UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS
ESCUELA DE INGENIERÍA INFORMÁTICA

PRÁCTICA PROFESIONAL DIRIGIDA

Para optar por el grado de Bachillerato en Sistemas de Información

Desarrollo de un prototipo funcional de un plugin para Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones (SAST)

Michael Hidalgo Fallas

AUTOR

Ing. Leonardo Delgado Arroyo, MAP

TUTOR

LECTOR

San José, Costa Rica

Octubre, 2014

TABLA DE CONTENIDOS

Michael Hidalgo 11/1/2014 5:25 PM
Eliminado: 115

INDICE DE CUADROS

INDICE DE GRÁFICOS

INDICE DE FIGURAS

TRIBUNAL EXAMINADOR

Esta Práctica Profesional fue aprobada por el Tribunal Examinador de la Carrera de Ingeniería en Sistemas de Información de la UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS, como requisito para optar por el Grado de Bachillerato.

_____ _____

Máster Olda Bustillos Ortega — Directora Escuela de Ingeniería Informática

Ing. Leonardo Delgado Arroyo, MAP — Tutor

_____

Lector

CARTA DEL TUTOR

San José, 11 de Diciembre del 2014

Máster Olda Bustillos Ortega
Directora de la Escuela de Ingeniería Informática
Universidad Internacional de las Américas

Estimada Directora:

Sirva la presente para saludarla y hacer de su conocimiento mi aprobación, en calidad de Tutor del trabajo realizado por el estudiante Michael Hidalgo Fallas, portador de la cédula número 1-12750522, en su Informe Final de Graduación titulado: Prototipo funcional de un plugin para Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones (SAST). Hago constar que se han revisado y corregido todos los aspectos referentes a este documento, por lo que manifiesto que se encuentra listo para ser presentado a la Universidad Internacional de las Américas como Informe Final de Graduación.

Atentamente,

_____

Ing. Leonardo Delgado Arroyo, MAP
Tutor

CARTA DEL FILÓLOGO

San José, 11 de Diciembre del 2014

Máster Olda Bustillos Ortega
Directora de la Escuela de Ingeniería Informática
Universidad Internacional de las Américas
S. O.

Estimada Directora:

Hago de su conocimiento que he recibido del estudiante Michael Hidalgo Fallas, cédula número 1-12750522, el Informe Final de Graduación**,** que lleva por título Prototipo funcional de un plugin para Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones (SAST), para su corrección filológica.

He procedido a revisar los aspectos de forma, redacción, estilo y otros vicios del lenguaje que se pudieron trasladar al texto.

Una vez comprobadas las correcciones por parte del interesado, expido esta carta de aprobación para lo que corresponda.

Atentamente,

XXXXXXXXXXXXXXXX
Carné 999999

CÓDIGO DE ÉTICA

Universidad Internacional de las Américas
Código de Ética

El suscrito Michael Hidalgo Fallas, número de carné: 090506, graduado del grado de Bachillerato de la carrera de Ingeniería en Sistemas de Información de la Universidad Internacional de las Américas, se compromete a cumplir, durante el ejercicio profesional, con el Código de Ética de la Institución, que se rige por los siguiente principios:

PROBIDAD: actuar siempre con rectitud y honradez.

PRUDENCIA: actuar con pleno conocimiento de la materia sometida a su consideración.

JUSTICIA: permanente disposición hacia las funciones de la profesión, bajo los lineamientos legales que debe respetar todo profesional.

RESPONSABILIDAD: cumplir con los deberes, tanto en calidad como en oportunidad.

DISCRECIÓN: guardar respeto sobre los hechos o informaciones de los que tenga conocimiento con motivo del ejercicio profesional, sin que esto perjudique las funciones y responsabilidades.

INDEPENDENCIA DE CRITERIO: no involucrarse o comprometerse con situaciones, intereses o actividades contrarias a la moral, a la sana crítica y que, por ley, sean incompatibles con las funciones profesionales correspondientes.

DIGNIDAD Y DECORO: actuar con sobriedad y moderación.

TOLERANCIA: evidenciar una actitud paciente y de comprensión ante las opiniones divergentes que puedan expresar otras personas.

EQUILIBRIO: desempeñar las funciones profesionales con sentido práctico, buen juicio y equidad.

ACTUALIZACIÓN: comprometer parte del tiempo en actualizar los conocimientos y adaptarlos en el desarrollo de la actividad profesional.

VOCACIÓN: mostrar siempre apego al trabajo y a la educación recibida, como fundamentos para el desempeño laboral.

BUENA FE: toda conducta o comportamiento, criterio emitido y labor desempeñada debe basarse en los más altos principios éticos y tendrá como fundamento la buena fe.

_____

Michael Hidalgo Fallas
Cédula: 1-12750522

CARTA DE LA DIRECTORA DE CARRERA

San José, 11 de Diciembre
del 2014

Señores
Universidad Internacional de las Américas

Estimados señores:

La suscrita, Máster Olda Bustillos Ortega, Directora de la Escuela de Ingeniería Informática, hace constar que ha revisado el Informe Final de Graduación del estudiante Michael Hidalgo Fallas, cédula número 1-12750522, que ha titulado: Prototipo funcional de un plugin para Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones (SAST).

El mencionado Informe Final, responde a los requisitos exigidos en el Reglamento que la Escuela de Ingeniería Informática tiene para estos efectos. Por tanto, se autoriza al autor para que lo presente ante el Tribunal Examinador nombrado para esta ocasión.

Atentamente,

_____

Máster Olda Bustillos Ortega
Directora de la Escuela de Ingeniería Informática
Universidad Internacional de Las Américas

DEDICATORIA

AGRADECIMIENTOS

RESUMEN EJECUTIVO

INTRODUCCIÓN

1. Tema

Prototipo funcional de un plugin para Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones (SAST).

2. Planteamiento del problema de estudio

Se vive en un mundo interconectado, donde las tecnologías de información y comunicación juegan un rol fundamental en la vida de los seres humanos simplificando la forma de hacer negocios y la forma de relacionarnos. No obstante esta dependencia en la tecnología es aprovechada de forma sustancial por usuarios maliciosos, comúnmente conocidos como atacantes o piratas informáticos, donde buscan explotar vulnerabilidades en las distintas aplicaciones de software.

Este grupo de personas comparten características comunes como lo son su exhaustivo y minucioso conocimiento en desarrollo de software, sistemas operativos, redes computacionales y seguridad, donde tratan de explotar alguna de las muchas vulnerabilidades que afectan la infraestructura tecnológica actual. En la medida en que los ataques informáticos tienen un trasfondo económico y social, la necesidad de desarrollar aplicaciones de software que no solamente sean confiables si no que sean resistentes a incidentes informáticos es clave.

Pese a el incremento en el crimen cibernético de últimos años y a las

pérdidas económicas significantes que estos incidentes traen a las organizaciones víctimas de una brecha de seguridad, se puede notar que la reacción de la industria informática es aún lenta, es decir, de forma cotidiana más y más organizaciones son comprometidas por ataques informáticos que datan de hace más de 10 años, por lo que se puede notar que es necesario de otros enfoques.

Esta propuesta tiene como objetivo fundamental resolver dos problemáticas actuales que están relacionadas y cuya evolución implica que se traten de forma conjunta.

El primer problema que se pretende resolver es un tema de educación y de visibilidad. Lamentablemente un alto porcentaje de empresas no siguen las buenas prácticas en el momento de desarrollar o de adquirir aplicaciones de software.

En un estudio reciente realizado por el Instituto Ponemon y la empresa Security Innovation bajo el título que reza El estado actual de la seguridad de las aplicaciones (2014)[1], donde se han entrevistado a 642 profesionales del sector de las tecnologías de información (entre ejecutivos y personal técnico) se encuentran hallazgos muy preocupantes como lo son:

1. El 71% de los ejecutivos entrevistados creen que tienen implementado un entrenamiento en seguridad informática, no obstante solamente 20% del personal técnico sabe que dicho entrenamiento existe y está actualizado.

[1] https://www.securityinnovation.com/security-lab/our-research/current-state-of-application-security.html

2. La mayoría de las organizaciones no tienen bien definido un ciclo de desarrollo de software.

3. La mayoría de las organizaciones no realizan pruebas de seguridad en las aplicaciones

4. La mayoría de las organizaciones no tienen un programa formal de entrenamiento de seguridad de la aplicaciones.

5. La mayoría de las organizaciones no identifican, miden, o no entienden los riesgos de seguridad de aplicaciones.

6. Existe una brecha significativa entre los ejecutivos y los profesionales (técnicos)con respecto a los niveles percibidos de madurez y las actividades de seguridad de la aplicación.

La principal problemática aquí se puede reducir a poder darle a los desarrolladores de software las herramientas para que comprendan cuales son los problemas de seguridad comunes, como poder identificar estos problemas y como poder resolverlos.

Este enfoque busca que los desarrolladores de software estén más comprometidos a crear aplicaciones seguras. No obstante durante muchos años, los entornos de desarrollo integrado (IDE), por ejemplo Visual Studio .NET, le han brindado a los desarrolladores muchas herramientas en un mismo lugar que le permiten de forma ágil desarrollar software de calidad. Sin

duda son un factor determinante cuando hablamos de productividad.

Sin embargo, los creadores de estos ambientes de desarrollo se han preocupado muy poco (o casi nada sin ánimos de entrar en controversias) por brindarle a los desarrolladores un mecanismo para desarrollar código seguro.

En otras palabras no ha habido por parte de esta industria un interés por brindar retroalimentación en el momento que las vulnerabilidades son creadas (cuando el código es desarrollado). Idealmente, cuando existe un error de sintaxis en el código, el IDE (Entorno de Desarrollo Integrado), indica en tiempo real que ha habido un error y en el mejor de los casos le brinda una sugerencia acerca de como resolverlo. Otras terceras partes han creado plugins, es decir complementos adicionales dentro de un ambiente integrado de desarrollo, que permiten dar formato y escribir código más eficiente, fácil de leer, fácil de mantener pero el rol de la seguridad en el software sigue siendo el gran ausente.

Desafortunadamente muchos profesionales en informática desconocen las causas más comunes que hacen que el software sea inseguro. Peor aún, algunas veces se siguen malas prácticas como lo son buscar soluciones a problemas de programación en Internet, encontrando segmentos de código inseguro pero que al fin y al cabo cumple con lo que se busca, y de esta forma se traslada el riesgo a la organización en el momento de incluir dicho código vulnerable.

Eventualmente un usuario malicioso podrá explotar alguna vulnerabilidad cuando la aplicación haya sido implementada.

El problema que se pretende resolver aquí entonces es desde la perspectiva del desarrollo seguro, poder darle a los desarrolladores de software un mecanismo para que puedan identificar cuando escriben código vulnerable a un ataque informático conocido, dentro del mismo ambiente de desarrollo y siguiendo las mejores prácticas, de forma que se puedan identificar en una etapa temprana del ciclo del desarrollo del software, donde aún es viable solucionar el problema.

Cuando se refiere a riesgos informáticos, organizaciones como OWASP (Proyecto Abierto para la Seguridad de las Aplicaciones) y MITRE han desarrollado proyectos para poder identificar vulnerabilidades comunes. Idealmente todas las personas que participan activamente en el ciclo del desarrollo del software deben conocer a cabalidad estas vulnerabilidades.

Por parte de la fundación OWASP, han creado una guía llamado OWASP Top 10, el cual es una guía basada en riesgos sobre los incidentes en seguridad informática más comunes que afectan nuestra infraestructura tecnológica y de servicios. OWASP es ampliamente avalada por la industria informática y los documentos, guías y herramientas son usados diariamente para desarrollar software seguro. MITRE por su parte ha definido un tipo diccionario de las debilidades del software conocido como CWE por sus siglas en inglés (Common Weakness Enumeration) o enumeración de debilidades comunes. CWE ofrece un conjunto unificado, medible de las debilidades de software que permite una discusión más efectiva, la descripción, la selección y el uso de herramientas y servicios que pueden encontrar estas debilidades

en el código fuente y los sistemas operativos, así como una mejor comprensión y gestión de las debilidades de software relacionado con la arquitectura y el diseño.

El objetivo con este componente de software a ser desarrollado es que tome en consideración los riesgos y las debilidades descritas por estos dos estándares y poder darle retroalimentación al desarrollador cuando accidentalmente se escribe código fuente vulnerable.

El segundo problema a abordar está enfocado hacia la empresa Security Innovation, la cual permitirá que este componente (plugin) se desarrolle en sus laboratorios y eventualmente se distribuya bajo una licencia de software respectiva. Security Innovation es una empresa estadounidense que a lo largo de los años se ha convertido en una autoridad en el campo de la seguridad en el software y ayuda a las organizaciones a construir aplicaciones de software más seguras. Actualmente, Security Innovation cuenta con un producto estrella llamado TEAM Mentor .

TEAM Mentor es una referencia en tiempo real de las normas de seguridad de desarrollo de software que se integra con herramientas de análisis estático y con los ambientes de desarrollo para brindar de forma completa una solución a los problemas de seguridad.

Actualmente TEAM Mentor se integra con herramientas de terceras partes que realizan análisis estático de código entre las que se encuentran HP Fortify, Checkmarx, CAT.NET. A su vez TEAM Mentor se ha integrado con diversos ambientes de desarrollo como Visual studio.NET, Eclipse, IntelliJ

entre otros.

Idealmente cuando se trabaja de forma integrada con dichas herramientas mencionadas anteriormente que realizan análisis estático (es decir un análisis al código fuente no ejecutado), cuando una vulnerabilidad es identificada, la referencia en tiempo real de TEAM Mentor es desplegada, ofreciéndole a los desarrolladores un mecanismo para comprender el problema de seguridad en cuestión, que se familiaricen con los problemas de seguridad y que puedan resolver los problemas de forma oportuna y expedita.

Actualmente Security Innovation no se ha movido al campo del análisis estático de código, razón por la cual TEAM Mentor se usa en gran parte como una integración con terceras partes que realizan un escáner del código fuente y cuando se encuentra una vulnerabilidad (basado en algún CWE mencionado anteriormente), los artículos de TEAM Mentor se muestran en las diferentes interfaces y en diferentes clientes.

De igual forma la plataforma TEAM Mentor es consumida por diferentes consultores, arquitectos de software y expertos en seguridad como un complemento para recomendar como se puede resolver un defecto en el área de seguridad. Esto significa que dentro de una organización, cuando se detectan vulnerabilidades en el software, los equipos de trabajo distribuyen los hipervínculos con el contenido de TEAM Mentor.

Se puede decir que el enfoque hasta este momento no ha sido en el campo del análisis estático, razón por la cual el desarrollo de este componente o plugin que se está proponiendo ahora, abre las puertas a un

mercado mayor, brindando un modelo para que se detecten vulnerabilidades en el momento en que son creadas. De esta forma Security Innovation podrá unificar la documentación sobre los defectos de seguridad comunes con una herramienta de análisis estático de código, proporcionando información oportuna cuando se detecta algún patrón de código fuente inseguro.

El objetivo del plugin, es hacer un análisis estático en el código compilado y basado en los árboles sintácticos del compilador de C# y VB.NET, se pueden identificar código vulnerable y utilizar el mismo Visual Studio .NET para mostrar mensajes utilizando el mismo mantra conocido como advertencia, error e información .

De forma tal que dependiendo del riesgo de la vulnerabilidad encontrada, se podrá brindar un tipo de mensaje específico dentro del IDE, sin necesidad de que el desarrollado abandone el trabajo para realizar investigaciones sobre como solucionar el problema.

Con este enfoque se pretende poder crear un motor de análisis estático de código enfocado en seguridad de aplicaciones dentro de un ambiente integrado y brindando retroalimentación en el momento en que se necesite, es decir cuando se detecta dichas alguna de las vulnerabilidades documentadas en el estándar de OWASP. De igual forma se busca que los desarrolladores se vayan familiarizando con las mejores prácticas en el momento de desarrollar cualquier aplicación.

El equipo de investigación y desarrollo de Security Innovation en su trayectoria liderando el mercado de la seguridad informática considera el

desarrollo de este plugin muy relevante pues permitirá a los desarrolladores software comprender con los estándares internacionales y desarrollar software seguro.

3. Justificación

A medida que los sistemas de información se vuelven más complejos e interconectados, la dificultad de desarrollar aplicaciones de software que sean seguras y resistentes a ataques informáticos crece de forma exponencial.

Durante los últimos años hemos visto como la cantidad de incidentes en seguridad han venido en aumento. Estos ataques no discriminan ningún sector social. Los encabezados de las noticias nos confirman una vez más que cualquier organización puede ser víctima de ataques cibernéticos, partiendo del hecho que los usuarios maliciosos que perpetran estos ataques cuentan con mucho tiempo para estudiar aplicaciones en Internet y eventualmente atacarla.

La lista de empresas que han sufrido algún tipo de incidente de seguridad crece considerablemente y nos revelan malas prácticas en el desarrollo de aplicaciones de software que terminan comprometiendo a organizaciones, datos sensitivos de las mismas y lamentablemente a usuarios de esas aplicaciones.

A propósito del rol fundamental del ciclo del desarrollo del software, la siguiente frase es rica en simbolismo y nos revela la situación actual en

cuanto a seguridad de aplicaciones "..expertos sostienen que en la actualidad, el 90% de los ataques informáticos ocurren en la capa de aplicación" Adams, E (2011,Marzo 8) tomado de http://goo.gl/6e4ClN. Esto significa, que los usuarios maliciosos utilizan la misma aplicación o sitio Web para explotar alguna vulnerabilidad conocida y poder robar información sensitiva y comprometer a otros usuarios.

Este dato es alarmante, y deja en claro las malas prácticas en el momento de desarrollar aplicaciones.

Pese a las pérdidas económicas, daños morales y efectos colaterales producto de un incidente cibernético (incluyendo comprometer otros servicios y aplicaciones como lo son las redes sociales como es el caso de Facebook, Twitter y cuentas de correo electrónico) vemos que pese al esfuerzo de la industria por mejorar este panorama, aún el sector de tecnologías de la información necesita invertir más tiempo y recursos en capacitar al personal. Hacer que las personas involucradas en el ciclo del desarrollo del software (y no solamente los desarrolladores) conozcan las principales causas que hacen que el software sea inseguro es vital.

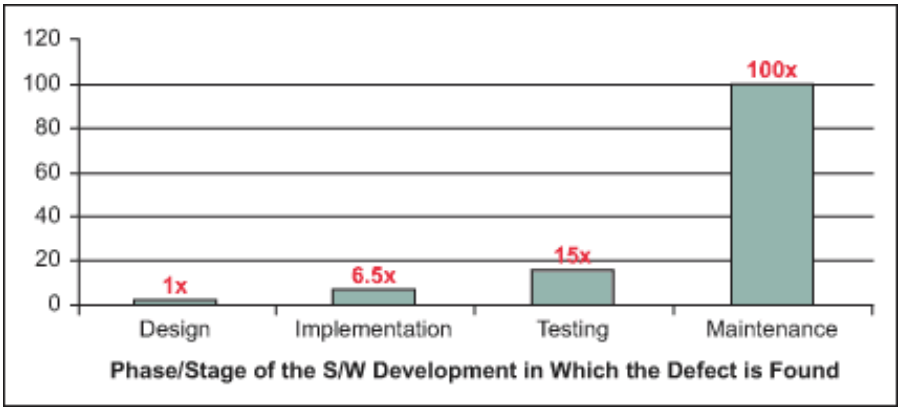
El tema de seguridad de la información, desafortunadamente no se le ha dado la importancia que realmente requiere.

Históricamente, seguridad en el software ha sido considerada como un requerimiento no funcional. Esta simple percepción tiene mucho peso en el desarrollo del software. De tal forma que muchas organizaciones conciben la seguridad como un segundo plano, algo que se debe implementar por ser

regulatorio, una obligación que se debe cumplir al final. Aunado a ello se puede comprender que no es fácil justificar el retorno de la inversión en seguridad de la información, tal como lo asegura el autor Mano Paul en su libro Official (ISC)2 Guide to the CSSLP.

Esta visión equivocada acarrea problemas más graves. Según los expertos[2] el costo relativo de resolver problemas de seguridad en un ambiente de producción va de 30 a 100 veces más caro que si se hubiera hecho en una etapa más temprana, dicha aseveración de muestra en el siguiente gráfico:

Gráfico 1 Costo relativo de arreglar defectos de código



Fuente: Paul (2011). Official (ISC2) Guide to the CSSLP

3.1 Estudio de Viabilidad de la propuesta

A continuación se presenta el estudio de viabilidad del proyecto propuesto.

[2] http://www.riceconsulting.com/public_pdf/STBC-WM.pdf

Según los autores Kendall, Kenneth & Kendall, Julie (2005) el concepto de viabilidad que ellos abordan es el siguiente:

> Nuestra definición de viabilidad es mucho más profunda que la que se da comúnmente ,puesto que la viabilidad de los proyectos de sistemas se evalúa de tres maneras principales: operativa, técnica y económica. El estudio de viabilidad no consiste en un estudio completo de los sistemas. Mas bien, se trata de recopilar sufrientes datos para que los directivos, a su vez, tengan los elementos necesarios para decidir si se debe proceder a realizar un estudio de sistemas.(p 52).

Tomando en cuenta la información anterior y con base en la definiciones presentadas por los autores, se presenta el estudio de viabilidad técnica, económica y operativo para el proyecto en cuestión.

3.2 Estudio de viabilidad técnica

Esta innovación tecnológica que supone el desarrollo de una extensión para el IDE (Entorno de Desarrollo Integrado) denominado Visual Studio .NET y que pertenece a la empresa tecnológica Microsoft, será desarrollado en Visual Studio .NET 2013, versión que es comercialmente estable en términos de que dicho software se puede adquirir de forma completa en el sitio de la misma compañía. De forma tal que el acceso alas herramientas de desarrollo

están disponibles en el mercado. Adicionalmente a esta cualidad, el desarrollo de la extensión de seguridad utilizará la plataforma de compilación abierta llamada Roslyn, la cual se puede obtener de forma gratuita desde el sitio oficial de Microsoft. Adicionalmente a la adquisición del compilador Roslyn, se instalarán las plantillas necesarias para desarrollar un proyecto denominado diagnóstico con solución de problemas.

Todas estas plantillas de proyectos están disponible de forma gratuita desde el sitio respectivo provisto por Microsoft. La adquisición de las mismas es trivial, ya que existe documentación basta para poder instalar los componentes requeridos.

Así mismo, el lenguaje de programación C# (pronunciado C Sharp), y que es el lenguaje de programación que será utilizado para desarrollar dicha extensión, viene incluido de forma intrínseca en Visual Studio. Adicionalmente, el plugin a ser desarrollado no requiere tener acceso a un motor de base de datos relacional, ya que no es necesario persistir datos durante la ejecución de la misma. Al contrario, la compilación en tiempo real permite que se evalúe la sintaxis de las expresiones y poder detectar patrones en el código fuente que guíen a determinar que existe código inseguro implementado. Las reglas respectivas para determinar los patrones de código inseguro serán parte de la extensión de seguridad que no necesita persistirse o almacenarse en ningún repositorio de datos, entendiéndose por repositorio cualquier mecanismo de almacenamiento incluyendo archivos en formato XML (acrónimo de Lenguaje de Marcas Extensible por su siglas en inglés), bases de datos transaccionales,

archivos planos de texto entre otros.

Con base en el proceso de identificar código inseguro, se tomará como referencia los estándares más importantes avalados en el sector de las tecnologías de información y comunicación referentes a seguridad informática, los cuales son el proyecto OWASP top 10 y como complemento la lista conocida como Enumeración de Vulnerabilidades Comunes (comúnmente conocida bajo el acrónimo de CWE por sus respectivas siglas en inglés).

Esta extensión de seguridad se basará en la plataforma Microsoft .NET 4.5.1, misma plataforma que se adquiere del sitio oficial de Microsoft y que viene integrada en las versiones de Visual Studio a ser desarrolladas. Las plantillas para desarrollar proyectos basados en Roslyn se adquieren al instalar la versión técnica para la comunidad (también conocido como CTP por sus siglas en inglés), dichas plantillas aparecerán disponibles en el ambiente integrado de desarrollo y permitirán crear la respectiva extensión.

3.3 Estudio de viabilidad económica

Según sostienen Kendall y Kendall "…La viabilidad económica es la segunda parte de la determinación de los recursos" Pág.82 y se fundamentan principalmente en el hecho de que se deben considerar aspectos como lo es el factor tiempo de los recursos entre otros elementos.

Dentro del análisis propuesto más abajo, se presentan los principales costos en los que se tendrá que incluir de forma preliminar para completar el prototipo funcional propuesto.

Cuadro 1 Costos

Gastos Operativos				
Cantidad	**Descripción**	**Horas**	**Costo por Hora**	**Sub Total**
1	Desarrollador de Software	154	$25.00	$3,850.00
1	Documentación del componente para su uso respectivo	10	$25.00	$250.00
1	Empaquetamiento y distribución del componente	10	$25.00	$250.00
1	Pruebas	30	$25.00	$750.00
			Total de Gastos Operativos	**$5,100.00**
Materiales y Misceláneos				
Cantidad	**Descripción**	**Costo de Mercado**	**Costo para el Proyecto**	**Sub Total**
1	Computadora personal, 8GB de Memoria RAM, Procesador Intel Core i5	$1,299.99	$0.00	$0.00
1	Office 365 Personal suscripción annual	$99.99	$0.00	$0.00
1	Suministros de oficina (cartuchos de tinta, papel)	$20	$20	$20
1	Tinta para impresora Epson XP-211	$50	$50	$50
1	Licencia de Microsoft Visual Studio Ultimate 2013	$486.96	$0.00	$0.00
1	Licencia de Microsoft Project Pro para Office 365	$300.00	$300.00	$300.00
	Total de Gastos Operativos			**$300.00**

Gastos Operativos	**Gastos Misceláneos**	**Total de Gastos Total**
$5,100.00	**$300.00**	**$5,400.00**

Fuente: Elaboración propia

Tal como se puede apreciar en el costo preliminar definido en la tabla anterior para este prototipo funcional, se estima que los gastos podrían rondar aproximadamente los $5,400.0. Con el objetivo de recuperar la inversión de este proyecto, la forma de comercializar el software supone dos modelos. Por una parte se propone que el precio unitario del componente que el costo de la extensión en su etapa inicial sea de $90 y que como parte de la descripción y de la retroalimentación se presenten artículos y referencias a TEAM Mentor.

Así mismo se requerían al menos de la venta de 60 licencias para

recuperar la inversión.

Por otra parte, a través de la adquisición del producto TEAM Mentor, se puede brindar la extensión de seguridad gratuita para 10 desarrolladores dentro de la organización. La comercialización de la licencia de TEAM Mentor, que es uno de los productos más importantes para la organización, permite que se brinde la extensión de forma gratuita, ya que se perciben más ganancias cuando se comercializa un producto tan maduro como lo es TEAM Mentor.

