

**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**

**Diciembre, 2014**

## TABLA DE CONTENIDOS

<b>TABLA DE CONTENIDOS .....</b>	<b>II</b>
<b>ÍNDICE DE CUADROS .....</b>	<b>XII</b>
<b>ÍNDICE DE GRÁFICOS .....</b>	<b>XIII</b>
<b>ÍNDICE DE FIGURAS.....</b>	<b>XIV</b>
<b>CARTA DEL TUTOR.....</b>	<b>XVIII</b>
<b>CARTA DEL FILÓLOGO.....</b>	<b>XIX</b>
<b>CÓDIGO DE ÉTICA .....</b>	<b>XX</b>
<b>CARTA DE LA DIRECTORA DE CARRERA .....</b>	<b>XXI</b>
<b>DEDICATORIA .....</b>	<b>XXII</b>
<b>AGRADECIMIENTOS .....</b>	<b>XXIII</b>
<b>RESUMEN EJECUTIVO .....</b>	<b>XXIV</b>
<b>INTRODUCCIÓN .....</b>	<b>27</b>
1. Tema.....	2
2. Planteamiento del problema de estudio.....	2
3. Justificación .....	10
3.1 Estudio de Viabilidad de la propuesta.....	12

<b>3.2 Estudio de viabilidad técnica.....</b>	<b>13</b>
<b>3.3 Estudio de viabilidad económica .....</b>	<b>15</b>
<b>3.4 Estudio de viabilidad operativa.....</b>	<b>17</b>
<b>4. Objetivos de la investigación .....</b>	<b>20</b>
<b>4.1 Objetivo General .....</b>	<b>20</b>
<b>4.2 Objetivos Específicos .....</b>	<b>20</b>
4.2.1 Realizar el levantamiento de requerimientos de cada una de las vulnerabilidades a identificar mediante el uso de estándares en la industria. ....	20
4.2.2 Elaborar el diseño del <i>software</i> que contempla el flujo de trabajo, la identificación de vulnerabilidades y la retroalimentación al usuario final.....	20
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.....	20
4.2.4 Implementar las reglas de diagnóstico para detectar vulnerabilidades en el código fuente utilizando estándares en la industria.....	21
4.2.5 Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo.....	21
<b>5. Alcances .....</b>	<b>21</b>
<b>6. Limitaciones .....</b>	<b>25</b>
<b>7. Antecedentes.....</b>	<b>25</b>
<b>7.1 El modelo de ejecución del CLR.....</b>	<b>28</b>
<b>7.2 Compiladores como cajas negras .....</b>	<b>30</b>
<b>7.3 El Proyecto Roslyn: Abriendo la caja negra.....</b>	<b>33</b>

<b>8. Referente Institucional.....</b>	<b>35</b>
<b>CAPÍTULO I.....</b>	<b>39</b>
<b>DIAGNÓSTICO.....</b>	<b>39</b>
<b>    1.1 Análisis FODA.....</b>	<b>40</b>
<b>    1.2 Análisis FODA para el prototipo funcional. ....</b>	<b>42</b>
<b>    1.3 Fortalezas.....</b>	<b>44</b>
1.3.1 Amplia experiencia en el mercado de la seguridad de las aplicaciones. ....	44
1.3.2 Mayor comercialización de los productos:.....	44
1.3.3 Integración con empresas en el mercado de la seguridad de las aplicaciones.	44
1.3.4 Empresa cuenta con áreas de investigación y desarrollo donde se produce tecnología de vanguardia. ....	45
1.3.5 Facultar a empresas a desarrollar aplicaciones de <i>software</i> más seguras. ....	45
<b>    1.4 Oportunidades.....</b>	<b>46</b>
1.4.1 Creciente demanda en seguridad de aplicaciones por parte de la industria....	46
1.4.2 Rápida evolución del lenguaje de programación C#. ....	47
1.4.3 Herramienta integrada en el ambiente de desarrollo .....	47
1.4.4 Acercamiento de nuevos clientes potenciales .....	48
<b>    1.5 Debilidades.....</b>	<b>48</b>
1.5.1 Poca o nula inserción en el campo del análisis estático de código .....	48
1.5.2 Dependencia de terceras empresas para realizar el análisis estático de código. .....	49
1.5.3 Proyectos de código abierto y gratuito ofrecen productos similares a muy bajo costo. ....	49

1.5.4 <i>Plugin</i> limitado a un lenguaje de programación y a un entorno integrado de desarrollo.....	50
<b>1.6 Amenazas .....</b>	<b>50</b>
1.6.1 La competencia ofrece productos similares e integrados .....	50
1.6.2 Clientes prefieren productos unificados .....	51
<b>CAPÍTULO II.....</b>	<b>57</b>
<b>MARCO TEÓRICO.....</b>	<b>57</b>
<b>2.1 Sistemas de Información.....</b>	<b>58</b>
<b>2.2 Desarrollo de Sistemas .....</b>	<b>60</b>
2.2.1 Modelo en Cascada (waterfall) .....	61
2.2.2 Desarrollo Incremental.....	63
2.2.3 Ingeniería de <i>software</i> orientada a reutilización.....	64
<b>2.3 <i>Plugin</i> .....</b>	<b>65</b>
<b>2.4 Prototipo .....</b>	<b>66</b>
<b>2.5 Tecnologías de Información y Comunicaciones.....</b>	<b>67</b>
<b>2.6 <i>Software</i> .....</b>	<b>67</b>
<b>2.7 <i>Hardware</i>.....</b>	<b>68</b>
<b>2.8 Computadora .....</b>	<b>69</b>
<b>2.9 Aplicaciones de <i>software</i>.....</b>	<b>70</b>
<b>2.10 Usuario malicioso .....</b>	<b>71</b>
<b>2.11 Hacker.....</b>	<b>71</b>
<b>2.12 Activo .....</b>	<b>72</b>
<b>2.13 Vulnerabilidad .....</b>	<b>72</b>

<b>2.14 Ataque informático.....</b>	<b>73</b>
<b>2.15 Probabilidad .....</b>	<b>73</b>
<b>2.16 Impacto.....</b>	<b>74</b>
<b>2.17 Factor de exposición .....</b>	<b>74</b>
<b>2.18 Controles.....</b>	<b>74</b>
<b>2.19 Políticas de seguridad .....</b>	<b>74</b>
<b>2.20 Crimen cibernético .....</b>	<b>75</b>
<b>2.21 Microsoft.....</b>	<b>76</b>
<b>2.22 Hewlett Packard .....</b>	<b>76</b>
<b>2.23 HP Fortify.....</b>	<b>77</b>
<b>2.24 IBM .....</b>	<b>78</b>
<b>2.25 IBM Security AppScan.....</b>	<b>78</b>
<b>2.26 Checkmarx.....</b>	<b>79</b>
<b>2.27 Instituto Ponemon .....</b>	<b>80</b>
<b>2.28 Security Innovation.....</b>	<b>81</b>
<b>2.29 MITRE.....</b>	<b>82</b>
<b>2.30 CWE.....</b>	<b>82</b>
<b>2.31 TEAM Mentor.....</b>	<b>84</b>
<b>2.32 Entorno de desarrollo Integrado .....</b>	<b>84</b>
<b>2.33 Visual Studio .Net .....</b>	<b>85</b>
<b>2.34.NET Framework.....</b>	<b>85</b>
<b>2.35 Lenguaje de programación C# .....</b>	<b>87</b>
<b>2.36 Eclipse .....</b>	<b>87</b>
<b>2.37 SQL (Structured Query Language).....</b>	<b>88</b>

<b>2.38 Inyección de SQL .....</b>	<b>89</b>
<b>2.39 Pérdida de Autenticación y Gestión de Sesiones.....</b>	<b>90</b>
<b>2.40 Secuencia de Comandos en Sitios Cruzados (XSS) .....</b>	<b>91</b>
<b>2.41 Exposición de datos sensibles .....</b>	<b>92</b>
<b>2.42 Configuración Incorrecta de Seguridad .....</b>	<b>93</b>
<b>2.43 JavaScript.....</b>	<b>94</b>
<b>2.44 HTTP.....</b>	<b>94</b>
<b>2.44 <i>Cookies</i> .....</b>	<b>95</b>
<b>2.45 Cabeceras HTTP.....</b>	<b>96</b>
<b>2.44 Refactorización.....</b>	<b>97</b>
<b>2.45 HTTP GET.....</b>	<b>97</b>
<b>2.46 HTTP POST .....</b>	<b>97</b>
<b>2.47 XML .....</b>	<b>98</b>
<b>2.48 Wi-Fi .....</b>	<b>98</b>
<b>CAPÍTULO III .....</b>	<b>99</b>
<b>MARCO METODOLÓGICO.....</b>	<b>99</b>
<b>3.1 Métodos de Investigación .....</b>	<b>101</b>
<b>3.1.1 Método Científico .....</b>	<b>101</b>
<b>3.1.2 Método Inductivo .....</b>	<b>102</b>
<b>3.1.3 Método Deductivo.....</b>	<b>103</b>
<b>3.1.4 Método Cuantitativo.....</b>	<b>104</b>
<b>3.1.4 Método Cualitativo .....</b>	<b>106</b>
<b>3.1.4 Método de Investigación Utilizado.....</b>	<b>107</b>

<b>3.2 Tipos de Investigación .....</b>	<b>108</b>
<b>3.2.1 Descriptiva .....</b>	<b>108</b>
<b>3.2.2 Exploratoria .....</b>	<b>109</b>
<b>3.2.3 Correlacional .....</b>	<b>110</b>
<b>3.2.4 Explicativo .....</b>	<b>110</b>
<b>3.2.5 Tipo de Investigación Seleccionada.....</b>	<b>111</b>
<b>3.3 Fuentes de Información.....</b>	<b>112</b>
<b>3.3.1 Fuentes Primarias.....</b>	<b>112</b>
<b>3.3.2 Fuentes Secundaria .....</b>	<b>113</b>
<b>3.3.3 Fuentes Terciaria .....</b>	<b>113</b>
<b>3.3.4 Fuente de Información Seleccionada .....</b>	<b>113</b>
<b>3.4 Descripción de Variables .....</b>	<b>114</b>
<b>3.4.1 Definición Conceptual .....</b>	<b>115</b>
<b>3.4.2 Definición Operacional .....</b>	<b>115</b>
<b>3.4.3 Definición Instrumental .....</b>	<b>115</b>
<b>3.5 Cuadro de Variables.....</b>	<b>116</b>
<b>Plataforma para realizar pruebas unitarias NUNIT.....</b>	<b>117</b>
<b>3.6 Población y Muestra .....</b>	<b>118</b>
<b>3.6.1 Población .....</b>	<b>118</b>
<b>3.6.2 Muestra.....</b>	<b>119</b>
<b>3.6.3 Selección de la Población y de la muestra .....</b>	<b>119</b>
<b>3.7 Instrumentos de recolección de datos .....</b>	<b>121</b>

<b>3.7.1 Entrevista .....</b>	<b>122</b>
<b>3.7.2 Cuestionario.....</b>	<b>122</b>
<b>3.7.3 Instrumento de recolección de datos seleccionado .....</b>	<b>123</b>
<b>3.7.4 Relación entre objetivos, definición instrumental y fuentes de información.</b>	
.....	<b>124</b>
<b>Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo. ....</b>	<b>125</b>
Fuente: Propia.....	125
<b>3.7.5 Relación entre objetivos, entregables y las herramientas.....</b>	<b>125</b>
<b>Desarrollar pruebas funcionales, pruebas de integración y pruebas unitarias del prototipo. ....</b>	<b>127</b>
<b>CAPÍTULO IV .....</b>	<b>128</b>
<b>DISEÑO .....</b>	<b>128</b>
<b>1. DISEÑO .....</b>	<b>129</b>
<b>1.1 Descripción de la arquitectura física y lógica .....</b>	<b>129</b>
<b>1.2 Análisis de Requerimientos .....</b>	<b>135</b>
<b>4.2.4 Identificación de vulnerabilidades .....</b>	<b>136</b>
<b>4.2.4.1 El Proyecto OWASP Top 10 .....</b>	<b>137</b>
<b>1.3 Diseño de la solución.....</b>	<b>139</b>
<b>1.3.1 UML .....</b>	<b>140</b>
<b>1.3.2 Casos de uso .....</b>	<b>141</b>
<b>1.3.2.1 Creación de un proyecto nuevo o selección de uno existente.....</b>	<b>141</b>

<b>Fuente: Propia .....</b>	143
1.3.2.2 Compilación como servicio.....	143
<b>1.3.3 Diagrama de Clases.....</b>	162
<b>1.3.4 Desarrollo del Prototipo Funcional .....</b>	164
<b>Fuente: Propia .....</b>	196
<b>1.4 Pruebas .....</b>	<b>199</b>
<b>1.4.1 Pruebas Unitarias.....</b>	201
<b>1.4.2 Pruebas Unitarias para el Prototipo.....</b>	202
<b>1.4.3 Pruebas Funcionales del Prototipo .....</b>	204
<b>1.4.3.4 Detección de Vulnerabilidades de Secuencia de Comandos entre Páginas.....</b>	207
<b>1.4.3.5 Detección de Vulnerabilidades de Inyección de SQL .....</b>	208
<b>1.4.3.6 Detección de Vulnerabilidades de Autenticación Rota y Manejo de Sesiones.....</b>	210
<b>1.4.3.7 Detección de Vulnerabilidades de Exposición de datos Sensibles.....</b>	212
<b>1.4.3.8 Detección de Vulnerabilidades de Configuración Incorrecta de Seguridad. ....</b>	213
<b>2. INTERPRETACIÓN DE RESULTADOS.....</b>	<b>215</b>
<b>2.1 Observación como instrumento de análisis .....</b>	<b>215</b>
<b>2.1.1 Observación del estado de la seguridad.....</b>	218
<b>2.2 Resultados de la encuesta.....</b>	<b>220</b>
<b>2.2.1 Pregunta 1.....</b>	220
<b>2.2.2 Pregunta 2.....</b>	221
<b>2.2.3 Pregunta 3.....</b>	223

<b>2.2.4 Pregunta 4.....</b>	225
<b>2.2.5 Pregunta 5.....</b>	225
<b>2.2.6 Pregunta 6.....</b>	226
<b>2.2.7 Pregunta 7.....</b>	227
<b>2.2.8 Pregunta 8.....</b>	227
<b>2.2.9 Pregunta 9.....</b>	228
<b>2.2.10 Pregunta 10.....</b>	230
<b>3. Aportes .....</b>	Error! Marcador no definido.
<b>CONCLUSIONES .....</b>	<b>231</b>
<b>RECOMENDACIONES .....</b>	<b>238</b>
<b>BIBLIOGRAFÍA Y OTRAS FUENTES .....</b>	<b>247</b>
<b>ANEXOS .....</b>	<b>252</b>
<b>Anexo 1: .....</b>	<b>252</b>
<b>Anexo 2: .....</b>	<b>253</b>
<b>Anexo 3: .....</b>	<b>254</b>
<b>Anexo 4: .....</b>	<b>255</b>

# ÍNDICE DE CUADROS

CUADRO 1 COSTOS .....	16
CUADRO 2 ANÁLISIS FODA .....	43
CUADRO 3 RELACIÓN ENTRE OBJETOS, DEFINICIÓN INSTRUMENTAL Y FUENTES DE INFORMACIÓN.....	124
CUADRO 4 RELACIÓN ENTRE OBJETOS, ENTREGABLES Y HERRAMIENTAS.....	125
CUADRO 5 CREACIÓN O SELECCIÓN DE UN PROYECTO EXISTENTE.....	141
CUADRO 6 COMPILACIÓN DEL CÓDIGO FUENTE COMO SERVICIO.....	144
CUADRO 7 MÓDULO DE VULNERABILIDADES DE INYECCIÓN DE SQL.....	148
CUADRO 8 MÓDULO DE VULNERABILIDADES DE XSS. ....	151
CUADRO 9 VULNERABILIDADES DE PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIONES ..	155
CUADRO 10 CONFIGURACIÓN INCORRECTA DE SEGURIDAD .....	159
CUADRO 11 INSTALACIÓN DEL COMPONENTE EN VISUAL STUDIO.NET .....	205
CUADRO 12 DESHABILITAR EL COMPONENTE DENTRO DE VISUAL STUDIO.NET .....	206
CUADRO 13 DESINSTALACIÓN DEL COMPONENTE .....	207
CUADRO 14 PRUEBA DE DETECCIÓN DE SECUENCIAS DE COMANDOS ENTRE PÁGINAS.....	208
CUADRO 15 DETECCIÓN DE VULNERABILIDADES DE INYECCIÓN DE SQL.....	209
CUADRO 16 DETECCIÓN DE VULNERABILIDADES DE AUTENTICACIÓN ROTA .....	211
CUADRO 17 PRUEBA EXPOSICIÓN DE DATOS SENSIBLES.....	212
CUADRO 18 DETECCIÓN DE CONFIGURACIONES INCORRECTAS DE SEGURIDAD .....	214
CUADRO 19 RESULTADOS DE LA PREGUNTA 1 .....	221
CUADRO 20 RESULTADOS DE LA PREGUNTA 2 .....	222
CUADRO 21 RESULTADO DE LA PREGUNTA 3 .....	223
CUADRO 22 RESPUESTAS DE LA PREGUNTA 8 .....	227
CUADRO 23 RESPUESTAS DE LA PREGUNTA 9 .....	228

## ÍNDICE DE GRÁFICOS

GRÁFICO 1 COSTO RELATIVO DE ARREGLAR DEFECTOS DE CÓDIGO .....	12
GRÁFICO 2 UBICACIÓN DE LA EXTENSIÓN DE SEGURIDAD .....	171
GRÁFICO 3 RESULTADOS DE LA PREGUNTA NÚMERO 2 .....	222
GRÁFICO 4 RESULTADO DE LA PREGUNTA 3 .....	224
GRÁFICO 5 RESPUESTAS DE LA PREGUNTA 5 .....	229

## ÍNDICE DE FIGURAS

FIGURA 1 COMPILANDO CÓDIGO FUENTE EN MÓDULOS MANEJADOS .....	30
FIGURA 2 PROCESO DE COMPILACIÓN EN UN AMBIENTE ADMINISTRADO .....	31
FIGURA 3 ELEMENTOS DE LA PLATAFORMA ROSLYN.....	34
FIGURA 4 TRES PILARES DEL DESARROLLO DE SOFTWARE SEGURO. ....	36
FIGURA 5 ORGANIGRAMA SECURITY INNOVATION .....	38
FIGURA 6 LISTADO DE DEFECTOS O ASUNTOS CREADOS PARA LA PLATAFORMA ROSLYN.....	55
FIGURA 7 LICENCIA APACHE 2.0 DE LA PLATAFORMA DE COMPILACIÓN ROSLYN.....	56
FIGURA 8 MODELO EN CASCADA (WATERFALL).....	63
FIGURA 9 INGENIERÍA DE SOFTWARE ORIENTADA A LA REUTILIZACIÓN .....	65
FIGURA 10 EXTENSIONES DISPONIBLES PARA FIREFOX.....	66
FIGURA 11 EJEMPLO DE HARDWARE.....	69
FIGURA 12 COMPUTADORA ORDINARIA .....	70
FIGURA 13 FLUJO DE TRABAJO DE IBM APPSCAN.....	79
FIGURA 14 CHECKMARX EN EL CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE .....	80
FIGURA 15 FLUJO DE TRABAJO DE UN CWE.....	83
FIGURA 16 ARQUITECTURA DE MICROSOFT .NET 4.5.....	86
FIGURA 17 VERSIONES DE ECLIPSE DURANTE LOS ÚLTIMOS AÑOS .....	88
FIGURA 18 TRANSMISIÓN DE COOKIES.....	96
FIGURA 19 INVESTIGACIÓN CUANTITATIVA.....	106
FIGURA 20 INVESTIGACIÓN CUALITATIVA Y CUANTITATIVA .....	107
FIGURA 21 ALCANCES DE LA INVESTIGACIÓN .....	111
FIGURA 22 POBLACIÓN Y MUESTRA.....	119
FIGURA 23 EVOLUCIÓN DE C# Y VISUAL BASIC .....	131
FIGURA 24 PASOS DEL PROCESO DE COMPILACIÓN.....	132
FIGURA 25 ARQUITECTURA DEL SISTEMA.....	133

FIGURA 26 MODELO DE RIESGOS .....	138
FIGURA 27 CASO DE USO 1 - CREACIÓN O SELECCIÓN DE UN PROYECTO .....	141
FIGURA 28 CASO DE USO 2 - COMPILACIÓN COMO SERVICIO .....	144
FIGURA 29 MÓDULO DE VULNERABILIDADES DE INYECCIÓN DE SQL .....	148
FIGURA 30 VULNERABILIDADES DE SECUENCIA DE SITIOS CRUZADOS (XSS) .....	151
FIGURA 31 PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIONES .....	155
FIGURA 32 CASO DE USO CONFIGURACIÓN INCORRECTA DE SEGURIDAD.....	159
FIGURA 29 DIAGRAMA DE CLASES .....	163
FIGURA 30 DIAGRAMA DE CLASES CONTINUACIÓN .....	164
FIGURA 35 PLANTILLAS DE DESARROLLO .....	167
FIGURA 36 BARRA DE HERRAMIENTAS.....	169
FIGURA 37 PLUGIN INSTALADO DENTRO DE VISUAL STUDIO.....	170
FIGURA 38 INYECCIÓN DE SQL MODELO DE RIESGO.....	174
FIGURA 39 INYECCIÓN DE SQL EN UN FORMULARIO HTML.....	176
FIGURA 40 INYECCIÓN DE SQL EN EL CÓDIGO FUENTE.....	177
FIGURA 41 PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIONES MODELO DE RIESGO .....	180
FIGURA 42 DIAGRAMA PÉRDIDA DE AUTENTICACIÓN .....	181
FIGURA 43 MANIPULACIÓN DE VARIABLES EN SESIÓN.....	183
FIGURA 44 SECUENCIA DE COMANDOS EN SITIOS CRUZADOS MODELO DE RIESGO .....	185
FIGURA 45 INGRESO DE DATOS NO CONFIABLES .....	186
FIGURA 46 ALTERACIÓN DE UNA PÁGINA WEB POR MEDIO DE XSS.....	188
FIGURA 47 CÓDIGO FUENTE VULNERABLE A XSS.....	189
FIGURA 48 MATERIALIZACIÓN DE UN ATAQUE DE SECUENCIA DE COMANDOS ENTRE PÁGINAS .....	190
FIGURA 49 ENVÍO DE CÓDIGO JAVASCRIPT EN LA APLICACIÓN .....	191
FIGURA 50 DETECCIÓN DE CÓDIGO FUENTE VULNERABLE POR MEDIO DEL PLUGÍN .....	192
FIGURA 51 VISTA PREVIA DE LA CORRECCIÓN DEL PROBLEMA DE XSS .....	193

FIGURA 52 CORRECCIÓN DEL PROBLEMA DE SEGURIDAD EN EL CÓDIGO FUENTE.....	193
FIGURA 53 CONFIGURACIÓN DE SEGURIDAD INCORRECTA MODELO DE RIESGO.....	195
FIGURA 54 CONFIGURACIÓN DE SEGURIDAD INCORRECTA POR MEDIO DE MENSAJES DE ERROR .....	196
FIGURA 55 EXPOSICIÓN DE DATOS SENSIBLES MODELO DE RIESGO .....	198
FIGURA 56 DIAGRAMA DE EXPOSICIÓN DE DATOS SENSIBLES.....	199
FIGURA 57 PROYECTO DE PRUEBAS UNITARIAS.....	202
FIGURA 58 ESCENARIO DE PRUEBA DE UNIDAD .....	203
FIGURA 59 PRUEBAS UNITARIAS EJECUTADAS .....	204
FIGURA 60 CHECKMARX MOSTRANDO VULNERABILIDADES DENTRO DE VISUAL STUDIO .....	217
FIGURA 61 MOSTRANDO UN ARTÍCULO DE TEAM MENTOR DENTRO DEL IDE .....	217

## **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, 19 de diciembre de 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, 2 de diciembre de 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 1-1275-0522, 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,

Andrea Araya Fonseca  
Carné ACFIL 0087

# 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, 19 de diciembre de 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 1-1275-0522, 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**

A mi esposa Heilen Rojas por todo su amor incondicional y por todo su apoyo, no hay palabras para describir lo afortunado que soy de compartir nuestras vidas juntos.

