las versiones de Internet Information Server.

PORTREPORTER

Herramienta activa, es decir es ejecutada todo el tiempo, como si fuera un servicio, realiza las siguientes tareas:

- Puertos usadas
- El proceso que usa el puerto
- Si el proceso es un servicio
- Los módulos (principalmente DLLs) cargados por el proceso
- La cuenta de usuario bajo el cual el proceso se ejecuta
- Genera archivos de LOG
- PRParser ayuda a interpretar LOGs, existe también una herramienta que ayuda a leer e interpretar estas bitácoras (logs).

DEFENDER EL APLICATIVO

Mantener el aplicativo seguro implica que debemos tener en cuanta los siguientes elementos:

- Nunca confiar en entradas de usuario (Hay que validarlas)
- Acceder a base de datos con seguridad
- Evitar vulnerabilidades en Autenticación por Formularios
- Proteger el estado de sesión del ASP.NET
- Almacenar datos confidenciales con seguridad
- Impedir errores y trátelos apropiadamente

VALIDAR LA ENTRADA

Filtrar las entradas para prevenir ataques del tipo de Sql Inyection o Cross Site Scripting.

- Filtrar caracteres y strings potencialmente peligrosos
- Usar HTML-encode para salidas en páginas Web, utilizar Encode antes de enviar la página suministra protección adicional contra ataques CSS.
- Usar codificaciones de caracteres "seguras", reduce la posibilidad de la utilización de Unicode y codificaciones multi bytes, para disfrazar caracteres peligrosos, esto es conocido como *Canonización*, que son representaciones diferentes para un mismo carácter.

```
<globalization requestEncoding="ISO-8859-1"
  responseEncoding="ISO-8859-1" />
```

- Si es posible, evitar uso de nombres de archivos como entrada

La entrada es conveniente validarla en varios niveles, y en cada etapa de un aplicativo multi capas, los ingresos deben ser validados nuevamente en función de la seguridad.

HERRAMIENTAS PARA LA VALIDACION DE ENTRADA

Existen varias rutinas que ayudan a la validación de entrada, estas son:

Tabla 8

Herramienta	Descripción
Regex	Clase, en el namespace, System.Text.RegularExpressions que implementa el mecanismo de expresiones regulares del .NET Framework
Controles de validación	Conjunto de seis controles que validan la entrada en el cliente y en el servidor: RequiredFieldValidator, RegularExpressionValidator, RangeValidator, etc.
HttpUtility.HtmlEncode	Codifica la entrada, convirtiendo caracteres potencialmente peligrosos, como "<“, en secuencias de escape inofensivas.
Validación de solicitudes de ASP.NET 1.1	Recurso de ASP.NET 1.1 que rechaza automáticamente solicitudes que contienen determinados caracteres y secuencia de caracteres (por ejemplo, "<script").

ACCEDIENDO A BASE DE DATOS CON SEGURIDAD

Realizar acceso seguro a base de datos significa que en la programación se denoto mal algunas providencias, por ejemplo:

- Use Procedimientos almacenados (stored procedures) o comandos parametrizados en vez de comandos SQL dinámicos
- Nunca use **sa** para acceder Base de datos Web
- Almacenar strings de conexión con seguridad
- Aplicar protecciones administrativas al SQL Server
- Usar opcionalmente SSL/TLS o IPSec para proteger la conexión con el servidor de base de datos

Crear comando para acceder a datos en Sql Server utilizando datos ingresados por el usuario, eso es crear comando sql dinámicamente es realmente muy peligroso, dado que facilita el ataque de inyection sql. Consultas como la del siguiente ejemplo a continuación debe ser utilizada para validar nombres de usuario, credenciales, contraseñas, en login de bases de datos, un valor de retorno cero indica que no hay un registro correspondiente, o sea nombre de usuario, la contraseña o ambos son inválidos.

```
// Peligro! La entrada de usuario está siendo usada
// directamente para generar la consulta a la base
de dados
string sql = String.Format ("select count (*) " +
    "from users where username=\'{0}\' and cast " +
    "(password as varbinary)=cast (\'{1}\' as " +
    varbinary)", username, password);
SqlCommand command = new SqlCommand (sql,
connection);
int count = (int) command.ExecuteScalar ();
```

COMANDOS CON PARÁMETROS

Los comando con parámetros son utilizados para realizar consultas parametrizadas, ya que estas son menos vulnerables.