Basado en este enfoque de comercialización, las organizaciones con muchos empleados se ven a su vez beneficiadas, principalmente porque TEAM Mentor es una aplicación Web que se instala en un servidor dentro de la organización y permite que los empleados puedan tener acceso a todo el contenido de seguridad dentro de la intranet corporativa.

3.4 Estudio de viabilidad operativa

Desde una perspectiva operativa, el desarrollo de la extensión de seguridad integrada en el ambiente de desarrollo, busca que los desarrolladores de software tengan ése acercamiento al desarrollo de software seguro que es tan necesario, en una época donde los incidentes en seguridad informática causan pérdidas incalculables para las organizaciones sin importar a fondo el sector donde se desenvuelven.

En un sentido más estricto se busca proveer una herramienta oportuna

e integrada para que los desarrolladores puedan escribir aplicaciones de software que sean resistentes a ataques informáticos y desde esta perspectiva permitir que el modelo extensible del software permita tomar ventaja de tal capacidad para el desarrollo de la extensión. Al estar integrada en el ambiente de desarrollo, la información presentada al usuario bajo el mantra información, advertencia y error aprovecha el modelo tradicional que el desarrollador está familiarizado.

Por ejemplo, cuando existe un error de sintaxis en el código, el proceso de compilación de la plataforma, en este caso Microsoft .NET, le brindará al desarrollador la retroalimentación necesaria para que este último pueda solucionar el problema. Dicha retroalimentación aparece con un mensaje que le brinda a la persona usuaria del ambiente, la información concreta del porqué existe un error de compilación.

A medida que los errores se presentan de forma inmediata, el desarrollador de software puede comprender cual es el problema que está causando que el compilador detecte errores tanto sintácticos como semánticos. Aunado a ello, la retroalimentación en tiempo real o en tiempo de compilación es un activo, puesto que podemos inferir que el desarrollador de software es más productivo ya que la mayor parte del tiempo se invierte en desarrollado código nuevo o dando el respectivo mantenimiento al código fuente existente, en lugar de invertir largas horas detectando errores y tratando de solventarlos.

Como consecuencia de la necesidad de ayudar a los desarrolladores

de software a ser mas productivos, son muchas las extensiones que se han desarrollado y he ahí la razón de que el mismo entorno de programación Visual Studio proponga un modelo de extensión, donde se pueden desarrollar los componentes necesarios. La industria por otra parte, ha aprovechado de forma exitosa esta fortaleza, de forma tal que se pueden adquirir ya sea de forma comercial o gratuita componentes para mejorar el código en términos de fácil lectura, eficiencia y de fácil mantenimiento.

En otras palabras, muchos de los usuarios de Visual Studio ya conocen y se han visto beneficiados por las extensiones de terceros, particularmente porque la forma de instalar dichas extensiones es un tanto trivial.

A su vez considerando que la extensión no lleva implícita ninguna configuración para su funcionamiento, el usuario final podrá comenzar a utilizarla de forma inmediata que ha sido instalado y luego de que se haya producido una compilación del código fuente en el sistema.

Otro de los beneficios que esta extensión aporta es que su desinstalación es simple, en dadas circunstancias que el usuario quiera prescindir de la extensión, de forma fácil podrá hacer la desinstalación de la misma.

El código fuente que se mostrará como recomendación en el momento de que la extensión detecte un patrón de código malicioso sigue a su vez los estándares de mejores prácticas tomando como referencia los estándares mencionados anteriormente. En la medida en que el usuario comprenda como la extensión detecta vulnerabilidades en el código y como brinda una solución

oportuna, permitirá que éste comprenda la raíz del problema y a su vez ayudarle a que mejore la calidad del software en términos de seguridad.

4. Objetivos de la investigación

A continuación se detalla el objetivo general y los respectivos objetivos específicos, los cuales fijarán la hoja de ruta necesaria para el desarrollo del prototipo funcional.

4.1 Objetivo General

Elaborar un prototipo funcional para realizar pruebas estáticas de seguridad de aplicaciones en el ambiente integrado de desarrollo Visual Studio .NET.

4.2 Objetivos Específicos

Un objetivo específico, según lo define Miranda, Juan José (2005) "..Corresponde a la solución concreta que el proyecto debe alcanzar" (pág. 45). Así mismo el autor continúa su definición concreta cuando sostiene que "…El objetivo específico es el logro de una situación deseable."(pág.45).

Tomando en consideración la definición, seguidamente se presentan los objetivos específicos para el prototipo funcional propuesto.

4.2.1 Realizar el levantamiento de requerimientos de cada una de las vulnerabilidades a identificar mediante el uso de estándares en la industria.

4.2.2 Elaborar el diseño del software que contempla el flujo de trabajo, la identificación de vulnerabilidades y la retroalimentación al usuario final.

4.2.3 Desarrollar el prototipo funcional de la extensión de seguridad para el ambiente de desarrollo Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones.

4.2.4 Implementar las reglas de diagnóstico para detectar vulnerabilidades en el código fuente utilizando estándares en la industria.

4.2.5 Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo.

5. Alcances

El prototipo funcional que consiste en un componente , plugin o extensión de seguridad, será desarrollado bajo el lenguaje de programación C# (pronunciado C Sharp) y bajo la plataforma Microsoft .NET 4.5. C# es un lenguaje de programación moderno e innovador que forma parte de la plataforma de Microsoft .NET y que cuya evolución y mejoras constantes hace que sea uno de los lenguajes de programación preferidos.

Se utilizará a su vez el ambiente integrado de desarrollo conocido como Visual Studio .NET 2013 y la plataforma de compilación de Microsoft

.NET llamada Roslyn, la cual brinda a los desarrolladores una serie de interfaces programables para poder interactuar con el compilador de C# y Visual Basic. NET.

Aún cuando la plataforma de compilación Roslyn de Microsoft .NET permite interactuar de forma conjunta con los lenguajes de programación C# y Visual Basic. NET, el enfoque de este prototipo funcional será detectar vulnerabilidades en código fuente desarrollado principalmente en C# y las pruebas de concepto evaluarán los resultados en dicho lenguaje.

En etapas posteriores que están fuera del alcance de este prototipo, se dará soporte al lenguaje Visual Basic .NET y de forma paulatina, que también está fuera de este alcance, a medida que Roslyn soporte más lenguajes dentro de la plataforma .NET , idealmente se buscaría que el plugin sea compatible con dichos lenguajes.

Pese a que es posible poder desarrollar este componente para que pueda detectar código vulnerable tanto en proyectos C# y Visual Basic .NET, este plugin propuesto se enfocará en detectar vulnerabilidades únicamente en aplicaciones C# .NET .

Considerando que la cantidad de lenguajes de programación incorporados bajo la sombrilla de la plataforma .NET es muy grande, incluso cuando los desarrolladores pueden crear su propio lenguaje que se ejecute bajo la plataforma, se hace necesario limitarse a un solo lenguaje, C# en este caso, para poder realizar el análisis estático de código en aplicaciones de software.

El desarrollo de este prototipo funcional de una extensión de seguridad en el ambiente integrado de desarrollo Visual Studio .NET se basa en el estándar conocido como OWASP Top 10 2013, el cual es una guía agnóstica de los principales riesgos que afectan las aplicaciones Web. Entiéndase por guía agnóstica a una guía que es independiente del lenguaje de programación y del sistema operativo, lo que quiere decir que se aplica para cualquier tecnología Web. Dicho informe se basa en 8 conjuntos de datos de siete firmas de seguridad informática donde los datos involucran más de 500 mil vulnerabilidades en cientos de organizaciones y miles de aplicaciones, información suministrada por el mismo informe.

Debido a que OWASP considera la seguridad en el software como un problema que involucra a personas, procesos y organizaciones, algunos de los riesgos presentes en este informe no están relacionadas directamente con el código fuente, tal es el caso de la utilización de componentes con alguna vulnerabilidad conocida u otros riesgos que pueden ser detectados mediante un exhaustivo proceso de pruebas como lo es el caso del riesgo A4-Referencia directa insegura a objetos.

Esto quiere decir que la identificación del riesgo no está atada directamente al código fuente per se, mas bien a un proceso como tal donde por ejemplo no se cumplen algunas condiciones como lo son que se exponen datos sensibles en forma de parámetros en el navegador , fáciles de predecir, y donde no hay un proceso de autenticación y autorización por cada solicitud recibida para constatar si el dato suministrado.

Partiendo el hecho de que este informe se basa en riesgos, el desarrollo del componente de seguridad se enfoca en la identificación de los 5 riesgos más importantes del informe y que puedan ser identificados en el código fuente .

La versión de OWASP Top 10 2013 y que es la más reciente contempla los siguientes riesgos informáticos:

- A1-Inyección
- A2-Pérdida de autenticación y gestión de sesiones.
- A3- Secuencia de comandos en sitios cruzados (XSS).
- A4- Referencia directa insegura a objetos
- A5- Configuración incorrecta de Seguridad.
- A6-Expocisión de datos sensibles
- A7- Ausencia de control de acceso a nivel de funciones.
- A8- Falsificación de peticiones en sitios cruzados (CSRF).
- A9- Uso de componentes con vulnerabilidades conocidas.
- A10- Redirecciones y reenvíos no válidos.

Tal como se muestra la lista anterior, la prioridad en la que se han identificado estos riesgos está basado en una serie de estudios que involucra la cantidad de aplicaciones Web que se han encontrado vulnerables a estos ataques, la incidencia en la industria y el impacto para las organizaciones.

El prototipo funcional para Visual Studio que se desarrollará tendrá su foco de estudio en los siguientes cinco riesgos :

- A1-Inyección
- A2-Pérdida de autenticación y gestión de sesiones.
- A3- Secuencia de comandos en sitios cruzados (XSS).
- A5- Configuración incorrecta de Seguridad.
- A6-Expocisión de datos sensibles

Nótese que el riesgo A4- Referencia directa insegura a objetos, está fuera de alcance de este prototipo debido a que la identificación del mismo está sujeto a más de una variable que no es fácilmente predecible en el código fuente y que necesita de procesos exhaustivos de pruebas para determinar si es o no vulnerable.

Para cada una de las vulnerabilidades mencionadas anteriormente, se desarrolla la extensión por medio de un proyecto o plantilla conocido como diagnóstico con solución de código, de forma tal que cuando se detecta alguna de las vulnerabilidades, se despliega un mensaje en el ambiente de Visual Studio donde el desarrollador podrá seleccionar ahí mismo corregir la vulnerabilidad. Así mismo se le mostrará por cada hallazgo un link directo a la guía de TEAM Mentor, donde el desarrollador podrá tener una mejor idea del problema de seguridad que enfrenta. TEAM Mentor con su enfoque de solucionar el problema de forma fácil y mejor ayudará de forma considerable a mejorar la calidad del software en términos de seguridad.

6. Limitaciones

Para la realización de este prototipo funcional no se han encontrado limitaciones relevantes que impidan la realización del mismo.

7. Antecedentes

La industria informática ha utilizado el concepto de análisis estático de software durante muchos años para definir un proceso mediante el cual se realiza un análisis del software sin ser ejecutado (como aclaración cuando se trabaja con código que es ejecutado se le llama análisis dinámico), mediante la revisión de código fuente, y generalmente este proceso se realiza por medio herramientas automatizadas aunque también se puede realizar de forma manual. El autor Esposito (2011) , cuando se refiere al rol del análisis estático código sostiene que :

> Para muchos, es difícil de creer que alguna vez hubo un tiempo cuando se puede imprimir todo el código fuente de una aplicación, la lectura de arriba a abajo y detectar los errores o mal comportamiento posible. La complejidad del software moderno que este enfoque poco práctico para los seres humanos, pero no para algunos tipos de software inteligente, como, por ejemplo, herramientas de análisis estático (Esposito, 2011).

Según el autor, el análisis estático de código tiene como objetivo fundamental deducir el comportamiento de una sección de código y sus instrucciones. El autor también sostiene que las herramientas que realizan

este análisis básicamente buscan hacer un examen del código fuente mediante la creación de forma progresiva de una base de datos de conocimiento, con el objetivo de probar las instrucciones y las salidas correctas e incorrectas. Cuando se realiza un análisis estático, el código fuente no es ejecutado. El objetivo de este proceso consiste en identificar errores potenciales y eventualmente poder brindar recomendaciones.

El análisis estático enfocado en seguridad se basa en el mismo principio. La primera vez que este concepto sale a relucir es mediante una publicación de la firma Gartner, titulado Cuadrante Mágico para las Pruebas de Seguridad de Aplicaciones Estáticas(2010) donde se analiza la evolución del mercado de las herramientas de análisis estático de código, los principales productores, la visión del negocio y la tecnología.

Este informe sostiene que debido al aumento considerable y a las variantes de vectores de ataques informáticos, el rol de SAST es fundamental, incluso los autores indican que tales tecnologías deberían ser obligatorias para toda organización de IT que desarrolla procesos y procedimientos.

En contraste, el estudio más reciente del año 2013 muestra como ha sido la evolución de SAST, se pude ver claramente como hay nuevos líderes incursionando y como más y más productos se suman a este mercado.

Los primeros pasos en implementar análisis estático de código dentro del entorno de Visual Studio .NET enfocados en seguridad datan de la creación del complemento llamado CAT.NET creado por Microsoft. Dicho componente fue descontinuado desde las versiones de Visual Studio.NET

2008 . CAT.NET permitía identificar por medio de errores en tiempo de compilación cuando existía código vulnerable, no obstante el hecho de que haya sido descontinuado y que no se pueda utilizar en las versiones más recientes de Visual Studio .NET abre la oportunidad de poder innovar y desarrollar componentes que puedan ser utilizados por la industria como es el caso de la extensión propuesta.

Además de CAT.NET, figura otras herramientas de análisis estático como lo es FoxCop , tal como lo define Microsoft “FoxCop es una aplicación que analiza ensamblados de código manejado (código destinado a la plataforma Microsoft .NET) y reporta información acerca de dicho ensamblado tal como alguna mejora en el diseño, desempeño y seguridad”. Nótese que FoxCop no está enfocada directamente al tema de seguridad si no por el contrario toma en consideración elementos claves de diseño.

7.1 El modelo de ejecución del CLR

El entorno en tiempo de ejecución del lenguaje o CLR (Common Language Runtime) por sus siglas en inglés , es como su nombre lo indica, un entorno en tiempo de ejecución que Microsoft .NET proporciona y donde según Microsoft (2014) “Ejecuta el código fuente y proporciona servicios que hacen que el proceso de desarrollo más fácil”.

De esta misma forma, Microsoft(2014) indica que:

> Los compiladores y herramientas exponen la funcionalidad del entorno en tiempo de ejecución y le permiten a usted escribir código que se beneficia de este ambiente manejado. El código que usted desarrolla

> con un compilador del lenguaje y que se ejecuta en este entorno se le denomina código manejado.

Así mismo, Richter (2010) sostiene que:

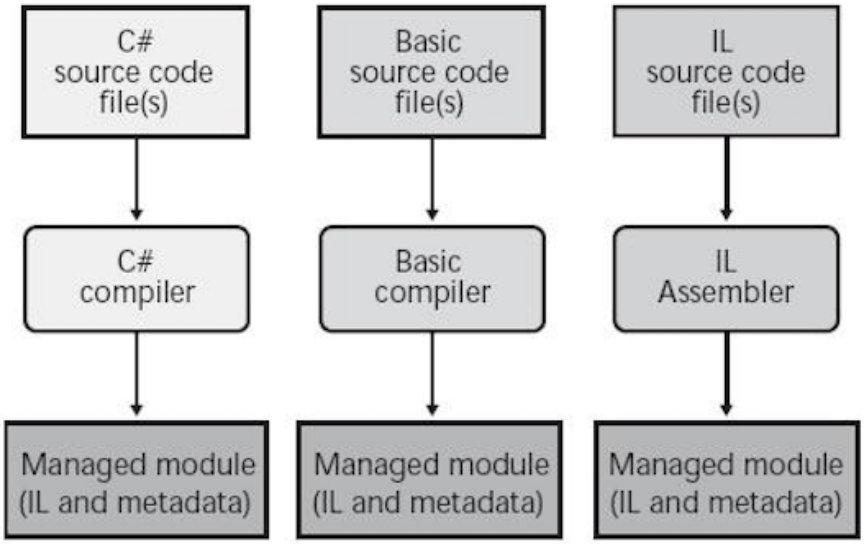
> Las funcionalidades principales del CLR (tales como manejo de memoria, carga de ensamblados, seguridad, manejo de excepciones y sincronización de hilos) están disponibles para todos los lenguajes que tienen como destino ser ejecutados bajo este entorno.
> De hecho, en tiempo de ejecución el CLR no sabe cual lenguaje de programación el desarrollador usó para escribir el código fuente. Usted puede desarrollar su código fuente en cualquier lenguaje de programación que usted desee siempre y cuando siga los lineamientos de CLR. (p. 2).

Microsoft (2014) también indica que algunas de las características que ofrece CLR son:

1. Mejoras en el rendimiento.
2. La habilidad de usar de forma fácil componentes desarrollados en otros lenguajes de programación.
3. Tipos de datos extensos proporcionados por la librería de clases.
4. Características del lenguaje entre las que figuran la herencia, interfaces, sobrecarga y programación orientada a objetos.
5. Soporte estructurado para manejo de excepciones.
6. Recolección de basura.
7. Soporte a atributos personalizados.

En la siguiente imagen se aprecia el proceso de compilar código fuente en módulos manejaros.

Figura 1 Compilando código fuente en módulos manejados.



Fuente: Richter (2010) CLR via C# Third Edition.

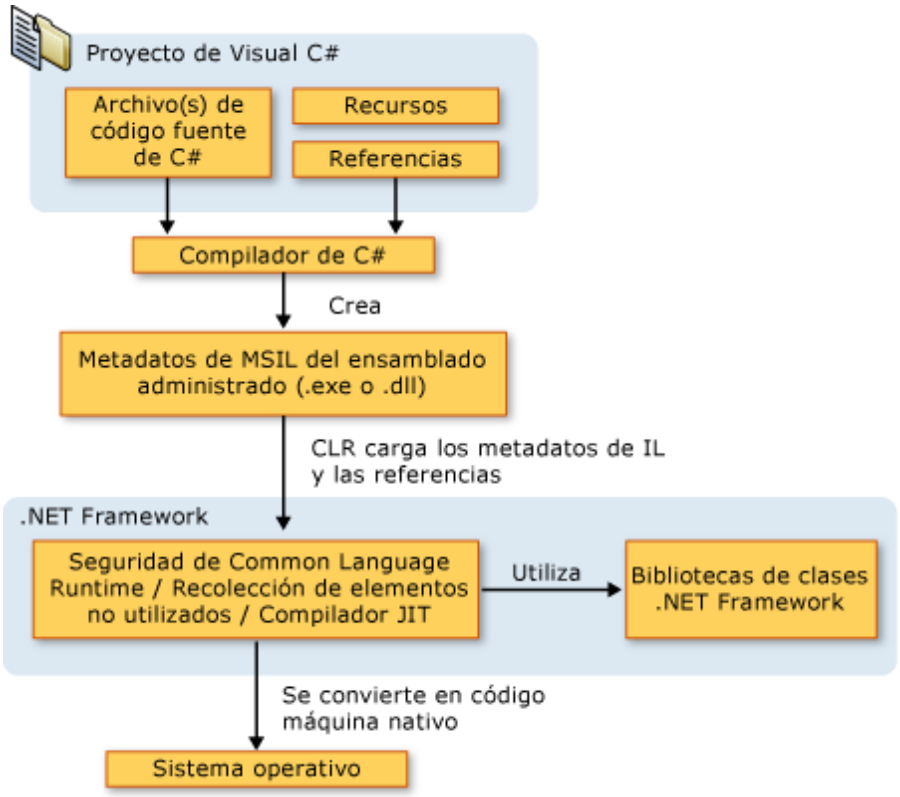
7.2 Compiladores como cajas negras

Stroustrup (2009) a propósito del proceso de compilación indica que :

> Para ejecutar un programa, usted debe primero traducirlo de una forma que es legible para los seres humanos a algo que una máquina pueda "entender". Esa traducción es hecha por un programa denominado compilador. Lo que usted lee y escribe se denomina código fuente o programa, y lo que la computadora ejecuta se le denomina ejecutable, código objeto o código máquina". (p. 47).

En la siguiente imagen se ilustran los componentes que forman parte de la compilación del código fuente en C#.

Figura 2 Proceso de compilación en un ambiente administrado



Fuente : Microsoft http://msdn.microsoft.com/es-es/library/z1zx9t92.aspx

Al igual que se demuestra en la imagen anterior, el proceso de compilación siempre se ha concebido como una caja negra, es decir el código

fuente ingresa por un lado, existe un proceso de compilación intermedio y luego se tiene la salida. Ese proceso intermedio ha sido un poco desconocido, pese a que se puede garantizar que funciona adecuadamente.

Hazzard & Bock (2013) se refieren a la caja negra del proceso de compilación de la siguiente forma :

> Desde la primera versión de .NET, el compilador ha existido como un ejecutable sencillo. Usted lo invoca para cambiar el código fuente, el cual está contenido en archivos de texto, en un ensamblado.
> A primera vista, un compilador parece algo trivial, pero lo que sucede adentro es increíblemente complejo. Las reglas que un escritor de compiladores debe seguir para cambiar el código fuente escrito en C# en metadatos y en código fuente intermedio pueden ser desalentadoras. (p. 288).

Tal como se puede distinguir en las afirmaciones de los autores anteriores, el proceso de compilación si bien es cierto es exitoso (puesto que hace lo que tiene que hacer) de alguna forma limita a los desarrolladores a tener acceso a lo que sucede durante tal proceso.

Ahora bien quizá la pregunta que es necesario formular en este momento sea Cual es el objetivo principal de poder tener acceso al proceso de compilación y a las desconocidas actividades que esta actividad constituye?. La respuesta mas asertiva a esta interrogante es porque las cosas se pueden hacer mejor; no obstante el término mejor es amplio en su naturaleza pero el punto de partida es que se puede mejorar la experiencia del

usuario brindándole por ejemplo una mejor retroalimentación cuando ocurre un error.

Hazzard & Bock (2013) apoyan este punto de vista pues afirman que :

> Aún cuando la mayoría del trabajo ocurre en un segundo plano, usted no tiene muchas opciones para controlar lo que el compilador hace. De hecho, usted tiene menos de 50 opciones, algunas de las cuales no tiene que ver con directamente con el proceso de compilación; ellas le dicen al compilador que cree archivos ajenos (tales como los archivos /doc). (p. 288).

7.3 El Proyecto Roslyn : Abriendo la caja negra.

El nuevo compilador de Microsoft, conocido bajo el código de “Roslyn”, abre la caja del compilador de forma tal que se puedan ver todos los pasos y rutas que el compilador toma en cuenta cuando convierte el código fuente en ensamblados.

Somasegar (2011) refiriéndose al proyecto Roslyn indica:

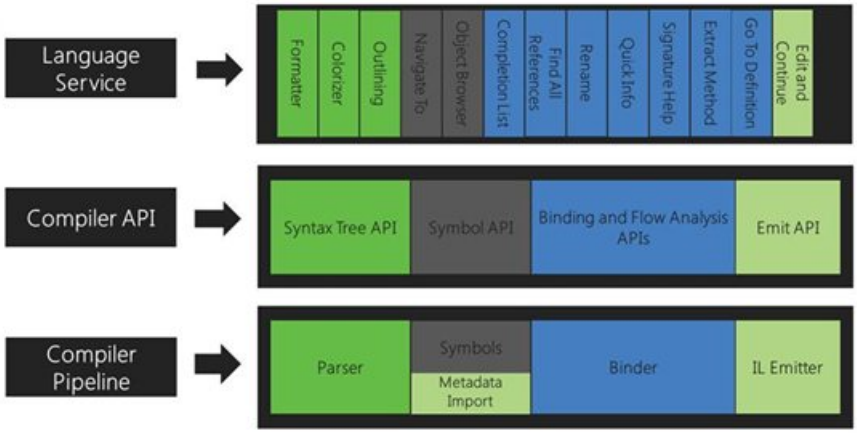
> Se acaba de lanzar Microsoft “Roslyn” CTP, el cual permite que los compiladores de C# y Visual Basic sean usados como servicio. Mientras que hoy los compiladores están implementados en C++, en Roslyn hemos reescrito los compiladores, implementando el compilador de C# en C# y el compilador de Visual Basic en Visual Basic.
>
> Históricamente, los compiladores manejados provistos en Visual Studio se consideran cajas opacas: usted provee el código fuente, y estos son convertidos en ensamblados.

Basado en la definición anterior, se muestran nuevas oportunidades de innovación pues de cierta forma se podrá tener acceso a estaciones de

compilación lo que eventualmente habilita al desarrollador a crear reglas específicas. Osenkov (2011) expresa que “Esto abre nuevas oportunidades para las personas que extienden la funcionalidad de Visual Studio para escribir poderosas herramientas para hacer análisis de código”.

En la siguiente imagen se identifican los principales componentes de la plataforma Roslyn.

Figura 3 Elementos de la plataforma Roslyn



Fuente: http://blogs.msdn.com/b/bryang/archive/2011/11/01/roslyn-ctp-released.aspx

8. Referente Institucional

Security Innovation es una empresa ubicada en Estados Unidos que tiene sus edificios corporativos en Wilmington, Massachusetts y además una oficina en Seattle, Washington en Estados Unidos. Radicada en el sector de la seguridad informática, es una autoridad en la seguridad del software que ayuda a las organizaciones a construir e implementar software más seguro. Entre sus clientes figuran proveedores de la industria informática y organizaciones de tecnologías de la información como lo son Microsoft, IBM, FedEx, ING, Symantec, Coca-Cola y General Electric. Estas empresas confían en la experiencia de Security Innovation para entender los riesgos de seguridad en sus sistemas .

Security Innovation se especializa en la seguridad del software y ha estado analizando vulnerabilidades y riesgos de las aplicaciones durante casi una década, siendo uno de los primeros proveedores de soluciones de riesgo de software para las empresas que figuran dentro de la lista de Fortune 500.

Fundada en el año 2001 por el doctor James Whittaker, Security Innovation está integrada por pioneros en el campo de la seguridad de las aplicaciones capaces de resolver cualquier problema de seguridad en las organizaciones. La empresa Security Innovation se compone de un equipo de ingenieros de primera clase, desarrolladores de software, analistas de control de calidad, analistas de seguridad y analistas del negocio que de forma colectiva resuelven problemas de negocio ofreciendo soluciones técnicas.