## **AGRADECIMIENTOS**

A todas las personas que son y siguen siendo mi inspiración para ser cada día una mejor persona.

Al profesor Leonardo Delgado por su excelencia profesional y por su nivel de compromiso en cada tarea, por sus recomendaciones y consejos que siempre me ha brindado y que me hacen ser un mejor profesional.

A la fundación OWASP y a todas las personas que trabajan en la comunidad para mejorar la seguridad del software.

## RESUMEN EJECUTIVO

Existe hoy en día una alta dependencia en las aplicaciones de *software* a tal punto que se puede afirmar que existen más computadoras en el mundo que personas. Según un estudio que data del año 2005, generado por la Asociación de la Industria de los Semiconductores (Semiconductor Industry Association) y que lleva por título “**2020 Is Closer Than You Think**”<sup>1</sup> indica que: “El año pasado, más transistores fueron producidos- y a más bajo costo- que granos de arroz”.

Esta aseveración, aunque impactante en naturaleza, ha sido una constante durante los últimos años ya que cada vez más dispositivos incorporan computadoras como parte de sus operaciones.

Esta tendencia ha venido en aumento y se ha respaldado con la popular pero cierta ley de Gordon Moore, también denominada Ley de Moore, la cual establece que: “El número de componentes en un circuito integrado se duplica aproximadamente cada 12 meses con una reducción considerable en el costo por componente”.<sup>2</sup>

Sin embargo, aunado a este crecimiento en el uso de aplicaciones de *software* y de computadores, el crimen cibernetico tiene también un crecimiento exponencial. La proliferación de aplicaciones web transaccionales, dispositivos móviles y puntos de acceso a través de redes inalámbricas

---

<sup>1</sup> [http://www.semiconductors.org/clientuploads/SIA\\_AR\\_2005.pdf](http://www.semiconductors.org/clientuploads/SIA_AR_2005.pdf)

<sup>2</sup> [http://www.semiconductors.org/clientuploads/SIA\\_AR\\_2005.pdf](http://www.semiconductors.org/clientuploads/SIA_AR_2005.pdf)

denominadas Wi-Fi, ayudan sustancialmente a que usuarios maliciosos comprometan la seguridad de las empresas utilizando en un alto porcentaje la misma aplicación para perpetrar el crimen.

En esta tesis, se propone el desarrollo de una extensión o *plugin* para el ambiente integrado Visual Studio.NET 2013; el cual realiza técnicas de análisis estático del código fuente a fin de encontrar vulnerabilidades o código bajo el lenguaje de programación C#. Con el desarrollo de esta extensión, se busca que los ingenieros comprendan mejor las causas que hacen que el *software* sea inseguro y se puedan solucionar en una etapa temprana del ciclo de vida del desarrollo del *software*.

En el capítulo I de esta tesis, se hace un análisis FODA del prototipo funcional propuesto en el marco de la industria, a fin de encontrar oportunidades de mejora. Se han identificado las fortalezas, oportunidades, debilidades y amenazas para el prototipo funcional.

En el capítulo II, que corresponde al marco teórico, se contemplan las definiciones y terminologías utilizadas en este documento, de tal forma que sirva como referente en el momento de encontrar algún término técnico dentro del mismo documento. En esta sección, se han incorporado las definiciones más acertadas de los autores con el fin de que el concepto en sí sea lo más explicativo posible.

Por su parte, el capítulo III incluye el marco metodológico donde se expresan las herramientas, instrumentos y procedimientos utilizados en el momento de realizar la investigación para esta tesis. Además, se explica el

tipo de investigación que ha sido utilizada, las técnicas para recolección de información y fuentes de información que forman parte del documento.

El capítulo IV corresponde al diseño del prototipo funcional propuesto. Utilizando las diferentes etapas que conforman el ciclo del desarrollo del *software*; se ha identificado los requerimientos funcionales, requerimientos técnicos y las herramientas necesarias para una adecuada recopilación de la información. Haciendo uso de diagramas se han ilustrado los casos de uso identificados, los diagramas de arquitectura del *plugin* y los diagramas de clases respectivos.

En esta sección, se hace un análisis exhaustivo de las vulnerabilidades en el código fuente que la extensión de seguridad pretende diagnosticar, se analizan los vectores de ataque, modelo de riesgo, impacto para el negocio y por su puesto la solución al problema desde la perspectiva del código.

La segunda parte de este capítulo corresponde a la interpretación de los resultados donde se hace un análisis de los datos recolectados por medio de la encuesta aplicada, se grafican aquellos casos donde se ha considerado oportuno mostrar el comportamiento mismo de los datos. Se explica además los resultados encontrados luego de la aplicación del instrumento.

## **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 relacionarse. 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 al 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)*<sup>3</sup>, 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.

---

<sup>3</sup> <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 las 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 cuáles son los problemas de seguridad comunes, cómo poder identificar estos problemas y cómo 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 se habla 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 cómo 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 y 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 cumplen 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 llamada OWASP Top 10, la cual está 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* es que tome en consideración los riesgos y las debilidades descritas por estos dos estándares y poder darle retroalimentación al desarrollador cuando accidentalmente 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 resolverlos 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 manera, la plataforma TEAM Mentor es consumida por diferentes consultores, arquitectos de *software* y expertos en seguridad como un complemento para recomendar cómo 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 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 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* cumplir 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 se ha visto cómo la cantidad de incidentes en seguridad han venido en aumento. Estos ataques no discriminan ningún sector social. Los encabezados de las noticias confirman una vez más que cualquier organización puede ser víctima de ataques ciberneticos, 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 revela la malas prácticas en el desarrollo de aplicaciones de *software* que terminan comprometiendo a organizaciones, los datos sensitivos de estas 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/6e4CIN>. 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.

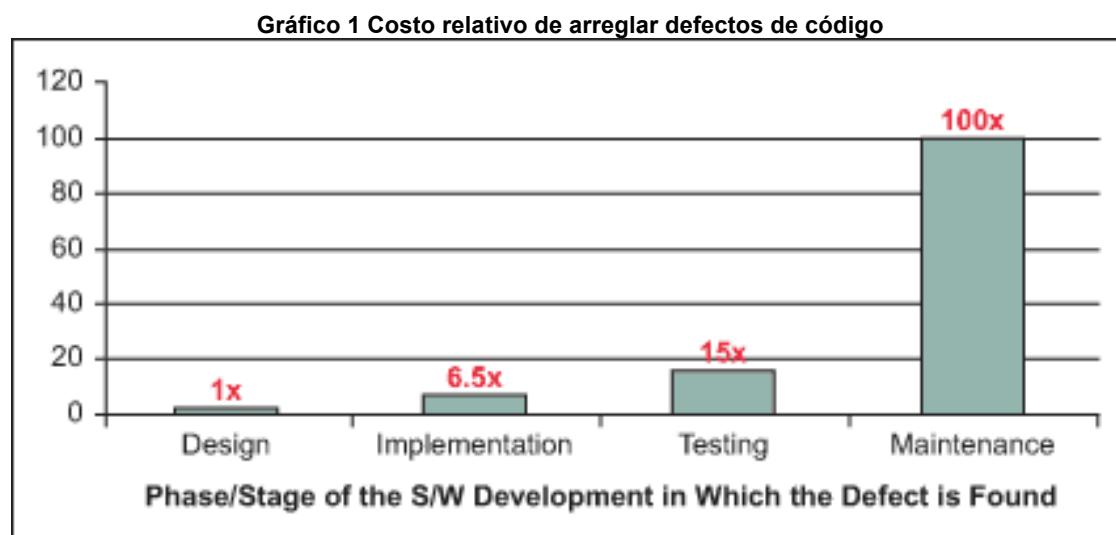
Pese a las pérdidas económicas, daños morales y efectos colaterales producto de un incidente cibernetico (incluyendo comprometer otros servicios y aplicaciones como lo son las redes sociales como es el caso de Facebook, Twitter y cuentas de correo electrónico) se ve que pese al esfuerzo de la industria 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.

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

Históricamente, la 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<sup>4</sup> 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:



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

---

<sup>4</sup> [http://www.riceconsulting.com/public\\_pdf/STBC-WM.pdf](http://www.riceconsulting.com/public_pdf/STBC-WM.pdf)

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 suficientes 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 las 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 a las 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 disponibles de forma gratuita desde el sitio respectivo provisto por Microsoft. La adquisición de estas es trivial, ya que existe documentación vasta para poder instalar los componentes requeridos.

Así mismo, el lenguaje de programación C# (pronunciado C Sharp), 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

<b>Gastos Operativos</b>				
<b>Cantidad</b>	<b>Descripción</b>	<b>Horas</b>	<b>Costo por Hora</b>	<b>Sub Total</b>
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
<b>Total de Gastos Operativos</b>				<b>\$5,100.00</b>

<b>Materiales y Misceláneos</b>				
<b>Cantidad</b>	<b>Descripción</b>	<b>Costo de Mercado</b>	<b>Costo para el Proyecto</b>	<b>Sub Total</b>
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
<b>Total de Gastos Operativos</b>				<b>\$300.00</b>

<b>Total de Gastos</b>			
<b>Gastos Operativos</b>	<b>Gastos</b>	<b>Misceláneos</b>	<b>Total</b>
<b>\$5,100.00</b>		<b>\$300.00</b>	<b>\$5,400.00</b>

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 \$5,400.00. 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 ese 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 cuál 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 se puede inferir que el desarrollador de *software* es más productivo ya que la mayor parte del tiempo invierte en desarrollar un código nuevo o da 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 más 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 exitosamente esta fortaleza, de manera 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 una vez ha sido instalada 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 esta.

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 cómo la extensión detecta vulnerabilidades en el código y cómo brinda una solución oportuna, permitirá que este comprenda la raíz del problema y, a su vez, le ayuda 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.

Aun 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*, más 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 del 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-Exposición 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 basada 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-Exposición 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 está sujeta 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 de este.

## 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 cómo ha sido la evolución de SAST; se pudo ver claramente cómo hay nuevos líderes incursionando y cómo 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.

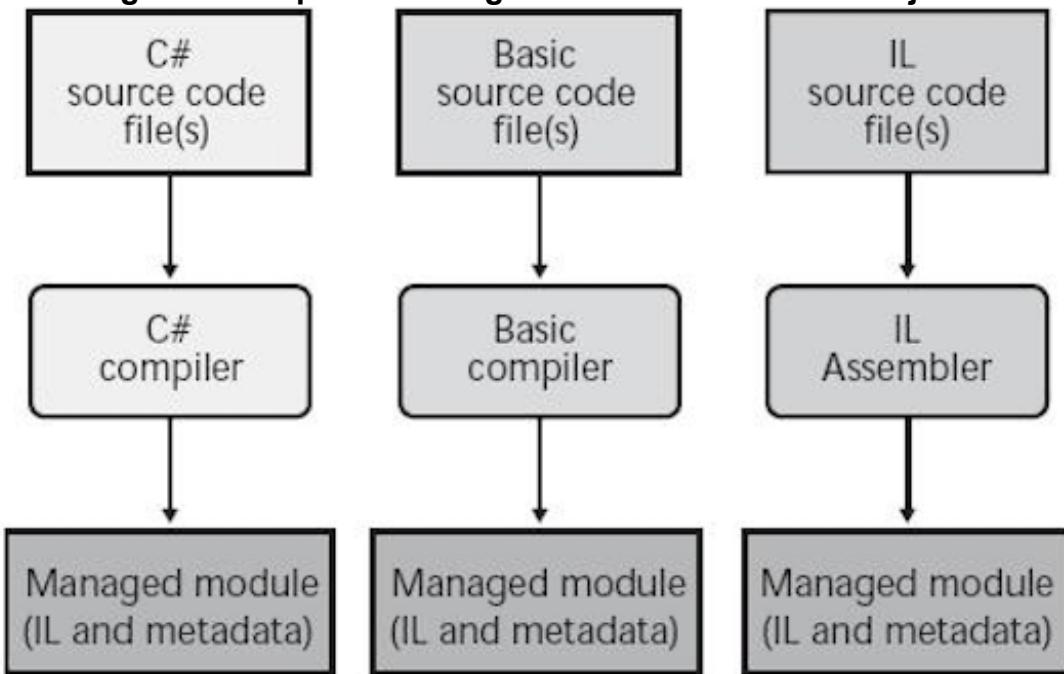
De hecho, en tiempo de ejecución el CLR no sabe cuál lenguaje de programación el desarrollador usó para escribir el código fuente. Usted puede desarrollar su código fuente en cualquier lenguaje de programación que usted desee siempre y cuando siga los lineamientos de CLR. (p. 2).

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 manejados.

**Figura 1 Compilando código fuente en módulos manejados**



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

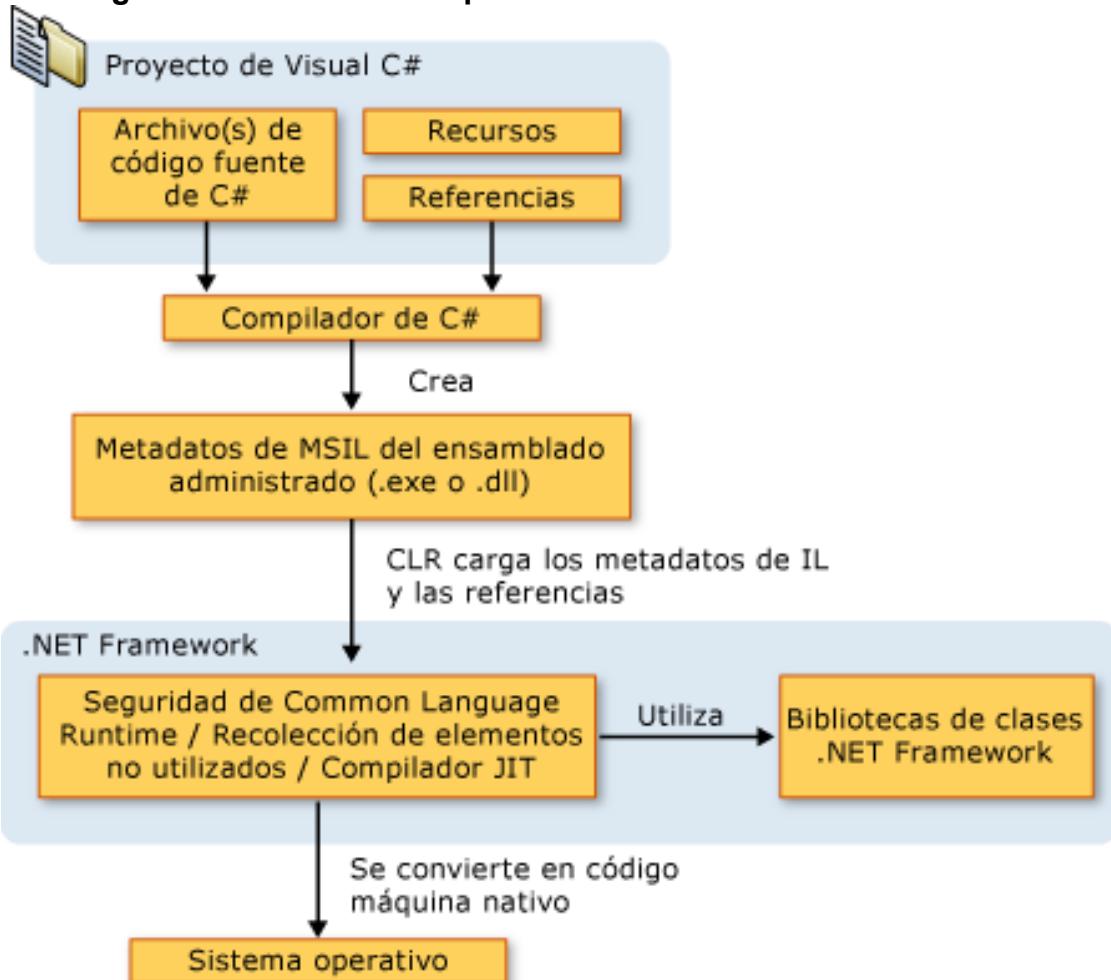
## 7.2 Compiladores como cajas negras

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 ¿Cuál es el objetivo principal de poder tener acceso al proceso de compilación y a las desconocidas actividades que esta actividad constituye? La respuesta más asertiva a esta interrogante es porque las cosas se pueden hacer mejor; no obstante, el mejor término 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:

Aun 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.

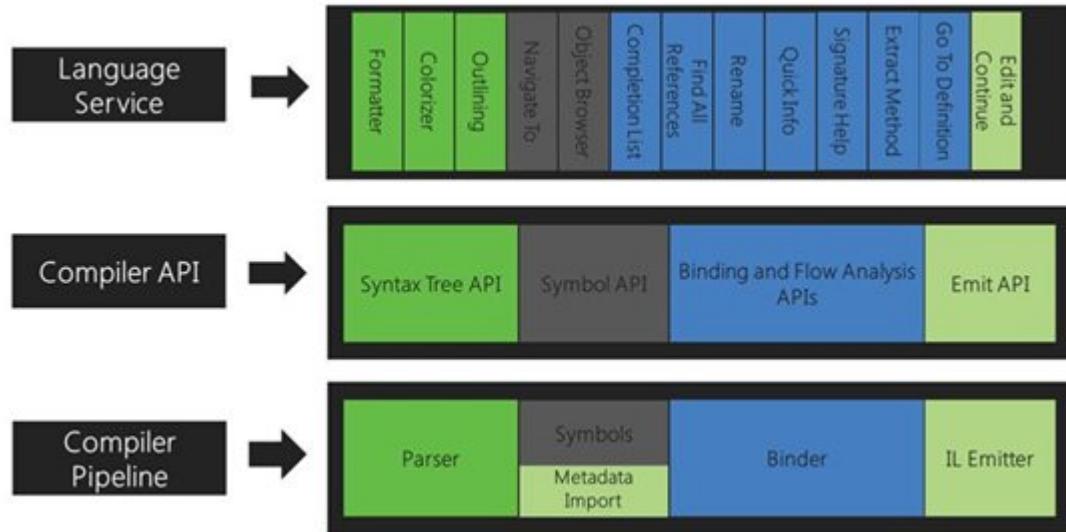
Históricamente, los compiladores manejados provistos en Visual Studio se consideran cajas opacas: usted provee el código fuente, y estos son convertidos en ensamblados.

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 basan 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 auditán aplicaciones y su ciclo de vida del desarrollo de *software* utilizando estándares internacionales.

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.**

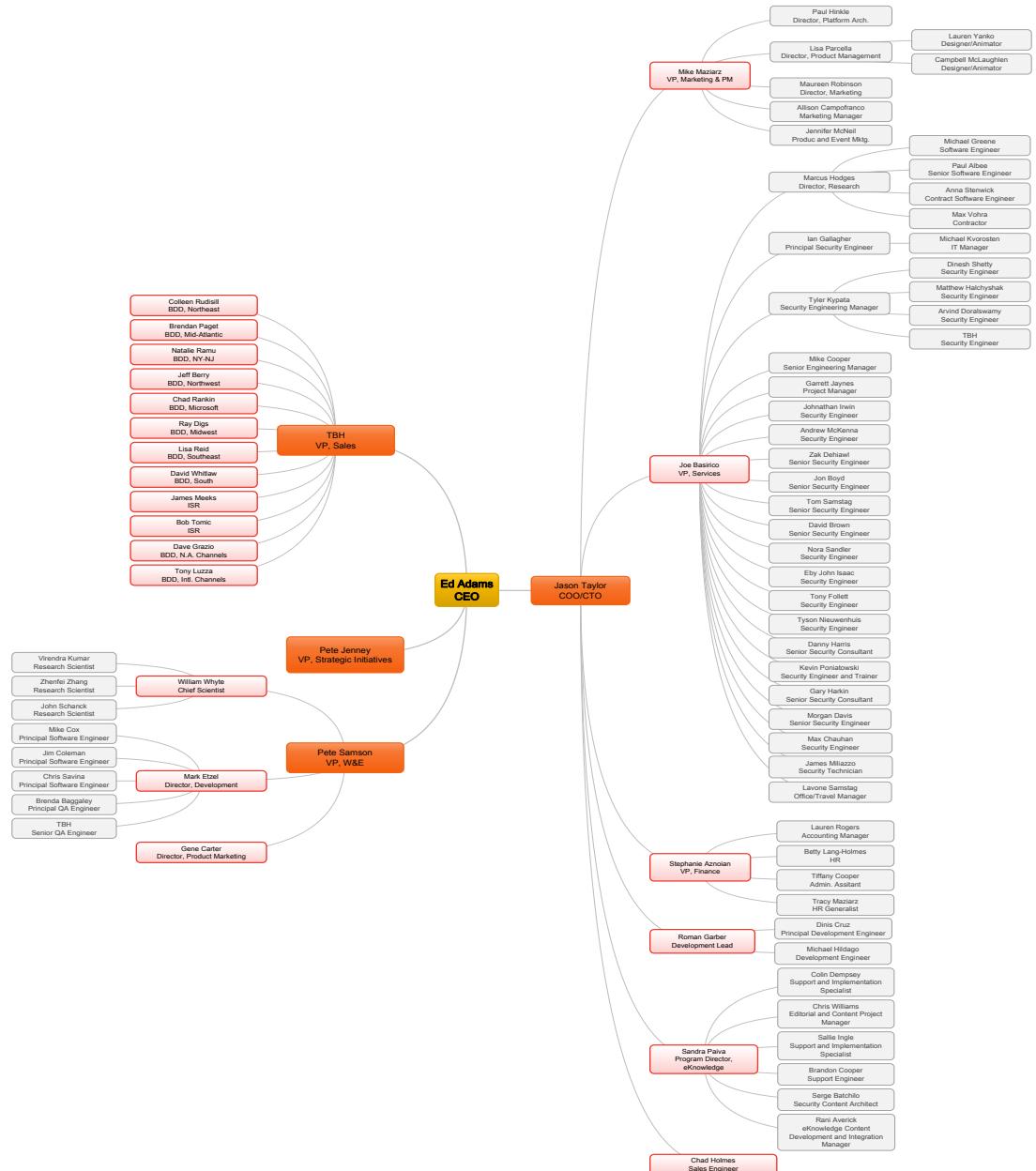


**Fuente:** <http://goo.gl/LfsVMX>

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 definida 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. Incluso 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 cómo 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 que las debilidades existen en toda organización y es necesario determinar cómo 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 manera más asertiva.