- Menos vulnerable a ataques de inyección de SQL

El siguiente ejemplo utiliza una consulta parametrizada que es menos vulnerable a la inyección de sql. En las ultimas dos líneas de código se crean objetos sql parameters, esta es una protección adicional contra entradas mal intencionadas, dado, que estos objetos controlan en mucha mejor medida, los datos que se están ingresando y los asignan dentro de la cadena de consulta original sin la posibilidad de que haya encadenamiento de otros tipos de valores.

```
// La entrada es utilizada en una consulta
parametrizada

SqlCommand command = new SqlCommand
    ("select count (*) from users where " +
    "username=@username and cast (password as " +
    "varbinary)=cast (@password as varbinary)",
     connection);
command.Parameters.Add ("@username",
    SqlDbType.VarChar).Value = username;
command.Parameters.Add ("@password",
    SqlDbType.VarChar).Value = password;
int count = (int) command.ExecuteScalar ();
```

PROCEDIMIENTO ALMACENADOS (STORED PROCEDURES)

Los procedimientos almacenados son procesos que es encuentran en el servidor de base de datos y que pueden ser ejecutados desde la aplicación.

- Menos vulnerable a ataques de inyección de SQL
- Agrega seguridad por medio de permisos de EJECUTE

La utilización de procedimientos almacenados simplemente impide la ejecución de un comando sql distinto de lo que realmente se desea. Una vez creado el usuario apenas recibe los permisos para ejecutar los procedimientos almacenados, si el usuario puede ejecutar procedimientos almacenados de la base de datos, pero no puede ejecutar comando de tipo SELECT, INSERT, UPDATE o DELETE, la inyección sql será difícil por no decir realmente imposible.

```
// Parámetros pasados para un procedimiento almacenado
SqlCommand command =
    new SqlCommand ("proc_IsUserValid", connection);
command.CommandType = CommandType.StoredProcedure;
command.Parameters.Add ("@username",
    SqlDbType.VarChar).Value = username;
command.Parameters.Add ("@password",
    SqlDbType.VarChar).Value = password;
command.Parameters.Add ("@return", SqlDbType.Int).
    Direction = ParameterDirection.ReturnValue;
int count = (int) command.ExecuteScalar ();
```

LA CUENTA SA

ES la cuenta por defecto administrador de Sql Server, sólo para administración; nunca usarla para acceder base de datos programáticamente:

- En vez de eso, use una o más cuentas que tienen permisos limitados a la base de datos
- Para consultas, use una cuenta con derecho SELECT

Mejor aún, usar procedimientos almacenados y conceder a la cuenta el permiso EJECUTE.

Hay que hacer más difícil para el invasor ejecutar comandos perjudiciales (por ejemplo, DROP TABLE)

AUTENTICACIÓN EN SQL SERVER

Microsoft SQL Server da soporte a dos tipos de autenticación:

- Autenticación con logins de SQL Server
- Autenticación con identidades de Windows

La autenticación Windows reduce la superficie amenazada, retirando nombres de usuario y contraseñas del strings de conexión, ya que estos no hacen parte de la cadena de conexión, por lo tanto siempre es preferible utilizar esta autenticación dentro de las aplicaciones de tipo web.

El siguiente código a continuación es un ejemplo de la cadena de conexión o Connetion String dentro de las aplicaciones de tipo web.

```
server=localhost;database=pubs;Trusted_Connection=yes
```

¿QUIÉN DEBE SER AUTENTICADO?

Se pueden autenticar diversas identidades para un aplicativo.

Tabla 9

Identidad	Pros	Contras
Proceso de trabajo (Worker process)	* Pool de conexiones * Ausencia de problemas con “one-hop”	La cuenta ASPNET queda desprovista de credenciales de red Todos los clientes comparten un conjunto de permisos de base de datos
Identidad personificada (impersonated)	* Políticas de la empresa a veces requieren auditoria en el nivel de usuario para accesos a base de datos * Aumento de granularidad de permisos de base de datos	▪ Impide pooling de conexión ▪ Aumenta problemas con “one-hop” ▪ Agrega administración ▪ Concede acceso a base de datos a más cuentas
Identidad COM+	* Pooling de conexión * Ausencia de problemas con “one-hop”	▪ Disminución de rendimiento ▪ Aumento de complejidad ▪ Clientes comparten un conjunto de permisos de base de datos

AUTENTICACION DEL PROCESO DE TRABAJO DE ASPNET

Existirá la ventaja de requerir y concebir acceso a recursos como: Base de datos, etc... A una sola cuenta, simplificando la administración y la posibilidad de conceder privilegio por accidente. Sin embargo como ASPNET en Internet Information Server 5 es una cuenta local la autenticación remota puede ser hecha configurando una cuenta idéntica (mismo nombre de usuario, idéntica contraseña, iguales privilegios o análogos) en el servidor remoto, lo cual es un problema de seguridad.