Security Innovation se enfoca en el problema más difícil de las tecnologías de información y la causa de la mayoría de los incidentes informáticos, es decir las aplicaciones de software inseguras. Las soluciones que ofrece Security Innovation, se basa en los tres pilares de un ciclo de vida del desarrollo de software seguro, las cuales abarcan estándares, educación y evaluaciones. De tal forma que se ofrecen diferentes productos a la medida para fortalecer cada uno de estos pilares:

- Estándares: TEAM Mentor mejores prácticas para el desarrollo de software seguro.
- Educación : TEAM Professor eLearning y entrenamiento.
- Evaluaciones: Se auditan aplicaciones y su ciclo de vida del desarrollo de software utilizando estándares internaciones.

En el siguiente cuadro se ilustra los pilares del desarrollo de software seguro implementados por la empresa Security Innovation:

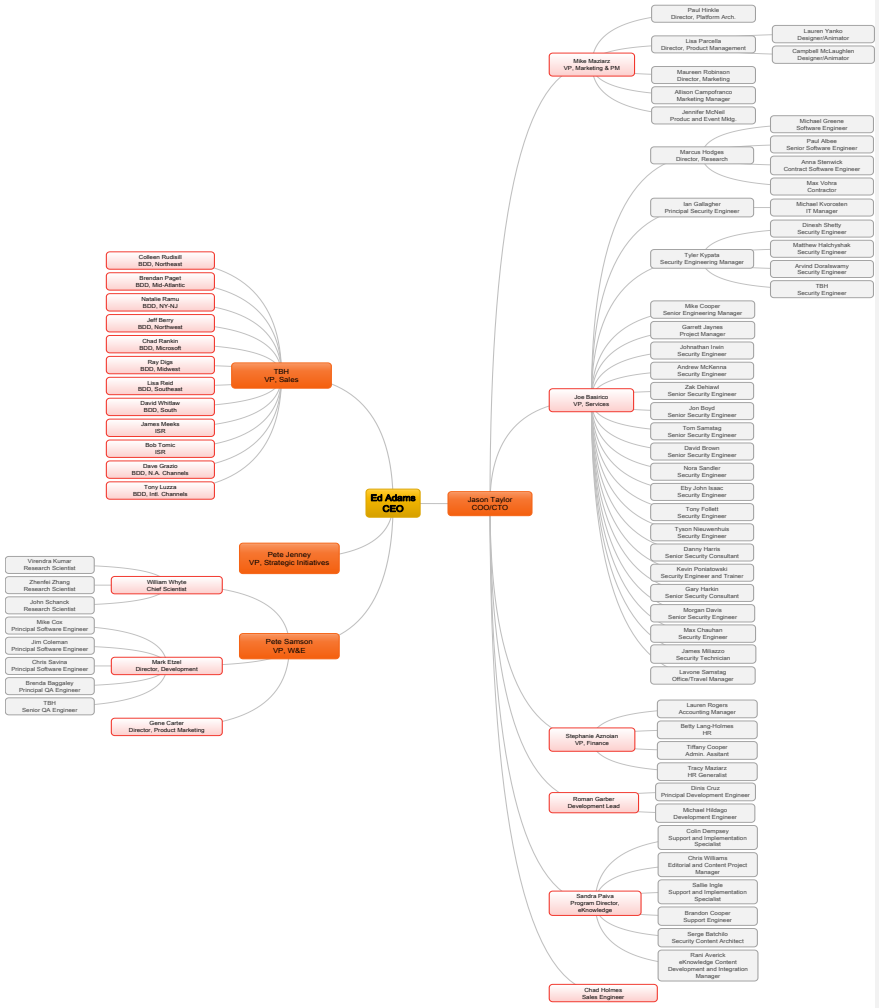
Figura 4 Tres pilares del desarrollo de software seguro.

Fuente: http://goo.gl/LfsVMX

La misión de la empresa Security Innovation consiste en trabajar incansablemente en su nombre para ofrecer soluciones de seguridad que responden a sus objetivos de negocio y que se alinean con su tolerancia al riesgo.

El presidente y director ejecutivo de Security Innovation es Ed Adams, el cual es un ejecutivo de software con amplia experiencia en el liderazgo exitoso de organizaciones de diversos tamaños en el campo de las tecnologías de información y seguridad. El señor Adams es a su vez un miembro del Instituto Ponemon. A continuación se presentan el organigrama de Security Innovation, actualizado a Julio de 2014.

Figura 5 Organigrama Security Innovation



Fuente: Security Innovation

CAPÍTULO I

DIAGNÓSTICO

1.1 Análisis FODA

Las siglas FODA hacen referencia a los conceptos de fortalezas, oportunidades, debilidades y amenazas respectivamente. El autor Ioannou (2012), refiriéndose a este tema detalla que “El análisis FODA, también conocido como matriz FODA es un método usado para determinar las fortalezas, debilidades, oportunidades y amenazas dentro de un proyecto o dentro del entorno de una empresa.”. Este mismo autor además define un punto verdaderamente importante para poder identificar cada uno de estos elementos al decir que :

> Las fortalezas y las debilidades son factores internos, los cuales son atribuidos a las acciones directas o a control o a la compañía o a los empleados. Las oportunidades y las amenazas por otro lado, son factores externos y están fuera de su control. (p. 2).

La importancia de este análisis es definido ampliamente por Lawrence G. Fine (2011) cuando indica que:

> Decisiones importantes son tomadas todos los días por cada uno de nosotros. Hay momentos en que necesitamos hacer un rápido juicio y basamos esas decisiones en la información que tenemos disponible. Sin embargo, hay otras situaciones donde hay que observar diferentes factores disponibles y en estos casos es cuando se necesita usar el análisis FODA (p. 1).

Partiendo de los dos enfoques propuestos por los autores, se puede comprender que el análisis FODA es una herramienta o técnica muy

importante para el proceso de toma de decisiones. Inclusive Fine (2011) sostiene que “El resultado del análisis FODA es ver la realidad de su negocio o sus ideas. También le dará una lista de puntos de acción, los cuales usted deberá seguir”.

Es relevante comprender cada uno de los elementos que forman parte del análisis FODA, a fin de proceder de forma correcta en el momento de recopilar la información y de principalmente identificar como afectan dichos elementos el prototipo funcional propuesto.

- Fortaleza: Las fortalezas forman parte del análisis interno de la organización. Una definición más amplia brindada por Ioannou (2012) detalla que “La fortaleza de una compañía son sus recursos y las capacidades que pueden ser usadas para desarrollar una ventaja competitiva”.
- Debilidades o Limitaciones: Las debilidades también forman parte del análisis interno. Claramente Ioannou (2012) dice que una debilidad es “la ausencia de una fortaleza en la organización”. Es muy importante reconocer las debilidades, principalmente porque en términos de Fine (2011), se puede concluir en que las debilidades existen en toda organización y es necesario determinar como se va a lidiar con ellas.
- Oportunidades : Las oportunidades son situaciones que se presentan y que pueden generar un beneficio para la organización. Identificar las oportunidades es un proceso importante, ya que palabras de

Ioannou (2012), el proceso de identificar oportunidades consiste en "Identificar oportunidades de crecimiento o de mejorar la rentabilidad de un negocio". Así mismo, refiriéndose a oportunidades Fine (2011) dice que "Hay oportunidades alrededor de nosotros, pero la mayoría del tiempo las perdemos porque estamos demasiado enfocados en aspectos negativos".

- Amenazas: Una amenaza es un factor externo que puede causar un efecto adverso o negativo a la organización. En términos de riesgo, el experto en seguridad Paul(2011) se refiere a una amenaza como "… es meramente la posibilidad de que un evento potencialmente dañino o no deseado ocurra". Para Ioannou (2012), algunos ejemplos de amenazas lo constituyen los competidores que venden el mismo producto, cambios bruscos en los impuestos, trabajadores habilidosos escasos.

Tomando en consideración el punto de vista de los autores y de los elementos que constituyen el análisis FODA, se puede rescatar que realizar este análisis de forma correcta ayuda de forma sustancial en el proceso de tomar decisiones de forma más asertiva.

1.2 Análisis FODA para el prototipo funcional.

A continuación se presenta en forma de matriz los principales elementos que forman parte del FODA para el prototipo funcional. Dicha matriz agrupa los factores internos y externos de forma tal que su apreciación sea clara.

Cuadro 2 Análisis FODA

Internas	
Fortalezas	Debilidades
• Amplia experiencia en el mercado de la seguridad de las aplicaciones. • Mayor comercialización de los productos. • Integración con empresas en el mercado de la seguridad de aplicaciones. • Empresa cuenta con áreas de investigación y desarrollo donde se produce tecnología de vanguardia. • Facultar a empresas a desarrollar aplicaciones de software más seguras.	• Poca o nula inserción en el campo del análisis estático de código. • Dependencia de terceras empresas para realizar el análisis estático de código. • Proyectos de código abierto y gratuito ofrecen productos similares a muy bajo costo. • Plugin limitado a un lenguaje de programación y a un entorno integrado de desarrollo.
Externas	
Oportunidades	Amenazas
• Creciente demanda en seguridad de aplicaciones por parte de la industria. • Rápida evolución del lenguaje de programación C#. • Herramienta integrada en el ambiente de desarrollo.	• Competencia ofrece productos similares e integrados. • Clientes prefieren productos unificados. • Uso de tecnologías que no son suficientemente maduras.

• Acercamiento de nuevos clientes potenciales.	

Fuente : Propia.

1.3 Fortalezas

1.3.1 Amplia experiencia en el mercado de la seguridad de las aplicaciones.

La empresa Security Innovation desde su fundación en el año 2001, se ha convertido en una autoridad en el tema de seguridad en el software, brindando herramientas y mejores prácticas para facultar a la industria de mecanismos para desarrollar software que sea seguro. La elaboración del prototipo funcional acá propuesto cuenta con todas las lecciones aprendidas a lo largo de los años en la organización, incluyendo el talento de profesionales en materia de seguridad informática, a los cuales se acudirá de forma recurrente, que constituye un pilar fundamental para desarrollar los principios básicos de seguridad de forma efectiva.

1.3.2 Mayor comercialización de los productos :

La elaboración del prototipo funcional permitirá que los productos desarrollados por la organización, principalmente las guías y estándares, puedan ser comercializados más fácilmente, ya que el prototipo hará referencia a las mejores prácticas desarrolladas por la organización así como las recomendaciones pertinentes, de forma tal que clientes potenciales puedan a su vez conocer más de los productos que la empresa ofrece.

1.3.3 Integración con empresas en el mercado de la seguridad de las aplicaciones.

La organización ha trabajado en paralelo con gigantes de la industria informática en el mercado de la seguridad de la aplicaciones entre las que se encuentran HP Fortify, IBM y Checkmarx. Estas empresas a su vez tienen trayectoria elaborando herramientas para mejorar la seguridad en las aplicaciones de software. El hecho de trabajar en conjunto con terceras partes faculta que el conocimiento en áreas de investigación y desarrollo sea incorporado en los nuevos productos que se desarrollan.

1.3.4 Empresa cuenta con áreas de investigación y desarrollo donde se produce tecnología de vanguardia.

La empresa Security Innovation ha estado inmersa en proyectos de investigación y tecnología de punta en temas como lo son criptografía y algoritmos de cifrado modernos y eficientes hasta trabajar en conjunto con la industria de los sistemas de transporte inteligente, donde se busca disminuir la cantidad de accidentes de tránsito en carretera por medio de la tecnología.

El hecho de participar en proyectos innovadores ha hecho que la empresa se reconocida aún más, de forma tal que el prototipo funcional propuesto busca formar parte de la lista de aplicaciones innovadoras ofrecidos por la organización.

1.3.5 Facultar a empresas a desarrollar aplicaciones de software más seguras.

El prototipo funcional propuesto viene a solventar una necesidad recurrente en el mercado, la cual consiste en brindarle a los desarrolladores de software retroalimentación en términos de seguridad cuando desarrollan aplicaciones. A lo largo de los años, Security Innovation ha tenido como objetivo brindar herramientas, guías y recursos educación (entre los que se destacan entrenamientos en línea y bases de datos de conocimiento) para que las organizaciones implementen software seguro. El prototipo funcional para realizar análisis estático de código se alinea perfectamente con los objetivos de la empresa y sus áreas de acción, facultando a los clientes a optar por nuevos productos.

1.4 Oportunidades

1.4.1 Creciente demanda en seguridad de aplicaciones por parte de la industria.

En un panorama como el actual donde la cantidad de incidentes en seguridad informática va en aumento, surge la necesidad de implementar mecanismos para evitar ser víctima de un ataque cibernético. Paul, Mano (2011) es claro cuando indica que :

> En una época cuando las brechas de seguridad en el software le están costando a las compañías multas más grandes y cargas regulatorias, desarrollar software que sea confiable en su funcionalidad, resistente a ataques, y que sea recuperable cuando se interrumpen las operaciones del negocio es una necesidad. (p 1).

A medida que esta necesidad de desarrollar software que sea resistente a ataques informáticos se vuelve imperativa, las empresas están solicitando herramientas que les ayuden a protegerse y evitar ser víctimas de un eventual ataque cibernético. El prototipo funcional propuesto para desarrollar análisis estático de código se ajusta perfectamente a esta realidad.

1.4.2 Rápida evolución del lenguaje de programación C#

El lenguaje de programación C# (pronunciado C Sharp) ha tenido una constante y rápida evolución, permitiendo incorporar nuevas características donde se desarrolla aplicaciones de software más rápido que a su vez presenta código más atractivo semántica y sintácticamente. Aunado a la rápida evolución se destaca la apertura del compilador de C# por medio de una interfaz de programación de aplicaciones (API por sus siglas en inglés), facultando a los ingenieros a desarrollar aplicaciones que anteriormente no eran posibles de construir, abriendo de esta forma un nuevo camino en el campo de la investigación y desarrollo de aplicaciones y componentes dentro del ambiente de desarrollo integrado. El prototipo funcional utilizará las características proporcionadas por el lenguaje de programación a fin de poder crear un producto que sea eventualmente extensible.

1.4.3 Herramienta integrada en el ambiente de desarrollo

El componente de seguridad se instalará en el ambiente de desarrollo integrado denominado Visual Studio .NET. Tradicionalmente, los desarrolladores de software están familiarizados con componentes de terceras empresas que les permitan personalizar el ambiente, entre los que se incluyen componentes para generación de código, reingeniería, búsquedas, ordenamiento entre otras. Este componente utiliza el mismo concepto de las aplicaciones mencionadas anteriormente de forma tal que para el desarrollador de software será transparente el uso del mismo, reduciendo de esta forma la resistencia al cambio.

1.4.4 Acercamiento de nuevos clientes potenciales

La empresa Security Innovation recientemente adquirió a la empresa Safelight[3], lo cual le permite aumentar la cartera de clientes y la diversificación de los productos. Este enfoque permite que clientes potenciales puedan evaluar los productos existentes y posteriormente evaluar y utilizar el complemento de seguridad a desarrollarse. De forma periódica nuevos clientes solicitan periodos de evaluación de los productos desarrollados en Security Innovation y muchos de ellos se convierten a su vez en nuevos clientes.

[3]https://www.securityinnovation.com/company/news-and-events/press-releases/safelight-acquisition.html

1.5 Debilidades

1.5.1 Poca o nula inserción en el campo del análisis estático de código

Pese a que la empresa Security Innovation ha sido una institución respetable que ofrece los mejores productos de la industria para mejorar la seguridad en las aplicaciones de software, ha tenido poca si no es que nula participación en el campo de el análisis estático de código. Debido a ello, se ha integrado con herramientas de terceros que son los que verdaderamente realizan el análisis del código fuente y las mejores prácticas de la empresa son luego despegadas. Esta dependencia en herramientas de terceros es importante pero a su vez es una limitación, puesto que no se pueden comercializar los productos de forma independiente. Debido a la poca participación en este mercado, existen otras empresas que han formado un sólido ecosistema y que de forma más madura ofrecen sus productos a los clientes en varias industrias.

1.5.2 Dependencia de terceras empresas para realizar el análisis estático de código.

Existe una dependencia con otras empresas en el mercado para integrar productos en conjunto. Este enfoque es necesario pero no es suficiente. Cuando se trabaja en conjunto con otras organizaciones existen contratos de comercialización del producto que se deben respetar, al igual que el tema de licenciamiento. Esto obliga a que los clientes tengan dos productos instalados en lugar de tener uno solo. Muchas veces incluso relucen

problemas de compatibilidad entre las partes que generan la tendencia a errores y al trabajo coordinado. De igual forma si las herramientas de terceros de las cuales se depende tienen un costo elevado, pequeñas y medianas empresas no podrían cubrir el precio de las respectivas licencias y por ende no pueden hacer uso de la integración.

1.5.3 Proyectos de código abierto y gratuito ofrecen productos similares a muy bajo costo.

En algunos casos el factor económico es determinante para decidir si se adquiere o no un producto. Este proceso se convierte en un poco más complicado si existen herramientas de código abierto y gratuito, que si bien es cierto no se asemejan a los productos comerciales, tienen un funcionamiento un tanto parecido. Este factor genera que se prefiera implementar controles internos y políticas basados en estándares gratuitos . Las organizaciones que comercializan productos se ven confrontados cuando la calidad de los estándares y herramientas gratuitas es ampliamente reconocida.

1.5.4 Plugin limitado a un lenguaje de programación y a un entorno integrado de desarrollo.

El prototipo funcional del componente o plugin a desarrollarse está destinado al ambiente de desarrollo Visual Studio.NET y al lenguaje de programación C# . Esto significa que se limita a estas dos tecnologías y no abarca otras plataformas ampliamente utilizadas como lo es Java por ejemplo. Incluso impide que se realice el análisis de código en otros lenguajes dentro

de la plataforma de programación de Microsoft, como lo es el caso del lenguaje Visual Basic.NET.

1.6 Amenazas

1.6.1 La competencia ofrece productos similares e integrados

A pesar de los retos en el momento de implementar seguridad en las organizaciones, la necesidad de proteger los activos de la empresa es vital.

Por esta razón, se cuenta hoy con variedad de productos similares que le facultan a la industria mejorar e incluso identificar defectos de seguridad en el software que se desarrolla. Muchas de estas organizaciones, con años de prevalencia en el mercado, crean productos innovadores con el respaldo de calidad del fabricante y que son atractivas por su grado de confianza.

Dentro de la lista de organizaciones líderes en el mercado de la seguridad de las aplicaciones figuran IBM con el producto denominado AppScan, Hewlett-Packard ofrece el producto Fortify Software Security Center, Checkmarx a su vez de forma emergente ha ganado territorio incursionando con la herramienta llamada Static Code Analysis y la empresa WhiteHat ofrece el producto denominado WhiteHat Sentinel.

El enfoque proporcionado por las herramientas mencionadas anteriormente, brindan la posibilidad de realizar un análisis del código fuente, con el objetivo de encontrar vulnerabilidades y generan de esta forma un reporte con los hallazgos, eventualmente el desarrollador analiza

detalladamente el informe y procede con las correcciones dictadas por la herramienta.

Así mismo los clientes, usuarios de estas aplicaciones, se preocupan por implementar las herramientas que sean efectivas y que les brinden el grado de protección que ellos requieren. Por tal razón es normal que una empresa pueda escoger y evaluar más de un producto donde deberá decidir cuál le bridará una mejor opción.

1.6.2 Clientes prefieren productos unificados

Siempre existe la opción de que un cliente elija otro producto que satisfaga las necesidades y que le permita cumplir a cabalidad con sus objetivos estratégicos de la organización. De igual forma buscará aquellas soluciones de software que puedan ser implementadas y adaptadas fácilmente. Una herramienta unificada es aquella que ofrece múltiples y comunes operaciones sin necesidad de que el usuario necesite adquirir componentes de terceros, lo cual facilita entre muchas otras cosas los costos de licenciamiento.

En seguridad de aplicaciones por ejemplo, herramientas como lo son HP Fortify son ampliamente utilizadas a nivel corporativo porque aparte del respaldo comercial que existe, las empresas encuentran en esta herramienta un producto unificado donde pueden hacer análisis periódico del código fuente, dentro del mismo entorno de desarrollo integrado, incluso permitiendo que aplicaciones en tecnologías no adyacentes puedan ser analizadas de

forma transparente. El usuario a su vez puede decidir la periodicidad con la que el análisis es realizado, definir ciertas reglas intrínsecas de la organización, extender la funcionalidad de la herramienta y predefinir la forma en que los reportes son generados, todo esto desde la misma configuración de la herramienta.

De esta forma existe gran variedad de productos que tienen como objetivo brindar un ecosistema donde todas las tareas relacionadas con seguridad dentro del ciclo de vida del desarrollo del software estén disponibles ahí mismo, sin necesidad de adquirir otros productos.

Cuando existe complejidad en el momento de implementar una solución de software en la organización, se pierde la funcionalidad de la misma, ya que se crea una atmósfera negativa donde se ve como un verdadero reto poder utilizarla de forma adecuada. Esto se evidencia cuando no existe un producto unificado y problemas de compatibilidad.

Bajo esta perspectiva es claro notar que una empresa tienda a preferir productos maduros especializados en brindar soluciones a aplicaciones de software vulnerables a ataques informáticos.

1.6.3 Uso de tecnologías que no son suficientemente maduras.

El prototipo funcional centra sus características únicas en el nuevo compilador de C# denominado Roslyn. No obstante Roslyn es un producto relativamente nuevo que todavía se encuentra en una etapa conocida como vista preliminar de tecnología para la comunidad o CTP por sus siglas en inglés que obedece a Community Technology Preview. En dicha etapa del

software, la cual es provista a la comunidad, se pueden encontrar errores de diseño e implementación así como errores inesperados, que si bien es cierto son corregidos por el proveedor, puede que tarde mucho tiempo. También puede suceder que alguna funcionalidad requerida para el proyecto no esté todavía implementada ni disponible para el público en general.

Chaudhary (2010) cuando se refiere a Community Technology Preview indica:

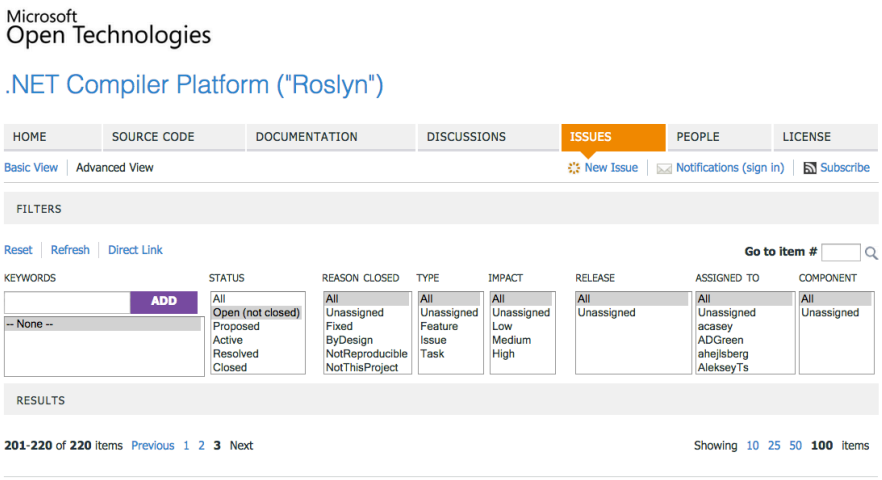
> Típicamente, una versión de software se compone de las siguientes fases : Definición, Planeamiento, Desarrollo, Estabilización e Hitos. Muchos productos de software utilizan variaciones de este proceso y la mayoría de las veces existen versiones públicas (alfa, beta, candidato a versión, versión final) que permiten que la calidad del producto aumente hasta llegar a los niveles aceptados. Algunas veces, existe un lazo estrecho con empresas pequeñas para recibir retroalimentación de forma temprana mientras las versiones públicas están disponibles para recibir retroalimentación de grandes audiencias.

En vista de la definición anterior se pude comprender que el hecho de que la plataforma Roslyn no esté lo suficientemente madura, existe el riesgo de que un algún error inesperado limite de cierta forma la funcionalidad del producto, esto se considera una amenaza para el proyecto, puesto que eventualmente podría causar un retraso considerado en el producto.

Incluso se podría suponer que la solución de un defecto técnico de la plataforma no se resolverá si no hasta que se siga el debido proceso y de esta forma se considera un factor determinante.

Tal como se ilustra en la siguiente imagen, con fecha Octubre de 2014 existen 220 asuntos reportados en el sitio oficial de Roslyn. Pese a que no todos esos asuntos listados corresponden a errores propiamente, si son solicitudes activas de usuarios que tratan de obtener un producto de mejor calidad.

Figura 6 Listado de defectos o asuntos creados para la plataforma Roslyn



Fuente : https://roslyn.codeplex.com/workitem/list/advanced?size=100

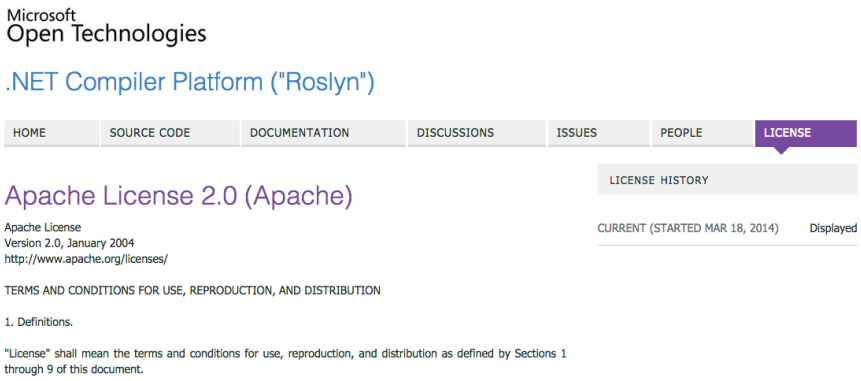
Aunado a ello se puede observar que la plataforma de compilación Roslyn es de código abierto; los desarrolladores en general pueden extender la plataforma y agregar mejoras sustanciales, no obstante existe el riesgo que

se incluya código fuente erróneo, sin pasar por alto las exhaustivas revisiones de código que puedan estar asociados a este proyecto.

El hecho de que permita la inclusión de código de otras personas y no necesariamente de Microsoft es un activo, pues se asegura que la versión a desarrollar es lo más cercano a las necesidades emergentes de los usuarios, no obstante siempre existe la posibilidad de que coexista algún evento inesperado, causado precisamente por un defecto en el código fuente.

En la imagen siguiente se muestra que la plataforma de compilación Roslyn se encuentra bajo la licencia de Apache 2.0 . Según se indica en el sitio oficial de Apache (2014) “La Fundación Apache Software usa varias licencias para distribuir software y documentación, aceptar colaboración regular de individuos y corporaciones y aceptar donaciones.”

Figura 7 Licencia Apache 2.0 de la plataforma de compilación Roslyn.