## **1.2 Análisis FODA para el prototipo funcional.**

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

**Cuadro 2 Análisis FODA**

Internas	
Fortalezas	Debilidades
<ul style="list-style-type: none"> <li>• Amplia experiencia en el mercado de la seguridad de las aplicaciones.</li> <li>• Mayor comercialización de los productos.</li> <li>• Integración con empresas en el mercado de la seguridad de aplicaciones.</li> <li>• Empresa cuenta con áreas de investigación y desarrollo donde se produce tecnología de vanguardia.</li> <li>• Facultar a empresas a desarrollar aplicaciones de <i>software</i> más seguras.</li> </ul>	<ul style="list-style-type: none"> <li>• Poca o nula inserción en el campo del análisis estático de código.</li> <li>• Dependencia de terceras empresas para realizar el análisis estático de código.</li> <li>• Proyectos de código abierto y gratuito ofrecen productos similares a muy bajo costo.</li> <li>• <i>Plugin</i> limitado a un lenguaje de programación y a un entorno integrado de desarrollo.</li> </ul>
Externas	
Oportunidades	Amenazas
<ul style="list-style-type: none"> <li>• Creciente demanda en seguridad de aplicaciones por parte de la industria.</li> <li>• Rápida evolución del lenguaje de programación C#.</li> <li>• Herramienta integrada en el ambiente de desarrollo.</li> <li>• Acercamiento de nuevos clientes potenciales.</li> </ul>	<ul style="list-style-type: none"> <li>• Competencia ofrece productos similares e integrados.</li> <li>• Clientes prefieren productos unificados.</li> <li>• Uso de tecnologías que no son suficientemente maduras.</li> </ul>

**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 las 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 facilita 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 sea reconocida aún más, de forma tal que el prototipo funcional propuesto busca formar parte de la lista de aplicaciones innovadoras ofrecidas 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 de este, 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<sup>5</sup>, 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 habitual 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.

### 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

---

<sup>5</sup><https://www.securityinnovation.com/company/news-and-events/press-releases/safelight-acquisition.html>

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 basadas 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 existencia 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 brindará 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 esta, 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 puede 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, sí 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**

The screenshot shows the Microsoft Open Technologies .NET Compiler Platform ("Roslyn") Issues page. At the top, there are navigation links for HOME, SOURCE CODE, DOCUMENTATION, DISCUSSIONS, ISSUES (which is highlighted in orange), PEOPLE, and LICENSE. Below these are links for Basic View, Advanced View, New Issue, Notifications (sign in), and Subscribe. A FILTERS section includes dropdowns for Keywords, Status (All, Open (not closed), Proposed, Active, Resolved, Closed), Reason Closed (All, Unassigned, Fixed, ByDesign, NotReproducible, NotThisProject), Type (All, Unassigned, Feature, Issue, Task), Impact (All, Unassigned, Low, Medium, High), Release (All, Unassigned), Assigned To (All, Unassigned, acasey, ADGreen, ahejlsberg, AlekseyTs), and Component (All, Unassigned). The RESULTS section shows 201-220 of 220 items, with pagination links for Previous, 1, 2, 3, Next, and a search bar. The bottom right shows options for Showing 10, 25, 50, 100 items.

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.**

Microsoft  
Open Technologies

[.NET Compiler Platform \("Roslyn"\)](#)

The screenshot shows the project page for ".NET Compiler Platform ("Roslyn")" on the Microsoft Open Technologies website. The navigation bar at the top includes links for HOME, SOURCE CODE, DOCUMENTATION, DISCUSSIONS, ISSUES, PEOPLE, and LICENSE. The LICENSE link is highlighted with a purple background. Below the navigation bar, the title "Apache License 2.0 (Apache)" is displayed in purple. To the right of the title, there are two status indicators: "CURRENT (STARTED MAR 18, 2014)" and "Displayed". The main content area contains the text of the Apache License 2.0, starting with "TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION". It includes a section titled "1. Definitions." and a note about the definition of "License".

Apache License 2.0 (Apache)

Apache License  
Version 2.0, January 2004  
<http://www.apache.org/licenses/>

CURRENT (STARTED MAR 18, 2014)      Displayed

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

**Fuente:** <https://roslyn.codeplex.com/license>

**CAPÍTULO II****MARCO TEÓRICO**

En el capítulo presente, se consigna 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 personas 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 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:

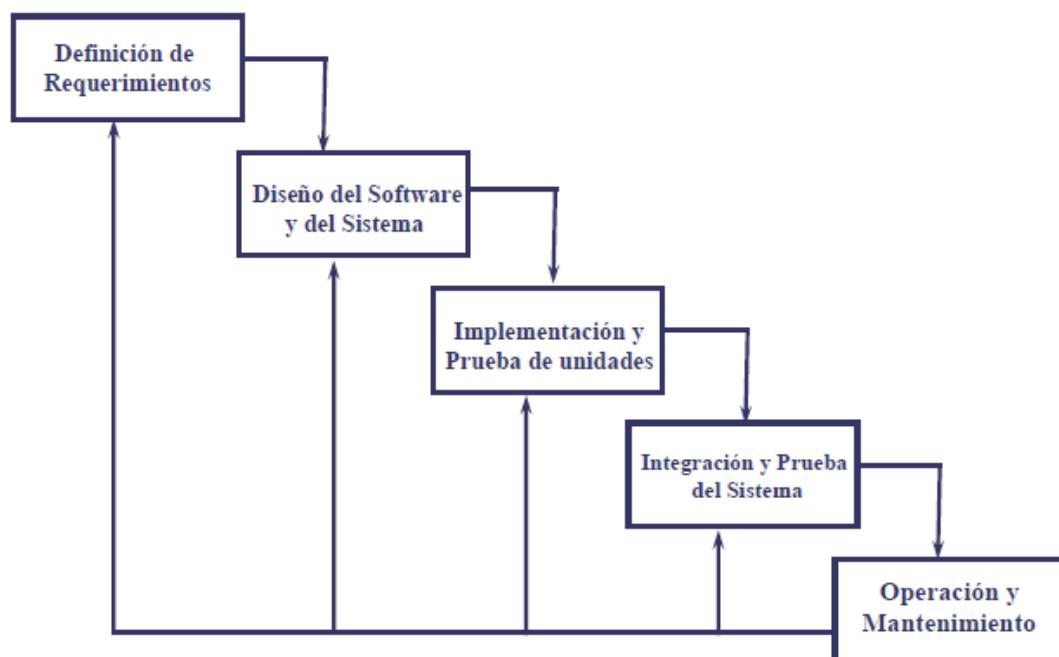
- a) Análisis y definición de requerimientos: Tomando en consideración a los usuarios del sistema, se definen las restricciones y los objetivos del sistema.

- 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).

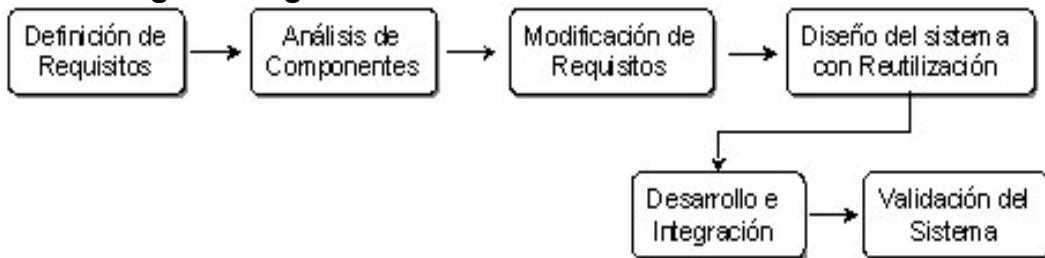
### 2.2.3 Ingeniería de *software* orientada a reutilización

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, facultando 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 al 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**

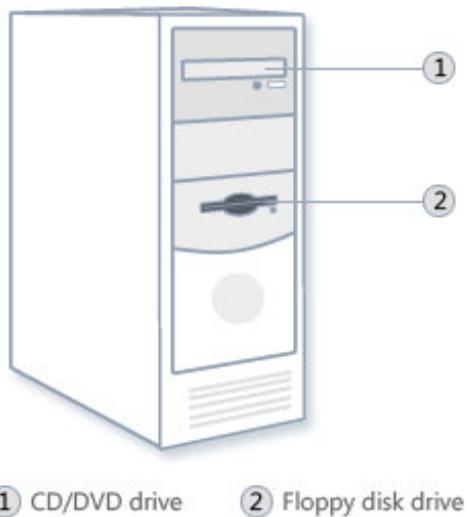
Se entiende por *hardware* todos los elementos tecnológicos que pueden ser percibidos por medio de los sentidos, tal es el caso de los servidores físicos, computadoras portátiles, impresoras, teclados, monitores dispositivos de red entre otros. El *hardware* tiene un rol fundamental en la tecnología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

Microsoft (2014)<sup>6</sup> define *hardware* al indicar que "Las partes físicas, las cuales usted puede ver y tocar, se denominan de forma colectiva como *hardware*."

---

<sup>6</sup> <http://windows.microsoft.com/en-us/windows/computer-parts#1TC=windows->

**Figura 11 Ejemplo de *Hardware***

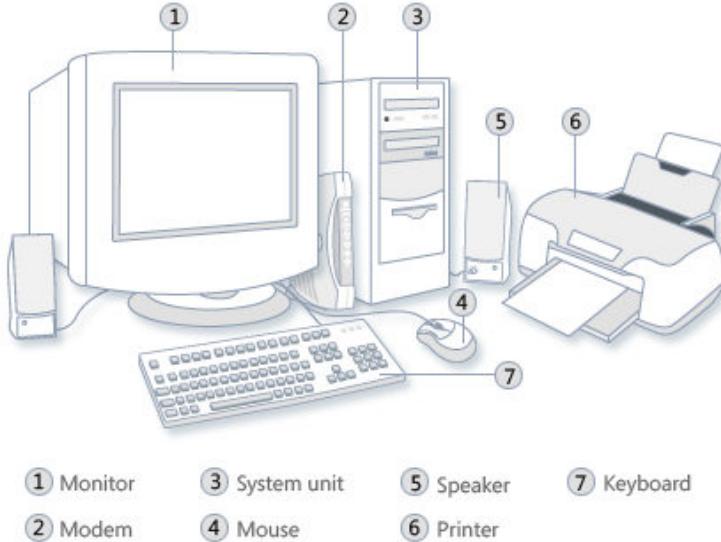


**Fuente:** Microsoft <http://goo.gl/8BrfXK>

## 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 señala 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 apoyar a individuos, grupos, y organizaciones. Las 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 de este o el deterioro causan una interrupción de las operaciones e impiden 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 re hace referencia 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 agentes 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 cómo 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 ciberneticas 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

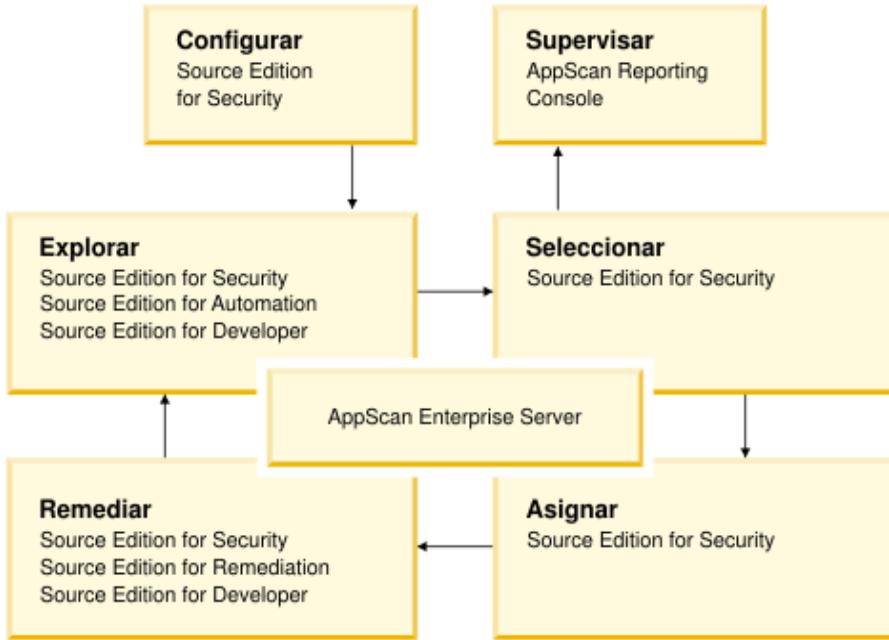
Es una herramienta creada por la empresa IBM, según el sitio oficial<sup>7</sup> del producto lo definen como “IBM Security AppScan mejora la seguridad de las aplicaciones Web y aplicaciones móviles, mejora los programas de seguridad y fortalece los cumplimientos de las normativas”.

---

<sup>7</sup> <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

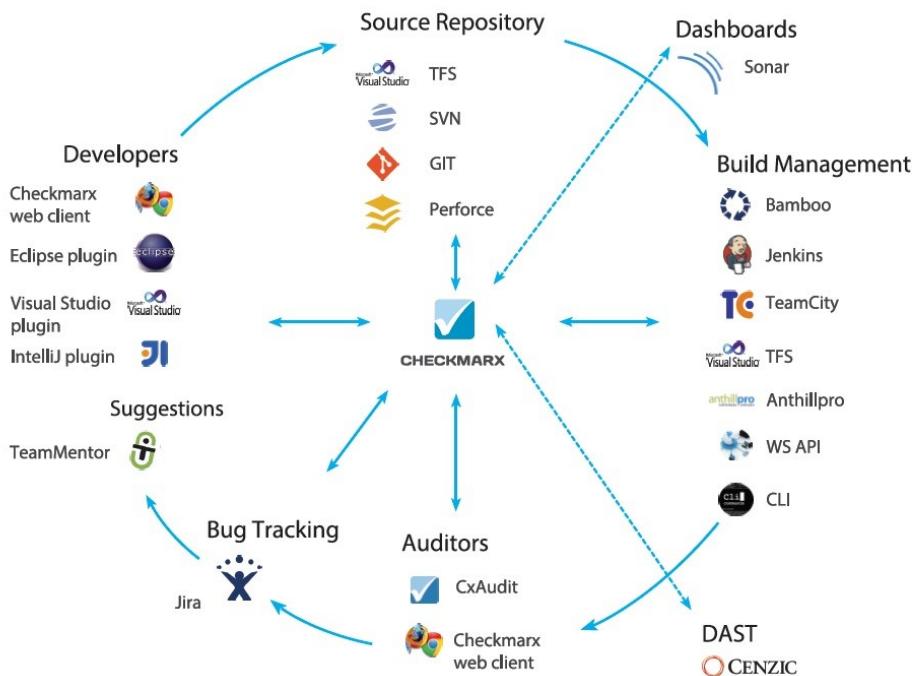
Es una empresa basada en Tel Aviv, Israel, la cual se dedica a proveer herramientas para realizar análisis de código fuente a fin de encontrar vulnerabilidades y que sean corrijan en etapas tempranas. En términos de análisis estático de código, Checkmarx ofrece el producto denominado Checkmarx Suite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

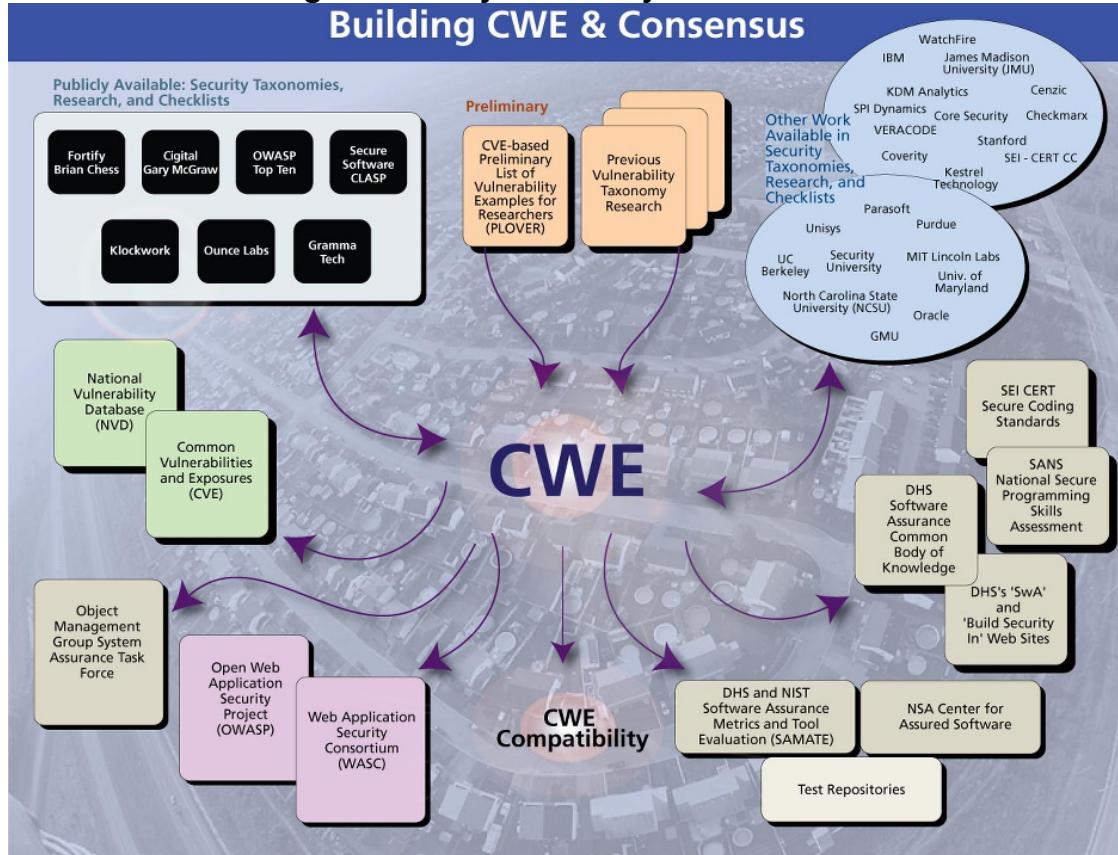
Tal como se indica en el sitio oficial de CWE (2014):

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 las 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  
Building CWE & Consensus**



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

## 2.31 TEAM Mentor

TEAM Mentor es una herramienta que funciona bajo el enfoque de ser una guía especializada en mejores prácticas para el desarrollo de aplicaciones de *software* resistentes a ataques informáticos. 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 sinfín 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). Con base 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).

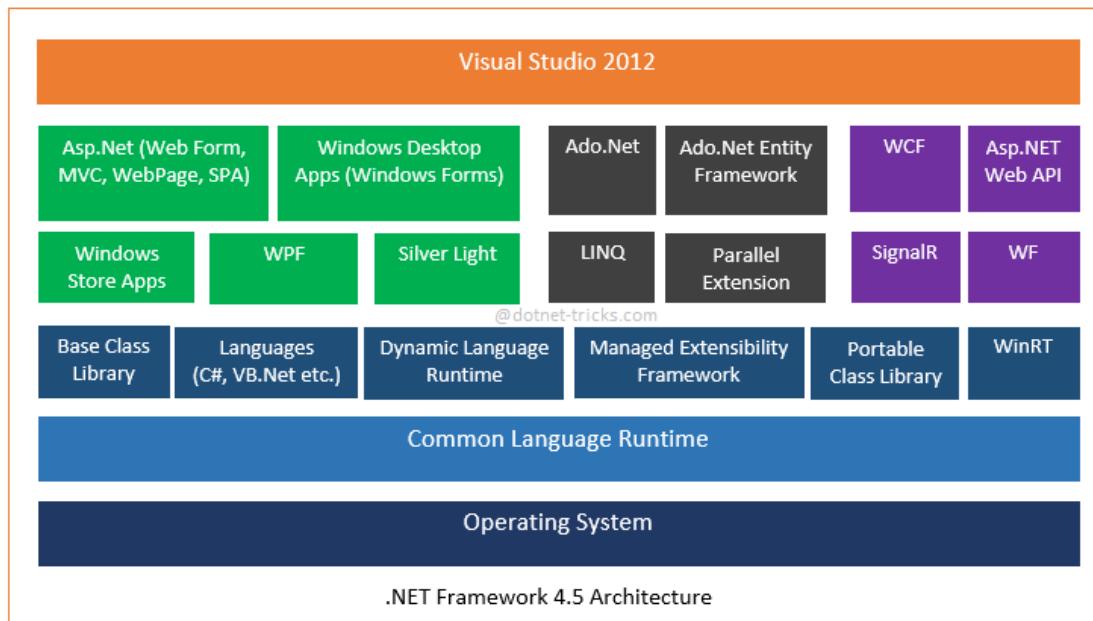
Según los mismos autores, la plataforma .NET consta de cuatro grupos de productos separados entre los que se destacan:

- 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:

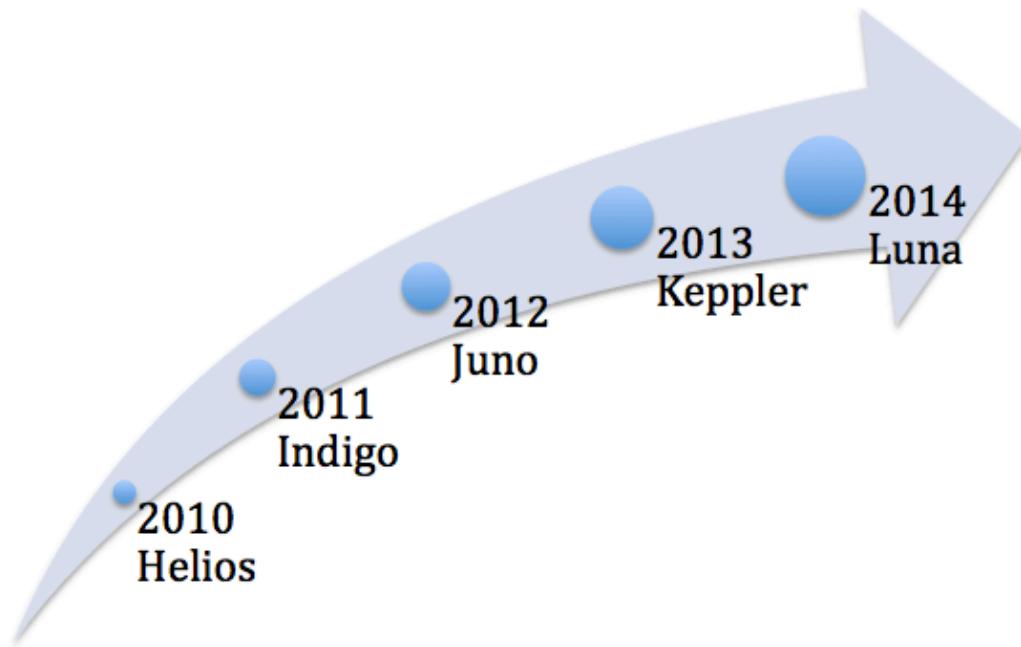
C# es un lenguaje de programación simple, moderno, orientado a objetos y de tipos de datos seguro. C# tiene sus raíces en la familia de lenguajes C y es inmediatamente familiar a desarrolladores de C, C++ y Java. C# está estandarizado por ECMA internacional por medio del estándar ECMA-334 y por ISO/IEC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

## 2.38 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).

### **2.39 Pérdida de Autenticación y Gestión de Sesiones**

La pérdida de autenticación y gestiones incorrectas de sesiones son un tipo de vulnerabilidad muy frecuente en sitios transaccionales comunes. En el marco de las configuraciones incorrectas y de la falta de controles para proteger datos sensibles para la organización, los usuarios maliciosos tratan de explotar tales problemas.