AUTENTICACIÓN POR FORMULARIOS

En la autenticación por formularios debemos seguir las siguientes claves:

- Proteger credenciales de login con SSL/TLS
- No almacenar claves directamente; almacene un hash
- Limitar vida útil de los cookies de autenticación para evitar la apertura a ataques de repetición
- Presumir que cookies de autenticación son falsos o robados al ejecutar operaciones sensibles
- No confiar en Autenticación por Formularios para proteger recursos no pertenecientes a ASP.NET

Partir del hecho que las cookies son falsas, pedir a los usuarios que asocien su login nuevamente antes de ejecutar operaciones altamente peligrosas, tales como: efectuar una compra con tarjeta de crédito, aun cuando el pedido ya contenga una cookie de autenticación que parezca valida.

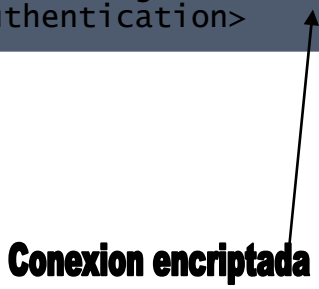
La autenticación por formularios no protege recursos que no pertenezcan a ASPNET, por ejemplo archivos de tipo HTML, JPG, JPEG, etc... Ni otros archivos que no estén asignados al ASPNETISAPI.DLL en la metabase de Internet Information Server. La solución es asignarlos para que la clase ASPNETISAPI.DLL los ejecute en la metabase de Internet Information Server.

PROTEGIENDO CREDENCIALES DE LOGIN

Enviar nombres de usuario y contraseña en formato de texto simple en una conexión que no este encriptada deja vulnerabilidades en la interceptación de la comunicación.

Coloque los formularios de login en directorios protegidos por SSL/TLS para resguardarlos de análisis

```
<authentication mode="Forms">
  <forms loginUrl="https://.../LoginPage.aspx" />
</authentication>
```



COOKIES DE AUTENTICACIÓN POR FORMULARIO

Las cookies de autenticación por formulario son encriptados de forma predeterminada, esto impide "Lectura y alteración" pero no impide "robo y repetición". La mejor protección es usar SSL/TLS, pero también se puede limitar la vida útil del cookie y desactivar la opción *sliding renewal*, con esto el usuario perderá el login después de un tiempo absoluto desde el login inicial, aun cuando este navegando directamente en el sitio.

Tabla 10

Defensa	Comentarios
Solo transmite cookies en SSL	Impide robo de cookies (defensa más fuerte)
Limita vida útil del cookie	Limita ataques de repetición al tiempo válido del cookie
Desactiva renovación periódica (*sliding renewal*)	También limita ataques de repetición al tiempo en el que cookie permanecer válido

TRATAMIENTO DE ERRORES

Los errores y excepciones son importantes tratarlaslos:

- Anticipar errores y tratarlos inteligentemente
- Usar <customErrors> para exhibir páginas de error personalizadas
- No revelar mucha información en páginas de error
- Hacer log de excepciones no tratadas
- Ser agresivo al hacer log de fallas

EXCEPCIONES NO TRATADAS

Las excepciones no manejadas en la aplicación, también son un tópico importante que no se puede descuidar, porque estás podrían abrir una brecha en nuestra aplicación.

El evento ApplicationError de Global.asax será automáticamente llamado por el ASPNET, si se genera una excepción no administrada. Cuando el ApplicationError es llamado, en realidad es tarde, pero no para que el administrador tenga conocimiento de la excepción. La presencia de muchas excepciones no tratadas significa que el aplicativo no esta bien hecho.

El siguiente ejemplo es sobre el método ApplicationError que debiera estar en forma predeterminada en cualquier aplicativo ASPNET. En este ejemplo se requiere que el aplicativo tenga permiso para grabar en el log de Windows, en forma predeterminada los aplicativos ASPNET carecen de privilegios suficientes para crear nuevas categorías en el log de eventos, de este modo el siguiente ejemplo presume que la categoría myaspnetapplication fue creada en forma anticipada, y esto si es permitido para el aplicativo escribir en el log de eventos.

```
void Application_Error (Object sender, EventArgs e)
{
    // Formulate message to write to event log
    string msg = "Error accessing " + Request.Path +
        "\n" + Server.GetLastError ().ToString ();
    // Write the message to Windows event log
    EventLog log = new EventLog ();
    log.Source = "My ASP.NET Application";
    log.WriteEntry (msg, EventLogEntryType.Error);
}
```

RECOMENDACIONES

Identificar que tipo de herramienta de seguridad debe ser instalada, teniendo en cuenta los requerimientos necesarios.

Recordar defender el aplicativo y validar la entrada seleccionando la herramienta adecuada.

Realizar acceso seguro a base de datos.