Fuente : https://roslyn.codeplex.com/license

CAPÍTULO II

MARCO TEÓRICO

En el capítulo presente, se consignan una serie de términos y conceptos fundamentales para el prototipo propuesto, cuyo significado es relevante para comprender mejor aspectos técnicos y específicos al prototipo.

Así mismo, este capítulo tiene como objetivo servir de referencia ante cualquier eventual término utilizado en este documento, de forma tal que exista una definición clara y concisa que ayude a trazar una hoja de ruta y un panorama explicativo de los términos aquí utilizados.

2.1 Sistemas de Información

Un sistema de información se compone de un conjunto de elementos, que tienen como objetivo común recolectar, almacenar procesar y analizar datos. Por medio de un sistema de información es que los datos almacenados y recopilados son debidamente procesados y convertidos en información, la cual es utilizada como base durante el proceso de toma de decisiones.

Los autores Kendall & Kendall (2011), refiriéndose a este tema indican que “Los sistemas de información se desarrollan para distintos fines, dependiendo de las necesidades de los humanos y la empresa.” (p. 2). En una sociedad globalizada como en la que se vive actualmente, los sistemas de información juegan un rol fundamental; tareas que en antaño se hacían de forma manual, con el advenimiento de los sistemas de información se han podido automatizar, permitiendo que exista un aprovechamiento consistente de recursos dentro de la organización.

Existen varios tipos de sistemas de información entre los que se destacan :

1) Sistemas de procesamiento de transacciones:

Dentro de esta categoría se agrupan los sistemas que realizan volúmenes inmensos de transacciones para la organización, como por ejemplo sistemas de procesamiento de datos de una entidad financiera. Los autores Kendall & Kendall (2011) sostienen que estos sistemas "…eliminan el tedio de las operaciones transaccionales necesarias y reduce el tiempo que se requeriría para hacerlo de forma manual" (p. 2).

2) Sistemas de automatización de oficinas :

Tienen como objetivo servir de soporte a aquellas persas que analizan la información y hacen ciertas transformaciones antes de poder distribuirlos. Por ejemplo se podría mencionar el caso de una persona que aprueba el pago de horas extra, ya que él debe transformar y manipular los datos para que el pago sea efectivo o no.

3) Sistemas de información administrativa:

Dentro de la categoría de sistemas de información administrativa se citan los sistemas que sirven de soporte a los empleados de una organización con el objetivo de hacer el trabajo más eficiente. Kendall & Kendall (2011) cuando se refieren a este tipo de sistemas indican que "Los sistemas de información administrativa producen información que se utiliza en el proceso de toma de decisiones." (p. 3).

4) Sistema de soporte de decisiones:

Son sistemas de información especializados cuyo objetivo es brindarle al usuario información de forma tal que él pueda tomar decisiones basado en la información disponible. Un aspecto importante a mencionar es que según los autores Kendall & Kendall (2011) estos sistemas "… está enfocado a brindar respaldo a la toma de decisiones en todas las fases, aunque el la decisión misma le corresponde de manera exclusiva al usuario". (p. 3).

5) Inteligencia artificial y sistemas expertos:

Este tipo de sistemas utilizan las técnicas utilizadas por la ingeniería artificial y que tienen como función ayudar de forma sustancial a las empresas en sus negocios. Kendall & Kendall (2011) establecen que "Los sistemas expertos son una clase muy especial de sistemas de información que han demostrado su utilidad comercial gracias a la disponibilidad extendida de software y hardware." (p. 2).

2.2 Desarrollo de Sistemas

El proceso de desarrollo de sistemas, considerado como una forma de arte para algunos autores entre los que figuran Donald E. Knuth en su reconocida obra de cuatro volúmenes denominada "The Art of Computer Programming", Addison-Wesley Professional 1997, contempla las diferentes etapas y metodologías necesarios para desarrollar una aplicación de software. Pese a que la ingeniería en computación es una rama de la ciencia relativamente nueva, el auge de sistemas de información en la industria y en la sociedad, amerita que exista un proceso unificado para el desarrollo de

sistemas eficientes que no solamente cumpla con los objetivos para los cuales fue diseñado si no que también sea resistente a fallos y recuperable ante cualquier eventualidad.

En el desarrollo de sistemas existen varias metodologías, a veces llamadas también procesos por varios autores tal es el caso de Sommerville(2011) el cual argumenta que “Un proceso de software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software.” (p. 28). El objetivo de la metodología es definir las actividades y la secuencia necesarias para la correcta implementación de sistemas. A continuación se muestran varios enfoques.

2.2.1 Modelo en Cascada (waterfall)

Este modelo propone una planificación detallada de las diversas actividades del proceso antes de trabajar en ellas. De esta forma se identifican las actividades fundamentales del desarrollo de sistemas tales como desarrollo, validación de datos, recopilación de requerimientos, ejecución de pruebas y luego se representan como fases separadas. El autor Sommerville (2011) identifica cinco fases o etapas en el modelo en cascada, las cuales son:

a) Análisis y definición de requerimientos : Tomando en consideración a los usuarios del sistema, se definen las restricciones y los objetivos del sistema.

b) Diseño del sistema : Sommerville (2011) afirma que “El proceso de diseño de sistemas asigna los requerimientos, para sistemas de hardware o de software, al establecer una arquitectura global” (p. 31). En este proceso de diseño se busca crear una abstracción e identificación de los componentes del sistema y como interactúan entre ellos.

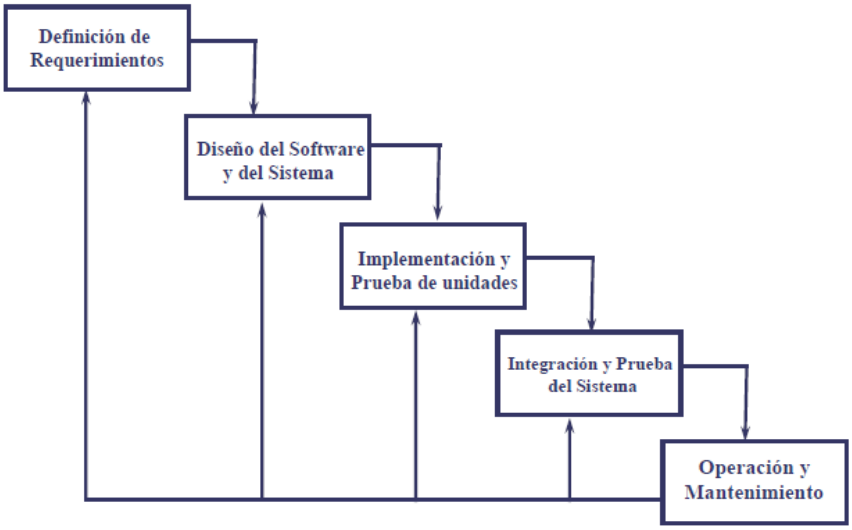
c) Implementación y prueba de unidad : En esta fase el diseño de software se implementa a manera de un conjunto de programas. Osherove (2009) cuando se refiere al concepto de pruebas de unidad afirma que “Una prueba de unidad es una pieza de código automatizada que invoca el método o clase que está siendo probada y luego valida algunas asunciones del comportamiento de la clase” (p. 11). Así mismo Sommerville (2011) supone que “La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.” (p. 31).

d) Integración y prueba de sistema : Durante la fase de integración los diferentes módulos o subprogramas se conciben como un único sistema y se realiza un proceso de pruebas completo a fin de determinar que se haya cumplido con los requerimientos iniciales.

e) Operación y Mantenimiento : En esta fase el sistema está listo para ser implementado y utilizado por los usuarios. Para Sommerville (2011) “…ésta es la fase más larga del ciclo de

vida, donde el sistema se instala y se pone en práctica" (p. 31). Durante las etapas de mantenimiento se busca solventar errores presentados en el momento de utilizar el sistema y además se utiliza para agregar mejoras sustanciales y desarrollo de nuevos requerimientos.

Figura 8 Modelo en cascada (Waterfall)



Fuente : Ingeniería de Software, Novena Edición Pearson 2011

2.2.2 Desarrollo Incremental

Este modelo aboga por realizar una implementación inicial, la cual es expuesta para que el usuario brinde retroalimentación y comentarios; una vez recibida esta retroalimentación, se desarrollan diferentes versiones del producto hasta que se consigue un producto adecuado.

Según Sommerville (2011) “El desarrollo incremental refleja la forma en la que se resuelven los problemas”.(p. 33). Este mismo autor también indica que el desarrollo incremental es mejor que el modelo en cascada debido a que “… resulta más barato y fácil de realizar cambios en el software conforme este se diseña.” (p. 33).

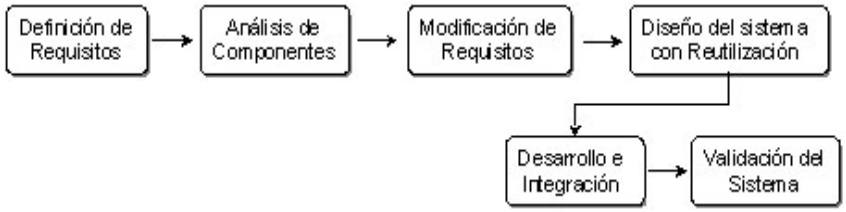
2.2.3 Ingeniería de software orientada a reutilización

Este modelo de desarrollo de sistemas se asemeja bastante a otros modelos existentes, con la diferencia de que aboga por la reutilización de componentes, de forma que no exista duplicidad. Sommerville (2011) es claro cuando indica que:

> La ingeniería de software orientada a la reutilización tiene la clara ventaja de reducir la cantidad de software a desarrollar y, por lo tanto, la de disminuir costos y riesgos; por lo general, también conduce a entregas más rápidas de software. Sin embargo, son inevitables los compromisos de requerimientos y esto conducirá hacia un sistema que no cubra las necesidades reales de los usuarios. (p. 36).

En la siguiente imagen se aprecia la ingeniería de software orientada a reutilización, nótese que existe una fase denominada diseño de sistemas con reutilización, donde se contempla poder reutilizar código, diseños, flujos de trabajo entre otros.

Figura 9 Ingeniería de software orientada a la reutilización



Fuente : Ingeniería de Software, Novena Edición Pearson 2011

2.3 Plugin

Un plugin es una extensión o complemento desarrollado para solventar una necesidad específica dentro de una tecnología determinada. Es generalmente utilizado en los entornos de desarrollo integrado como Eclipse o Visual Studio y además en navegadores o agentes de usuario más populares entre los que figuran Google Chrome y Mozilla Firefox. El objetivo de estos complementos es extender la funcionalidad o proveer más herramientas de las que están comúnmente disponibles a fin de realizar tareas específicas. En la actualidad existe gran diversidad de estas extensiones que permiten por ejemplo buscar y reemplazar texto, convertir texto de minúscula a mayúscula y viceversa, encontrar texto con contenido similar entre muchas otras cosas.

La Fundación Eclipse(2003) argumenta que un plugin es "…un componente que provee cierto tipo de servicios dentro del contexto de Eclipse."

En la siguiente imagen se muestran algunos ejemplos de extensiones para el navegador Firefox.

Figura 10 Extensiones disponibles para Firefox

Fuente: https://addons.mozilla.org/es/firefox/

2.4 Prototipo

Sommerville (2011) define prototipo como "Una versión inicial de un sistema de software que se usa para demostrar conceptos, tratar opciones de diseño y encontrar más sobre el problema y posibles soluciones." (p. 45).

Partiendo de la definición anterior se observa que el prototipo es una herramienta de suma importancia pues permite tener una mejor percepción

del problema a resolver y de esta forma encontrar soluciones a problemas de diseño e incluso encontrar en una etapa temprana alguna complicación en el futuro. Según afirma Somerville (2011) “Los prototipos no tienen que ser ejecutables para ser útiles” (p. 46).

2.5 Tecnologías de Información y Comunicaciones

Son un conjunto de normas que agrupan elementos de infraestructura que permiten hacer más ágil y eficiente la comunicación entre los seres humanos, facultándolos para que puedan realizar sus actividades diarias de forma oportuna. Dentro de este modelo se sitúan las personas, los elementos de infraestructura tales como servidores, redes inalámbricas y telefonía entre otros elementos cuyo objetivo común es poner al alcance de los seres humanos mecanismos para que se comuniquen mejor y por ende mejorar las relaciones humanas.

Los autores Monge, R., Hewitt, J (2004) refiriéndose a las tecnologías de información y comunicaciones indican que “… las Tecnologías de Información y Comunicaciones (TICs) hacen alusión a los medios e instrumentos que se emplean para ser posible la transmisión de voz, datos, texto e imágenes en forma digital.”

2.6 Software

Es un término genérico que hace referencia a el conjunto de programas que forman parte de una computadora. Pese a ser un anglicismo, el

diccionario de la Real Academia de la Lengua Española (2001) define el software como el “Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora”. Basado en esta definición, se puede interpretar que el software son aquellos componentes intangibles de una computadora.

Para Tanenbaum (1998) “El programa de sistema más fundamental es el sistema operativo que controla todos los recursos de la computadora”. Partiendo de la definición que brinda el autor, se reconoce que el sistema operativo dentro de una computadora es el sistema más importante ya que sus tareas están relacionadas con el manejo de los recursos y con permitir que nuevas aplicaciones sean instaladas y ejecutadas.

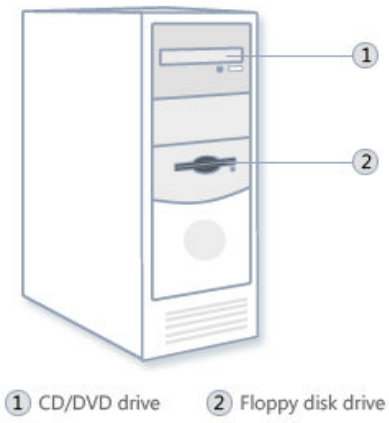
2.7 Hardware

Se entiende por hardware todos los elementos tecnológicos que pueden ser percibidos por medio de los sentidos, tal es el caso de los servidores físicos, computadoras portátiles, impresoras, teclados, monitores dispositivos de red entre otros. El hardware tiene un rol fundamental en la tecnología pues habilita al usuario a realizar mejor sus tareas.

Microsoft (2014)[4] define Hardware al indicar que “Las partes físicas, las cuales usted puede ver y tocar, se denominan de forma colectiva como hardware.”

[4] http://windows.microsoft.com/en-us/windows/computer-parts#1TC=windows-

Figura 11 Ejemplo de Hardware

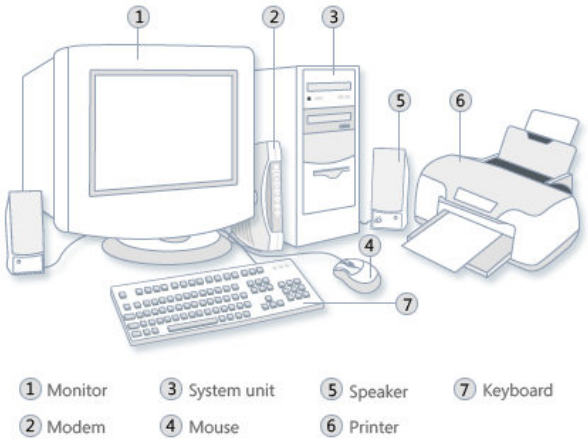


Fuente: Microsoft http://goo.gl/8BrfXK

2.8 Computadora

Una computadora es una máquina que tiene como función básica capturar datos de entrada, procesarlos y emitir una salida. Stroustrup (2009) afirma que “Las computadoras son construidas por personas para que sean usadas por personas” (p. 21), luego indica que “Una computadora es una herramienta bastante genérica; puede ser usada para un rango inimaginable de tareas. Hace que un programa sea útil para alguien” (p. 21). En la imagen siguiente se puede apreciar una computadora ordinaria que fácilmente se encuentra en un hogar.

Figura 12 Computadora Ordinaria



Fuente: Microsoft http://goo.gl/8BrfXK

2.9 Aplicaciones de software

Son programas informáticos desarrollados con el objetivo de ayudar a resolver un problema del negocio por medio de una computadora y que están desarrollados en una tecnología específica. Estas aplicaciones de software pueden ser desarrolladas a lo interno de las organizaciones o puede ser subcontratadas.

Mishra, J., Mohanty, A (2011) indican que :

> Las aplicaciones de Software aplican el poder de la computadora para resolver problemas al desempeñar tareas específicas. Pueden apoyara a individuos, grupos, y organizaciones. La compañías pueden personalizar las aplicaciones de software, comprar programas

existentes o usar una combinación de software personalizado y adquirido.

Hojas electrónicas de cálculo, procesadores de palabras, software gráfico y software integrado son aplicaciones de software de propósito general. Programas de cómputo para el pago de planillas, manejo de inventario y análisis de ventas son aplicaciones de software específicas. (p. 3).

2.10 Usuario malicioso

Es un usuario experto de la tecnología pero que busca la forma de darle un uso inapropiado a la misma, con el objetivo de descubrir alguna vulnerabilidad existente y comprometer la seguridad del sistema, alterar o robar información considerada sensitiva para la organización y comprometer a otros usuarios del sistema.

Los autores Elden, C. , Swaminatha, T (2003) acotan que “un usuario malicioso es un individual o grupo que tiene el conocimiento, las habilidades o el acceso a comprometer la seguridad de un sistema.”

2.11 Hacker

Según Erickson (2008) el término hacker se refiere a aquella persona que tiene un modo de pensar distinto al de la mayoría cuando indica que “La esencia de hacking consiste en la búsqueda de usos no previstos o pasados por alto por un usuario normal”. Tomando como base la definición que el autor

brinda, se puede determinar que un hacker no es un criminal que comete delitos informáticos, si no es más bien una persona que tiene un modo de pensar distinto al convencional y que aprovecha esta capacidad para alcanzar grandes logros.

2.12 Activo

Un activo son aquellos componentes que tienen un valor para la organización y que la pérdida del mismo o el deterioro causa una interrupción de las operaciones e impide que una organización pueda alcanzar sus metas.

En una época como la actual la información de una empresa es quizá el activo más importante y por ende es necesario implementar controles para proteger el acceso no autorizado a la información . Según afirma Paul (2011) “Los activos pueden ser tangibles e intangibles en su naturaleza”, luego el mismo autor indica que “Los activos tangibles son aquellos que pueden ser percibidos por los sentidos”.(p. 16).

2.13 Vulnerabilidad

Paul (2011) define el concepto de vulnerabilidad como “Una debilidad que puede ser ejecutada accidentalmente o explotada intencionalmente por un atacante, resultando en la brecha de la política de seguridad”. Como lo indica el autor, una vulnerabilidad está asociada al concepto de debilidad o falla que es causada de forma accidental o intencional. Cuando se habla de vulnerabilidades en el código de fuente se habla por lo tanto a fallas

introducidas por los desarrolladores durante el proceso de desarrollo del software y que son aprovechadas por usuarios maliciosos para comprometer al sistema y a la organización.

2.14 Ataque informático

El autor y experto en seguridad Paul(2011) define el concepto de ataque al mencionar que

> Los agentes de riesgo pueden intencionalmente causar que un riesgo se materialice o que las amenazas puedan ocurrir como resultado de un error de usuario o que haya sido descubierto accidentalmente. Cuando los agente de amenaza de forma activa o intencional causan que un riesgo se materialice, se le conoce como un ataque. (p. 18).

En la definición anterior se aprecia como un ataque está relacionado a una actividad que se hace de forma intencional y que tendrá efectos adversos en las organizaciones son víctimas de estos ataques.

2.15 Probabilidad

En términos de riesgo, la probabilidad es la oportunidad de que una amenaza particular ocurra. Paul(2011) afirma que “La probabilidad se expresa como un percentil, no obstante algunas veces se utilizan técnicas meramente heurísticas por lo cual algunas organizaciones utilizan categorías como medio alto y bajo” (p. 19).

2.16 Impacto

Es el resultado de una amenaza materializada, la cual puede tener efectos adversos para una empresa como lo son la interrupción temporal de los servicios, pérdidas económicas y el robo de información sensitiva.

Para Paul(2011) el impacto se define como “La extensión de la gravedad en la alteración de la capacidad de la organización para lograr los objetivos se denomina impacto”.

2.17 Factor de exposición

Paul (2011) define el término de factor de exposición como “… la oportunidad de una amenaza de causar pérdidas” (p.18). Este parámetro juega un papel muy importante en el momento de calificar el riesgo.

2.18 Controles

Un control es un mecanismo para mitigar una amenaza, el cual puede ser técnico, administrativo o físico. En términos de seguridad de aplicaciones Paul(2011) indica que algunos ejemplos de controles lo constituyen “…validación de datos de entrada, control del código fuente, controles de acceso controlados y supervisados” (p. 19).

2.19 Políticas de seguridad

Paul (2011) se refiere a una política de seguridad cuando expresa que “…Es el instrumento por medio del cual los activos digitales que requieren

protección pueden ser identificados". (p. 26). Según la definición anterior se observa que el objetivo de una política de seguridad es responder a la pregunta de qué es lo que se quiere proteger y las consecuencias de no protegerlos. Bajo esta perspectiva se puede concluir que identificando los activos más críticos de la organización es vital.

2.20 Crimen cibernético

Se refiere a acciones poco éticas y criminales dirigidas hacia una entidad u organización a fin, con el objetivo de causar una interrupción de las operaciones, generalmente asociadas con pérdidas económicas, robo de información sensitiva y afectación de la imagen, sin dejar de lado las repercusiones legales en las cuales las empresas pueden verse inmersas luego de un ataque cibernético.

La empresa de seguridad y proveedora de software antivirus denominada Norton, en su sitio oficial brinda una definición acertada de crimen cibernético al indicar que :

> La definición de crimen cibernético que se incluye en el manual de Prevención y Control de los Crímenes Informáticos de las Naciones Unidas engloba fraude, falsificación y acceso no autorizado [Naciones Unidas,1995].
>
> Como puede apreciarse en dichas definiciones, el crimen cibernético puede englobar un abanico muy amplio de ataques. Es importante comprender esta amplia variedad de tipos de crímenes cibernéticos, ya que es necesario adoptar distintos planteamientos ante estos distintos

> tipos de actividades criminales cibernéticas a fin de poder mejorar la seguridad de su equipo. (Symantec 2014).

2.21 Microsoft

Empresa norteamericana dedicada al desarrollo y comercialización de aplicaciones de software y sistemas operativos. La empresa fue fundada en 1985 por Bill Gates y Paul Allen. La empresa Microsoft desarrolla y mantiene la plataforma de desarrollo denominada Microsoft .NET, la cual permite que ingenieros en computación puedan crear software escalable.

La revista Forbes (2014) la ubica dentro de la lista de denominada Global 2000 Leading Companies, y brinda la siguiente definición de la empresa:

> Microsoft Corp. desarrolla y comercializa software, servicios y hardware que entrega nuevas oportunidades, mayor conveniencia y mejorado valor en la vida de las personas. Los productos de la compañía incluyen sistemas operativos para computadoras personales, servidores, teléfonos y otros dispositivos inteligentes.

2.22 Hewlett Packard

Es una empresa estadounidense la cual también se conoce bajo el acrónimo de HP, fue fundada en 1939. La empresa ha sido un gigante en el mercado informático creando computadoras personales, servidores,

impresoras y dispositivos móviles entre otros. HP también participa activamente en el mercado de seguridad de aplicaciones.

La revista Forbes (2014) define los productos y servicios de la empresa HP al indicar que:

> Hewlett Packard Co. brinda productos, tecnología, software, soluciones y servicios para clientes individuales, negocios pequeños y medianos y empresas grandes , incluyendo clientes en el sector del gobierno, salud y educación. La empresa opera bajo siete segmentos del negocio: Sistemas Personales, Impresión, Grupo Empresarial, Servicios Empresariales, Software, HP Servicios Financieros e Inversión Corporativa.

2.23 HP Fortify

Es una herramienta que permite hacer análisis estático de código fuente, brindándole al usuario un reporte detallado de los problemas de seguridad que tiene la aplicación antes de que dicha aplicación sea publicada en un ambiente de producción. Una vez que se han direccionado las vulnerabilidades mostradas por la herramienta, se proceden con las distintas etapas del ciclo de vida del desarrollo de software. El objetivo principal de la herramienta es servir de ayuda a mitigar riesgos asociados al diseño y a la programación de la aplicación.

En el sitio oficial de HP Fortify (2014) se define la herramienta como :

HP Fortify analizador de código le ayuda a verificar que el software es confiable, reduce costos, incrementa la productividad e implementa mejores prácticas de programación segura.

El analizador estático de código fuente realiza un escáner del código fuente, identifica las causas de la vulnerabilidades de seguridad y correlaciona y prioriza los resultados.

2.24 IBM

IBM es el acrónimo de International Business Machines Corporación, una empresa norteamericana la cual provee productos de software y hardware para la industria informática. Fue fundada en el año 1911 y ha sido una empresa muy importante en la evolución de la computación moderna.

El diario estadounidense The New York Times define a IBM como “Es una compañía de tecnología de información, la cual opera en cinco segmentos. El software que produce consiste principalmente en software para sistemas operativos y middleware.”

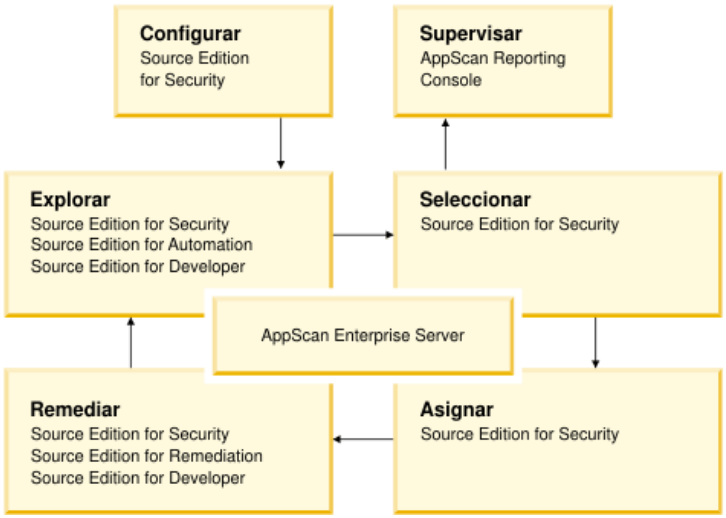
2.25 IBM Security AppScan

Es una herramienta creada por la empresa IBM, según el sitio oficial [5] del producto lo definen como “IBM Security AppScan mejora la seguridad de las aplicaciones Web y aplicaciones móviles, mejora los programas de seguridad y fortalece los cumplimientos de las normativas”.

[5] http://www-03.ibm.com/software/products/en/appscan

La imagen siguiente muestra el flujo de trabajo que ejecuta la herramienta para determinar los problemas de seguridad en el código fuente.