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).

## 2.40 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 (Lenguaje de marcas de hipertexto). XSS es la más prevalente 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 phishing, 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 víctima es un potencial objetivo para este ataque.

Tal como se ha definido, cuando una vulnerabilidad de este tipo se materializa, el impacto para el negocio es muy grande ya que puede permitir que exista una modificación del sitio web de la organización, redireccionamiento del usuario a un sitio potencialmente dañino y la instalación de *software* no deseado, entre muchos otros efectos adversos.

## 2.41 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.

En el CWE (Common Weakness Enumeration) número 20 que reza bajo el título de CWE-200 Exposición de Información<sup>8</sup>, se establece que:

La exposición de información la revelación de información de forma intencionada o mal intencionada de un actor que no está explícitamente autorizado a tener acceso a dicha información.

Tales datos pueden ser filtrados diversos lugares entre los que figuran los mensajes de error que emiten las aplicaciones donde se demuestra de forma detallada, información técnica rica en contenido para un atacante malicioso, el cual puede determinar la tecnología adyacente en la cual se ha construido el sistema así como las respectivas versiones. De igual forma las bitácoras de los sitios web pueden eventualmente almacenar información confidencial tal es el caso de actividades, usuarios, contraseñas, llaves criptográficas entre otro tipo de información que en el momento de ser expuesta o accedida por un usuario no autorizado pone en riesgo a la organización.

---

<sup>8</sup> <http://cwe.mitre.org/data/definitions/200.html>

## 2.42 Configuración Incorrecta de Seguridad

La configuración incorrecta de seguridad corresponde a un tipo de vulnerabilidad donde las aplicaciones web han sido publicadas en un ambiente de producción (es decir donde todas las personas las pueden acceder); pero donde existen configuraciones que revelan información confidencial.

Un ejemplo de esta vulnerabilidad ocurre cuando la aplicación web guarda en un archivo plano datos como contraseñas o claves criptográficas pero tal directorio no ha sido protegido contra navegación. En principio, un usuario malicioso encuentra el directorio donde estos archivos han sido almacenados y posteriormente tiene acceso a la información que ha sido almacenada en ellos.

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 seguridad *per se*, se aplica la frase que indica que una cadena se rompe por el eslabón más débil. Una configuración incorrecta de las aplicaciones es un reflejo del eslabón débil en la cadena de la seguridad.

## 2.43 JavaScript

JavaScript es un lenguaje de programación que cuenta con soporte nativo por parte de los navegadores y que permite escribir aplicaciones que son interpretadas por los navegadores las cuales se referencian comúnmente como aplicaciones del lado del cliente.

Reid & Valentine (2013) establecen que:

JavaScript es un lenguaje de programación que fue introducido en 1995. A pesar de su nombre no tiene ninguna relación con el lenguaje de programación conocido como Java. (p. 5).

Luego ambos autores también indican que

Hasta el momento la implementación más común de JavaScript está en los navegadores. Un motor de JavaScript en el navegador típicamente implementa la mayoría de las características especificadas en el estándar ECMA-263. (p. 5)

## 2.44 HTTP

HTTP es el acrónimo de Protocolo de Transferencia de Hipertexto por sus siglas en inglés (Hypertext Transfer Protocol), uno de los protocolos más utilizados a nivel mundial.

Gourley & Totty indican que:

El Protocolo de Transferencia de Hipertexto es el programa utilizado para permitir la comunicación en World Wide Web. Hay muchas aplicaciones para HTTP, pero HTTP es famoso para realizar una comunicación de dos vías entre un navegador y un servidor.

Debido a que HTTP hace uso de protocolos confiables para la transmisión de datos, garantiza que sus datos no serán dañados o mezclados en tránsito, aun cuando éstos provengan desde el otro lado del globo. (p .3)

## 2.44 Cookies

Las *cookies* o conocidas también como HTTP Cookies, son archivos en texto plano que son almacenadas por un navegador en la computadora de un cliente y son usadas por los servidores a fin de identificar a un cliente. Microsoft (2014)<sup>9</sup> indica que:

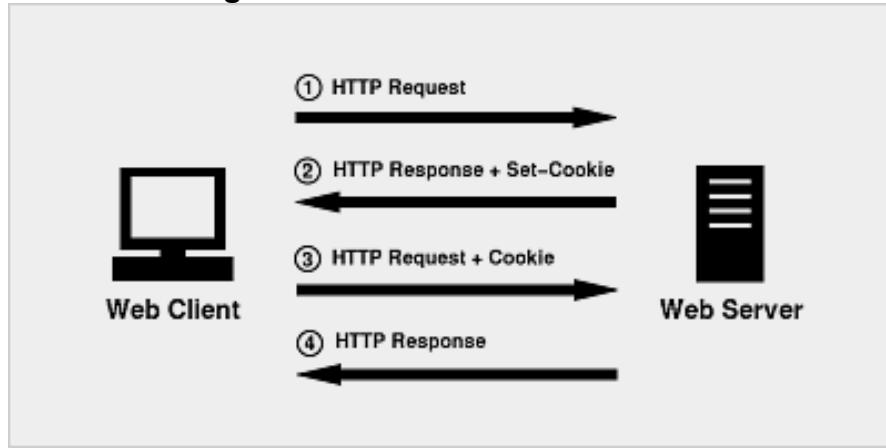
Las cookies HTTP le proveen al servidor un mecanismo para almacenar y recuperar información del estado en el sistema de un cliente. Este mecanismo le permite a las aplicaciones basadas en Web la habilidad para almacenar información acerca de elementos seleccionados, preferencias de usuario, información de registro, y cualquier otro tipo de información que pueda ser recuperada más tarde.

En la imagen que se presenta a continuación, se muestra la forma en que las *cookies* son transmitidas entre un cliente y un servidor.

---

<sup>9</sup> [http://msdn.microsoft.com/en-us/library/windows/desktop/aa384321\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/aa384321(v=vs.85).aspx)

**Figura 18 Transmisión de cookies**



Fuente: <http://goo.gl/m11LA>

## 2.45 Cabeceras HTTP

Las cabeceras HTTP son un mecanismo por medio del cual un cliente y un servidor pueden hacer envío de información adicional, misma que es necesaria durante la negociación de contenido. De esta forma, un cliente por medio de una cabecera HTTP puede indicar el formato en que el servidor debe retornar el recurso solicitado por ejemplo.

Gourley & Totty indican:

Las cabeceras y los métodos trabajan en conjunto para determinar lo que un cliente y un servidor hacen. Hay cabeceras que son específicas para cada tipo de mensaje y cabeceras que son de propósito general, proporcionando información en mensajes de solicitud y de respuesta. (p. 67).

## 2.44 Refactorización

La refactorización, del inglés *refactoring*, es actividad relevante en toda aplicación de *software*, ya que busca oportunidades de mejora.

El autor Fowler (1999) se refiera al proceso de refactorización de la siguiente forma:

Refactorización es el proceso de cambiar un sistema de software de forma tal que no afecte el comportamiento externo del código, mas sin embargo mejora su estructura interna. Es una forma disciplinada de limpiar el código donde se minimizan las oportunidades de introducir defectos. En esencia cuando usted refactoriza usted está mejorando el diseño del código luego de que se ha escrito. Por medio de la refactorización, usted puede tomar un mal diseño, caótico incluso, y convertirlo en código bien diseñado. (p. 5).

## 2.45 HTTP GET

HTTP Get es un método utilizado por el protocolo HTTP y que fue diseñado para traer un recurso y mostrarlo en un navegador. Gourley & Totty (2002) indican que:

GET es uno de los métodos más comunes. Usualmente se usa para pedir a un servidor que envíe un recurso. La especificación HTTP/1.1 requiere que los servidores implementen este método. (p. 53).

## 2.46 HTTP POST

En términos del protocolo HTTP, POST es un método que se usa con el objetivo de crear un recurso nuevo en el servidor. Los parámetros enviados a

una aplicación por medio de este método son enviados como parte del cuerpo del mensaje. Gourley & Totty (2002) indican que:

El método POST fue diseñado para el envío de datos a un servidor. En la práctica se usa para soportar formularios HTML. Los datos de un formulario completo típicamente es enviado al servidor, el cual los redirecciona y envía hacia el destino. (p. 54).

## **2.47 XML**

XML es el acrónimo de Lenguaje Extensible de Marcas, por sus siglas en inglés (Extensible Markup Language), y es un formato ampliamente utilizado para el intercambio de datos. Flanders (2009) señala que:

XML es probablemente el formato más popular para la representación de los recursos. Es un formato bien conocido, y hay librerías para procesar XML en cada plataforma. El tipo de medio formal para XML es application/xml. (p. 9).

## **2.48 Wi-Fi**

Duntemann's (2004) define Wi-Fi de la siguiente forma:

El término Wi-Fi es una palabra informal para definir a toda una especificación de familia de redes inalámbricas con nombres prohibidos como lo son 802.11a, 802.11b, 802.11g. Usualmente se le conoce como “Ethernet inalámbrico” o Wi-Fi. Dentro de la lista de usuarios entusiastas de las redes Wi-Fi figuran las personas que viajan por asuntos de negocios, quienes toman ventaja de la proliferación de redes inalámbricas en cafeterías, hoteles, centros de conferencias e incluso parques públicos con el objetivo de conectar sus computadoras a Internet mientras se movilizan. (p. 5).

**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 y 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 manera, 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 tener mayor 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

Tal como su nombre parece indicar, el método científico es un proceso mediante el cual se hace uso de la ciencia a través de la observación con el objetivo de llegar a resultados concluyentes. El autor indio Pathak (2011) cuando se refiere al método científico señala que:

El proceso del uso científico para la recolección de datos y la verificación de conocimiento científico se le denomina método científico. Este concepto es usado frecuentemente en estudios investigativos que son empíricos en el sentido de que lidian con evidencia observable – evidencia que es la experiencia común de investigadores entrenados. Un dominio crítico de la investigación científica moderna es la construcción de teorías que organizan los hechos disponibles de tal forma que cede una explicación general acerca de un rango amplio de fenómenos (p. 200).

Según se indica en un artículo publicado por la Universidad de Murcia en España y que reza bajo el título de *Ciencia y método científico* (2014), el método científico se compone de las siguientes etapas:

- a) Plantear un problema.
- b) Observar algo.
- c) Buscar una teoría que lo explique.
- d) Hacer predicciones en base a esa teoría.
- e) Comprobar esas predicciones haciendo experimentos u observaciones.
- f) Si los resultados están de acuerdo con la teoría, volver al cuarto paso, si no volver al tercero.

### 3.1.2 Método Inductivo

El autor Sánchez (2012) se refiere al concepto de método inductivo cuando afirma que:

Consiste en basarse en enunciados singulares, tales como descripciones de los resultados de observaciones o experimentos para plantear enunciados universales, tales como la hipótesis o teorías. Ello es como decir que la naturaleza se comporta siempre igual cuando se dan las mismas circunstancias, lo cual es como admitir que bajo las mismas condiciones experimentales se obtienen los mismos resultados, base de la repetitividad de las experiencias, lógicamente aceptadas. Por otra parte, la inducción equivale a la extrapolación, lo cual puede ser un recurso en el campo experimental, que no se confirma siempre. Por todo ello, mediante solo la inducción, o sea, una colección de datos experimentales para construir leyes y teorías científicas auxiliadas por la lógica, es difícil elaborar una teoría científicamente admisible. (p. 32)

### **3.1.3 Método Deductivo**

El método de investigación deductivo, en principio, es el que usa el ser humano de forma recurrente ante la toma de decisiones y cuando se afronta a la resolución de un problema específico. Sánchez (2012) brinda una explicación clara en naturaleza acerca de este método al sostener que:

El método hipotético-deductivo se emplea corrientemente tanto en la vida ordinaria como en la investigación científica. Es el camino lógico para buscar la solución a los problemas que se plantean. Consiste en emitir hipótesis acerca de las posibles soluciones al problema trazado y en comprobar con los datos disponibles si estos están de acuerdo con aquellas. Cuando el problema está próximo al nivel observacional, en el caso más simple, las hipótesis se pueden clasificar como empíricas, mientras que en los casos más complejos, las hipótesis son de tipo abstracto. (p. 13).

### 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).

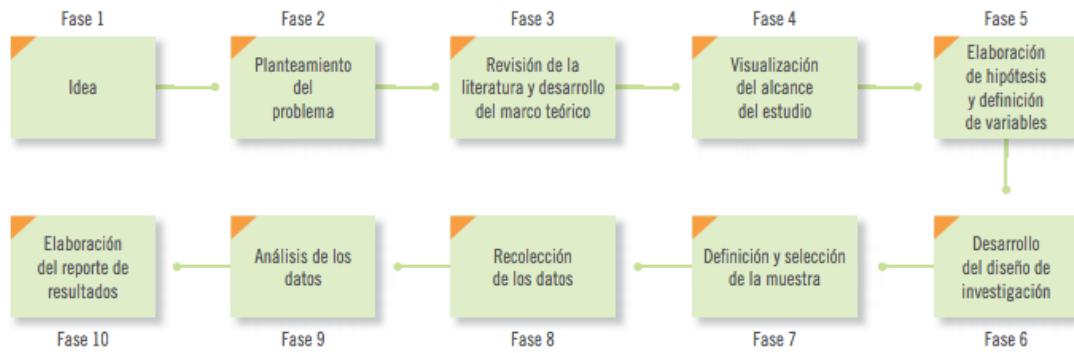
Como complemento, Kendall & Kendall (2011) también indican que “Hay muchos documentos cuantitativos disponibles para su interpretación en cualquier empresa” (p. 136). Precisamente, los autores Kendall & Kendall sostienen que dentro de tales documentos se encuentran:

- 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 extraída del libro Metodología de la Investigación del autor Sampieri e ilustra el proceso de la investigación cuantitativa.

**Figura 19 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 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).

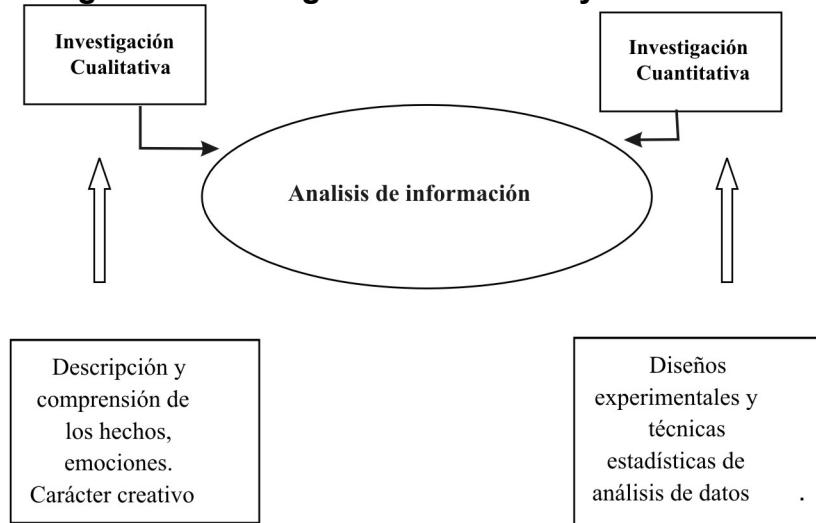
Incluso Sánchez (2008) explica cuáles son los métodos cualitativos al inferir que “Los cuestionarios, la observación descriptiva, las entrevistas y otros métodos cualitativos son muy antiguos.” (p. 7).

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, señala claramente algunas de las diferencias sustanciales entre ambos enfoques:

**Figura 20 Investigación cualitativa y cuantitativa**



Fuente: [http://bvs.sld.cu/revistas/spu/vol33\\_3\\_07/spu20207.htm](http://bvs.sld.cu/revistas/spu/vol33_3_07/spu20207.htm)

### 3.1.4 Método de Investigación Utilizado.

Con base en las definiciones anteriores y con apoyo 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 este 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 estas. (p. 80).

### 3.2.2 Exploratoria

La Universidad Nacional Abierta y a Distancia de Colombia (UNAD)<sup>10</sup>, 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 estudios exploratorios se realizan cuando el objetivo es examinar un tema o problema de investigación poco estudiado, del cual se tiene muchas 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

---

<sup>10</sup> [http://dataoteca.unad.edu.co/contenidos/100104/100104\\_EXE/leccin\\_6\\_investigacin\\_exploratoria\\_descriptiva\\_correlacional\\_y\\_explicativa.html](http://dataoteca.unad.edu.co/contenidos/100104/100104_EXE/leccin_6_investigacin_exploratoria_descriptiva_correlacional_y_explicativa.html)

relacionadas con el problema de estudio, o bien, si se desea 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).

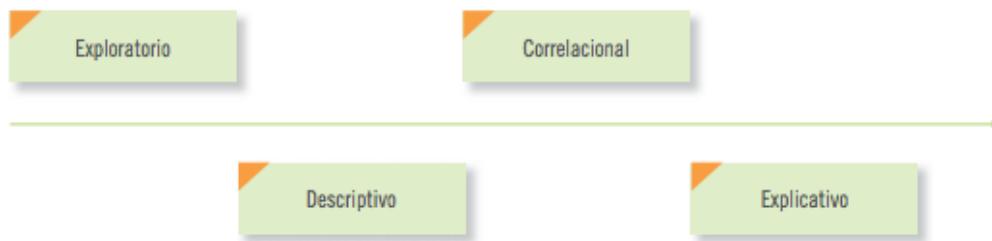
### **3.2.4 Explicativo**

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 21 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, además 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<sup>11</sup>, el cual a su vez es el cimiento para el prototipo funcional propuesto, hace referencia a las distintas áreas de innovación y a su relevancia en la computación moderna al indicar que:

---

<sup>11</sup> <https://roslyn.codeplex.com/wikipage?title=Overview&referringTitle=Home>

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 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 plataforma educativa denominada TEAM Professor de la empresa Security Innovation contiene cursos virtuales, asistidos por computadora, enfocados en mejores prácticas de seguridad y en ayudar al desarrollador a identificar código fuente vulnerable a ataques informáticos. Dicha base de datos de conocimiento será usada como material de apoyo en el momento de determinar patrones de código fuente vulnerable y en el momento de brindar una recomendación al problema propuesto. 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.

### **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 qué actividades u operaciones deben realizarse para medir una variable.”

### **3.4.3 Definición Instrumental**

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

Aquí se aclara cómo 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.

### 3.5 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	Diseño de un prototipo funcional	Definición de los componentes de la extensión y su arquitectura, diagramas de	Basado en los requerimientos se procede a la elaboración de diagramas de arquitectura, componentes,	Diagramas de casos de uso, de componentes, de clases, diagrama de arquitectura

vulnerabilidades y la retroalimentación al usuario final		clase, objetos, casos de uso.	UML, casos de uso	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 vulnerabilidades en el código fuente utilizando estándares en la industria.	Desarrollo de las reglas de diagnóstico en el código fuente.	Identificación de los patrones de código fuente vulnerables y de las soluciones a tales problemas.	Utilizando guías, metodologías, mejores prácticas y estándares en la industria del desarrollo del software se implementarán las reglas de diagnóstico.	Guía OWASP Top 10. TEAM Mentor base de datos 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	Plataforma para realizar pruebas unitarias NUNIT. Plantillas para escenarios de pruebas.

			problema de seguridad ha sido resuelto.	
--	--	--	---	--

**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:

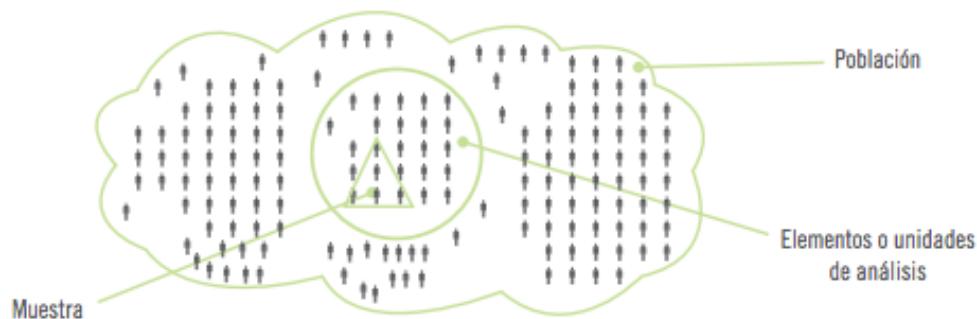
Al recolectar datos que determinan las características de un grupo de individuos u objetos, por ejemplo, las alturas y los pesos de los estudiantes de una universidad o la cantidad de piezas defectuosas y no defectuosas producidas en una fábrica en un día determinado, muchas veces es imposible o impráctico 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 22 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 y 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.  $q = (1-p) 0.95$

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

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

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

$$\mathbf{n = 40.88}$$

Con base en el cálculo anterior se puede determinar el valor de la muestra de personas a entrevistar corresponde a 40.88 o 41.

No obstante, por decisión del director del proyecto, se ha establecido que solo el punto de vista de dos personas, ambos con vasta experiencia en el modelo de negocios de la empresa y en la industria de la seguridad de las aplicaciones, se considerará vinculante para la recopilación de información y la colaboración en el levantamiento de requerimientos. Esas dos 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 considerada como vital ya que permite 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 mencionan 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. El objetivo fundamental de la encuesta es recopilar información acerca de un componente de análisis estático de código en el marco del desarrollo seguro de *software*. Aun cuando por decisión del administrador del proyecto se ha establecido que solamente se aplicará la encuesta a dos de los miembros del equipo de desarrollo, específicamente a Roman Garber, basado en la ciudad de Nueva York, Estados Unidos y a Dinis Cruz ubicado en Londres, Reino Unido.

Se pretende contar con el criterio experto de estas dos personas y tener una mejor visión en el momento de desarrollar el componente.

Además de la encuesta se hará uso del método de observación como instrumento de análisis, dicha observación se centra en ver el comportamiento y tendencia de ataques informáticos actuales, los cuales han tenido repercusión financiera para múltiples entidades a nivel mundial. Así mismo mediante la utilización de datos fiables (provenientes de fuentes diversas encargadas de realizar investigaciones), se pretende identificar oportunidades potenciales y tomar como base ese comportamiento para asegurar que la

extensión de seguridad a ser desarrollada contempla los problemas de seguridad más comunes.