GLOSARIO

Saturación de Buffer: la saturación del búfer se produce cuando los datos proporcionados por el intruso tienen un tamaño mayor que el que espera la aplicación y, consecuentemente, se desbordan en el espacio de la memoria interna. Las saturaciones del búfer son, fundamentalmente, un problema relacionado con C/C++. Constituyen una amenaza, pero generalmente tienen fácil solución. El desbordamiento daña otras estructuras de datos en la memoria y estos daños pueden permitir que el intruso ejecute código malintencionado. También se pueden producir saturaciones y subdesbordamientos del búfer debido a errores en la indización de matrices, aunque son menos frecuentes.

Internet Information Services (o Server): IIS, es una serie de servicios para los computadores que funcionan con Windows. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Este servicio convierte a un computador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

Código Manejado: Cuando trabajamos con este tipo de código, lo que se genera al compilar el proyecto no es código nativo entendible por el hardware y el sistema operativo del equipo, sino que es un código que es entendible por un aplicativo intermedio entre nuestro programa y el hardware, llamado Máquina Virtual. Esta máquina virtual interpreta el código manejado y lo convierte en tiempo real (Just in Time) a código nativo subordinado al sistema y hardware en que se encuentra.

Assemblies: Es una librería dinámica (DLL) o programa ejecutable en la cual pueden existir distintos espacios de nombres. Los ensamblados componen la unidad fundamental de implementación, control de versiones, reutilización, ámbito de activación y permisos de seguridad en una aplicación basada en .NET.

FrameWork: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

STRCPY: Función de C y C++, copia una cadena y retorna la dirección de la cadena destino.

Listas de Control de Acceso (ACL): Es una tabla que le dice a un sistema los derechos de acceso que cada usuario posee para un objeto determinado, como directorios, ficheros, puertos, etc. Técnicas para limitar el acceso a los recursos según la información de autenticidad y las normas de acceso.

Global Assembly Cache: es un almacén del equipo que se utiliza para contener ensamblados que van a compartir varias aplicaciones en el equipo. .NET Framework proporciona dos herramientas para trabajar con la caché. Una de ellas es una extensión Shell de Windows que permite trabajar con la caché mediante una Interfaz gráfica de usuario (Graphical User Interface, GUI). La otra es una herramienta de línea de comandos denominada Global Assembly (Gacutil.exe), que normalmente se utiliza en secuencias de comandos de generación y de prueba.

HTML Encode**:** Se encarga de limpiar las entradas, para luego mostrarlas en pantalla. Html Encode trata el código solamente como texto en formato simple y no como java script por ejemplo, que es el objetivo que perciben los hackers, no permite que el código se ejecute en el cliente o en la página web.

BIBLIOGRAFIA

- Daniel Alfredo Seara, Revista MSDN abril de 2006
- Mauro Sant'Anna, Revista MSDN abril de 2006
- Hector Minaya, Revista MSDN abril de 2006
- Mundie, Craig, Notas del producto de cómputo confiable
- Howard, Michael, Superficie de ataque: Mitigue los riesgos de seguridad al minimizar el código que expone para los usuarios no confiables, Revista MSDN noviembre de 2004
- Howard, Michael, Consejos expertos para encontrar defectos de la seguridad en su código, Revista MSDN, noviembre 2003
- Howard, Michael y David LeBlanc, Escribir código seguro, Segunda Edición, Microsoft Press, Redmond, Washington, 2003
- Swiderski, Frank y Window Snyder, Modelo contra amenazas, Microsoft Press, Redmond Washington, 2004
- **http://msdn2.microsoft.com/en-us/library/cb6t8dtz.aspx**
- **http://www.elguille.info/NET/vs2005/clickonce_pub/ClickOnce.aspx**
- **https://www.microsoft.com/spanish/msdn/articulos/archivo/070205/voices/realworld12012004.asp**
- http://www.microsoft.com/spanish/msdn/articulos/archivo/030505/voices/sdl.mspx
- http://www.microsoft.com/spain/technet/seguridad/default.mspx, boletines de seguridad de Microsoft TechNet
- http://msdn.microsoft.com/security/securecode/threatmodeling/acetm/
- http://msdn.microsoft.com/security/
- http://www.microsoft.com/spanish/msdn/latam/netpro
- http://blogs.msdn.com/fxcop/archive/2006/03/11/549611.aspx
- http://www.gotdotnet.com/Team/FxCop/
- http://msdn.microsoft.com/library/default.asp?url=/library/enus/wcepbguide5/html/wce50conprefastoverview.asp

- http://www.microsoft.com/technet/desktopdeployment/appcompat/toolkit.mspx
- http://msdn.microsoft.com/library/default.asp?url=/library/enus/vccore/html/vclrfGSBufferSecurity.asp
- http://msdn.microsoft.com/library/default.asp?url=/library/enus/dncode/html/secure06112002.asp