Figura 13 Flujo de trabajo de IBM AppScan



Fuente: http://goo.gl/rJPIVj

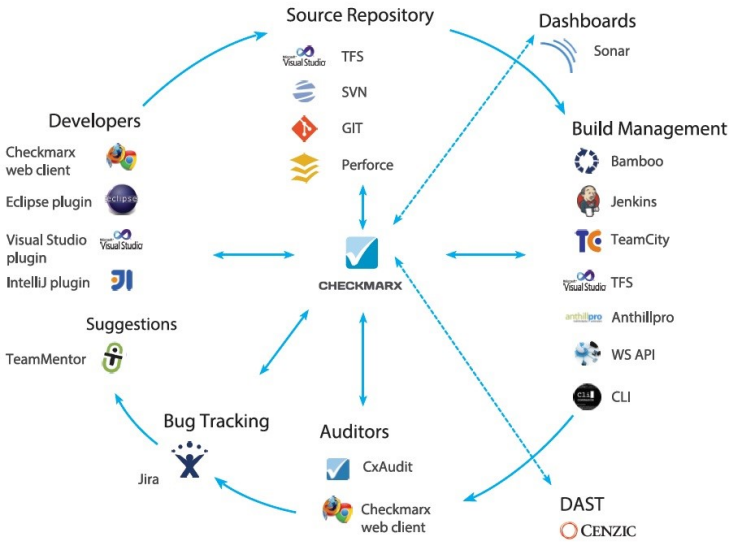
2.26 Checkmarx

Es una empresa basada en Tel Aviv, Israel la cual se dedica a proveer herramientas para realizar análisis de código fuente a fin de encontrar vulnerabilidades y que sean se corrijan en etapas tempranas. En términos de análisis estático de código, Checkmarx ofrece el producto denominado Checkmarx Suite el cual brinda extensiones para diversos entornos de desarrollo integrados.

En el artículo que lleva por nombre Checkmarx CxSuite Overview (2014) se define al producto Checkmarx Suite como “Es una solución única de análisis de código fuente que provee herramientas para identificar, seguir y reparar fallas lógicas en el código fuente tales como vulnerabilidades de seguridad.”

La siguiente imagen muestra la participación de Checkmarx en las diversas etapas del desarrollo de software.

Figura 14 Checkmarx en el ciclo de vida del desarrollo del software.



Fuente : http://goo.gl/uq9KNv

2.27 Instituto Ponemon

Es un instituto fundado en el año 2002 por el doctor Larry Ponemon y es considerado como una de las instituciones más importantes dedicado a temas como lo son privacidad, protección de datos y políticas de seguridad.

Fraim, D., Kane, K (2014),bajo la publicación que lleva por título Global Cost of Data Breach Increases by 15 percent, According to Ponemon Institute, cuando se refieren al Instituto Ponemon lo definen como :

> El Instituto Ponemon está dedicado a investigación independiente y educación que los avances de seguridad de información, protección de datos, la privacidad y las prácticas responsables dentro de la empresa y l los gobiernos de todo el mundo.

Tal como se aprecia, dicho Instituto es ampliamente reconocido por los estudios que constantemente genera y así brinda un panorama claro de la situación actual de temas como la seguridad de aplicaciones.

2.28 Security Innovation

Es una empresa estadounidense enfocada en ayudarle a las organizaciones a construir y mantener software seguro, dicha empresa se dedica a la seguridad del software brindando una gama de productos facultando a organizaciones a comprender, manejar y evaluar el riesgo asociado a aplicaciones de software inseguro, así como brindar herramientas para resolver problemas de seguridad..

El sitio oficial Security Innovation cuando describe a la compañía añade que “Somos un equipo de ingenieros de primera clase, desarrolladores, analistas de seguridad y analistas de negocio los cuales de forma conjunta solucionamos problemas del negocio mediante soluciones técnicas. ”.

2.29 MITRE

La fundación MITRE es una organización internacional sin fines de lucro que opera, investiga y desarrolla centros patrocinados por el gobierno federal de los Estados Unidos.

En el sitio oficial de la fundación MITRE (2014) se especifica que “MITRE es una corporación privada, sin fines de lucro que opera los FFRDCs (federally founded research and developments centers)”.

En el sitio oficial de la corporación se indica que MITRE apoya al gobierno de los Estados Unidos mediante los siguientes mecanismos :

- Investigación científica y análisis.
- Desarrollo y adquisición.
- Ingeniería en sistemas e integración.

MITRE por su parte, es la organización encargada de mantener el diccionario para vulnerabilidades de software denominado CWE.

2.30 CWE

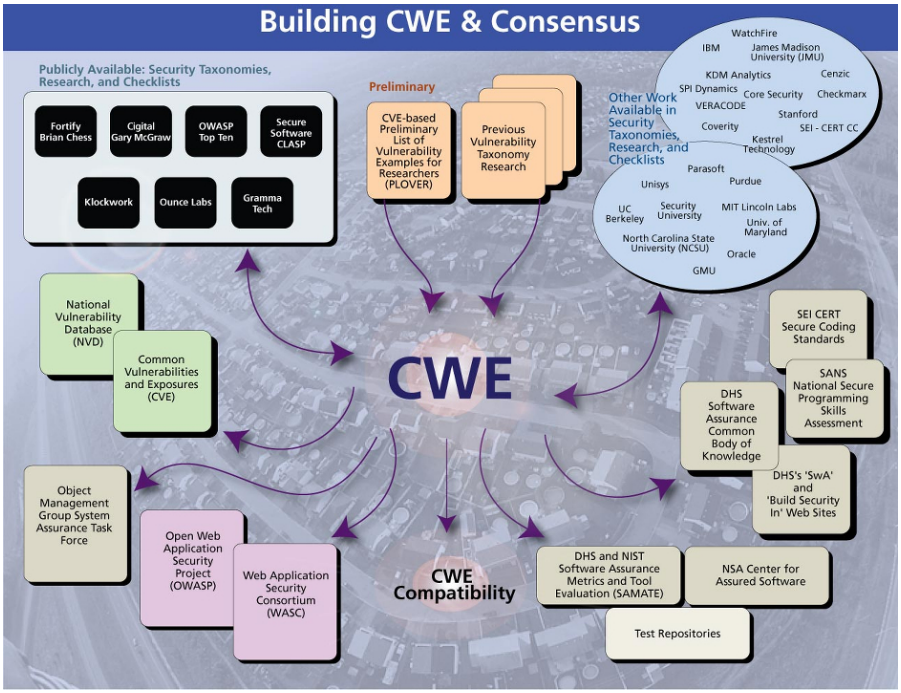
CWE es el acrónimo de Enumeración de Debilidades Comunes, por sus respectivas siglas en Inglés. El cual es un diccionario creado por la comunidad donde se hospedan todos los tipos de debilidades del software.

Tal como se indica en el sitio oficial de CWE (2014) :

> CWE proporciona un conjunto unificado y medible de debilidades que están habilitando discusiones efectivas, descripción, selección, y uso de las herramientas de seguridad del software y servicios que puedan encontrar estas debilidades en el código fuente y sistemas operacionales así también como un mejor entendimiento y manejo de l as debilidades del software relacionadas a arquitectura y diseño.

Según se aprecia en la definición anterior, este diccionario es muy importante en el marco del desarrollo seguro, puesto que su mantenimiento es proporcionado por la comunidad.

Figura 15 Flujo de trabajo de un CWE



Fuente: http://cwe.mitre.org/

2.31 TEAM Mentor

TEAM Mentor es una herramienta que funciona majo el enfoque de ser una guía especializada en mejores prácticas para el desarrollo de aplicaciones de software resistentes a ataques informáticas. Esta plataforma es desarrollada y soportada por la empresa Security Innovation.

La empresa Security Innovation (2014) define TEAM Mentor al indicar que :

> TEAM Mentor es una referencia en tiempo real, el repositorio más grande del mundo en el conocimiento del software seguro, se integra con herramientas de análisis estático de código fuente, ayudando organizaciones a desarrollar software más seguro, fácil de adaptarse a las políticas de seguridad y reduciendo vulnerabilidades en aplicaciones, reduciendo de esta forma el riesgo de la organización.

2.32 Entorno de desarrollo Integrado

Un entorno de desarrollo integrado o comúnmente denominado IDE por sus siglas en inglés, es una aplicación de software desarrollada para que ingenieros en sistemas puedan a su vez escribir aplicaciones de software.

Como su nombre lo refleja, es integrado y los usuarios encuentran en la herramienta un sin fin de módulos pequeños que cumplen tareas particulares entre las que se identifican el poder contar las líneas de código, hacer reingeniería, buscar ocurrencias, dar formato al código fuente, agregar componentes de terceros entre otros.

El experto Lair (2012) cuando se refiere al concepto de IDE expresa que “Un IDE es una aplicación de software que contiene facilidades comprensivas para ayudar a los desarrolladores a construir sus aplicaciones” (p. 19). Apoyándonos en la definición previa del autor se confirma la importancia de estas herramientas durante el proceso del desarrollo del software.

2.33 Visual Studio .Net

Visual Studio .Net es un entorno integrado de desarrollo creado por Microsoft por medio del cual los desarrolladores pueden escribir aplicaciones empresariales distribuidas y escalables bajo diversos lenguajes de programación. La versión más reciente de este ambiente integrado es Visual Studio .Net 2014.

Lair (2012) define Visual Studio como “Visual Studio es un IDE que permite a los desarrolladores .NET implementar una variedad de soluciones dentro de los confines de un editor.” (p. 19).

2.34.NET Framework

Thai y Lam (2003) cuando hacen referencia a la plataforma .NET Framework indican que:

> .NET Framework es una plataforma de desarrollo que provee una nueva interface de programación de servicios Windows y que integra un número de tecnologías que emergieron de Microsoft durante la década de 1990. Microsoft anunció la iniciativa .NET en Julio de 2000. En Abril de 2003, la versión 1.1 integral de .NET Framework fue liberada. (p. 6).

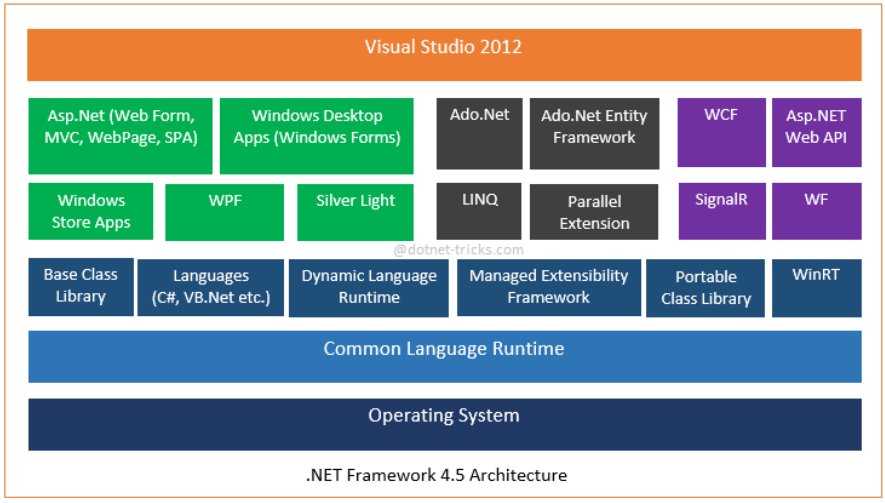
Según los mismos autores, la plataforma .NET consta de cuatro grupos de productos separados entre los que se destacan :

a) Herramientas de desarrollo y librerías.
b) Servicios Web.
c) Servidores especializados.
d) Dispositivos.

En vista de que es una plataforma de desarrollo, incluye herramientas predefinidas para tareas habituales y provee un modelo extensible para que la industria cree sus propios productos utilizando los fundamentos básicos previstos por la tecnología.

La imagen siguiente muestra una hoja de ruta de la plataforma Microsoft .NET 4.5.

Figura 16 Arquitectura de Microsoft .NET 4.5



Fuente : DotNet-Tricks http://goo.gl/UwZi8J

2.35 Lenguaje de programación C#

C# (léase C Sharp) es un lenguaje de programación de alto nivel, de propósito general y orientado a objetos que permite desarrollar múltiples tipos de aplicaciones entre las que se destacan aplicaciones de escritorio,

aplicaciones Web, aplicaciones móviles entre otras. La sintaxis y semántica de este lenguaje de programación hace que sea similar a otros lenguajes de propósito general como lo es Java.

Hejlsber, Torgersen, Wiltamuth & Golde (2010) brindan una explicación más amplia al establecer que :

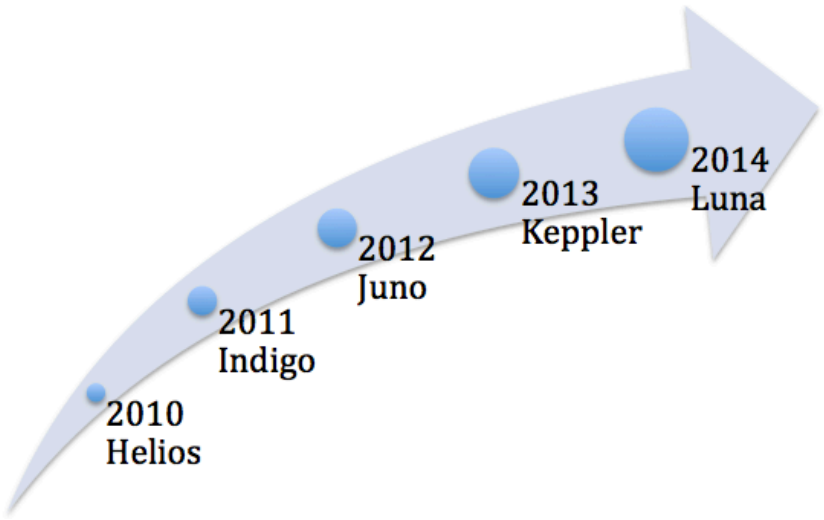
> C# es un lenguaje de programación simple, moderno, orientado a objetos y de tipos de datos seguro. C# tiene sus raíces en la familia de lenguajes C y es inmediatamente familiar a desarrolladores de C, C++ y Java. C# está estandarizado por ECMA internacional por medio del estándar ECMA-334 y por ISO/IEC por medio del estándar ISO/IEC 23270 .(p. 1)

2.36 Eclipse

Eclipse es un entorno integrado de desarrollo (IDE), el cual es soportado por la fundación Eclipse. Actualmente en el mercado se encuentran diferentes versiones de Eclipse entre las que figuran Luna, Kepler, Índigo, Helios, Galileo, Juno entre otras. Además el nombre Eclipse corresponde a una comunidad, según la Fundación Eclipse (2014) “Eclipse es una comunidad para individuos y organizaciones que desean colaborar en software de código abierto que sea comercialmente amigable.”

En la siguiente imagen se muestra la evolución y las versiones del IDE Eclipse de los últimos años.

Figura 17 Versiones de Eclipse durante los últimos años



Fuente: Propia

2.37 SQL (Structured Query Language)

El Lenguaje de Consulta Estructurado SQL por sus siglas en inglés corresponde a un lenguaje de programación utilizado en bases de datos transaccionales. Utilizando sintaxis y semántica muy similar al lenguaje humano, se puede interactuar con datos almacenados en un repositorio o base de datos SQL.

IBM (2014) define SQL como:

SQL es un lenguaje estandarizado para la definición y manipulación de datos en una base de datos relacional. De acuerdo con el modelo relacional de los datos, la base de datos se trata como un conjunto de tablas, las relaciones se representan por los valores en las tablas, y los datos se recuperan especificando una tabla de resultados que se pueden derivar de una o más tablas base.

Las sentencias SQL son ejecutadas por un administrador de base de datos. Una de las funciones del administrador de base de datos es transformar la especificación de una tabla de resultados en una secuencia de operaciones internas que optimizan la recuperación de datos .

La transformación se produce en dos fases: preparación y vinculante. Todas las sentencias de SQL ejecutables deben prepararse antes de que puedan ser ejecutados . El resultado de la preparación es la forma ejecutable u operacional de la declaración.

CAPÍTULO III

MARCO METODOLÓGICO

Las aplicaciones de software juegan un papel fundamental en la sociedad moderna en que se vive actualmente. Cuesta trabajo identificar algún sector social que no se vea beneficiado con el advenimiento de la

computación y aplicaciones a la medida. Es claro que el software evoluciona y de una forma muy acelerada.

En vista de tal evolución, se hace estricto la utilización de mecanismos que ayuden a crear software que se ajuste a las necesidades de los clientes y de tal forma les permita solucionar una problemática de negocio. Esposito, Saltarello (2009) citan una frase de la doctora Pamela Zave donde se indica que “El propósito de la ingeniería del software es controlar la complejidad, no crearla”.

En el presente capítulo se tiene como objetivo dar a conocer las diferentes metodologías existentes para, de una forma asertiva, desarrollar aplicaciones que se adapten a las exigencias y requerimientos de los usuarios finales. De igual forma se define el método de investigación, el tipo de investigación, las variables , población, muestra e instrumentos de recolección de datos utilizados en el presente proyecto informático, los cuales son un pilar fundamental en el momento de brindar una solución de software.

3.1 Métodos de Investigación.

Kendall & Kendall (2011) brindan una descripción bastante interesante de lo que es una investigación al establecer que :

> Investigar es describir y analizar información. Al investigar evidencia en una organización, el analista actúa como Sherlock Holmes, el famoso detective.
>
> A medida que el analista de sistemas trabaja para entender a los usuarios, su organización y sus requerimientos de información, debe examinar los distintos tipos de datos que ofrecen información no disponible por otro método de recolección. (p. 136).

Es interesante notar que, aunque la investigación se atribuye a muchas áreas a fines, el desarrollo de software también se beneficia con dicha práctica donde se tiene como objetivo poder tener mejor certeza del software que se desarrolla.

El autor Leedy (1993) indica que “La metodología de la investigación trasciende las limitaciones de un área específica, es un acercamiento a la conducción de un proyecto de investigación.”

3.1.1 Método Científico

3.1.2 Método Inductivo

3.1.3 Método Deductivo

Laica 10/23/2014 8:15 PM
Comentario [1]: Debe completar

3.1.4 Método Cuantitativo

Briones (1996) a propósito del método de investigación cuantitativo asegura que :

> El término investigación, que en general, significa indagar o buscar, cuando se aplica a las ciencias sociales, toma la connotación específica de crear conocimientos sobre la realidad social, es decir, sobre su estructura, las relaciones entre sus componentes, su funcionamiento, los cambios que experimenta el sistema en su totalidad o en esos componentes.

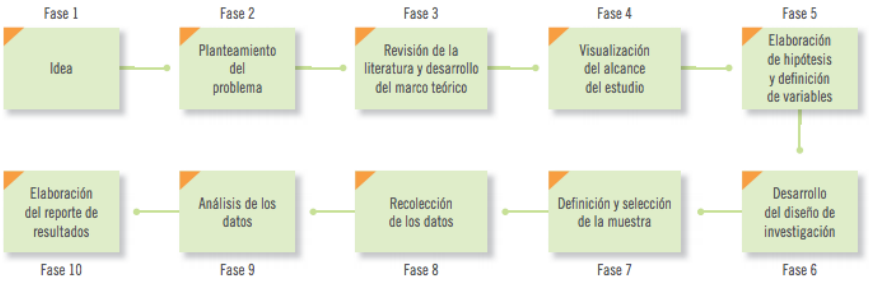
Luego el mismo autor interviene para aclarar que la investigación cuantitativa "…utiliza preferentemente información cuantitativa o cuantificable para describir o tratar de explicar los fenómenos que estudia, en las formas que es posible hacerlo en el nivel de estructuración lógica en el cual se encuentran las ciencias sociales actuales." (p. 17).

Como complemento, Kendall & Kendall (2011) también indican que "Hay muchos documentos cuantitativos disponibles para su interpretación en cualquier empresa" (p. 136). Precisamente, los autores Kendall & Kendall sostienen que dentro de tales documentos se encuentran :

- Informes para la toma de decisiones : Incluye aquellos informes donde se muestra información acerca del estado actual de la empresa, como es el caso de inventarios, reportes de ventas y producción.
- Informes de Rendimiento : Kendall & Kendall (2011) indican que “La mayoría de informes de rendimiento consisten en una comparación entre el rendimiento actual y el esperado” (p.136).
- Registros : El rol fundamental del registro es que permite tener una recopilación histórica de datos, los cuales si se hace de forma correcta, son actualizados de forma periódica.
- Formularios de Captura de Datos : Son los distintos formularios que utiliza la organización para recopilar datos, es importante comprender los tipos de datos que se solicitan para así poder modelar un sistema que de igual forma permita la recolección de tales piezas de información.

La siguiente figura ha sido extraía del libro Metodología de la Investigación del autor Sampieri e ilustra el proceso de la investigación cuantitativa.

Figura 18 Investigación cuantitativa



Fuente : Metodología de la investigación, Roberto Hernández Sampieri

3.1.4 Método Cualitativo

Con el objetivo de brindar una definición amplia de lo que en realidad es el método cualitativo es necesario citar a Sánchez (2008), donde según su punto de vista nos muestra la siguiente definición :

> Cuando hablamos de metodología cualitativa nos referimos, en su más profundo sentido, a la investigación que origina datos descriptivos : las propias palabras de las personas, habladas o escritas, y la conducta observable. En analogía con la metodología cuantitativa, consiste en más que un conjunto de técnicas para recabar información. (p. 10).

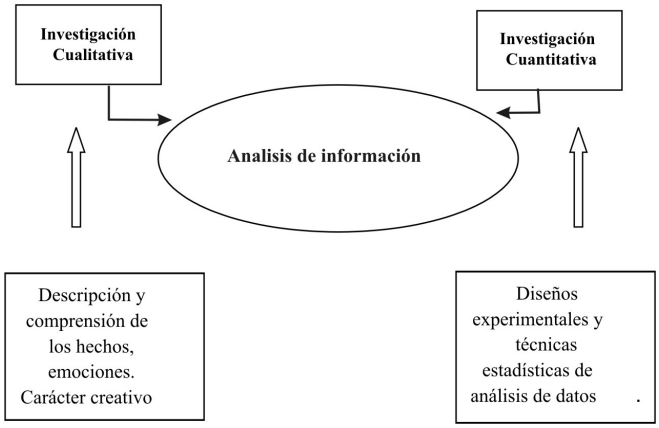
Incluso Sánchez (2008) explica cuales son los métodos cualitativos al inferir que “Los cuestionarios, la observación descriptiva, las entrevistas y otros métodos cualitativos son muy antiguos.” (p. 7).

Desde el punto de vista de Kendall & Kendall (2011) “Los documentos cualitativos incluyen mensajes de correo electrónico, memorandos, anuncios en tableros y áreas de trabajo y páginas Web.”.

Tomando en consideración la perspectiva de los autores citados anteriormente, se comprende que en el enfoque cualitativo cuenta con varias técnicas para la recolección de datos entre los que se encuentran la observación, entrevistas, análisis de documentos y los cuestionarios entre otros.

La siguiente imagen muestra una comparación entre el método cuantitativo y el método cualitativo, el cual de forma resumida, muestra claramente algunas de las diferencias sustanciales entre ambos enfoques:

Figura 19 Investigación cualitativa y cuantitativa



Fuente : http://bvs.sld.cu/revistas/spu/vol33_3_07/spu20207.htm

3.1.4 Método de Investigación Utilizado.

Laica 10/23/2014 8:16 PM
Comentario [2]: Amplíe más su justificación

Basado en las definiciones anteriores y apoyado en el punto de vista de los autores anteriormente mencionados, se puede comprender que para el proyecto propuesto se aplica un enfoque cualitativo donde se hace uso de herramientas como la encuesta para la recopilación de datos.

3.2 Tipos de Investigación

Sampieri (2010) cuando se refiere a la investigación cuantitativa sostiene que existen varios alcances de investigación, entre los que se encuentran: exploratorio, correlacional, descriptivo, explicativo. El mismo autor hace énfasis en estos enfoques al argumentar que “Del alcance del estudio depende la estrategia de investigación.”

3.2.1 Descriptiva

Los estudios de alcance descriptivo según Sampieri (2010) se alinean a los objetivos del investigador donde la mayor parte del tiempo éste busca describir fenómenos, situaciones, contextos y eventos.

Así mismo Sampieri (2010) indica que :

> Los estudios descriptivos buscan especificar las propiedades, las características y los perfiles de las personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, únicamente pretende medir o recoger información de manera independiente o conjunta sobre los conceptos o variables a las

> que se refieren, esto es, su objetivo no es indicar cómo se relacionan éstas. (p. 80).

3.2.2 Exploratoria

La Universidad Nacional Abierta y a Distancia de Colombia (UNAD)[6], refiriéndose a la investigación exploratoria comenta que :

> Cuando no existen investigaciones previas sobre el objeto de estudio o cuando nuestro conocimiento del tema es tan vago e impreciso que nos impide sacar las más provisorias conclusiones sobre qué aspectos son relevantes y cuáles no, se requiere en primer término explorar e indagar, para lo que se utiliza la investigación exploratoria.
>
> Para explorar un tema relativamente desconocido se dispone de un amplio espectro de medios y técnicas para recolectar datos en diferentes ciencias como son la revisión bibliográfica especializada, entrevistas y cuestionarios, observación participante y no participante y seguimiento de casos.
>
> La investigación exploratoria terminará cuando, a partir de los datos recolectados, haya sido posible crear un marco teórico y epistemológico lo suficientemente fuerte como para determinar qué factores son relevantes al problema y por lo tanto deben ser investigados.

La definición anterior propuesta por la entidad educativa concuerda con el punto de vista de Sampieri (2010) el cual aduce que:

Los estudio exploratorios se realizan cuando el objetivo es examinar un tema o problema de investigación poco estudiado, del cual se tiene muchas

[6] http://datateca.unad.edu.co/contenidos/100104/100104_EXE/leccin_6_investigacin__exploratoria_descriptiva_correlacional_y_explicativa.html

> dudas o no se ha abordado antes. Es decir, que cuando la revisión de la literatura reveló que tan solo hay guías no investigadas e ideas vagamente relacionadas con el problema de estudio, o bien, si deseamos indagar sobre temas y áreas de nuevas perspectivas.(p. 79).

3.2.3 Correlacional

> Sampieri (2010) define el estudio correlacional de la siguiente forma:
>
> Este tipo de estudio tiene como finalidad conocer la relación o grado de asociación que existe entre dos o más conceptos, categorías o variables en un contexto particular.
> En ocasiones solo se analiza la relación entre dos variables pero con frecuencia se ubican en el estudio relaciones entre tres, cuatro o más variables. (p. 81).

3.2.4 Explicativo

Sampieri (2010) argumenta que :

> Los estudios explicativos van más allá de la descripción de conceptos o fenómenos o del establecimiento de relaciones entre conceptos; es decir, están dirigidos a responder a las causas de los eventos y fenómenos físicos o sociales. Como su nombre lo indica, su interés se centra en explicar porqué ocurre un fenómeno y en qué condiciones se manifiesta, o por qué se relacionan dos o más variables. (p. 83).