### **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.

**Cuadro 3 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 observación.	- Fuentes primarias: Cuestionario realizado a los miembros del equipo de trabajo. Libros técnicos enfocados en análisis estático de código fuente y meta programación. Observación de reportes, estudios e investigaciones recientes en temas de seguridad.
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

Objetivo	Definición instrumental	Fuente de información
		<i>software.</i>
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 diagnóstico 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 <i>software</i> .

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.

**Cuadro 4 Relación entre objetos, entregables y herramientas**

Objetivo	Entregables	Instrumentos 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 <i>software</i> 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 estáticas de seguridad de aplicaciones.	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.
Implementar las reglas de diagnóstico para detectar vulnerabilidades	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.

<b>Objetivo</b>	<b>Entregables</b>	<b>Instrumentos y herramientas</b>
en el código fuente utilizando estándares en la industria.		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

### DISEÑO

## 1. DISEÑO

En esta sección se abarcan los elementos fundamentales del diseño del prototipo funcional. Se brindan metodologías y herramientas con el objetivo de modelar el problema a resolver.

Steve Jobs, una de las personas más influyentes en la computación, solía decir que “El diseño es una palabra divertida. Algunas personas piensan que el diseño significa cómo se ve. Pero, por supuesto, cuando uno profundiza, es realmente cómo funciona.”<sup>12</sup>

### 1.1 Descripción de la arquitectura física y lógica

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 detallarán los principales componentes de arquitectura del prototipo propuesto. Se utilizarán diversos diagramas con el objetivo principal de explicar mejor el prototipo a ser implementado.

---

<sup>12</sup> [http://archive.wired.com/wired/archive/4.02/jobs\\_pr.html](http://archive.wired.com/wired/archive/4.02/jobs_pr.html)

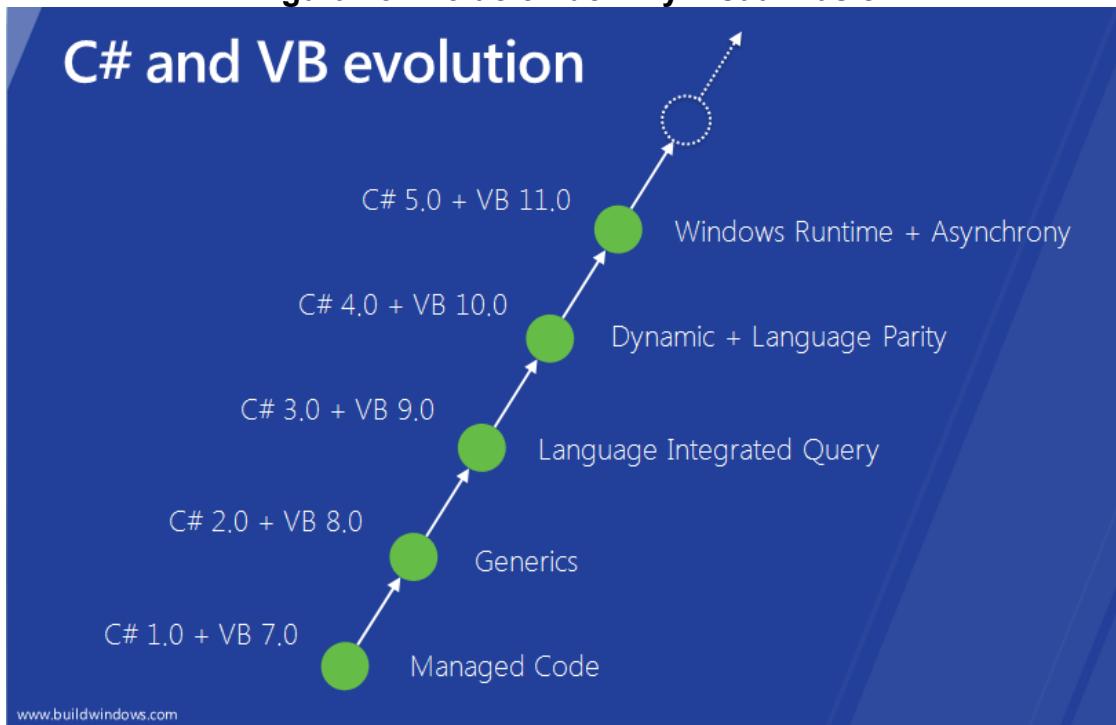
Según Sommerville (2011):

El diseño arquitectónico se interesa por entender cómo 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).

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

**Figura 23 Evolución de C# y Visual Basic**



Fuente: <http://goo.gl/GSYjP>

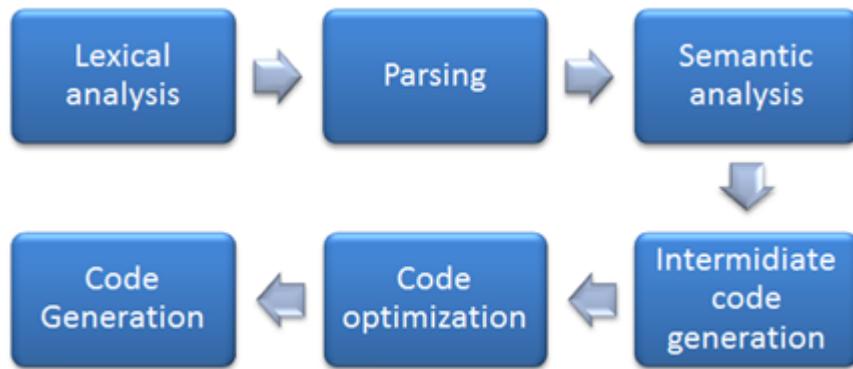
Es claro percibir que ambos lenguajes han incluido características fundamentales que permiten que el desarrollo de código sea un proceso más intuitivo, permitiendo que la forma de escribir código sea similar al pensamiento humano, con sintaxis y semántica más similar al lenguaje cotidiano. Tal como se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

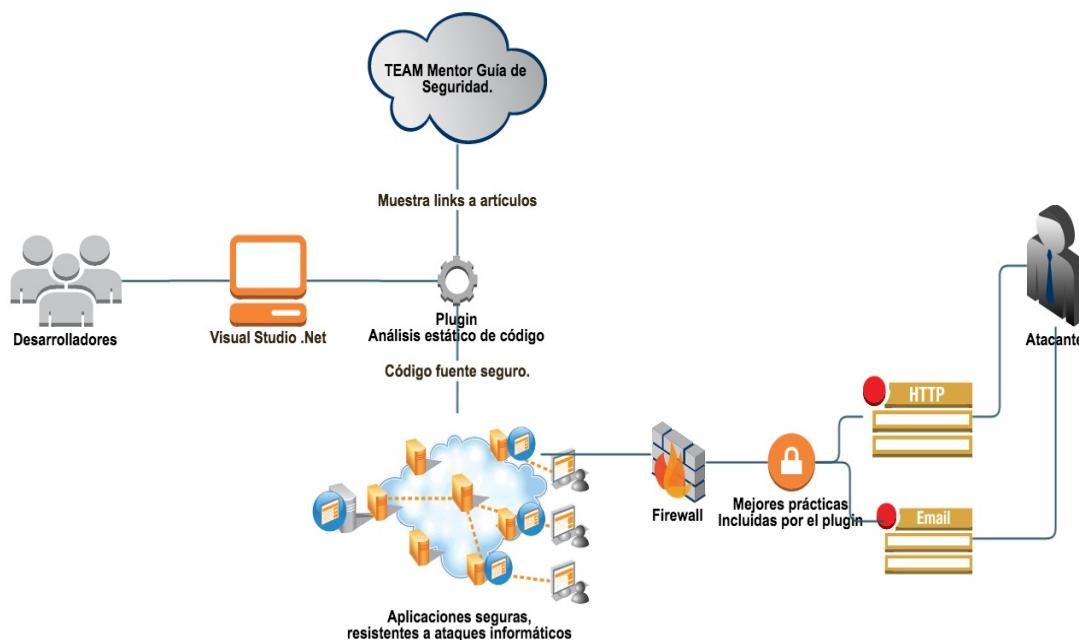
En la imagen siguiente se muestran los pasos del proceso de compilación descritos anteriormente.

**Figura 24 Pasos del proceso de compilación.**



Fuente: <http://goo.gl/PaOoD>

**Figura 25 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

este 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 (puede ser alertas, advertencias o errores), este se muestra 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.

## 1.2 Análisis de Requerimientos

El proceso de análisis de requerimientos juega un rol fundamental en el ciclo de vida del desarrollo de *software*. Una mala o incorrecta incepción e identificación de los problemas que el *software* en cuestión debe resolver guiará de forma inevitable al fracaso del mismo. Aun así cuando el *software* no presenta errores funcionales ni operacionales pero no hace lo que el usuario final realmente necesita, debido un pobre proceso de levantamiento de requerimientos, entonces habrá un desperdicio de recursos puesto que el sistema a fin no es funcional.

Durante esta etapa de incepción resulta indispensable contar con el apoyo de los interesados y de usuarios expertos en el negocio, los cuales brinden las pautas y las recomendaciones que el *software* debe incluir con el objetivo de resolver el problema presentado.

Esposito & Saltarello (2009) refiriéndose a los requerimientos del *software* aseveran que:

En términos un poco abstractos, un requerimiento es una característica del sistema que puede ser funcional o no funcional. Un requerimiento funcional se refiere al comportamiento que el sistema debe suplir a fin de solventar un escenario dado. Un requerimiento no funcional se refiere a los atributos del sistema explícitamente solicitado por los interesados. (p. 12).

#### **4.2.4 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 que parece indicar que los atacantes utilizan la misma aplicación, luego de explotar alguna vulnerabilidad en esta, 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, las cuales 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.

#### **4.2.4.1 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 se están provocando.

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

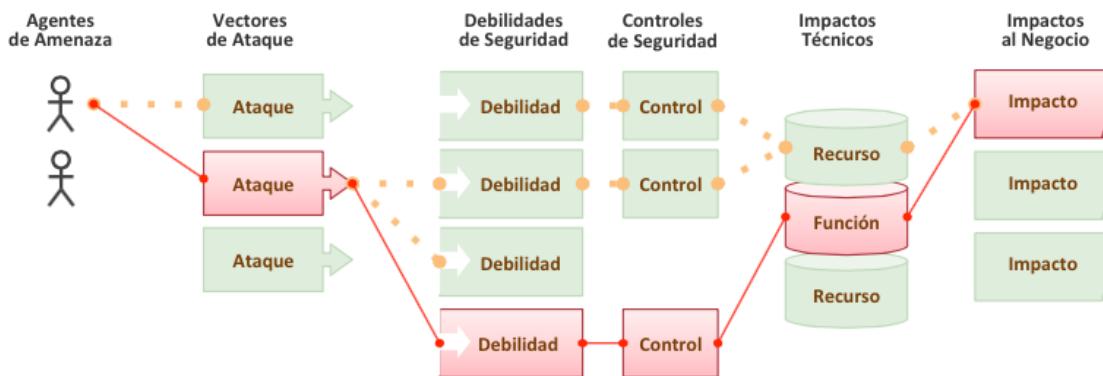
El objetivo del proyecto Top 10 es crear conciencia acerca de la seguridad en aplicaciones mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. El proyecto Top 10 es referenciado por muchos estándares, libros, herramientas, y organizaciones, incluyendo MITRE, 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 solo a la incidencia. La edición 2013 sigue el mismo enfoque. (p.2).

La guía del OWASP Top 10 es ampliamente utilizada en la creación de herramientas, estándares, manuales y componentes entre muchas 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 26 Modelo de riesgos**



Fuente: OWASP Top 10 2013 Español

### 1.3 Diseño de la solución

El diseño del *software*, parte fundamental del ciclo de vida del desarrollo, permite tener noción de los diversos componentes del *software*, sus interacciones y los diversos flujos de trabajo. El afamado escritor Antoine de Saint-Exúper en el libro titulado Viento, Arena y Estrellas, refiriéndose al 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*.

### 1.3.1 UML

UML, por sus siglas en inglés, es el acrónimo de Lenguaje de Modelado Unificado un lenguaje gráfico de propósito general el cual se ha convertido en un estándar en el momento de modelar un sistema.

Esposito & Saltarello (2009) afirman que:

Para diseñar un sistema, cualquier sistema en cualquier campo de la ciencia, usted primero necesita crear una abstracción de él. Una abstracción en esencia es un modelo que provee una representación conceptual del sistema en términos de vistas, estructuras, comportamiento, participación de entidades y procesos.

UML es un lenguaje de modelado gráfico de propósito general que, al pasar los años se ha convertido en un estándar. Basado en una familia de notaciones gráficas, UML está particularmente equipado para crear modelos en escenarios orientados a objetos. Aunque UML es un lenguaje de modelado de propósito general, también provee herramientas para personalizar a un dominio específico. (p. 31).

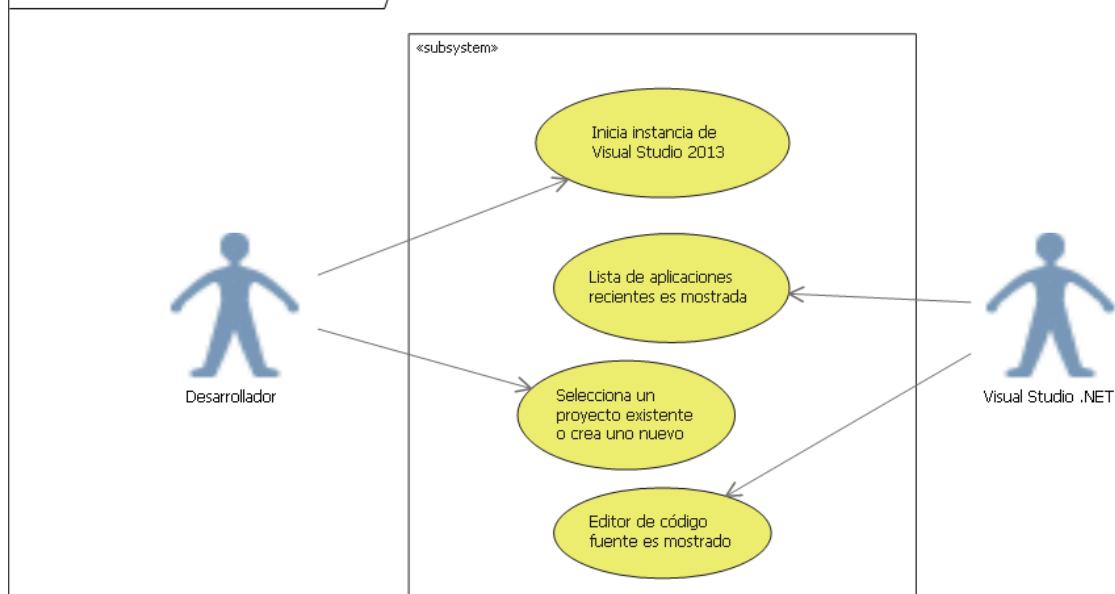
En esta investigación, se hará uso de las herramientas proporcionadas por UML entre las que figuran los casos de uso y diagramas de clases.

### 1.3.2 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).

#### 1.3.2.1 Creación de un proyecto nuevo o selección de uno existente

**Figura 27 Caso de uso 1 - Creación o selección de un proyecto**  
uc Visual Studio startup



**Fuente: Propia**

**Cuadro 5 Creación o selección de un proyecto existente.**

<b>CU:01</b>	Inicio de un proyecto nuevo o existente en Visual Studio
<b>Versión</b>	Versión 1. Viernes 24 de octubre de 2014
<b>Autores</b>	Michael Hidalgo Fallas

<b>Fuentes</b>		
<b>Objetivos asociados</b>		
<b>Descripción</b>	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.	
<b>Precondición</b>	El usuario deberá tener instalado una versión de Visual Studio .NET 2013 o superior en su ambiente.	
<b>Secuencia</b>	<b>Paso</b>	<b>Acción</b>
<b>Normal</b>	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 disponible en caso de tratarse de una aplicación existente.
<b>Poscondición</b>	Análisis de código fuente es ejecutado.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>

	1	La computadora del desarrollador no cuenta con los requisitos mínimos de <i>software</i> y <i>hardware</i> 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 <i>software</i> desde el sitio oficial de Microsoft.
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	De 1 a 3 minutos dependiendo de las características de <i>software</i> y <i>hardware</i> de la computadora del desarrollador.
<b>Frecuencia esperada</b>	<nº de veces> veces / <unidad de tiempo>	
<b>Importancia</b>	Importante	
<b>Urgencia</b>	Inmediatamente.	
<b>Comentarios</b>		

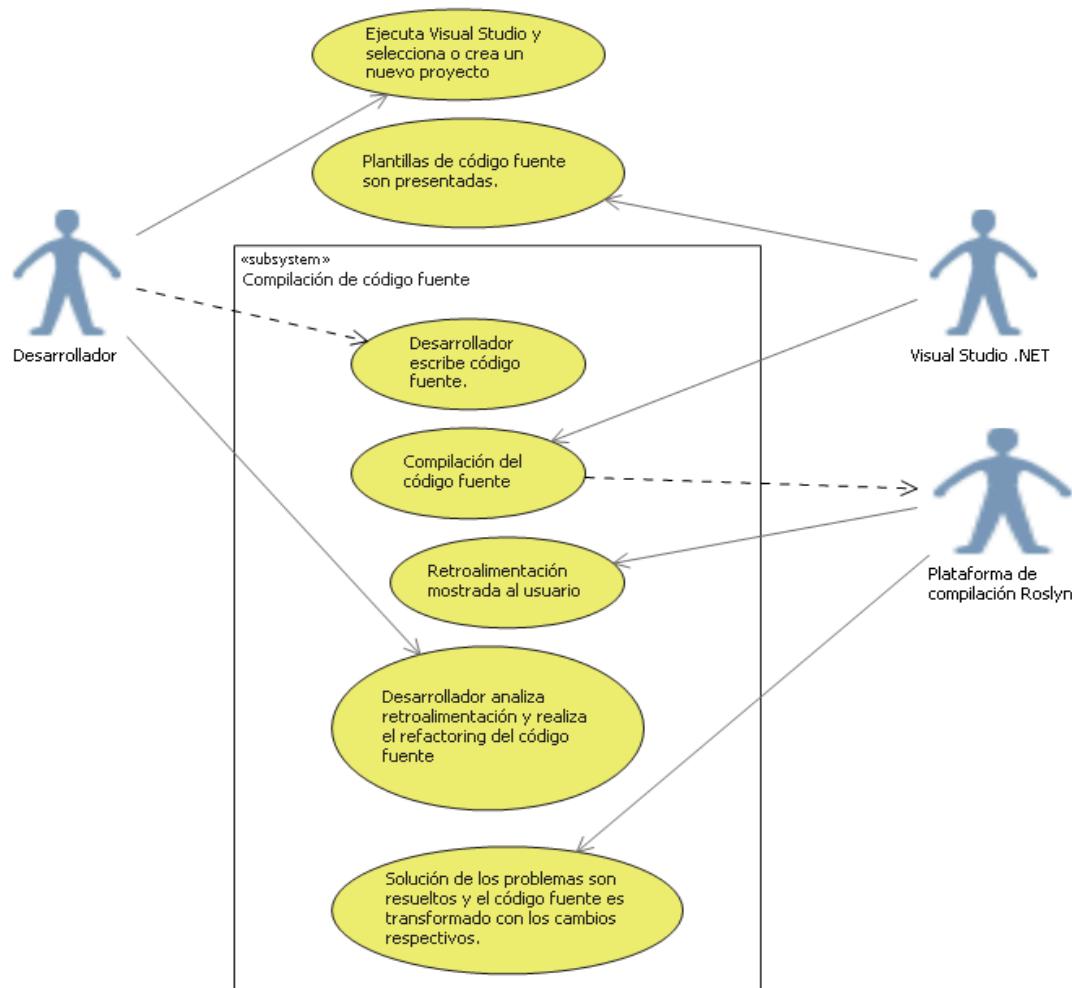
**Fuente: Propia**

### 1.3.2.2 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. Al 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*).

**Figura 28 Caso de Uso 2 - Compilación como servicio**



**Fuente: Propia**

**Cuadro 6 Compilación del código fuente como servicio.**

<b>CU:02</b>	Compilación del código fuente como servicio
<b>Versión</b>	Versión 1. Viernes 24 de Octubre de 2014
<b>Autores</b>	Michael Hidalgo Fallas
<b>Fuentes</b>	

<b>Objetivos asociados</b>			
<b>Descripción</b>	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 hacer 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.		
<b>Precondición</b>	UC:01		
<b>Secuencia</b>	<b>Paso</b>	<b>Acción</b>	
<b>Normal</b>	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.	