A manera de síntesis de los métodos de investigación descritos anteriormente, la imagen siguiente los identifica en forma esquematizada y concisa.

Figura 20 Alcances de la investigación

Fuente : Metodología de la investigación, Roberto Hernández Sampieri

3.2.5 Tipo de Investigación Seleccionada

En el presente proyecto investigativo se utilizará en forma conjunta los estudios descriptivos y exploratorios. La combinación de ambos modelos corresponde a que en principio, se busca tener un panorama claro de lo que se busca medir y sobre todo se pretende definir la forma de recolectar los datos.

Es relevante mencionar que la investigación propuesta, a demás de ser innovadora en su ámbito, hace uso de tecnologías de vanguardia, las cuales son relativamente nuevas por lo cual es necesario realizar un estudio exploratorio sobre las bondades que tales tecnologías ofrecen.

Incluso el proyecto de la empresa Microsoft denominado Roslyn[7] , el cual a su vez es el cimiento para el prototipo funcional propuesto, hace

[7] https://roslyn.codeplex.com/wikipage?title=Overview&referringTitle=Home

referencia a las distintas áreas de innovación y a su relevancia en la computación moderna al indicar que :

> La plataforma de compilación de .NET (“Roslyn”) expone un conjunto de interfaces de aplicación programables (APIs) y espacios de trabajo que proveen información valiosa acerca de su código fuente y que tiene un enfoque fidedigno con los lenguajes de programación C# y Visual Basic. La transición de compiladores como plataforma disminuye dramáticamente las barreras para crear código enfocado en herramientas y aplicaciones. Crea muchas oportunidades de innovación en áreas como meta programación, generación de código y transformación, uso interactivo de los lenguajes C# y Visual Basic en lenguajes de dominio específico.

3.3 Fuentes de Información

El proceso de recopilación de información es clave durante la etapa donde se realiza el descubrimiento de requerimientos. El autor Sommerville (2011) alega que:

Las fuentes de información durante la fase de descubrimiento de requerimientos incluyen documentación, participantes del sistema y especificaciones de sistema similares. La interacción con participantes es a través de entrevistas y observaciones, y pueden usarse escenarios y prototipos para ayudar a los participantes a entender como será el sistema.

3.3.1 Fuentes Primarias

Las fuentes de información primarias, según la biblioteca de la Universidad de Alcalá (2014) “Contienen nueva y original, resultado de un trabajo intelectual”. Dicha institución también comenta que son documentos primarios “libros, revistas científicas y de entretenimiento, periódicos, diarios, documentos oficiales de instituciones públicas, informes técnicos y de investigación de instituciones públicas o privadas, patentes, normas técnicas.”.

3.3.2 Fuentes Secundaria

Citando nuevamente a la biblioteca de la Universidad de Alcalá (2014) la cual argumenta que las fuentes secundarias “contienen información organizada, elaborada, producto de análisis, extracción o reorganización que refiere a documentos primarios originales.”. Con base en la definición y basándonos en la entidad educativa mencionada anteriormente se comprende que dentro de las fuentes de información secundarias se encuentran: enciclopedias, antologías, directorios, libros o artículos que interpretan otros trabajos o investigaciones.

3.3.3 Fuentes Terciaria

La Universidad Nacional Abierta y a Distancia de Colombia (2014) indica que las fuentes de información terciaria corresponde a “Información de tercera mano, pero resumida y organizada. La mayoría de los libros

publicados hoy en día son de fuentes "terciarias" porque usan las fuentes primarias y secundarias en su redacción para un público general.”

3.3.4 Fuente de Información Seleccionada

En la investigación desarrollada median dos tipos de fuentes de información:

Fuentes primarias: Se utilizan diversos libros en temas estrictamente relacionados con el área de investigación, páginas Web de Microsoft, blogs de expertos en seguridad y material de la empresa Security Innovation.

Fuentes secundarias: La empresa Security Innovation, enfocada en el mercado de la seguridad de las aplicaciones provee un catálogo único y enfoques específicos (tal es el caso de cursos virtuales) para sustentar la investigación desarrollada.

Laica 10/23/2014 8:20 PM
Comentario [3]: No queda clara esta fuente secundaria

3.4 Descripción de Variables

Los autores Spiegel & Stephens (2012) brindan la siguiente definición de lo que es una variable :

> Una variable es un símbolo, como X,Y,Z, x o B, que puede tomar cualquiera de los valores de un conjunto predeterminado llamado dominio de la variable. Si la variable toma solo un valor, entonces a esta variable se le llama constante.
> A una variable que, teóricamente, toma cualquier valor entre dos valores dados, se le llama variable continua. Si no es así, se le denomina variable discreta(p. 1).

3.4.1 Definición Conceptual

Sampieri (2010) refiriéndose a la definición conceptual o constitutiva afirma que :

> Trata a la variable con otros términos. Así inhibición proactiva se podría definir como “la dificultad de evocación que aumenta con el tiempo”; y poder como : “influir más en los demás que éstos influyen en uno”. Se trata de definiciones de diccionario o de libros especializados. (p. 110).

3.4.2 Definición Operacional

Reynolds (1986) se refiere a una definición operacional al establecer que:

> Una definición operacional constituye el conjunto de procedimientos que describe las actividades que un observador debe realizar para recibir las impresiones sensoriales, las cuales indican la existencia de un concepto teórico en mayor o menor grado. (p. 52).

Tomando en consideración la anterior definición, Sampieri (2010) indica que “especifica que actividades u operaciones deben realizarse para medir una variable.”

3.4.3 Definición Instrumental

El doctor Cáceres (2010) en relación a la definición instrumental señala que :

> Aquí se aclara como se estudiará la variable que se acaba de definir, los medios o instrumentos para recoger la información.

> Deben definirse y elaborarse los instrumentos y medios con que se recolectará la información. Los instrumentos nacen de las variables y de los objetivos. Nunca deberá elaborarse un instrumento sin tener definida la variable o variables.

En la definición anterior se infiere la importancia de definir los instrumentos con los cuales se medirá la información, lo cual permitirá que exista una adecuada definición de las variables a utilizar en el desarrollo del prototipo funcional.

3.4 Cuadro de Variables

Tabla 1 Identificación de variables

Objetivo Específico	Variable	Variable Conceptual	Variable Operacional	Variable Instrumental
Realizar el levantamiento de requerimientos de cada una de las vulnerabilidades a identificar mediante el uso de estándares en la industria.	Documentación y requerimientos para el desarrollo del sistema	Recopilación de información necesaria para una correcta implementación del sistema.	Recopilación de datos y de requerimientos funcionales con los usuarios definidos por el administrador del proyecto e identificar el modo operacional del prototipo.	Cuestionarios y entrevistas
Elaborar el diseño del software que contempla el flujo de trabajo, la identificación de vulnerabilidades y la retroalimentación al usuario final	Diseño de un prototipo funcional	Definición de los componentes de la extensión y su arquitectura, diagramas de clase, objetos, casos de uso .	Basado en los requerimientos se procede a la elaboración de diagramas de arquitectura, componentes, UML, casos de uso	Diagramas de casos de uso, de componentes, de clases , diagrama de arquitectura del sistema, UML.
Desarrollar el prototipo funcional de la extensión de seguridad para el ambiente de desarrollo Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones.	Desarrollo de la extensión de Visual Studio para realizar análisis estático de código.	Desarrollo de una extensión para Visual Studio que permita realizar análisis estático de código fuente y poder identificar problemas de seguridad en una etapa temprana.	Programación y desarrollo de cada una de las reglas y patrones de código fuente considerado como vulnerable y brindarle retroalimentación al usuario final.	Microsoft.NET Framework Visual Studio .NET C# Plataforma de compilación Roslyn. Sistema operativo Windows.
Implementar las reglas de diagnóstico para detectar	Desarrollo de las reglas de diagnóstico en el código	Identificación de los patrones de código fuente	Utilizando guías, metodologías, mejores prácticas y	Guía OWASP Top 10. TEAM Mentor base de datos

vulnerabilidades en el código fuente utilizando estándares en la industria.	fuente.	vulnerables y de las soluciones a tales problemas.	estándares en la industria del desarrollo del software se implementarán las reglas de diagnóstico.	de conocimiento. Guía Enumeración de Debilidades Comunes (CWE)
Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo.	Pruebas funcionales, unitarias.	Metodologías aplicadas con el objetivo de determinar que el prototipo funciona como debiera, libre de errores sintácticos y semánticos.	Elaboración de escenarios de pruebas donde se verifiquen las vulnerabilidades definidas en el alcance, validación del proceso de instalación de la extensión, que los mensajes informativos al usuario sean removidos cuando el problema de seguridad ha sido resuelto.	Plataforma para realizar pruebas unitarias NUNIT. Plantillas para escenarios de pruebas.

Fuente : Propia.

3.6 Población y Muestra

El proceso de recopilación de información es sumamente importante pues permite al analista tener un panorama claro en el momento de implementar un sistema. Kendall & Kendall advierten que :

> Si todos en la población vieran el mundo de la misma forma, o si cada uno de los documentos de una población tuviera exactamente la misma información que cualquier otro, sería suficiente una muestra con un tamaño de uno. Como no es así, es necesario establecer un tamaño de muestra mayor, pero menor que el de la población. (p. 132).

3.6.1 Población

Una población corresponde al total de elementos/miembros de un grupo. En una definición más profunda, Spiegel & Stephens (2009) añaden que :

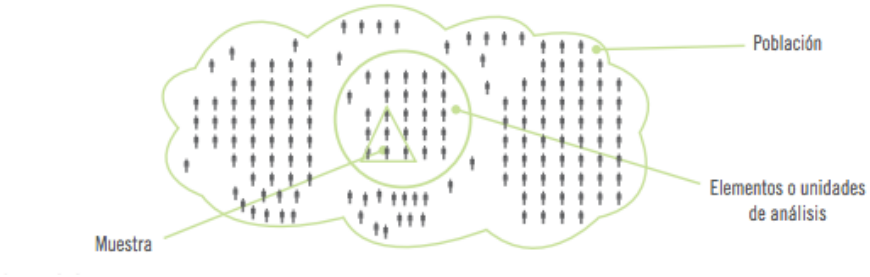
> Al recolectar datos que determinan las características de un grupo de individuos u objetos, por ejemplo, las alturas y los pesos de los estudiantes de una universidad o la cantidad de piezas defectuosas y no defectuosas producidas en una fábrica en un día determinado, muchas veces es imposible o impráctico observar a todo el grupo, especialmente si éste es grande. Este grupo se llama población, la cual puede ser finita o infinita. (p. 1).

3.6.2 Muestra

Según afirman Kendall & Kendall (2011) “El muestreo es el proceso de seleccionar sistemáticamente elementos representativos de una población.” (p.131).

Tal definición es acorde con el punto de vista de Spiegel & Stephens (2009) los cuales establecen que “En lugar de examinar a todo el grupo, llamado población o universo, se examina a una pequeña parte del grupo, a la que se llama muestra” (p. 1).

Figura 21 Población y Muestra



Fuente : Metodología de la investigación, Roberto Hernández Sampieri

3.6.3 Selección de la Población y de la muestra

En la empresa Security Innovation laboran un total de 51 ingenieros, ubicados estratégicamente en las oficinas de Boston, Seattle en Estados Unidos y otros ingenieros distribuidos en diferentes partes del mundo como Costa Rica, el Reino Unido, España. El objetivo principal es que dichos ingenieros, con mucha experiencia en el campo de la seguridad informática y áreas innovadoras como la seguridad en la industria automovilística, algoritmos criptográficos modernos y eficientes, bases de datos de conocimiento y cursos virtuales en temas de concientización y privacidad, sean los que eventualmente utilice, comercialicen y mejoren el prototipo funcional propuesto.

De acuerdo con la información anterior, se pueden recopilar los siguientes datos estadísticos:

1. Tamaño de la Población : 51
2. Error máximo aceptable : 5% (0.05)
3. Porcentaje estimado del muestreo : 70%
4. Nivel de confianza deseado : 95 % (1.96)
5. p = 0.05
6. q =(1-p) 0.95

$$n = \frac{K^2 N p q}{e^2(N-1) + k^2 \, pq}$$

$$n = \frac{(1.96)^2 * 51 * 0.05 * 0.95}{(0.03)^2(51 - 1) + (1.96)^2 * 0.05 * 0.95}$$

$$n = \frac{9.30}{0.045 + 0.182476}$$

n = 40.88

Basado en el cálculo anterior se puede determinar el valor de la muestra de personas a entrevistar corresponde a 40.88 ó 41. No obstante por decisión del administrador del proyecto se ha establecido que solo el punto de vista de tres personas se considerará vinculante para la recopilación de información y la colaboración en el levantamiento de requerimientos. Esas tres personas son las que brindarán guía y retroalimentación en la elaboración del prototipo propuesto.

3.7 Instrumentos de recolección de datos

La recopilación de datos es un proceso de suma importancia y es considerado como vital ya que permiten que exista una descripción más completa de los requerimientos.

Sampieri (2011) menciona que :

> La recolección de datos ocurre en los ambientes naturales y cotidianos de los participantes o unidades de análisis. En el caso de seres humanos, en su vida diaria: como hablan, en qué creen, qué sienten, cómo piensan, cómo interactúan etcétera. (p. 409).

3.7.1 Entrevista

Citando nuevamente a los autores Kendall & Kendall (2011) se comprende el rol fundamental de la entrevista, pues ambos autores indican que :

> Una entrevista para recopilar información es una conversación dirigida con un propósito específico, en la cual se usa un formato de preguntas y respuestas. En la entrevista hay que obtener las opiniones del entrevistado y lo que siente sobre el estado actual del sistema, los objetivos de la organización y los personales, y los procedimientos informales para interactuar con las tecnologías de información. (p. 103).

Los autores citados anteriormente además mocionan que es importante “Además de las opiniones hay que tratar de capturar los sentimientos del entrevistado.” (p. 104).

3.7.2 Cuestionario

Como su nombre lo parece indicar, se trata de un conjunto de interrogantes que tiene un sentido lógico y que tiene como objetivo poder sacar conclusiones con base en las respuestas proporcionadas.

Sampieri (2011) brinda esta definición

> Tal vez, el instrumento más utilizado para recolectar los datos es el cuestionario. Un cuestionario consiste en un conjunto de preguntas respecto de una o más variables a medir. Debe ser congruente con el planteamiento de la hipótesis. (p. 259).

3.7.3 Instrumento de recolección de datos seleccionado

Tal como se ha indicado, existen diversos mecanismos para realizar la recolección de datos. Para el prototipo propuesto, el método de recolección que ha sido seleccionado es el de la encuesta.

Dicha encuesta será aplicada a los miembros del equipo de desarrollo de la empresa Security Innovation, los cuales han sido seleccionados por el administrador del proyecto.

3.7.4 Relación entre objetivos, definición instrumental y fuentes de información.

En la tabla que se presenta a continuación, se puede observar la relación entre los objetivos, la definición instrumental y el tipo de fuente de información seleccionada.

Tabla 2 Relación entre objetos, definición instrumental y fuentes de información.

Objetivo	**Definición instrumental**	**Fuente de información**
Realizar el levantamiento de requerimientos de cada una de las vulnerabilidades a identificar mediante el uso de estándares en la industria.	Cuestionarios y Encuestas	- Fuentes primarias: Cuestionario realizado a los miembros del quipo de trabajo. Libros técnicos enfocados en análisis estático de código fuente y meta programación.
Elaborar el diseño del software que contempla el flujo de trabajo, la identificación de vulnerabilidades y la retroalimentación al usuario final	Diagramas UML Diseño de interfaces y clases. Pantallas y flujos de trabajo Arquitectura de componentes	Fuentes Primarias: Entrevista a las personas designadas por el administrador del proyecto y que pertenecen al área de ingeniería. Fuentes secundarias: Documentación disponible para un correcto diseño del sistema, incluidos blogs, revistas, mejores prácticas. Cursos virtuales de la empresa Security Innovation enfocados en seguridad del software.

Objetivo	Definición instrumental	Fuente de información
Desarrollar el prototipo funcional de la extensión de seguridad para el ambiente de desarrollo Visual Studio .NET que permita realizar pruebas estáticas de seguridad de aplicaciones.	Visual Studio .NET 2013 y C# 4.5 Plataforma de compilación Roslyn.	Fuentes primarias: Documentación de Microsoft acerca del proyecto Roslyn. Referencia del lenguaje de Programación C#. Fuentes secundarias: Blogs, foros, listas de correos donde se discuten temas relacionados con el enfoque denominado diagnóstico con solución de código.
Implementar las reglas de diagnóstico para detectar vulnerabilidades en el código fuente utilizando estándares en la industria.	Desarrollo de las reglas de dianóstico para detectar problemas de seguridad en el código fuente desarrollado en C#.	Fuentes Primarias: Estándares internacionales entre los que figuran OWASP Top 10, MITRE CWE. Mejores prácticas provistas por Microsoft.
Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo.	Casos de pruebas enfocados en la funcionalidad de la extensión.	Fuentes Primarias: Libros de ingeniería de software.

Fuente : Propia

3.7.5 Relación entre objetivos, entregables y las herramientas

A continuación se presenta un cuadro de la relación entre los entregables y los instrumentos utilizados.

Tabla 3 Relación entre objetos, entregables y herramientas

Objetivo	Entregables	Instrumentos y herramientas
Realizar el levantamiento de requerimientos de cada una de las vulnerabilidades a identificar mediante el uso de estándares en la industria.	Diagramas de casos de uso. Estudios de factibilidad (técnica, económica y operativa).	Microsoft Office para Mac Visual Paradigm Entrevistas.
Elaborar el diseño del software que contempla el flujo de trabajo, la identificación de vulnerabilidades y la retroalimentación al usuario final	Diagramas UML Diseño de flujos de trabajo Detalle de las vulnerabilidades en el código fuente a ser abarcadas y el vector de ataque.	Visual Paradigm Plataforma de compilación Roslyn. Visual Studio 2013.
Desarrollar el prototipo funcional de la extensión de seguridad para el ambiente de desarrollo Visual Studio .NET que permita realizar pruebas	Prototipo funcional de la extensión para Visual Studio que permita realizar pruebas estáticas de código fuente.	Visual Studio .NET 2013 Lenguaje de Programación C# Plataforma de compilación Roslyn. Microsoft .NET Framework 4.5 o superior.

Objetivo	Entregables	Instrumentos y herramientas
estáticas de seguridad de aplicaciones.		
Implementar las reglas de diagnóstico para detectar vulnerabilidades en el código fuente utilizando estándares en la industria.	Reglas de diagnóstico para detectar patrones de código fuente vulnerable, basado en estándares internacionales.	Reglas de diagnóstico implementadas donde se ha definido el mensaje de retroalimentación para el usuario final. Mensajes de información intuitivos dentro del ambiente de desarrollo de Visual Studio .NET.
Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo.	Casos de pruebas Resultados esperados luego de analizar aplicaciones vulnerables. Resultados de las pruebas unitarias.	Plantillas de casos de pruebas y resultados en formato Office.

Fuente : Propia

CAPÍTULO IV

DISENO

4.1 Análisis

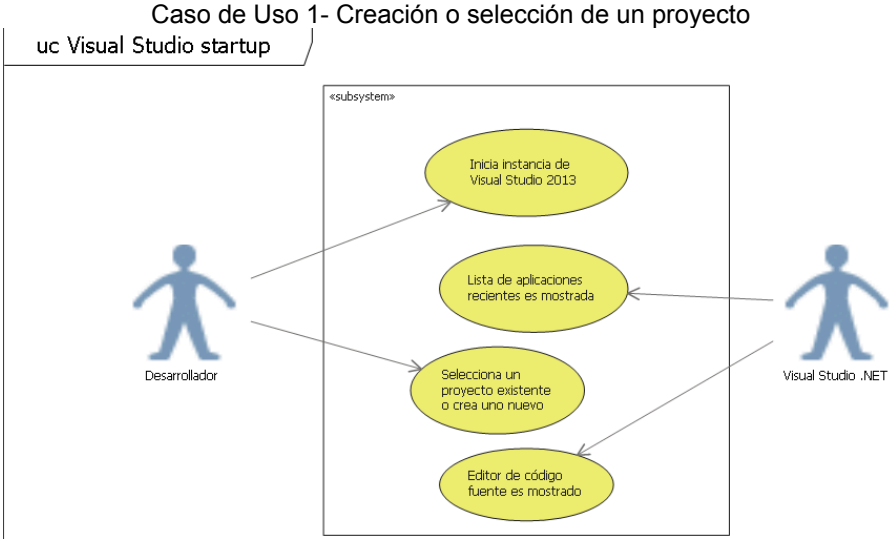
4.1.1 Casos de uso

En esta sección se detallan los casos de uso de la extensión de seguridad para Visual Studio .NET y la relación entre los componentes del prototipo. Según afirman Esposito & Saltarello (2009) "… un caso de uso es una interacción entre el sistema y uno de sus actores. Un caso de uso muestra lo que cada actor hace." (p. 43).

4.1.1.1 Creación de un proyecto nuevo o selección de uno existente

Ilustración 1

Caso de Uso 1- Creación o selección de un proyecto



Fuente : Propia

Tabla 4 Creación o selección de un proyecto existente.

CU:01	Inicio de un proyecto nuevo o existente en Visual Studio	
Versión	Versión 1. Viernes 24 de Octubre de 2014	
Autores	Michael Hidalgo Fallas	
Fuentes		
Objetivos asociados		
Descripción	Este caso de uso inicia cuando el usuario (desarrollador de software en el lenguaje C#),ejecuta una instancia nueva del IDE de Visual Studio, el cual muestra una pantalla inicial donde el usuario podrá seleccionar un proyecto (aplicación) existente o crear un nuevo proyecto.	
Precondición	El usuario deberá tener instalado una versión de Visual Studio .NET 2013 o superior en su ambiente.	
Secuencia Normal	**Paso**	**Acción**
	1	El desarrollador inicia una instancia de Visual Studio .NET
	2	Visual Studio muestra la pantalla principal donde el desarrollador puede seleccionar un proyecto.
	3	El desarrollador selecciona un proyecto Web (desarrollado en C#) existente o crea un proyecto nuevo.
	4	Visual Studio muestra el editor de código con plantillas precargadas o con el código fuente existente en caso de tratarse de una aplicación existente.
Postcondición	Análisis de código fuente es ejecutado.	

Excepciones	**Paso**	**Acción**
	1	La computadora del desarrollador no cuenta con los requisitos mínimos de software y hardware necesarios para instalar Visual Studio .NET.
	2	El desarrollador no cuenta con ninguna versión de Visual Studio .NET, deberá descargar la versión de pruebas (periodo de evaluación de 90 días) o comprar la licencia del software desde el sitio oficial de Microsoft.
Rendimiento	**Paso**	**Cota de tiempo**
	1	De 1 a 3 minutos dependiendo de las características de software y hardware de la computadora del desarrollador.
Frecuencia esperada	<nº de veces> veces / <unidad de tiempo>	
Importancia	Importante	
Urgencia	Inmnediatamente.	
Comentarios		

Fuente: Propia

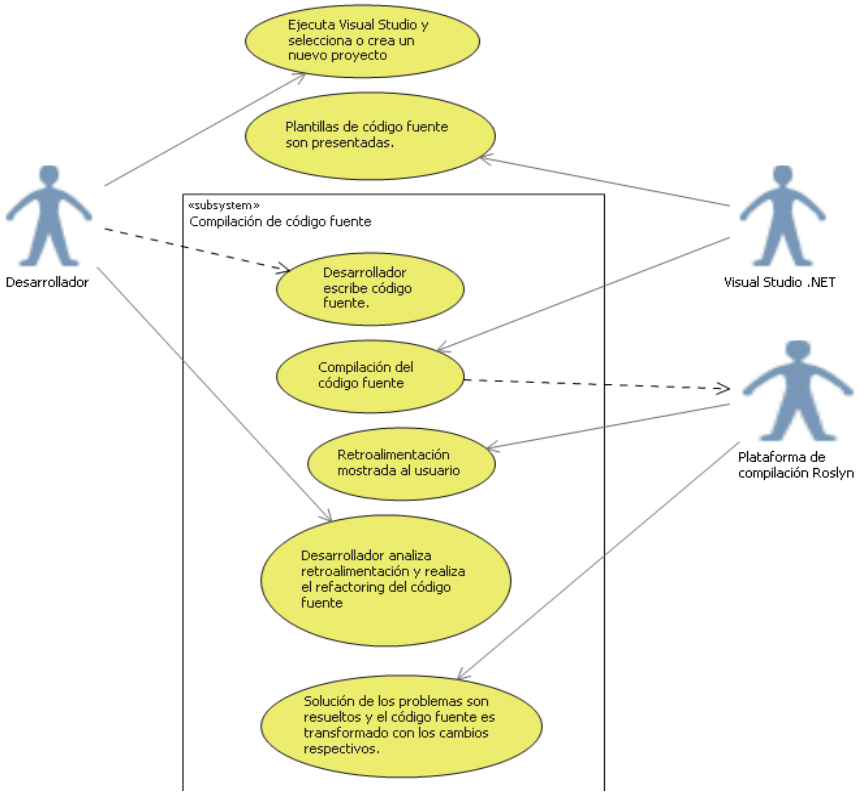
4.1.1.2 Compilación como servicio

Este caso de uso muestra el proceso de compilación del código fuente por medio de la plataforma Roslyn, donde se tiene mayor control sobre el proceso de compilación. El enfoque que proporciona la plataforma de compilación Roslyn se le denomina de forma genérica bajo el nombre de compilación como servicio, y tiene como función primordial habilitar al

desarrollador a poder extender de cierta forma el proceso de compilación, esto por medio de la implementación de reglas específicas o incluso cambios en el código fuente (denominado refactoring).

Ilustración 2

Caso de Uso 2 : Compilación como servicio



Fuente : Propia

Tabla 5 Compilación del código fuente como servicio

CU :02	Compilación del código fuente como servicio	
Versión	Versión 1. Viernes 24 de Octubre de 2014	
Autores	Michael Hidalgo Fallas	
Fuentes		
Objetivos asociados		
Descripción	Este caso de uso inicia cuando el desarrollador de código fuente ha seleccionado o creado una aplicación Web bajo el lenguaje de programación C#. Cuando se realiza la compilación del código fuente o en su defecto, cuando se escribe el código fuente en tiempo real, la plataforma de compilación Roslyn se encargará de realizar el análisis estático de código fuente. Tomando en consideración los patrones de código fuente vulnerable previamente definidos, se mostrará errores al usuario final de forma tal que éste pueda proceder a corregir los problemas del código fuente.	
Precondición	UC:01	
Secuencia Normal	**Paso**	**Acción**
	1	El desarrollador de software inicia un proyecto Web basado en el lenguaje de programación C#.
	2	Visual Studio.NET muestra el código fuente y hace análisis estático del mismo por medio de la plataforma de compilación Roslyn.
	3	El desarrollador escribe el código fuente necesario para resolver una necesidad de negocio

	4	Visual Studio.NET procede a la compilación el código fuente, por medio de la plataforma Roslyn, la cual ejecuta los patrones de diagnóstico de código fuente implementados, y muestra la retroalimentación al usuario en tiempo real dentro del ambiente de desarrollo.
	5	El desarrollador observa la retroalimentación proporcionada por la herramienta y procede aceptar o rechazar la sugerencia brindada por el plugin con el objetivo de tener código fuente más seguro.
	6	El plugin de Visual Studio realiza el refactoring del código fuente con el propósito de resolver el problema de seguridad encontrado en el código fuente.
Postcondición		
Excepciones	**Paso**	**Acción**
	1	El desarrollador omite las recomendaciones proporcionadas por el plugin de seguridad, las cuales tienen un propósito informativo, y de esta forma no permite que la extensión de seguridad haga el cambio respectivo en el código fuente.
	2	La aplicación no contiene código fuente vulnerable por lo que el plugin no muestra información al respecto, ya que el diagnóstico implementado no ha sido capas de encontrar patrones de código fuente con vulnerabilidades conocidas.
Rendimiento	**Paso**	**Cota de tiempo**
	1	3 segundos

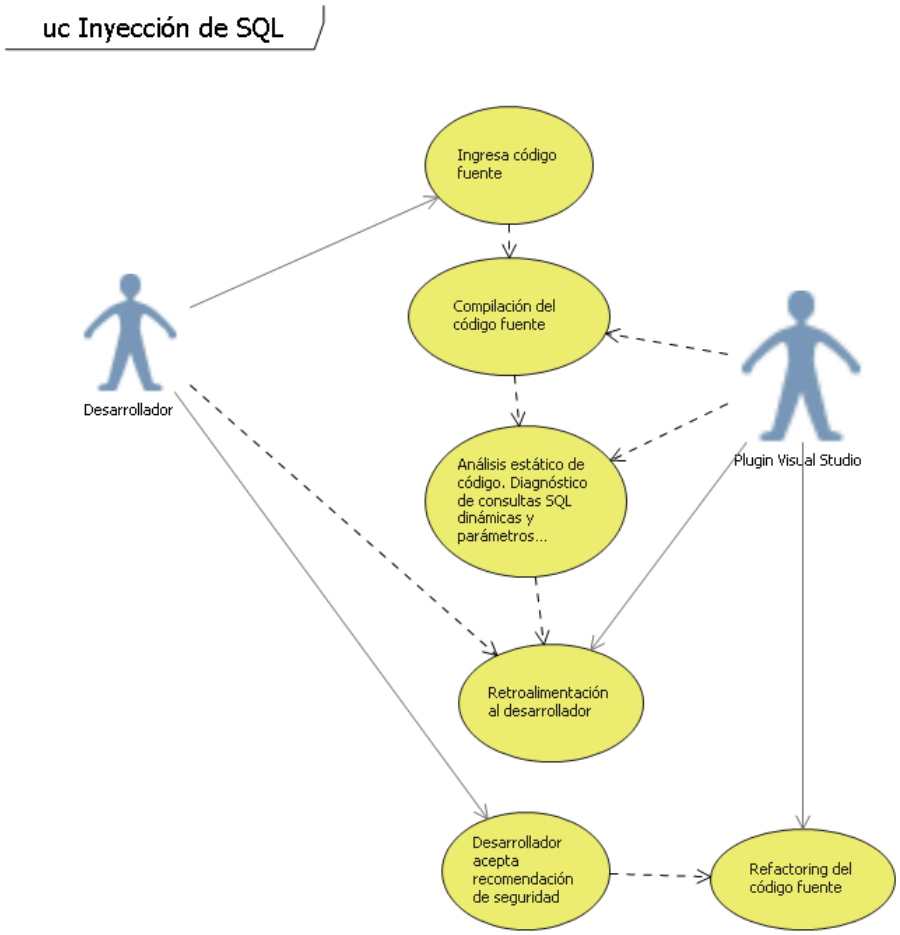
Frecuencia esperada	Múltiples ocasiones.
Importancia	Importante.
Urgencia	Inmediatamente
Comentarios	

Fuente : Michael Hidalgo Fallas

4.1.1.3 Módulo Vulnerabilidades de Inyección de SQL

En el presente caso de uso se demuestra el módulo de detección de vulnerabilidades de Inyección de SQL, donde las herramientas de diagnóstico implementadas en el plugin de Visual Studio harán el análisis estático de código fuente y le permitirán al desarrollador cambiar el código fuente para mitigar el riesgo de que tal vulnerabilidad se materialice.

Ilustración 3 Módulo de vulnerabilidades de inyección de SQL



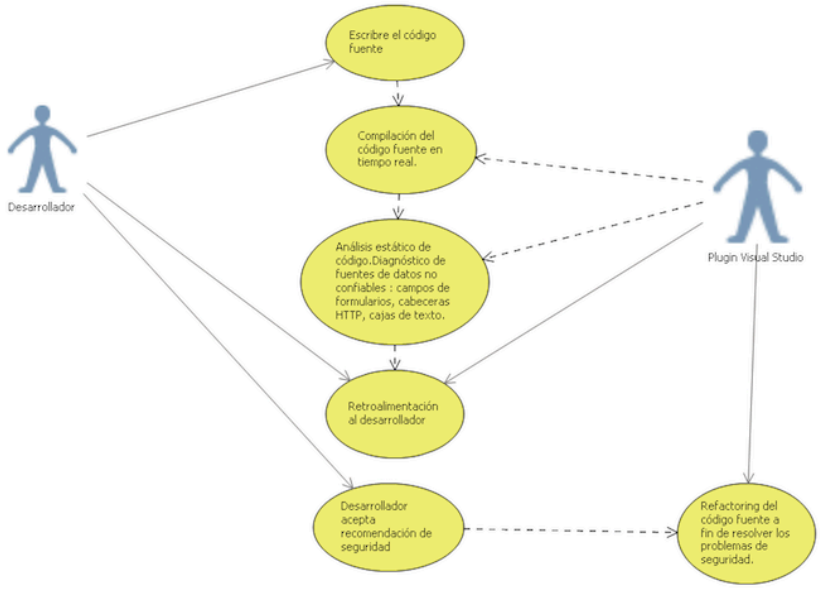
Fuente : Propia

CU:03	Módulo de vulnerabilidades de inyección de SQL.
Versión	Versión 1. Viernes 24 de Octubre de 2014
Autores	Michael Hidalgo Fallas
Fuentes	
Objetivos asociados	

Descripción	Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el plugin de Visual Studio .NET realiza el análisis del código a fin de encontrar vulnerabilidades o patrones inseguros de código e inmediatamente le muestra al desarrollador la retroalimentación en tiempo real. Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.	
Precondición	Se realiza el refactoring del código fuente con soluciones a los problemas identificados.	
Secuencia **Normal**	**Paso**	**Acción**
	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el plugin desarrollado realiza la compilación en tiempo real.
	3	El plugin realiza un diagnóstico del código fuente buscando vulnerabilidades de inyección de SQL, identificando consultas de SQL dinámicas y datos no confiables provenientes de formularios , cabeceras HTTP, cookies.
	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un refactoring del código fuente, siguiendo una mejor práctica.
	5	El usuario selecciona la opción de resolver el problema de seguridad.
	6	El componente de seguridad realiza la modificación del código fuente para mitigar el riesgo de que se explote la vulnerabilidad.
	n	

Postcondición	Código fuente es modificado para mitigar el riesgo de que una vulnerabilidad de inyección de SQL ocurra.	
Excepciones	**Paso**	**Acción**
	1	El plugin de Visual Studio .NET no encuentra vulnerabilidades en el código fuente.
	2	La aplicación a ser desarrollada en Visual Studio no corresponde al lenguaje de programación C# ni a un proyecto Web.
	3	
Rendimiento	**Paso**	**Cota de tiempo**
	1	3 segundos
Frecuencia esperada	Frecuente. Se ejecuta durante todo el desarrollo.	
Importancia	Importante.	
Urgencia	Urgente.	
Comentarios		

4.1.1.4 Módulo de vulnerabilidades de Secuencia de Sitios Cruzados (XSS).



Fuente : Propia

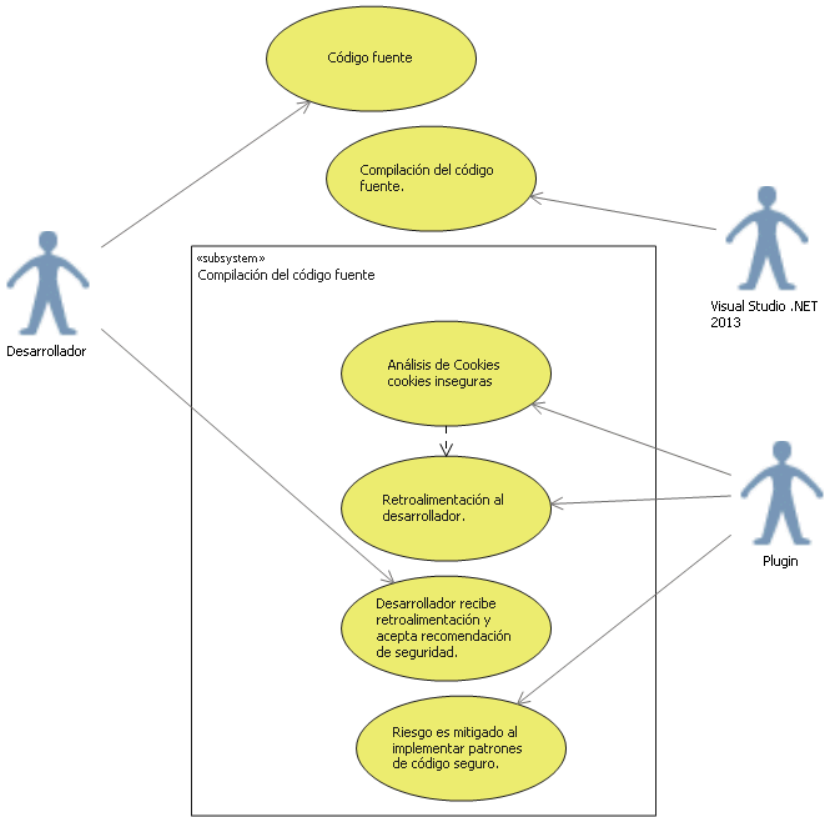
CU:04	Módulo de vulnerabilidades de Secuencia de Comandos Cruzados entre páginas (XSS)
Versión	Versión 1. Viernes 24 de Octubre de 2014
Autores	Michael Hidalgo Fallas
Fuentes	
Objetivos asociados	

Descripción	Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el plugin de Visual Studio .NET realiza el análisis del código a fin de encontrar vulnerabilidades o patrones inseguros de código e inmediatamente le muestra al desarrollador la retroalimentación en tiempo real. Dicho análisis involucra las fuentes de datos no confiables entre las que se encuentran los datos de entrada provistos por el usuario final, formularios de campo, cabeceras HTTP, cookies. Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.	
Precondición	Se realiza el refactoring del código fuente con soluciones a los problemas identificados.	
Secuencia **Normal**	**Paso**	**Acción**
	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el plugin desarrollado realiza la compilación en tiempo real.
	3	El plugin realiza un diagnóstico del código fuente buscando vulnerabilidades Secuencias de comandos en sitios cruzados (XSS) al analizar detalladamente datos no confiables que puedan generar un comportamiento inadecuado en la aplicación.

	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un refactoring del código fuente, siguiendo una mejor práctica. De igual forma se proporcionan links a TEAM Mentor con detalles de la vulnerabilidad con el propósito de que el desarrollador conozca un poco más del problema.
	5	El usuario selecciona la opción de resolver el problema de seguridad.
	6	El componente de seguridad realiza la modificación del código fuente para mitigar el riesgo de que se explote la vulnerabilidad.
	n	
Postcondición	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
Excepciones	**Paso**	**Acción**
	1	El plugin de Visual Studio .NET no encuentra vulnerabilidades en el código fuente.
	2	La aplicación a ser desarrollada en Visual Studio no corresponde al lenguaje de programación C# ni a un proyecto Web.
	3	El plugin de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
Rendimiento	**Paso**	**Cota de tiempo**
	1	3 segundos

Frecuencia esperada	Frecuente. Se ejecuta durante todo el desarrollo de la aplicación ya que el componente realiza el análisis estático de código a medida que el mismo es desarrollado.
Importancia	Importante.
Urgencia	Urgente.
Comentarios	

4.1.1.5 Módulo de vulnerabilidades de Pérdida de autenticación y gestión de sesiones

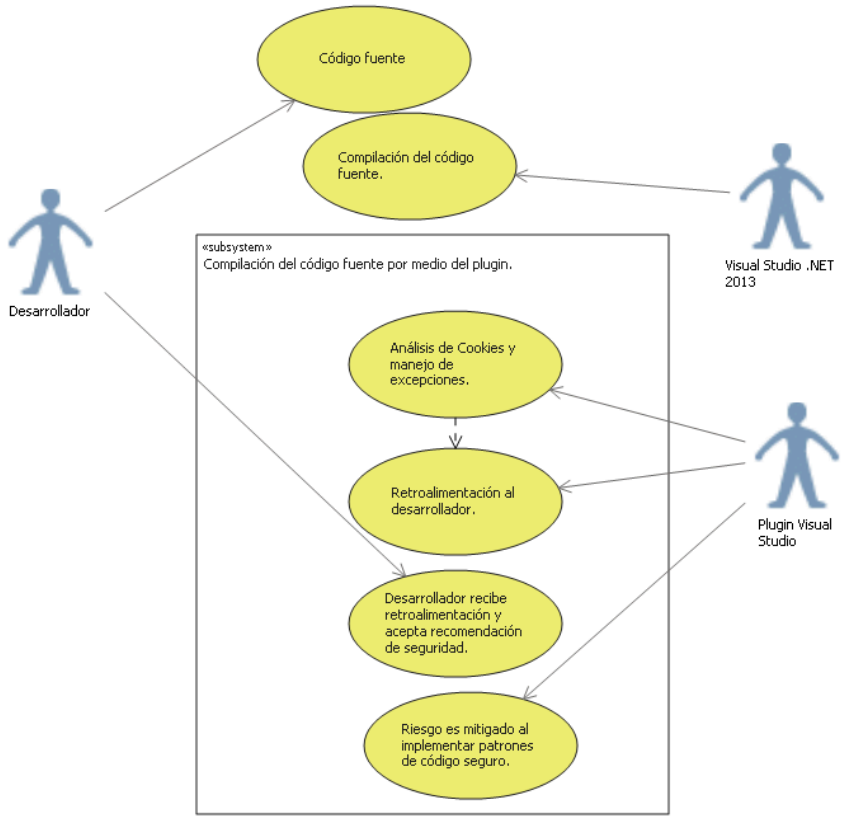


CU:05	Módulo de vulnerabilidades de Pérdida de autenticación y gestión de sesiones.	
Versión	Versión 1. Viernes 24 de Octubre de 2014	
Autores	Michael Hidalgo Fallas	
Fuentes		
Objetivos asociados		
Descripción	Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el plugin de Visual Studio .NET realiza el análisis del código a fin de encontrar vulnerabilidades o patrones inseguros de código e inmediatamente le muestra al desarrollador la retroalimentación en tiempo real. Dicho análisis involucra la verificación de las variables de sesión, cookies que puedan ser modificadas por medio de código del lado del cliente (propiamente por medio de JavaScript).las fuentes de datos no confiables entre las que se encuentran los datos de entrada provistos por el usuario final, formularios de campo, cabeceras HTTP, cookies. Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.	
Precondición	Se realiza el refactoring del código fuente con soluciones a los problemas identificados.	
Secuencia **Normal**	**Paso**	**Acción**
	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el plugin desarrollado realiza la compilación en tiempo real.

	3	El plugin realiza un diagnóstico del código fuente buscando Pérdida de autenticación y gestión de sesiones al analizar variables de sesión y falta de atributos que permitan que se puedan manipular por medio de código del lado del cliente.
	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un refactoring del código fuente, siguiendo una mejor práctica. De igual forma se proporcionan links a TEAM Mentor con detalles de la vulnerabilidad con el propósito de que el desarrollador conozca un poco más del problema.
	5	El usuario selecciona la opción de resolver el problema de seguridad.
	6	El componente de seguridad realiza la modificación del código fuente para mitigar el riesgo de que se explote la vulnerabilidad.
	n	
Postcondición	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
Excepciones	**Paso**	**Acción**
	1	El plugin de Visual Studio .NET no encuentra vulnerabilidades en el código fuente.
	2	La aplicación a ser desarrollada en Visual Studio no corresponde al lenguaje de programación C# ni a un proyecto Web.

	3	El plugin de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
Rendimiento	**Paso**	**Cota de tiempo**
	1	3 segundos
Frecuencia esperada	Frecuente. Se ejecuta durante todo el desarrollo de la aplicación ya que el componente realiza el análisis estático de código a medida que el mismo es desarrollado.	
Importancia	Importante.	
Urgencia	Urgente.	
Comentarios		

4.1.1.5 Módulo de vulnerabilidades de Configuración Incorrecta de Seguridad.



CU:06	Módulo de vulnerabilidades de Configuración Incorrecta de Seguridad.
Versión	Versión 1. Viernes 24 de Octubre de 2014
Autores	Michael Hidalgo Fallas
Fuentes	
Objetivos asociados	

Descripción	Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el plugin de Visual Studio .NET realiza el análisis del código a fin de encontrar vulnerabilidades o patrones inseguros de código e inmediatamente le muestra al desarrollador la retroalimentación en tiempo real. Como parte del análisis a ser desarrollado se contempla el uso inapropiado de excepciones que revelan información sensitiva, tales como plataforma de desarrollo, origen de los errores entre otros. Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.	
Precondición	Se realiza el refactoring del código fuente con soluciones a los problemas identificados.	
Secuencia **Normal**	**Paso**	**Acción**
	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el plugin desarrollado realiza la compilación en tiempo real.
	3	El plugin realiza un diagnóstico del código fuente buscando Configuraciones incorrectas de seguridad. Al analizar variables de sesión y falta de atributos que permitan que se puedan manipular por medio de código del lado del cliente.

	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un refactoring del código fuente, siguiendo una mejor práctica. De igual forma se proporcionan links a TEAM Mentor con detalles de la vulnerabilidad con el propósito de que el desarrollador conozca un poco más del problema.
	5	El usuario selecciona la opción de resolver el problema de seguridad.
	6	El componente de seguridad realiza la modificación del código fuente para mitigar el riesgo de que se explote la vulnerabilidad.
	n	
Postcondición	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
Excepciones	**Paso**	**Acción**
	1	El plugin de Visual Studio .NET no encuentra vulnerabilidades en el código fuente.
	2	La aplicación a ser desarrollada en Visual Studio no corresponde al lenguaje de programación C# ni a un proyecto Web.
	3	El plugin de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
Rendimiento	**Paso**	**Cota de tiempo**
	1	3 segundos

Frecuencia esperada	Frecuente. Se ejecuta durante todo el desarrollo de la aplicación ya que el componente realiza el análisis estático de código a medida que el mismo es desarrollado.
Importancia	Importante.
Urgencia	Urgente.
Comentarios	

4.1.2 Análisis de Requerimientos

4.1.2.1

4.1.2.2

4.2 Diseño de la solución

El diseño del software, parte fundamental del ciclo de vida del desarrollo de software, permite tener noción de los diversos componentes del software, sus interacciones y los diversos flujos de trabajo. El afamado escritor Antoine de Saint-Exúpery en el libro titulado Viento, Arena y Estrellas, refiriéndose a diseño escribió una frase que siempre se encuentra en la jerga de los arquitectos, la cual sostiene que “Usted sabe que usted ha alcanzado perfección en diseño, no cuando no hay nada más que agregar, sino cuando

no hay nada más que quitar". Dicha frase refleja el pensamiento simplista de la necesidad en el diseño de evitar complejidad y mostrar facetas para modelar un proceso o un sistema.

> Espostito & Saltarello (2009) indican que :
>
> En el inicio de la era de la computación, en los inicios de 1960, el costo de hardware era predominantemente mayor a los costos del software. Cuarenta años después encontramos que la situación es radicalmente diferente.
>
> Apropiadamente desde la industria de la construcción, el término arquitectura se ha convertido en la forma apropiada de describir el arte de planificar, diseñar e implementar intensivos sistemas. En software, arquitectura necesita menos arte que en construcción. En software, el término arquitectura precisamente se refiere a construir un sistema para un cliente. (p. 3).

Tal como se puede apreciar, el punto de vista de los autores refleja la importancia de aplicar el arte del diseño a la industria del software, lo cual permite que se creen aplicaciones que no solamente resuelven una problemática del negocio sino que también cuenta con cimientos muy fuertes en términos de arquitectura. Este enfoque, importante en naturaleza, facilita que el software evolucione ya que modificaciones o alteraciones a los flujos de trabajo establecidos se complementan de forma muy sencilla, sin afectar el sistema per se.

4.2.1 Arquitectura de software

> Según Sommerville (2011):

> El diseño arquitectónico se interesa por entender como debe organizarse un sistema y cómo tiene que diseñarse la estructura global de ese sistema. En el modelo del proceso de desarrollo de software, el diseño arquitectónico es la primera etapa en el proceso de diseño de software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. (p. 148).

Así mismo Esposito & Saltarello (2009), definiendo arquitectura desde el punto de vista de los estándares indican que “lo importante del estándar ANSI/IEEE para arquitectura de software es que un sistema existe para cumplir las expectativas de todos los interesados”. (p. 5).

4.2.2 Arquitectura del sistema

En su libro Patterns of Enterprise Application Architecture, Fowler (2002) indica que:

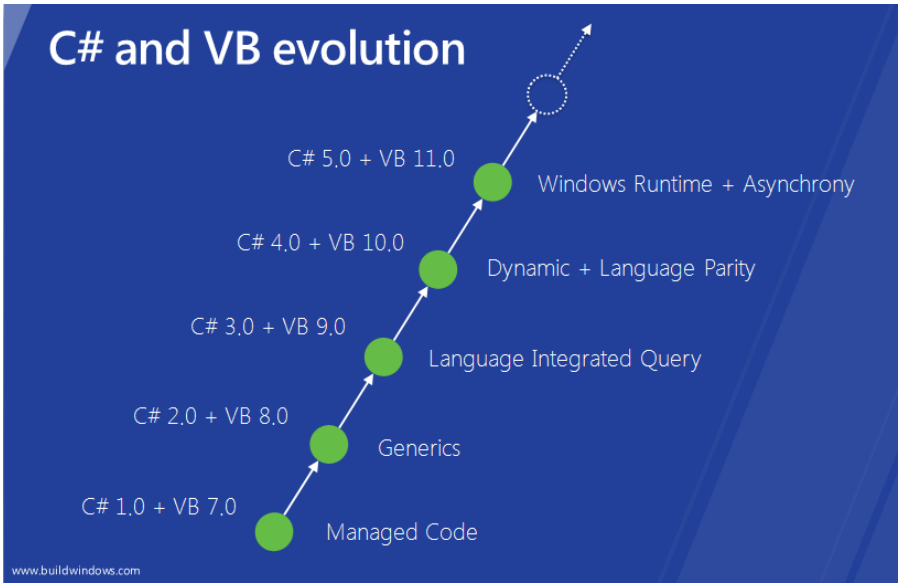
> Si usted encuentra que algo es más fácil de cambiar de lo que usted pensó, entonces ya no es arquitectura. Al final arquitectura se resume en la importancia de las cosas-cualquiera que éstas sean.

En esta sección se detallará los principales componentes de arquitectura del prototipo propuesto. Se utilizarán diversos diagramas con el objetivo principal de explicar mejor el prototipo a ser implementado.

Considerando que el prototipo funcional hace uso exhaustivo de la plataforma de compilación de Microsoft denominada bajo el nombre clave de Roslyn, es requerido tener noción del proceso evolutivo de los leguajes de

programación C# y Visual Basic. La imagen siguiente muestra dicha evolución.

Figura 22 Evolución de C# y Visual Basic



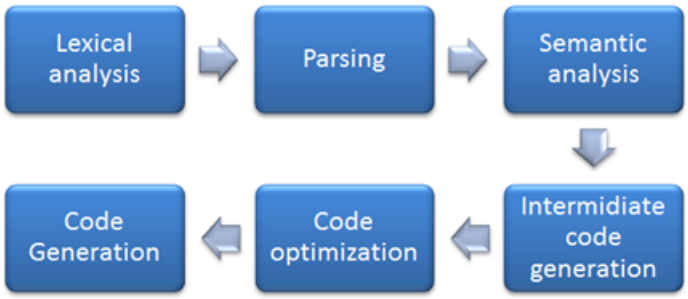
Fuente: http://goo.gl/GSYjP

Es claro percibir que ambos lenguajes han incluido características fundamentales que permiten que el desarrollo de código sea un proceso más intuitivo, permitiendo que la forma de escribir código sea similar al pensamiento humano, con sintaxis y semántica más similar al lenguaje cotidiano. Y tal como se ilustra en la imagen anterior el crecimiento exponencial continúa, expandiendo de tal forma las barreras tecnológicas actuales.

Utilizando la plataforma de compilación Roslyn se puede acceder a las cajas negras del proceso de compilación, el cual se puede visualizar como varias etapas que comprenden el análisis léxico del código fuente, analizadores, análisis semántico, generación de código, optimización de código y generación de código intermedio.

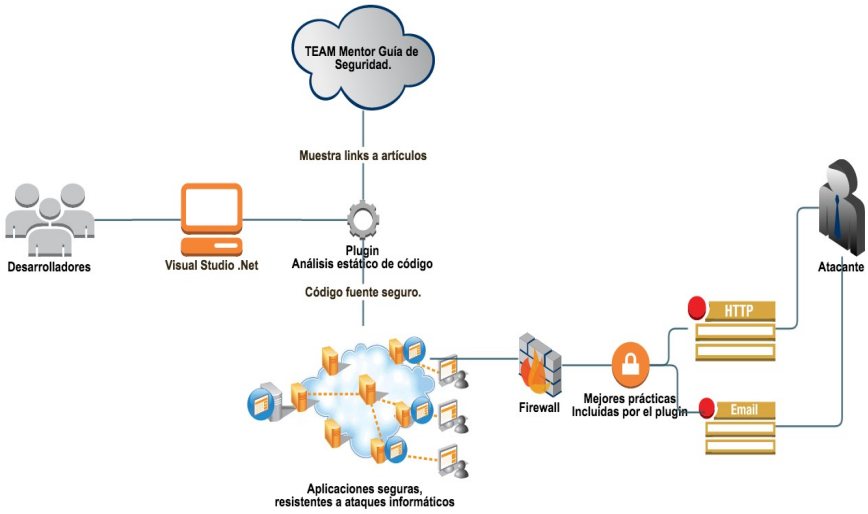
En la imagen siguiente se muestran los pasos del proceso de compilación descritos anteriormente.