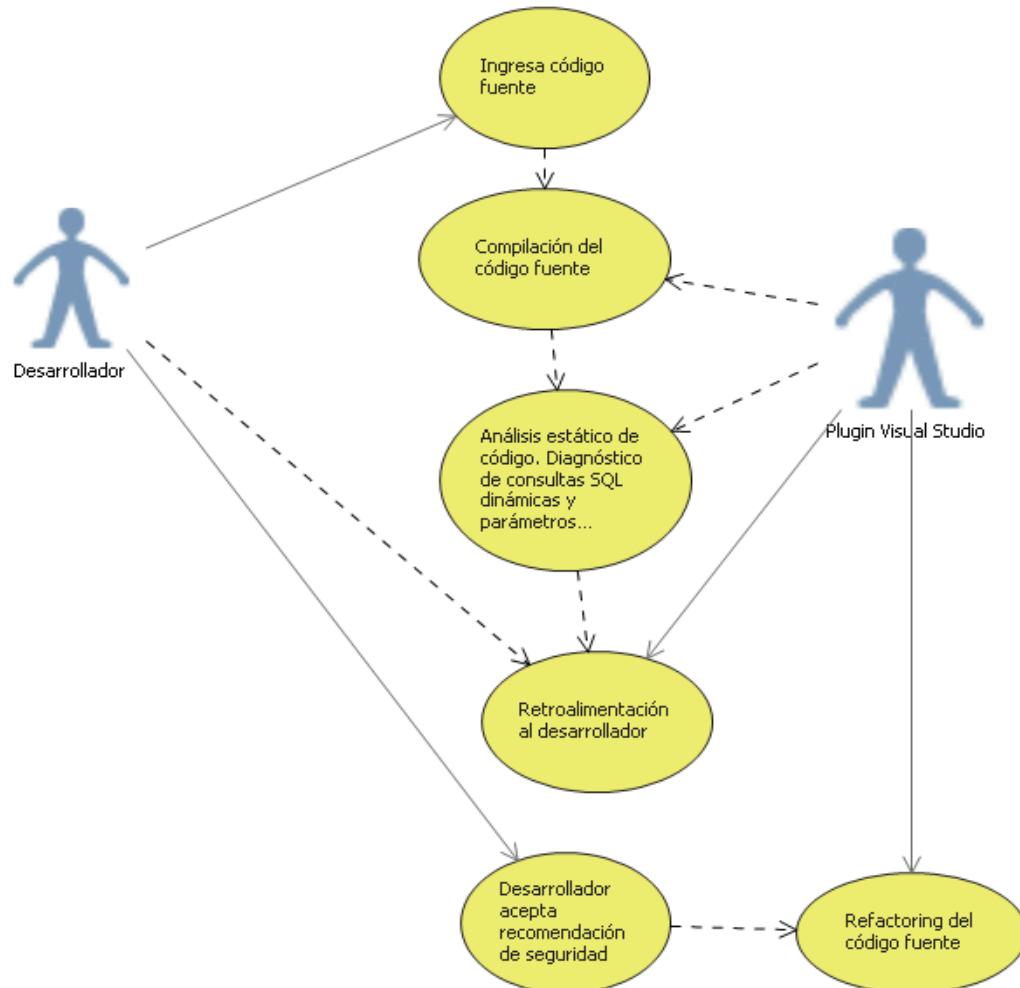
	5	El desarrollador observa la retroalimentación proporcionada por la herramienta y procede aceptar o rechazar la sugerencia brindada por el <i>plugin</i> con el objetivo de tener código fuente más seguro.
	6	El <i>plugin</i> de Visual Studio realiza el <i>refactoring</i> del código fuente con el propósito de resolver el problema de seguridad encontrado en el código fuente.
<b>Poscondición</b>		
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	El desarrollador omite las recomendaciones proporcionadas por el <i>plugin</i> 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 <i>plugin</i> no muestra información al respecto, ya que el diagnóstico implementado no ha sido capaz de encontrar patrones de código fuente con vulnerabilidades conocidas.
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	3 segundos
<b>Frecuencia esperada</b>	Múltiples ocasiones.	
<b>Importancia</b>	Importante.	
<b>Urgencia</b>	Inmediatamente	
<b>Comentarios</b>		

**Fuente: Propia**

### 1.3.2.3 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.

**Figura 29 Módulo de vulnerabilidades de Inyección de SQL**



**Fuente: Propia**

**Cuadro 7 Módulo de vulnerabilidades de Inyección de SQL.**

CU:03	Módulo de vulnerabilidades de inyección de SQL.
Versión	Versión 1. Viernes 24 de octubre de 2014
Autores	Michael Hidalgo Fallas

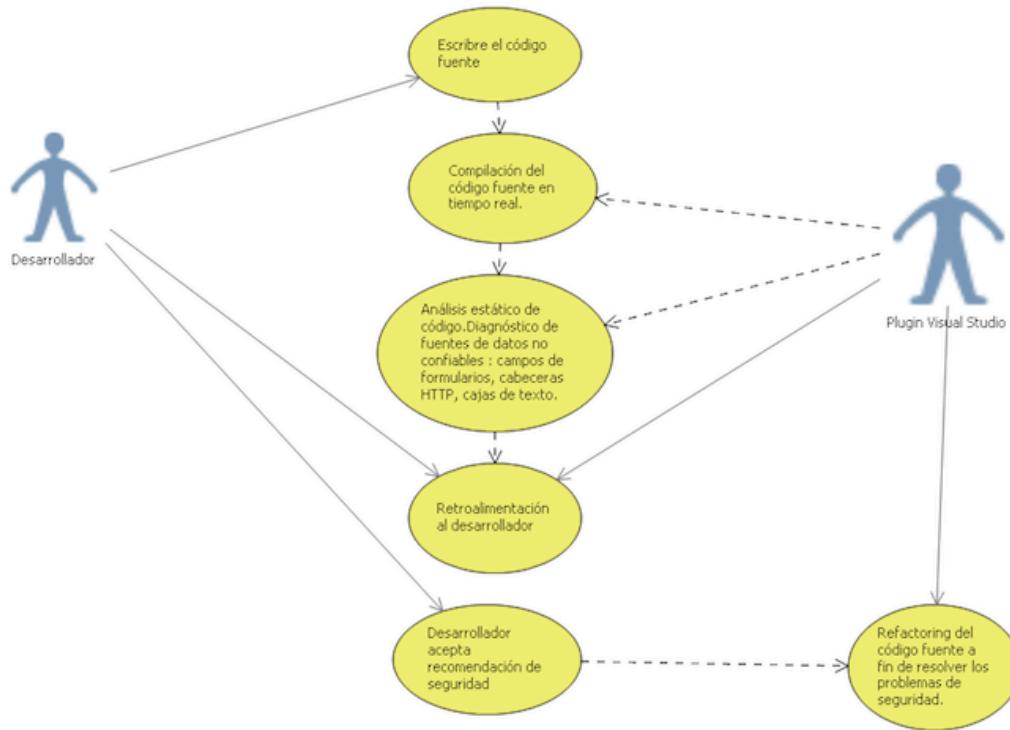
<b>Fuentes</b>		
<b>Objetivos asociados</b>		
<b>Descripción</b>	Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el <i>plugin</i> de Visual Studio .NET realiza el análisis del código a fin de encontrar vulnerabilidades o patrones 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.	
<b>Precondición</b>	Se realiza el <i>refactoring</i> del código fuente con soluciones a los problemas identificados.	
<b>Secuencia</b>	<b>Paso</b>	<b>Acción</b>
<b>Normal</b>	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.
	3	El <i>plugin</i> 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, <i>cookies</i> .
	4	Una vez identificados estos patrones vulnerables de código fuente, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un <i>refactoring</i> 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	
<b>Postcondición</b>	Código fuente es modificado para mitigar el riesgo de que una vulnerabilidad de inyección de SQL ocurra.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	El <i>plugin</i> 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	
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	3 segundos
<b>Frecuencia esperada</b>	Frecuente. Se ejecuta durante todo el desarrollo.	
<b>Importancia</b>	Importante.	
<b>Urgencia</b>	Urgente.	
<b>Comentarios</b>		

**Fuente: Propia**

### 1.3.2.4 Módulo de vulnerabilidades de Secuencia de Sitios Cruzados (XSS).

**Figura 30 Vulnerabilidades de Secuencia de Sitios Cruzados (XSS)**



**Fuente: Propia**

**Cuadro 8 Módulo de vulnerabilidades de XSS.**

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

<b>Descripción</b>	<p>Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación; y en tiempo de compilación, el <i>plugin</i> 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, <i>cookies</i>.</p> <p>Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.</p>	
<b>Precondición</b>	<p>Se realiza el <i>refactoring</i> del código fuente con soluciones a los problemas identificados.</p>	
<b>Secuencia</b>	<b>Paso</b>	<b>Acción</b>
<b>Normal</b>	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.
	3	El <i>plugin</i> 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.

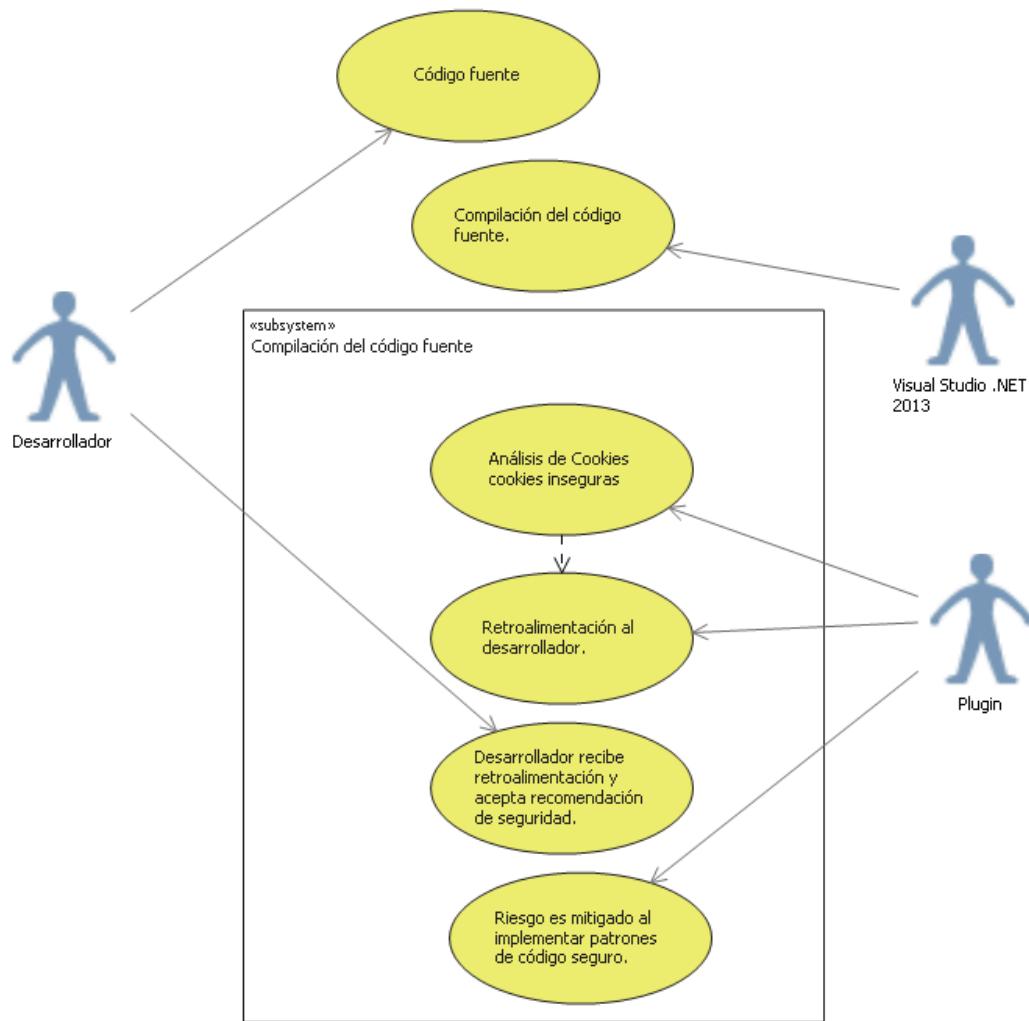
	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un <i>refactoring</i> del código fuente, siguiendo una mejor práctica. De igual forma, se proporcionan <i>links</i> a TEAM Mentor 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	
<b>Postcondición</b>	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	El <i>plugin</i> 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 <i>plugin</i> de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	3 segundos

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

**Fuente: Propia**

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

**Figura 31 Pérdida de autenticación y gestión de sesiones**



**Fuente: Propia**

**Cuadro 9 Vulnerabilidades de pérdida de autenticación y gestión de sesiones**

<b>CU:05</b>	Módulo de vulnerabilidades de pérdida de autenticación y gestión de sesiones.
<b>Versión</b>	Versión 1. Viernes 24 de octubre de 2014
<b>Autores</b>	Michael Hidalgo Fallas
<b>Fuentes</b>	
<b>Objetivos asociados</b>	

<b>Descripción</b>	<p>Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación, el <i>plugin</i> 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, <i>cookies</i> que puedan ser modificadas mediante un código del lado del cliente (propriamente 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, <i>cookies</i>.</p> <p>Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.</p>								
<b>Precondición</b>	Se realiza el <i>refactoring</i> del código fuente con soluciones a los problemas identificados.								
<b>Secuencia</b>	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>El desarrollador escribe el código fuente.</td></tr> <tr> <td>2</td><td>A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.</td></tr> <tr> <td>3</td><td>El <i>plugin</i> 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.</td></tr> </tbody> </table>	Paso	Acción	1	El desarrollador escribe el código fuente.	2	A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.	3	El <i>plugin</i> 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.
Paso	Acción								
1	El desarrollador escribe el código fuente.								
2	A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.								
3	El <i>plugin</i> 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.								

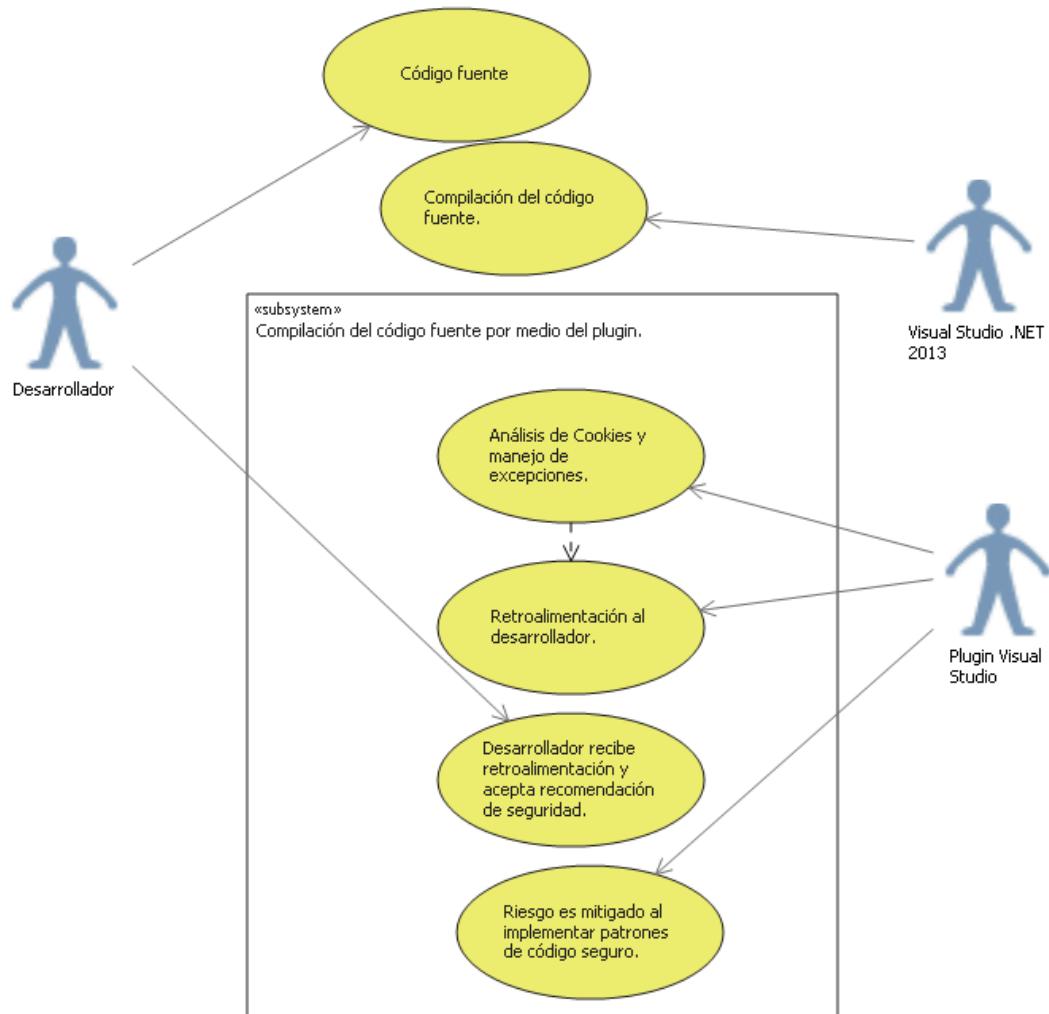
	4	Una vez identificados estos patrones de código fuente vulnerables, el usuario es alertado con un mensaje dentro del ambiente de desarrollo y con la opción de poder realizar un <i>refactoring</i> del código fuente, siguiendo una mejor práctica. De igual forma, se proporcionan <i>links</i> a TEAM Mentor 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	
<b>Poscondición</b>	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	El <i>plugin</i> 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 <i>plugin</i> de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	3 segundos

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

**Fuente: Propia**

### 1.3.2.6 Módulo de vulnerabilidades de Configuración Incorrecta de Seguridad.

**Figura 32 Caso de Uso Configuración Incorrecta de Seguridad**



**Fuente: Propia**

**Cuadro 10 Configuración Incorrecta de Seguridad**

<b>CU:06</b>	Módulo de vulnerabilidades de Configuración Incorrecta de Seguridad.
<b>Versión</b>	Versión 1. Viernes 24 de octubre de 2014
<b>Autores</b>	Michael Hidalgo Fallas
<b>Fuentes</b>	

Objetivos asociados		
Descripción	<p>Este caso de uso inicia cuando el desarrollador está escribiendo el código fuente de la aplicación, y en tiempo de compilación el <i>plugin</i> 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 sensible, tales como plataforma de desarrollo, origen de los errores, entre otros.</p> <p>Luego el desarrollador tendrá la opción de aceptar las recomendaciones brindadas por el componente.</p>	
Precondición	Se realiza el <i>refactoring</i> del código fuente con soluciones a los problemas identificados.	
Secuencia	Paso	Acción
<b>Normal</b>	1	El desarrollador escribe el código fuente.
	2	A medida que el código fuente es escrito, el <i>plugin</i> desarrollado realiza la compilación en tiempo real.
	3	El <i>plugin</i> 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 <i>refactoring</i> del código fuente, siguiendo una mejor práctica. De igual forma se proporcionan <i>links</i> 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	
<b>Postcondición</b>	El usuario es alertado dentro del mismo ambiente integrado y se le muestra la opción de mitigar el problema de seguridad.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	El <i>plugin</i> 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 <i>plugin</i> de Visual Studio .NET para realizar análisis estático de código fuente no se encuentra instalado en la máquina del desarrollador.
<b>Rendimiento</b>	<b>Paso</b>	<b>Cuota de tiempo</b>
	1	3 segundos

<b>Frecuencia esperada</b>	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.
<b>Importancia</b>	Importante.
<b>Urgencia</b>	Urgente.
<b>Comentarios</b>	

### **Fuente: Propia**

#### **1.3.3 Diagrama de Clases**

Kendall & Kendall (2011) refiriéndose a los diagramas de clases aportan que:

Las metodologías orientadas a objetos trabajan para descubrir las clases, atributos, métodos y relaciones entre las clases. Como la programación ocurre a nivel de clase, definir clases es una de las tareas más importantes del análisis orientado a objetos. Los diagramas de clases muestran las características estáticas del sistema y no representan ningún procesamiento en especial.

En un diagrama de clases, las clases se representan mediante un rectángulo. En el formato más simple, el rectángulo puede incluirse solamente el nombre de la clase, pero también se pueden incluir los atributos y los métodos. (p. 297).

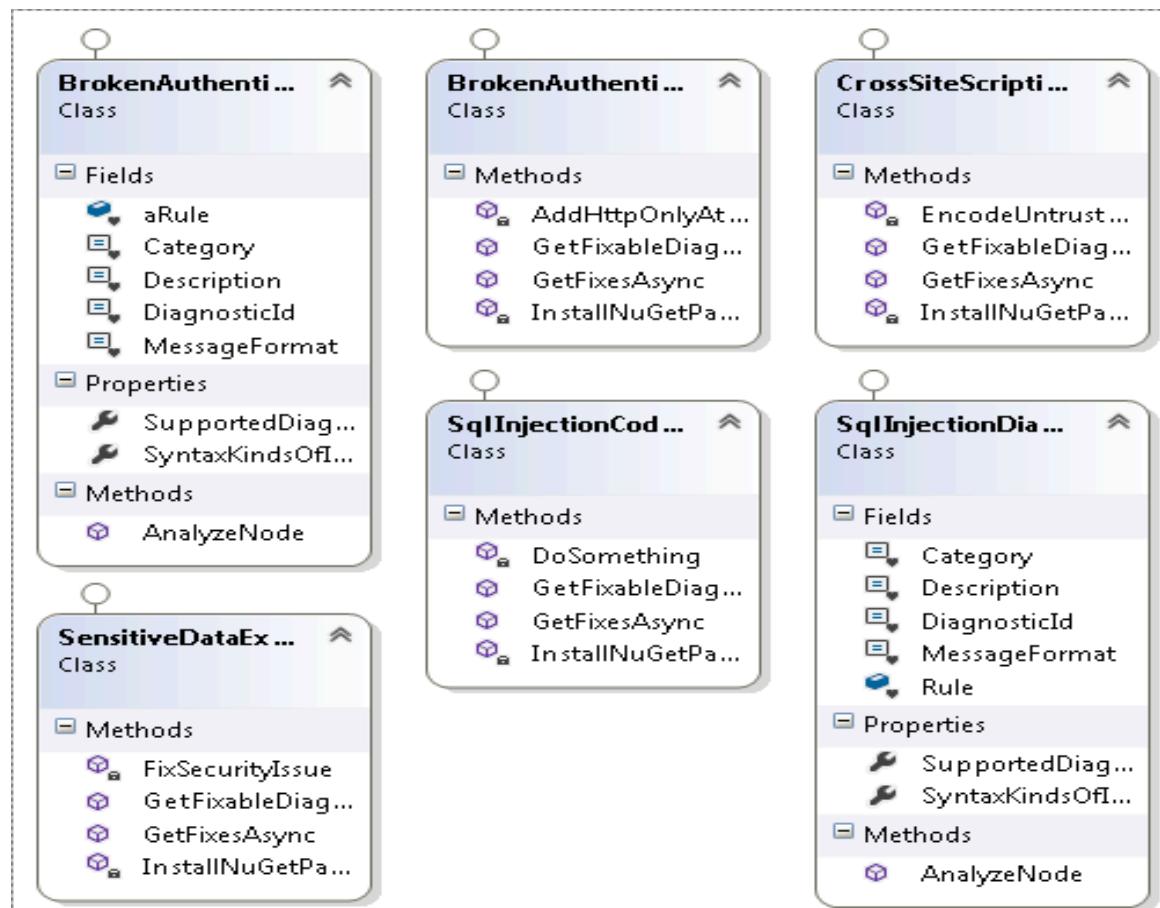
A su vez, Esposito & Saltarello (2009) también aportan que:

Un diagrama de clases representa la estructura estática del sistema. La estructura estática del sistema se compone de las clases y sus

estructuras. Las clases son exactamente las clases (e interfaces) a ser implementadas por el equipo de desarrollo. (p. 47).

A continuación, se presenta el diagrama de clases utilizado por el prototipo funcional propuesto.