Figura 23 Pasos del proceso de compilación.



Fuente : http://goo.gl/PaOoD

Figura 24 Arquitectura del sistema



Fuente: Propia

En el diagrama anterior se muestran los principales componentes de la arquitectura del sistema. Según se ilustra, los desarrolladores escriben el código fuente en el ambiente de desarrollo de Visual Studio, el cual realiza la compilación del código fuente por medio de la extensión de seguridad desarrollada. El plugin desarrollado muestra links a artículos de la base de datos de conocimiento denominada TEAM Mentor, cuyo objetivo primordial es la de servir como complemento al desarrollador y como guía en caso de que

éste quiera tener un conocimiento más detallado de los problemas de seguridad mostrados por el componente.

En esta etapa, el desarrollador podrá ver los mensajes de error mostrados en el momento de escribir código fuente inseguro, de forma tal que tendrá la opción de aceptar la mejor práctica proporcionada por la herramienta.

La retroalimentación al usuario sigue el mantra estipulado durante el proceso de compilación, el cual indica que cuando hay un error de sintaxis o semántica, inmediatamente el usuario es alertado con algún tipo de información (el cual puede ser alertas, advertencias o errores), mismos mostrados en una sección específica dentro del entorno de desarrollo.

Una vez que el usuario haya aceptado las recomendaciones propuestas se tendrán aplicaciones más seguras, resistentes a ataques informáticos comunes lo cual disminuye considerablemente los vectores de ataques.

En el diagrama se ilustra la interacción del usuario malicioso, el cual tratará de manipular los datos de entrada del sistema para causar un comportamiento distinto al esperado (con el objetivo de comprometerlo). En este momento los mecanismos de código seguro deberán frenar las malas intenciones de tales usuarios, realizando una correcta verificación de los datos de entrada.

4.2.3 Identificación de vulnerabilidades

Según estudios recientes del Instituto Ponemon, el cual es una de las referencias primarias para el enfoque que se propone, argumenta que cerca del 92% de los ataques informáticos que ocurren en la actualidad se perpetran a nivel de la capa aplicativa. Este estudio muestra un interesante patrón parece indicar que los atacantes utilizan la misma aplicación, luego de explotar alguna vulnerabilidad en el la misma, para comprometer la integridad del mismo sistema, el robo de información confidencial y generar daños irreparables a la reputación de la entidad. Bajo esta perspectiva es claro ver por ejemplo las pérdidas económicas para una institución financiera, sin dejar de lado la histeria colectiva que desencadena un ataque informático que haya comprometido cuentas bancarias e información confidencial de los empleados. Pero en una era donde la tecnología predomina, ningún sector de la industria está exento de ser víctima de crímenes cibernéticos. Esto se debe en parte a que tales incidentes pueden tener trasfondos políticos, económicos y sociales que indirectamente tienden a afectarnos a todos.

Volviendo al enfoque de la seguridad aplicativa, las vulnerabilidades encontradas en el código fuente, mismas que han sido producidas involuntariamente durante la fase de desarrollo (por desconocimiento mismo de los patrones de código inseguro así como la falta de controles y validaciones en los datos de entrada), generan un impacto severo en los modelos de negocios de las empresas. Es necesario por lo tanto un análisis y

un acercamiento a tales vulnerabilidades las cuales marcan una hoja de ruta en el momento de brindar un mecanismo de mitigación eficiente.

4.2.3.1 El Proyecto OWASP Top 10

La fundación OWASP (Proyecto Abierto para la Seguridad de las Aplicaciones Web) por sus siglas en inglés es una organización internacional sin fines de lucro, creada en el año 2001 con el objetivo principal de crear visibilidad sobre las causas que generan que el software sea inseguro. El modelo que la institución propone invita a todas las personas involucradas activamente en el ciclo de vida del software así como a expertos en seguridad a trabajar en conjunto para establecer los estándares que se necesitan para mitigar los problemas de seguridad que mas causa están provocando.

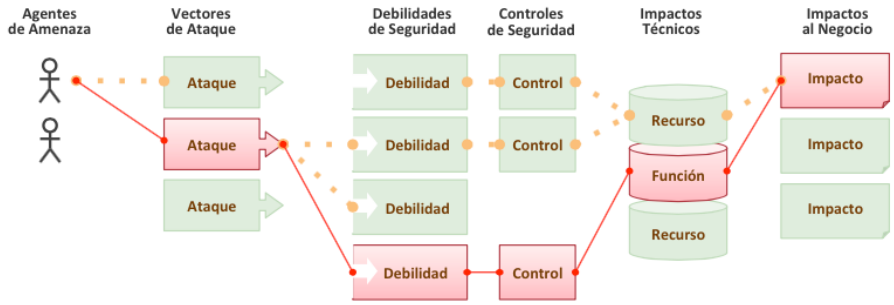
Dentro de las guías y materiales elaborados por la comunidad figura la guía agnóstica denominada OWASP Top 10, la cual es definida por OWASP (2013) como:

> El objetivo del proyecto Top 10 es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. El proyecto Top 10 es referenciado por muchos estándares, libros, herramientas, y organizaciones, incluyendo MITRE, PCI DSS, DISA, FCT, y muchos más . Esta versión de OWASP Top 10 marca el aniversario número diez de este proyecto, de concientización sobre la importancia de los riesgos de seguridad en aplicaciones. OWASP Top 10 fue lanzado por primera vez en 2003, con actualizaciones menores en 2004 y 2007. La versión 2010 fue renovada para dar prioridad al riesgo, no sólo a la prevalencia. La edición 2013 sigue el mismo enfoque.

La guía del OWASP top 10 es ampliamente utilizada en la creación de herramientas, estándares, manuales y componentes entre otros mucho otras ideas innovadoras.

Tal como se puede apreciar en la imagen siguiente, la modalidad del OWASP Top 10 brinda un modelo de riesgos en seguridad de aplicaciones el cual se define como “Los atacantes pueden potencialmente usar rutas diferentes a través de la aplicación para hacer daño a su negocio u organización. Cada una de estas rutas representa un riesgo que puede, o no, ser lo suficientemente grave como para justificar la atención.” (p. 1).

Figura 25 Modelo de riesgos



Fuente: OWASP Top 10 2013 Español

4.2.3.1 Inyección de SQL

Inyección es un término genérico aplicado a un tipo de vulnerabilidad donde el usuario malicioso manipula los datos de entrada de una aplicación Web. Debido a que un alto porcentaje de las aplicaciones Web funcionan con un motor de base de datos, el atacante entonces tratará de enviar como parte de los datos, caracteres u expresiones con semántica que es interpretado por el lenguaje Structured Query Language (SQL). Al analizar los mensajes de error o el comportamiento de la aplicación, el atacante podrá fácilmente darse cuenta si es posible o no perpetrar un ataque.

Paul (2011) establece que :

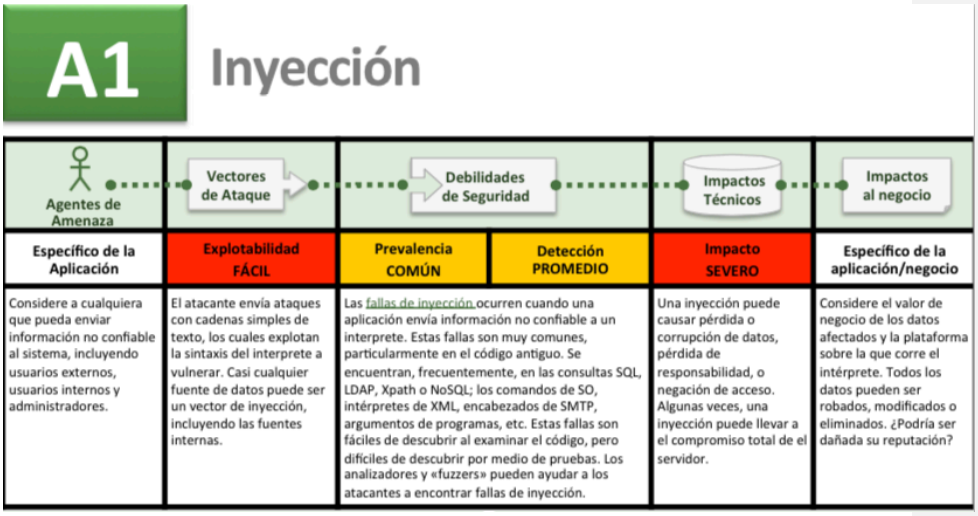
> Este es probablemente el ataque de inyección más conocido, debido a que las bases de datos que almacenan datos del negocio se están convirtiendo en el objetivo principal de los atacantes. En inyección de SQL (Structured Query Language), los atacantes explotan la forma en que las consultas a bases de datos son construidas. Ellos ingresan datos de entrada, la cual si no es correctamente validada, se convierte en parte de la consulta que la base de datos procesa como parte del comando. (p. 249).

Casualmente este tipo de ataque informático es ampliamente difundido de forma mundial y parte de este comportamiento se le atribuye a la facilidad con la que se puede materializar al igual que pobres prácticas de desarrollo que permiten que tal ataque se pueda ejecutar. De igual forma, la gran cantidad de información disponible en Internet, misma que no es siempre confiable, guía a que eventualmente se tome como referencia código fuente que es vulnerable y se introduzca en la organización, trasladando de esta manera el riesgo.

En la imagen siguiente se puede observar el flujo de trabajo y el vector de ataque de una inyección de SQL. Se puede observar el agente de amenaza, el vector de ataque , las debilidades y el impacto para el negocio e impactos técnicos.

Es importante recalcar que este enfoque basado en riesgos, lo cual permite que se analicen y se tomen medidas para poder mitigarlo.

Ilustración 4 Inyección de SQL



Fuente: OWASP 2013 Español

4.2.3.2 Pérdida de Autenticación y Gestión de Sesiones

Las aplicaciones Web modernas a lo largo de los años han implementado manejo de autenticación y sesiones precisamente para poder

implementar el concepto de estado. Esto quiere decir que el usuario pueda realizar varias acciones en un sitio transaccional sin perder los datos o las actividades realizadas. Es importante aclarar que los mecanismos de sesiones no forman parte innata del protocolo HTTP, sino que fue incluido con la intención de resolver una necesidad emergente.

Pobres mecanismos de autenticación y un manejo incorrecto de sesiones conlleva a que un usuario maliciosa pueda impersonalizar (actuar en nombre de otro usuario legítimo del sistema) y realizar transacciones a su nombre.

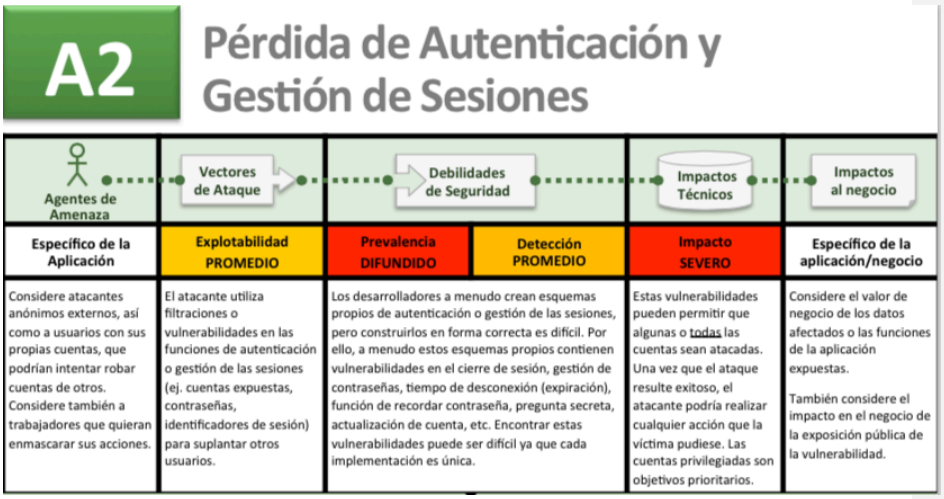
Paul (2011) afirma:

> Las aplicaciones funcionan relacionadas con autenticación y manejo de sesiones no siempre se implementan de forma correcta, permitiendo que los atacantes puedan comprometer contraseñas, llaves y tokens de sesiones, o explotar problemas de implementación para asumir la identidad de otro usuario. (p. 250).

La prevalencia de estos problemas de seguridad en la industria es muy alta debido a la confianza de las organizaciones en las aplicaciones Web como un mecanismo de negocios y a la proliferación de aplicaciones inseguras.

En la siguiente figura se ilustra el modelo de riesgo desarrollado por la fundación OWASP a fin de comprender los vectores de ataque y el impacto para el negocio. En tal ilustración se confirma que el impacto para la organización es alto, ya que puede comprometer información confidencial.

Ilustración 5 Pérdida de Autenticación y Gestión de Sesiones



Fuente : OWASP Top 10 Español

4.2.3.3 Secuencia de Comandos en Sitios Cruzados (XSS)

La Fundación OWASP (2008) al referirse a la Secuencia de Comandos en sitios cruzados afirma :

> La secuencia de comandos en sitios cruzados, más conocida como XSS, es en realidad un subconjunto de inyección HTML. XSS es la más prevaleciente y perniciosa problemática de seguridad en aplicaciones Web. Las fallas de XSS ocurren cuando una aplicación toma información originada por un usuario y la envía a un navegador Web sin primero validarla o codificando el contenido.
>
> XSS permite a los atacantes ejecutar secuencias de comandos en el navegador Web de la víctima, quienes pueden secuestrar sesiones de usuario, modificar sitios Web, insertar contenido hostil, realizar ataques de phising, y tomar control del navegador Web del usuario utilizando

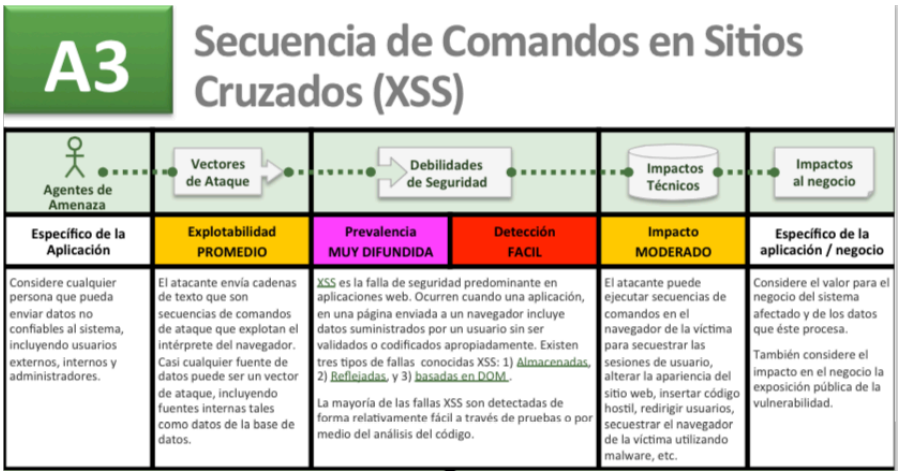
> secuencias de comando maliciosas. Generalmente JavaScript es utilizado, pero cualquier lenguaje de secuencia de comandos soportado por el navegador de la victima es un potencial objetivo para este ataque.

Esta vulnerabilidad tiene sus raíces en las fuentes de información no confiables , es decir cuya identidad no puede ser verificada, tal es el caso en datos provenientes del usuario final. No se puede confiar en los datos proporcionados por el usuario ya que éste podría ser un atacante buscando robar información sensitiva.

La recomendación emitida para disminuir el riesgo de XSS tiene su fundamento en la validación de los datos contra una lista de valores aceptables.

En la imagen siguiente se aprecia el modelo de riesgo proporcionado por la fundación OWASP para los problemas de XSS.

Ilustración 6 Secuencia de Comandos en Sitios Cruzados (XSS)



Fuente :OWASP Top 10 Español

4.2.3.4 Configuración de Seguridad Incorrecta

En seguridad informática se aplica la premisa que sostiene que una cadena es tan fuerte como el eslabón más débil. Lo difícil es poder encontrar a tal eslabón e incluso asegurar que todos los eslabones en la cadena de lo sistemas de información (incluyendo usuarios, infraestructura, sistemas) tengan el mismo nivel de seguridad.

El hecho de que participen varios actores en el proceso del software aunado a complejidades innecesarias al implementar soluciones de software permite que alguno de los componentes tenga alguna configuración incorrecta de seguridad.

Tales configuraciones incorrectas se reducen al uso de contraseñas por defecto (es decir las emitidas por el fabricante), puertos habilitados, software

desactualizado, mensajes de error que muestran información sensible, información de diagnóstico accesible por medio de internet entre otros.

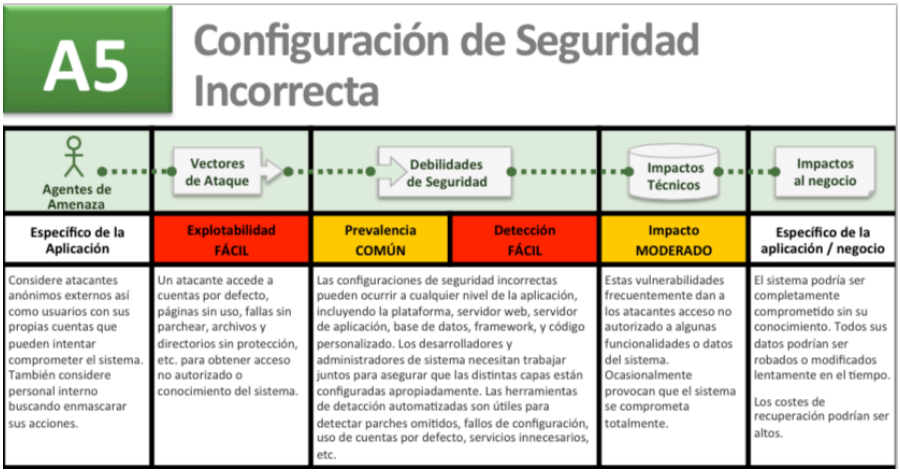
Es necesario comprender esta vulnerabilidad y hacer un esfuerzo por mejorar la seguridad. Es en este momento cuando se hace necesario un enfoque holístico en seguridad, es decir que los distintos componentes que forman parte de la infraestructura tecnológica, cuenten con el mismo nivel de seguridad, lo que permite que se disminuya la brecha de que alguno de tales componentes sea comprometido.

Para Paul (2011) :

> La seguridad depende de tener una configuración definida para la aplicación, plataforma, servidor Web y servidor de aplicaciones. Todas estas configuraciones deben estar definidas, implementadas y mantenidas ya que la mayoría no están incluidas por defecto. (p. 250).

En la imagen siguiente se detalla el modelo de riesgo y el impacto para la organización en cuanto a las configuraciones incorrectas de seguridad.

Ilustración 7 Configuración de Seguridad Incorrecta



Fuente :OWASP Top 10 Español

4.2.3.5 Exposición de datos sensibles

Los datos sensibles son aquellos cuya relevancia para la organización se ubica en los niveles más altos, tal es el caso de secretos comerciales, información financiera, información de usuarios o cualquier otro tipo de información que la organización considere que tiene un alto valor y que debe ser protegida.

Los atacantes o usuarios maliciosos buscan tener acceso a estos datos a fin de causar un daño irreparable a la organización o generar lucro de las actividades ilícitas. A manera de ejemplo se podría presentar el caso donde un usuario malicio ha tenido información a tarjetas de crédito que habían sido almacenadas de forma insegura por la organización. Esta persona

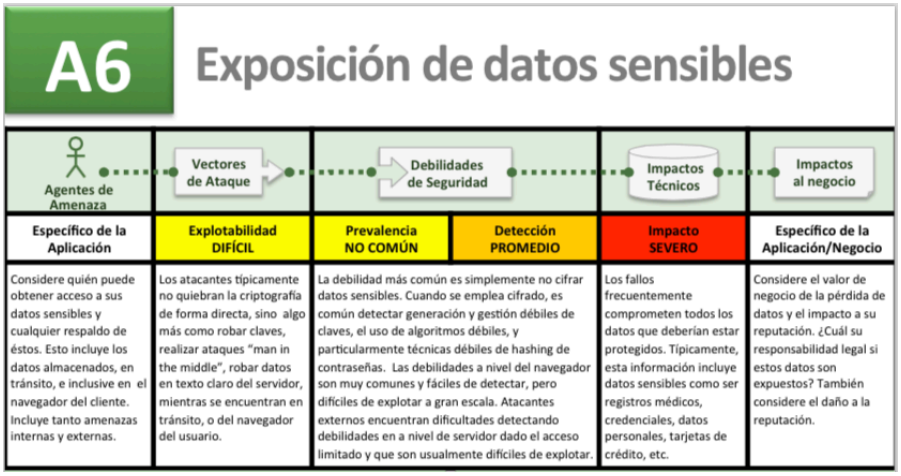
eventualmente venderá en el mercado negro toda la información de la brecha de seguridad permitiendo que haya exposición global a la información de clientes y a sus tarjetas de crédito, habilitando que hayan pérdidas millonarias tanto para los clientes como para las instituciones financieras como para la empresa víctima del ataque.

Se hace importante identificar cuales son estos datos sensibles para la empresa y poder definir si los controles aplicados actualmente son necesarios para protegerlos o si por el contrario se hace necesario implementar nuevos controles.

Cabe mencionar que mientras que la organización no tenga claramente definidos cuales son estos datos considerados sensibles ni el manejo que se les da desde las distintas aplicaciones, entonces la complejidad de que se protejan correctamente es aún mayor.

En la imagen siguiente se pude observar el modelo de riesgo desarrollado por la fundación OWASP y donde se puede apreciar el vector de ataque y el impacto para la organización.

Ilustración 8 Exposición de datos sensibles



Fuente: OWASP Top 10 2013 Español.

4.3 Diseño de entradas

Referencias Bibliográficas

Adams, Ed (Marzo 08, 2011). Q&A with Myself - Thoughts on Sony, DOD, RSA, IMF & Lockheed Martin. Recuperado el día 16 de Julio de 2014 de, http://goo.gl/74QBtY

Biblioteca Universidad de Alcalá (2014). *Tipos de fuentes de Información.* Recuperado el día Viernes 17 de Octubre de 2014 de http://goo.gl/fjjymp

Cáceres, I (2010). *Las variables*. Recuperado el día Viernes 17 de Octubre de 2014 de http://goo.gl/Ymm6np

Chaudhary, A. (2010). *What is a Community Technology Preview?.* Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/J5edZd.

Checkmarx CxSuit Overview (2014). Recuperado el 06 de Octubre de 2014 de http://goo.gl/HYxggf.

File, G. (2011). *The SWOT Analysis Using Strengths to overcome Weakness Using Opportunities to overcome Threats.* Kindle Edition. Recuperado de Amazon.com

Forbes (2014). *Hewlett-Packard on the Forbes Global 2000 List*. Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/NEIiiZ.

Forbes (2014). *Global 2000 Leading Companies. Microsoft*. Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/HOcYhk.

Fortify Static Code Analyzer (2014). Recuperado el 06 de Octubre de 2014 de http://goo.gl/P6m7Vf.

Fraim, D., Kane, K (2014). *Global Cost of Data Breach Increases by 15 percent, According to Ponemon Institute.* Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/8BpzLM.

Fundación Eclipse (2014). *About the Eclipse Foundation*. Recuperado el Lunes 06 de Octubre de 2014.

Hazzard, K., Bock, J. (2013). *Metaprogramming in .NET.* Shelter Island, New York: Manning Publications Co.

Hejlsberg, A., Torgerse, M., Wiltamuth, S., Golde, P. (2010). *The C# Programming Language (Covering C# 4.0).* Boston, MA: Pearson Education, Inc.

Ioannou, C. (2012). *SWOFT Analysis An easy to understand guide.* Kindle Edition. Recuperado de Amazon.com

IBM (2014). *Structured Query Language (SQL).* Recuperado el Viernes 24 de Agosto de 2014 de http://goo.gl/cEdyKT

Kendall, K., Kendall, J. (2011). *Análisis y Diseño de Sistemas.* Naucalpan de Juárez, Estado de México : Prentice Hall.

Lair, R (2012). *Beginning Silverlight 5 in C#, Fourth Edition*. Apress.

Lee, C. (2011). *How Do You Cite an E-Book (e.g., Kindle Book)?*. Recuperado de http://blog.apastyle.org/apastyle/2011/06/how-do-you-cite-an-e-book.html

Microsoft (2014). *Introducción al lenguaje C# y .NET Framework.* Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/c2ym0J.

Mishra, J., Mohanty, A (2011). *Software Engineering.* Pearson India.

MITRE Corporation (2014). Corporate Overview . Recuperado el Viernes 10 de Octubre de 2014 .

Monge, R., Hewitt, J (2004). *Tecnologías de Información y las Comunicaciones (TICs) y el futuro del desarrollo de Costa Rica.* Recuperado el Lunes 06 de Octubre de 2014 de http://www.caatec.org/CAATEC/publicaciones/crdigital/CR_Digital_3.pdf

Osenkov, K. (2011). *Today we are releasing the first Community Technology Preview of the Roslyn Project.* Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/UrseJn.

Osherove, R. (2009). *The art of Unit Testing with Examples in .NET.* Greenwich, CT: Manning Publications Co.

Paul, M. (2011). *Official (ISC)2 Guide to the CSSLP*. New York, New York: CRC Press Taylor & Francis Group.

Norton Symantec (2014). ¿Qué es el crimen cibernético?. Recuperado el Lunes 06 de Octubre de 2014 de http://mx.norton.com/cybercrime-definition.

Security Innovation (2014). *A Passion for Application Security*. Recuperado el Lunes 06 de Octubre de 2014.

Security Innovation (2014). *Secure Software Development Guidance.* Recuperado el Viernes 10 de Octubre de 2014 de http://goo.gl/bsPzER.

Somasegar (2011). *Roslyn CTP now Available.* Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/tYmDrd.

Sommerville, Ian. (2011). *Ingeniería de Software*. Naucalpan de Juárez, Estado de México: Addison Wesley.

Spiegel, M., Stephens, L (2001). *Estadística.* Distrito Federal, México: McGraw-Hill Interamericana.

Stroustrup, B. (2009). *Programming Principles and Practice Using C++.* Boston, MA: Addison Wesley.

Tanenbaum, A. (2010). *Computer Networks, Fifth Edition*. Pretince Hall.

The New York Times (2014). International Business Machines Corporation. Recuperado el Lunes 06 de Octubre de 2014 de http://goo.gl/5BKbDP.