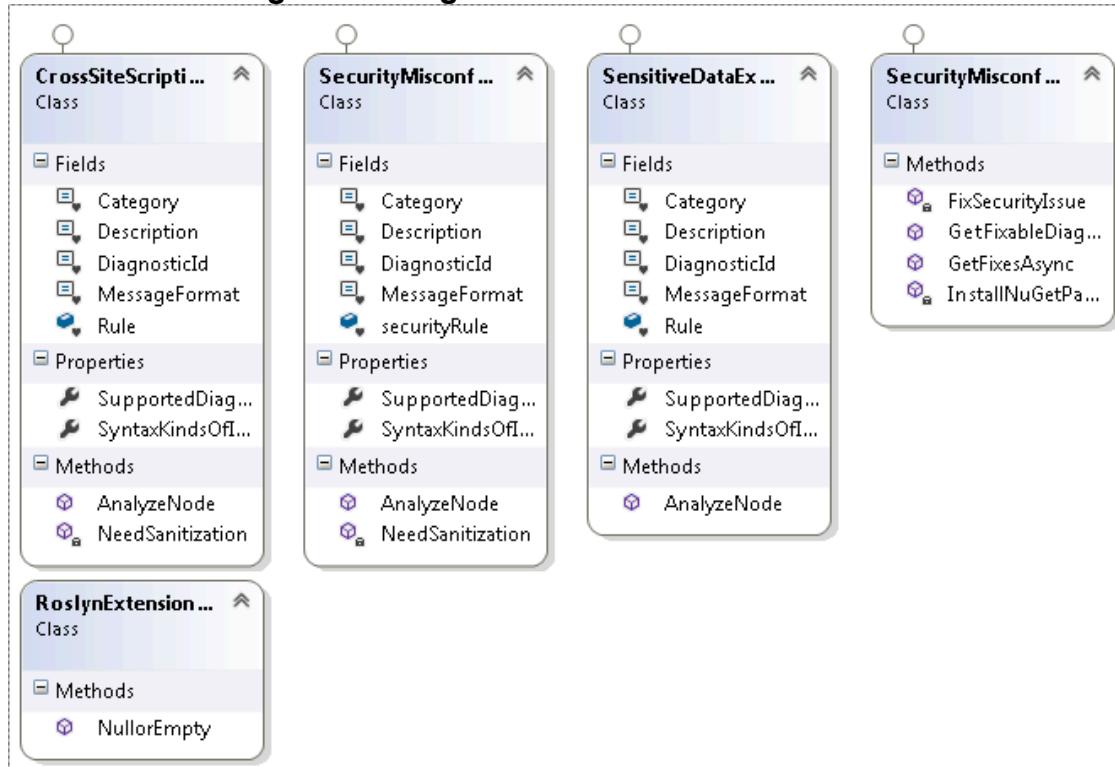
**Figura 33 Diagrama de clases**



Fuente: Propia

En la siguiente imagen, se aprecia la continuación del diagrama de clases utilizado en el desarrollo del prototipo funcional:

**Figura 34 Diagrama de clases continuación**



**Fuente: Propia**

#### 1.3.4 Desarrollo del Prototipo Funcional

Con el objetivo de reafirmar la investigación realizada y fundamentada en este documento, se ha elaborado un prototipo funcional que permite detectar, en tiempo de compilación, errores propiamente de seguridad que si bien es cierto no son detectados por el IDE de Visual Studio, guían a que se desarrolle aplicaciones de *software* que no sean seguras.

Así mismo se busca brindarle al desarrollador no solamente información de que existe código fuente que es vulnerable y que eventualmente permita o facilite que se dé la explotación de algún riesgo, si no que además se le brinda una opción para que corrija el problema de forma automática.

Utilizando la plataforma de compilación Roslyn se provee un enfoque donde se presenta una vista previa del problema resuelto y así el desarrollador se va familiarizando con las mejores prácticas que están siendo recomendadas por el componente. Además el hecho de mostrar vínculos con información con mejores prácticas presenta un valor agregado para cualquier organización. Se fomenta así mismo una cultura de concientización donde los desarrolladores se familiaricen con el código fuente vulnerable y sus posibles soluciones.

#### **1.3.4.1 Instalación de la Plataforma de Compilación Roslyn**

Tal como se ha indicado en secciones previas de este documento, en la investigación propuesta se utiliza un enfoque novedoso por medio de la utilización de la plataforma de compilación Roslyn. Dicha plataforma de compilación, nueva en naturaleza, tuvo su lanzamiento oficial a inicios del presente año 2013. Es imprescindible instalar esta plataforma a fin de poder crear y desarrollar las extensiones capaces de realizar el análisis de código al nivel que se ha propuesto.

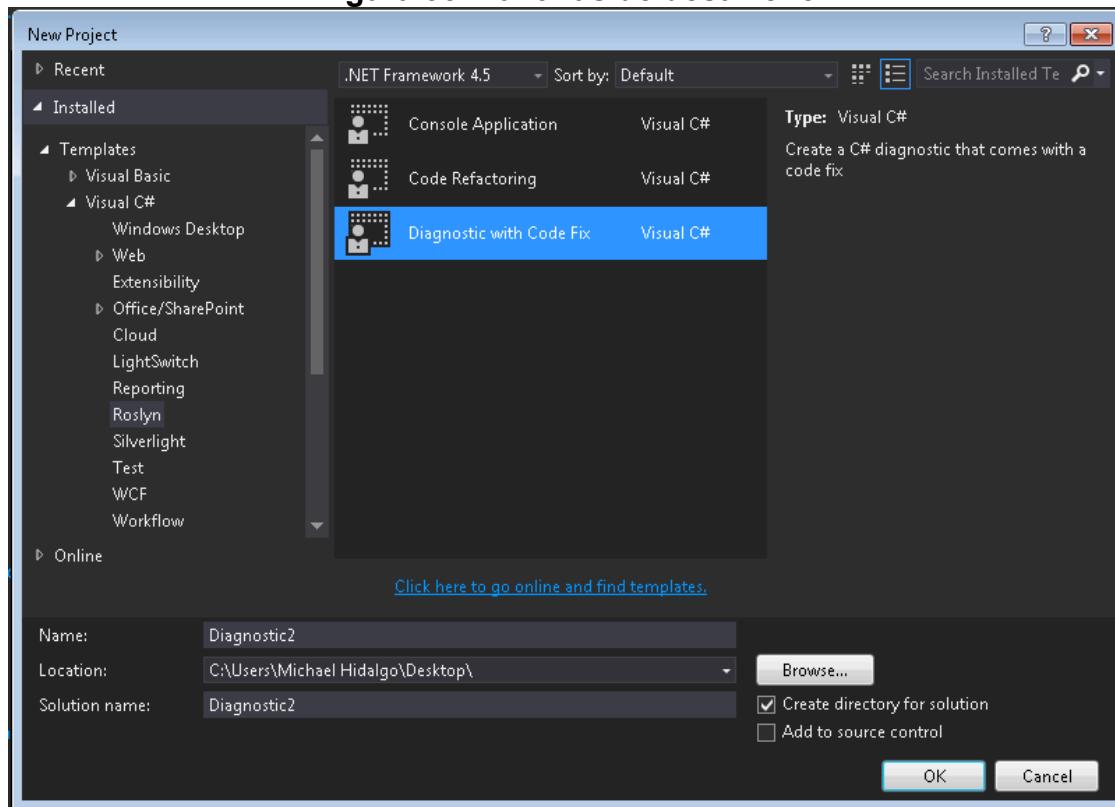
El requisito fundamental entonces es contar con Visual Studio .NET 2013 o superior instalado en el ambiente propuesto para el desarrollo de las extensiones.

A continuación, se enumeran los pasos necesarios para la correcta configuración de la plataforma:

1. Instalar el SDK (Software Development Kit) para Visual Studio.NET 2013. Este se encuentra en esta dirección para su descarga desde el sitio <http://www.microsoft.com/en-us/download/details.aspx?id=40758>
2. Extraer el contenido de los archivos comprimidos en una ruta específica dentro del computador.
3. Instalar la extensión Roslyn SDK Project Template.vsix. Esta extensión instalará seis nuevas plantillas de diseño en Visual Studio .NET.
4. Instalar la extensión Roslyn Syntax Visualizer.vsix. Esta extensión permite investigar los árboles de sintaxis de los archivos de código fuente.
5. Instalar el ambiente experimental denominado Visual Studio Experimental Hive, la cual permite probar en tiempo real las extensiones que están siendo desarrolladas.

Una vez que se han seguido los pasos anteriores, en el momento de crear un proyecto en Visual Studio, aparecerán las nuevas plantillas para el desarrollo de las extensiones.

**Figura 35 Plantillas de desarrollo**



### Fuente: Propia

Para el desarrollo de la extensión de seguridad propuesto, se utiliza la plantilla denominada Diagnóstico con Solución de Código; la cual, como su nombre implica, permite realizar el diagnóstico del problema a fin y luego mostrar la opción de cómo solucionar el problema.

Es importante recalcar que esta extensión provee una solución al problema siguiente estándares internacionales. No obstante, queda a discreción el desarrollador obedecer la recomendación propuesta o implementar su propio control. Pero la identificación del problema es clave, la

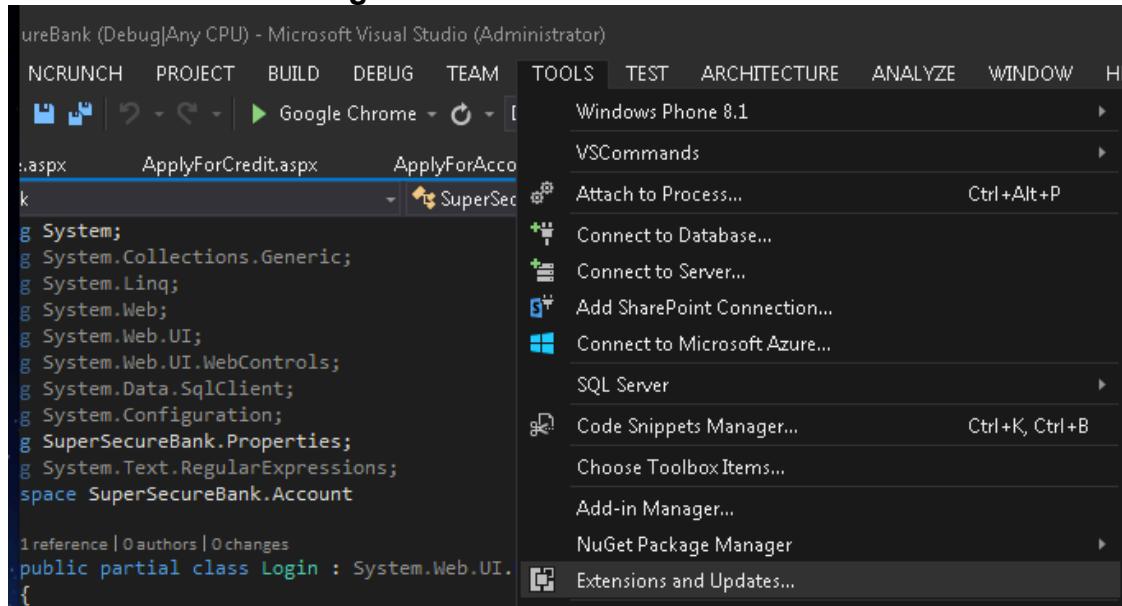
retroalimentación que el prototipo propone facilita que el desarrollador se dé cuenta del problema o problemas en los que se está incurriendo.

#### **1.3.4.2 Ubicación de la extensión en el modelo de desarrollo**

El componente de seguridad se instala en un ambiente de desarrollo que tenga instalada una versión de Visual Studio .NET 2012 o superior. Inmediatamente que se ha detectado que Visual Studio existe, se procede con la instalación. Nótese que una vez instalado el componente debe aparecer como parte de las extensiones de instaladas dentro del ambiente de desarrollo.

Esto quiere decir que dentro de Visual Studio, en la barra de herramientas bajo el submenú que reza bajo el nombre de Extensiones y Actualizaciones (Extension and Updates dependiendo del lenguaje predeterminado de Visual Studio), ahí aparece la extensión tal como se ilustra seguidamente:

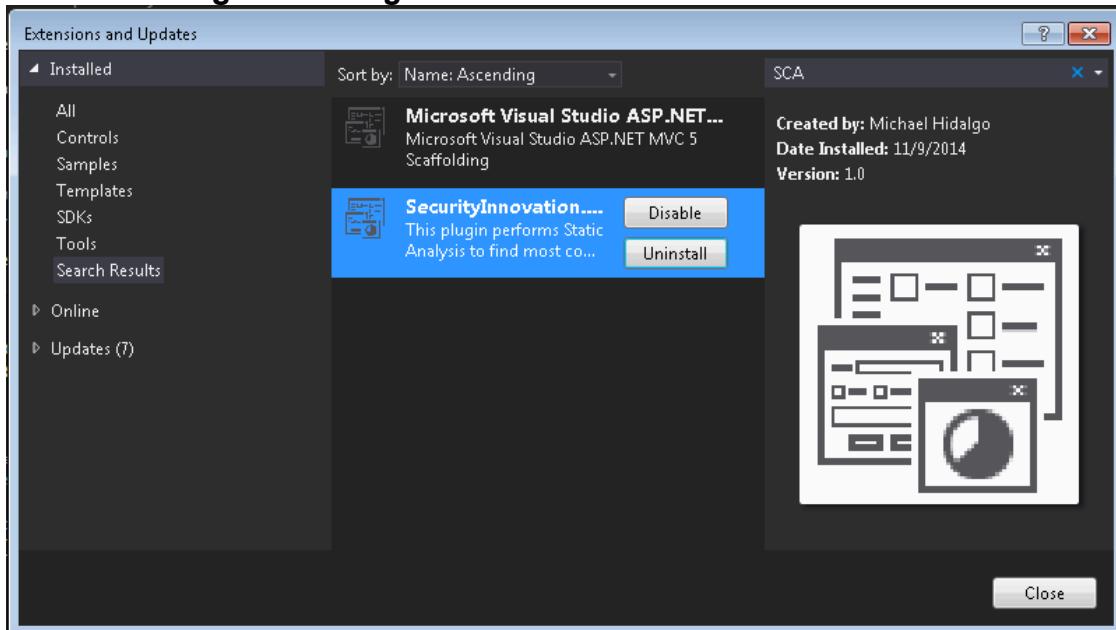
**Figura 36 Barra de Herramientas**



### Fuente: Propia

Seguidamente aparece una ventana donde se localizan las extensiones instaladas y dentro de la lista figura el componente desarrollado, el cual lleva por nombre SecurityInnovation.SCA. La abreviatura SCA obedece a Análisis Estático de Código por sus siglas en inglés (Static Code Analysis).

**Figura 37 Plugin instalado dentro de Visual Studio**



### Fuente: Propia

Nótese además que en esta ventana se podrá ya sea deshabilitar o desinstalar el componente.

De esta forma, el desarrollador escribe el código fuente de la aplicación web en el lenguaje de programación C#. En tiempo real, el código de forma transparente es compilado y analizado por la extensión justo después de que se ha digitado el código fuente. Nótese que el desarrollador no experimentará ningún problema de desempeño durante este proceso.

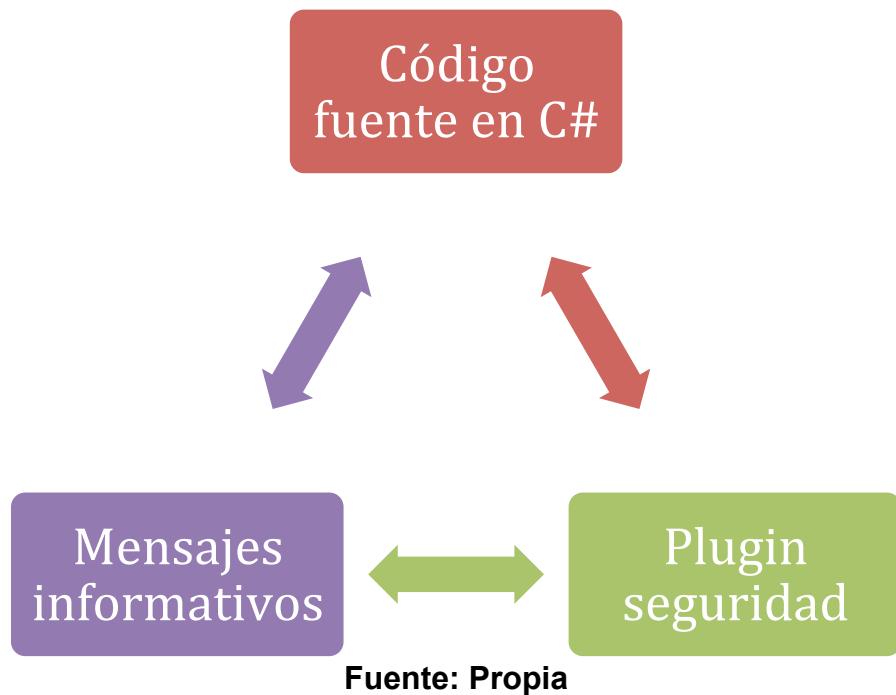
La lógica implementada dentro del componente de seguridad permite que se descarte cualquier código que no forme parte del diagnóstico, esto quiere decir que si hay secciones de código incluso de todo el archivo que no presentan ninguna sección del diagnóstico de interés, el componente no lo validará, haciendo un proceso amigable.

Como se ha indicado, el proceso ocurre en tiempo real una vez que el código fuente ha sido digitado, el análisis es ejecutado en busca de algún patrón de código fuente vulnerable.

Desde el punto de vista de experiencia de usuario, es decir, la percepción que el desarrollador tiene de la herramienta, para él será transparente pues la eficiencia del componente hace que no se note los procesos internos.

Se ha utilizado el mantra característico de los mensajes de error con el objetivo de mostrar la retroalimentación al desarrollador.

**Gráfico 2 Ubicación de la extensión de seguridad**



Tal como se ilustra existe una relación estrecha entre el componente dentro del ambiente de desarrollo. Para el usuario desarrollador, la instalación del componente es trivial en términos de que ya existe asimilación con las extensiones en un ambiente tan fácil de acoplarse como lo es Visual Studio.

En el sentido directo, se busca tomar ventaja de las características extensibles del ambiente de desarrollo y del IDE de Visual Studio.

#### **1.3.4.3 Identificación de código fuente vulnerable**

El uso de estándares internacionales, adoptados por la industria a lo largo de los años para detectar vulnerabilidades en el código fuente, tal es el caso de la guía agnóstica OWASP Top 10 2013 son fundamentales para comprender los problemas invisibles de seguridad en el código fuente.

Siguiendo las mejores prácticas propuestas por estos estándares es que se ha logrado desarrollar reglas de diagnóstico a fin de poder encontrar código fuente vulnerable en las aplicaciones de Internet que se desarrollan en la actualidad.

Cada regla de seguridad implementada consta de dos secciones: del diagnóstico del código fuente y la solución respectiva al problema. En el siguiente segmento de código fuente, se aprecia cómo se ha programado una regla de seguridad para identificar un uso incorrecto de los mensajes de error.

Los mensajes de error no manejados correctamente figuran como candidatos potenciales en el momento de revelar información sensitiva para la organización. De forma tal que el diagnóstico se hace para detectar

excepciones en el código que se hayan dejado por defecto, es decir donde muestran al usuario final todo el historial de compilación.

#### **1.3.4.4 Anatomía de un ataque de Inyección de SQL**

Los ataques de Inyección de SQL buscan poder explotar un sitio web vulnerable el cual se basa en un modelo de bases de datos relacionales para el almacenamiento de los datos. El atacante entonces aprovecha todas las fuentes de datos no confiables como lo son los campos dentro de un formulario, las cajas de texto que realizan búsquedas, servicios web de terceros para injectar como parte de los datos de entrada, sintaxis SQL que tiene como objetivo alterar el comportamiento de la aplicación.

En el momento en que el usuario malicioso determina que los datos inseguros que él está introduciendo en la aplicación están siendo interpretados por el motor de base de datos, este puede ganar acceso a todos los datos de almacenados en el repositorio e incluso realizar modificaciones a estos, claro está si no existen los roles y permisos que lo prevengan.

Un dato preocupante lo constituye el hecho de que 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 de *software*. De igual forma, la gran cantidad de información disponible en Internet, la cual 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 está basado en riesgos, lo cual permite que se analicen y se tomen medidas para poder mitigarlo.

**Figura 38 Inyección de SQL Modelo de Riesgo**



**Fuente: OWASP 2013 Español**

El anterior modelo de riesgo es claro al indicar que la explotación de una vulnerabilidad de inyección es sumamente fácil de perpetrar y trae consigo un efecto nefasto en la organización, ya que se puede dar el caso

donde la información confidencial de la empresa y de los clientes sea robada o alterada de forma no autorizada.

Tal como se puede apreciar en la siguiente ilustración, el atacante utiliza la misma aplicación web para inyectar comandos, es decir, sintaxis del lenguaje SQL, utilizando la caja de texto del usuario, con el objetivo de borrar la tabla en la base de datos. Nótese que el formulario de inicio de sesión mostrado seguidamente es el reflejo de miles de formulario similares, los cuales tiene el objetivo de verificar que el usuario es quien dice ser y así permitirle el acceso.

En el flujo normal de eventos el usuario de la aplicación ingresa los credenciales (compuestos por un usuario y una contraseña). Sin embargo, en un vector de ataque en lugar de ingresar el nombre del usuario, el atacante, en este caso, ingresa el texto ‘ OR 1=1; DROP TABLE STUDENTS; generando que el motor de base de datos interprete de forma assertiva los datos de entrada y provoca un efecto de lado en el servidor; es decir, elimina el objeto en el motor transaccional.

**Figura 39 Inyección de SQL en un formulario HTML****Ejemplo Inyección de SQL**

The figure consists of two side-by-side screenshots of a 'Login' interface. Both screenshots show a 'Username' field containing 'student' and a 'Password' field containing a series of dots. On the left, the 'Login' button is grey. On the right, the 'Login' button is yellow. Below each screenshot is a corresponding SQL query:

```
SELECT * FROM STUDENT WHERE USERNAME='student'
SELECT * FROM STUDENT WHERE USERNAME=" OR 1=1;DROP TABLE STUDENT;
```

**Fuente: Propia**

Desde la perspectiva del vector de ataque, mostrado en rojo en la pantalla de la derecha, se puede notar que la sintaxis proporcionada por el usuario malicioso como parte de los datos de entrada se traduce en el motor de base de datos relacional como una consulta que en principio es válida.

Nótese cómo el comando de borrado de la tabla o Drop ha sido introducido como parte del parámetro de entrada y es ejecutado secuencialmente por el motor. En un escenario más trágico (y perfectamente válido valga la aclaración), el atacante podrá borrar la tabla si los roles previamente definidos para la aplicación en cuestión le permiten hacer tales operaciones.

Es importante entonces explicar por qué ocurre una vulnerabilidad de este tipo y qué factores guían a que se materialice. En principio, se puede objetar que el problema fundamental se encuentra en utilizar consultas

dinámicas en el código fuente. Esto significa que el desarrollador va armando una consulta dependiendo de los datos introducidos, por ejemplo un rango de fechas, el código de cliente ingresado, el número de identificación personal entre otros.

En la medida en que los datos mencionados se van concatenando para crear una búsqueda en el motor de base de datos relacional, existe la posibilidad de que el usuario malicioso, consciente de que la aplicación se comporta según lo estipulado, ingrese en lugar de un rango de fechas un texto que es perfectamente válido como sintaxis de la base de datos. Debido a la naturaleza de la aplicación todos los datos concatenados se envían al intérprete de comandos y este evalúa en términos de sintaxis y semántica que la consulta sea válida y, posteriormente, se ejecuta causando alguna modificación o solamente mostrando la información solicitada.

Desde la perspectiva del código fuente, en la siguiente imagen, se ilustra un problema de inyección:

**Figura 40 Inyección de SQL en el código fuente**

```
String strID = Request.QueryString["ID"];
String strSQLQueryTemplate = "SELECT * FROM tblProduct WHERE ProductId = {0}";
String strSQLQuery = String.Format(strSQLQueryTemplate, strID);
DataTable dtProduct = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(strSQLQuery, new SqlConnection("connection string"));
adapter.Fill(dtProduct);

productListView.DataSource = dtProduct;
productListView.DataBind();
```

Fuente: <http://goo.gl/v1Cx5c>

Nótese que un parámetro denominado ID es esperado, lo cual significa que una dirección como es el caso de <http://example.com?ID=10> funciona correctamente. No obstante, el atacante podría manipular el parámetro ID, cuya modificación es trivial puesto que puede ser editado desde el navegador en la barra de navegación, y enviar algo como esto <http://wxample.com?ID='OR 1=1--;>.

En tal escenario, el motor de base de datos mostrará en su defecto todos los productos almacenados y no necesariamente uno a como está diseñada la aplicación.

Nótese que otra metodología que el atacante puede implementar consiste en enviar texto que provoque un error interno en el momento de evaluar el código por parte del motor de base de datos transaccional.

Debido a la falta de validación de datos de entrada, por ejemplo una validación creada para asegurar que efectivamente el parámetro proporcionado es una fecha válida, o que el número de identificación corresponda con el formato y longitud apropiados para un dato de esta naturaleza, genera que el atacante tenga acceso total a ingresar texto arbitrario, el mismo que no es validado. No obstante, un error recurrente lo constituye el factor de realizar las validaciones solamente en la capa de presentación del sistema.

Con mensajes informativos de que los datos ingresados no son válidos, pero solamente creados a nivel de presentación, se corre el riesgo de

que el usuario malicioso evite tales validaciones y permite a su vez que el riesgo se materialice.

En síntesis, la validación de datos de entrada así como la utilización de sentencias preparadas, es decir, donde se define los tipos de datos de los parámetros, se puede solucionar esta vulnerabilidad.

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

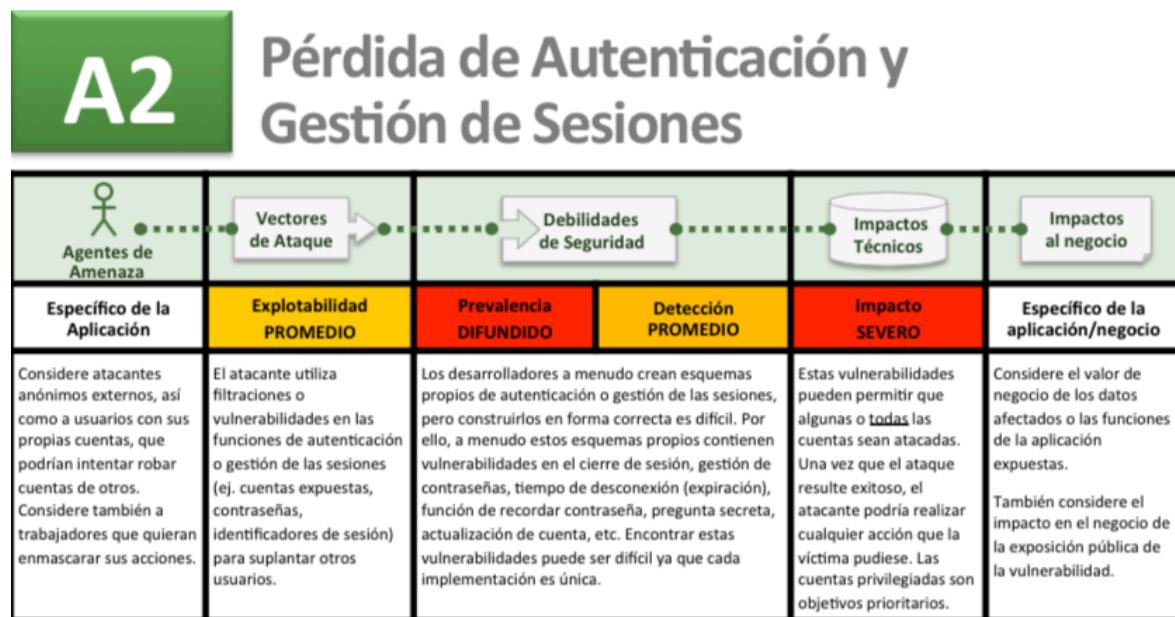
Pobres mecanismos de autenticación y un manejo incorrecto de sesiones conlleva a que un usuario malicioso pueda suplantar (actuar en nombre de otro usuario legítimo del sistema) y realizar transacciones a su nombre.

La incidencia 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.

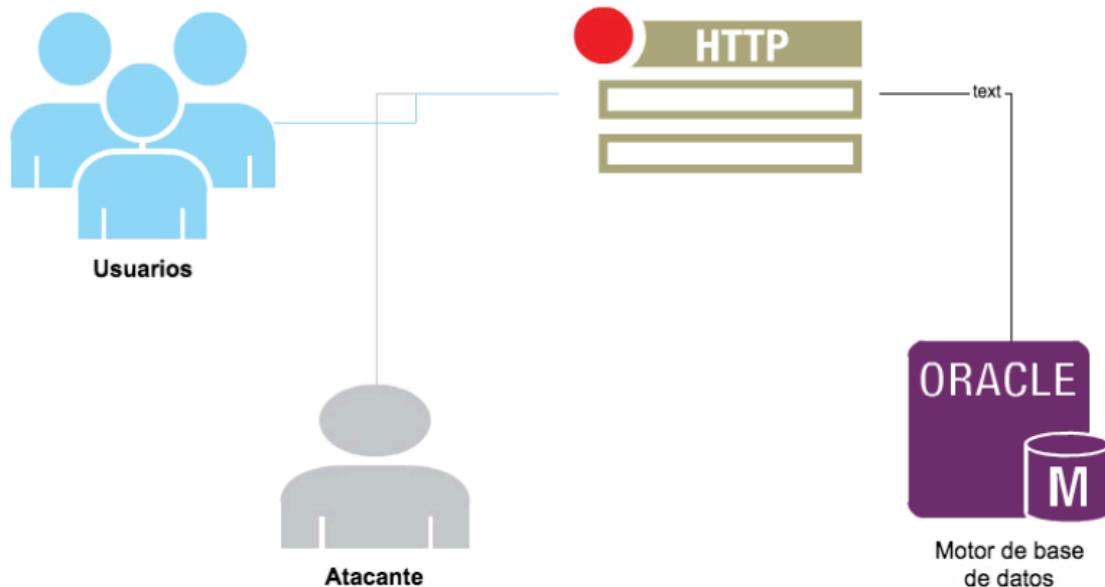
En tal ilustración, se confirma que el impacto para la organización es alto; ya que puede comprometer información confidencial.

**Figura 41 Pérdida de autenticación y gestión de sesiones modelo de riesgo**



**Fuente: OWASP Top 10 Español**

En el siguiente diagrama, se muestra cómo un usuario malicioso es capaz de robar información debido a un manejo incorrecto de la autenticación de la aplicación o por una gestión incorrecta de las sesiones.

**Figura 42 Diagrama pérdida de autenticación****Aplicación Web****Fuente: Propia**

Uno de los principales problemas que acarrea el uso de sesiones en las aplicaciones web, lo constituye el hecho de que estos valores están definidos en forma de *cookies*. Una *cookie* es una variable que es almacenada en el navegador del cliente y que es usada por la aplicación para identificarlo y diferenciarlo de otros usuarios que hacen uso concurrente de la aplicación.

Dependiendo de la naturaleza de la aplicación, estas variables pueden contener información relevante para el modelo de negocio de la empresa. En una tienda virtual, podría almacenar el monto total de los productos del cliente; identificar si el usuario conectado es un administrador del sistema e

incluso en implementaciones no tan bien diseñadas podrían incluso almacenar los niveles de acceso y permisos.

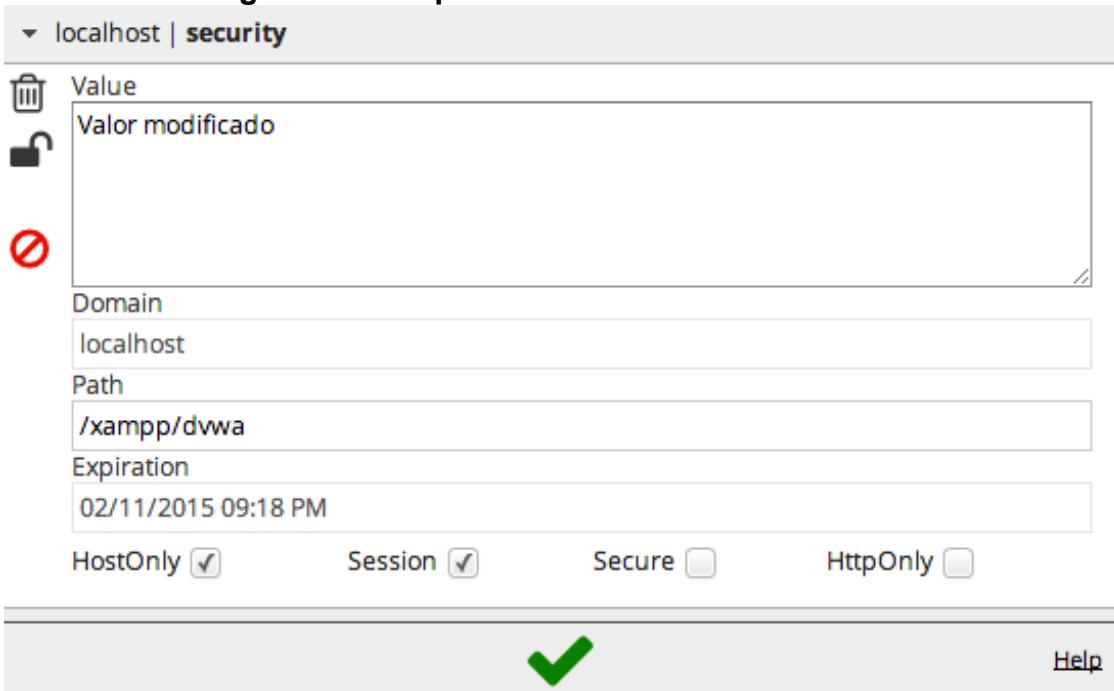
En estos casos específicos, si el atacante puede acceder a tales variables de sesión (las cuales son accesibles de forma intrínseca por medio de JavaScript) y modificarlas o ganar acceso a ellas podrá causar un efecto no deseado en el comportamiento de la aplicación. Utilizando algún lenguaje de programación del lado del cliente, tal es el caso de JavaScript, el atacante puede tener acceso directo a estas variables y a alterarlas a su antojo.

En la imagen siguiente, se demuestra cómo utilizando herramientas y lenguajes del lado del cliente se pueden alterar las variables en sesión. En el caso presentado, se puede modificar el nivel de seguridad de la aplicación al alterar la variable definida bajo el nombre de *security*.

Eventualmente, al realizar el cambio respectivo de dicha variable, por ejemplo cambiar el nivel de seguridad de alto a bajo, se cambia radicalmente el comportamiento de la aplicación, permitiendo que se acceda a información valiosa puesto a que se ha configurado la aplicación a utilizar los niveles más bajos de seguridad definidos.

De igual modo, si se confía plenamente en los datos almacenados en sesión para realizar transacciones que requieren un poco más de cuidado, tal es el caso de valores financieros como procesar un pago, y de esta forma el usuario malicioso ha podido alterar el monto de la compra o de la transferencia, se notará una interrupción tarde o temprano en el sistema y en la contabilidad de la organización.

**Figura 43 Manipulación de variables en sesión**



#### Fuente: Propia

Resulta entonces necesario cuestionarse cómo se podría evitar un problema de tal magnitud en las cientos de miles de aplicaciones que podrían presentar este mismo patrón de comportamiento y de operación. En tal caso y según se puede apreciar en la ilustración anterior, las variables en sesión cuentan con diferentes atributos entre los que se resaltan:

1. *HostOnly*: Esta propiedad es establecida en el momento en que la variable de sesión es creada y en términos generales garantiza que tal valor va a ser compartido únicamente entre el cliente y el servidor específicos durante la negociación de contenido HTTP.

2. *Session*: Indica si la variable definida es de sesión o no. Cuando un *cookie* es de sesión su ciclo de vida estará definido mientras el usuario navegue en el sitio. Cuando no se ha especificado la fecha de caducidad para este tipo de variables, generalmente el navegador elimina dicha variable luego de que el usuario ha cerrado el navegador.
3. *Secure*: Cuando este atributo es establecido, la variable en sesión va a ser transmitida y compartida entre el cliente y el servidor únicamente cuando exista un canal de comunicación seguro. Es decir, cuando se estén protegiendo los datos en tránsito por medio del protocolo SSL. Si no existe este nivel de cifrado de datos en tránsito, tal variable no será transmitida.
4. *HttpOnly*: Esta propiedad indica si la variable en sesión puede o no ser accedida por medio de alguna tecnología del lado del cliente como lo es el lenguaje JavaScript. Idealmente esta propiedad debe estar seleccionada de forma que un atacante no pueda alterar el valor usando alguna de estas técnicas.

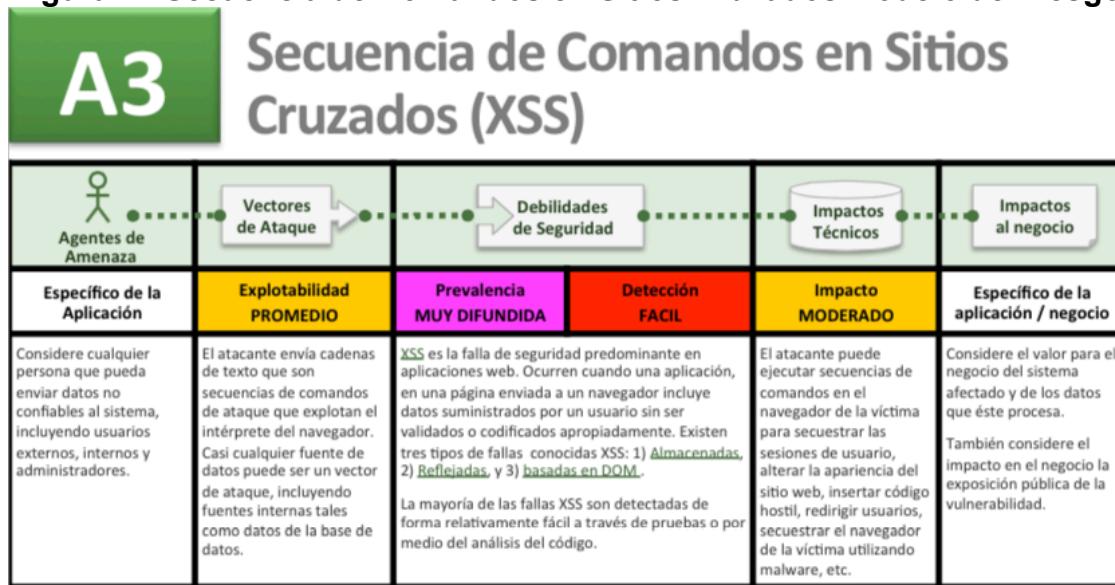
Como recomendación de seguridad, el componente desarrollado analizará las variables en sesión y verificará la existencia del atributo *HttpOnly*, el cual tiene mucha relevancia en el momento de restringir que la variable pueda ser modificada y alterada por algún *script* o código del lado del cliente.

### 1.3.4.7 Secuencia de Comandos en Sitios Cruzados (XSS)

Esta vulnerabilidad tiene sus raíces en las fuentes de información no confiables, es decir, fuentes 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 este podría ser un atacante buscando robar información sensitiva.

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

**Figura 44 Secuencia de Comandos en Sitios Cruzados Modelo de Riesgo**



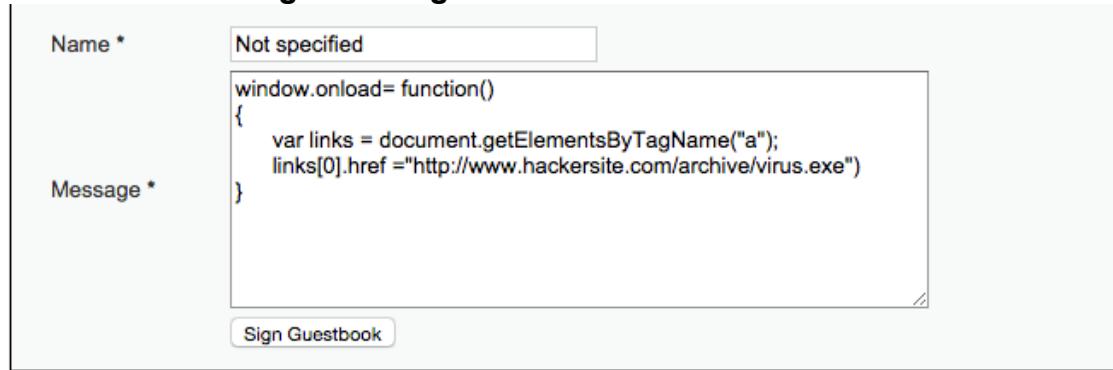
**Fuente: OWASP Top 10 Español**

En un ataque informático de Secuencia de Comandos en Sitios Cruzados, se identifican dos modalidades del ataque las cuales son almacenadas y reflejadas. Como su nombre indica un código malicioso es ingresado por el usuario malicioso, dicho código fuente se almacena en el

motor de base de datos y luego cuando otro usuario del sistema ingresa al sitio y visita alguna página donde esos datos son utilizados, el código malicioso es ejecutado sin el consentimiento ni el conocimiento de lo que está sucediendo precisamente porque los datos almacenados contienen el código dañino.

En tal vector de ataque las consecuencias radican en que el usuario, el cual está autenticado en el sistema, puede ser direccionado a un sitio malicioso donde los datos de su sesión van a ser robados. También puede darse el caso de que se altere la página web, cambiando imágenes, colores textos hipervínculos. En la imagen siguiente, se ilustra el proceso utilizado por un atacante al ingresar código fuente dañino en un sitio Web confiable:

**Figura 45 Ingreso de Datos No Confiables**



The screenshot shows a guestbook form with two fields: 'Name \*' and 'Message \*'. The 'Name' field contains the placeholder 'Not specified'. The 'Message' field contains the following JavaScript code:

```

window.onload= function()
{
    var links = document.getElementsByTagName("a");
    links[0].href ="http://www.hackersite.com/archive/virus.exe"
}

```

Below the message input is a 'Sign Guestbook' button.

### Fuente: Propia

En la imagen anterior, se puede observar que donde se debió haber ingresado el mensaje a ser enviado, el atacante ha injectado un código malicioso que se ejecutará cuando alguien visite el sitio por segunda vez. El objetivo del ataque es que luego de que un usuario normal del sistema ha

visitado el sitio, se localizará el primer *link* (vínculo) de la aplicación y se reemplazará el destino de forma tal que descargue un archivo potencialmente dañino para el computador como lo puede ser un virus. El usuario final al seleccionar el link alterado de forma inadvertida descargará el archivo arbitrario infectado en su computadora. Eventualmente el archivo malicioso podrá afectar a otras estaciones de trabajo dentro de la red corporativa o permitir que alguien externo a la organización pueda realizar ejecución remota de comandos o código fuente a fin de ganar acceso en los sistemas.

Por su parte, los vectores de ataque de la Secuencia de Comandos en Sitios Cruzados (XSS) tienen un trasfondo en ataques de *phishing*, puesto que el atacante utiliza la misma aplicación, con el mismo dominio para presentar algún elemento controlado como es el caso de un hipervínculo o cargando un formulario para realizar el cambio de contraseñas y de esta forma robar los credenciales.

El usuario final no tendrá sospechas del ataque pues pareciera ser que los elementos reflejados forman parte de la aplicación. En la siguiente imagen, se aprecia cómo se podría alterar una página con un formulario arbitrario para el robo de información. El atacante en el escenario propuesto ingresa en la caja de texto un formulario previamente desarrollado en el lenguaje HTML que una vez que es interpretado por el navegador, se muestra ante el usuario final. Nótese que todo esto sucede dentro del mismo dominio de la aplicación, es decir, cualquier persona supondría que el elemento desplegado o que el

cambio de contraseña proviene de la misma entidad ya que parece que forma parte de la aplicación.

En tal modelo se engaña al usuario con la intención de que este se formule la idea que la aplicación le está ayudando a prevenir fraude al realizar el cambio de contraseña.

**Figura 46 Alteración de una página web por medio de XSS**

La figura muestra dos capturas de pantalla de un formulario web. El formulario original (izquierda) tiene un campo de texto con placeholder 'What's your name?' y un botón 'Submit'. El formulario alterado (derecha) muestra el mismo diseño pero con contenido malicioso: 'Hello' aparece en el campo de nombre, y 'Login or Register' aparece en el botón 'Submit'. Los campos de 'Username or email' y 'Password' también están presentes.

**Fuente: Propia**

En la imagen mostrada anteriormente, se puede llegar a la conclusión de que todo texto incluido por el usuario en la caja de texto respectiva es enviado al navegador e interpretado por este. Esto quiere decir por ejemplo que si el usuario en lugar de enviar el nombre envía una etiqueta en HTML, esta será mostrada en el navegador. Tenemos aquí un claro ejemplo de la falta de validación de datos de entrada.

Tal falencia de validación queda plasmada en la imagen siguiente, nótese que el segmento de código mostrado procesa un parámetro

denominado Id, el cual luego es asignado a un control etiqueta (*label*) el cual se encarga de mostrarlo en pantalla.

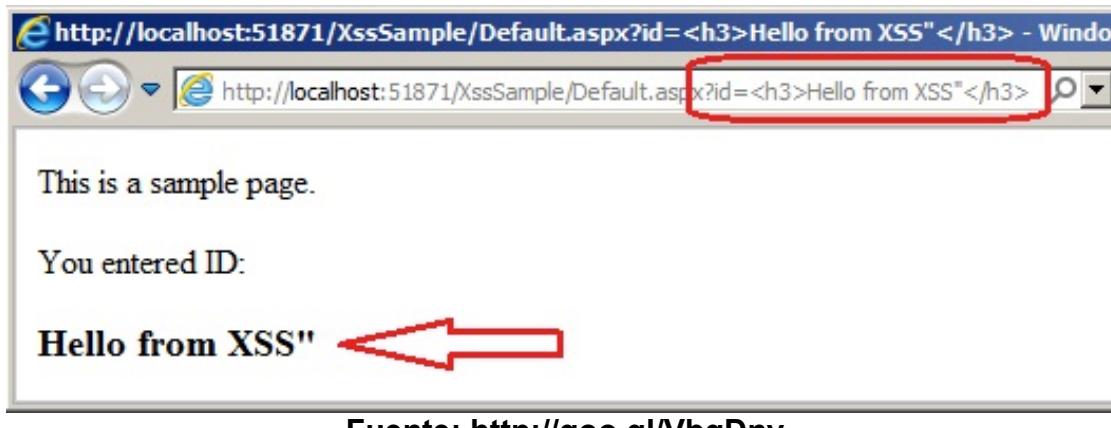
**Figura 47 Código fuente vulnerable a XSS**

```
protected void Page_Load(object sender, EventArgs e)
{
    string id = Request.QueryString["id"] as string;
    if (id == null)
    {
        lblId.Text = "NA";
    }
    else
    {
        lblId.Text = id;
    }
}
```

Fuente: <http://goo.gl/VbgDnv>

Esto significa por ejemplo que un potencial usuario malicioso altera, de forma arbitraria, el parámetro que está siendo enviado, a fin de determinar si es posible que la etiqueta enviada pueda ser interpretada. En la imagen siguiente es posible ver que el atacante envía un texto que al ser interpretado en el navegador, y debido a las etiquetas HTML que contiene, mostrará un título de tercer orden denominado h1.

**Figura 48 Materialización de un ataque de Secuencia de Comandos entre Páginas**

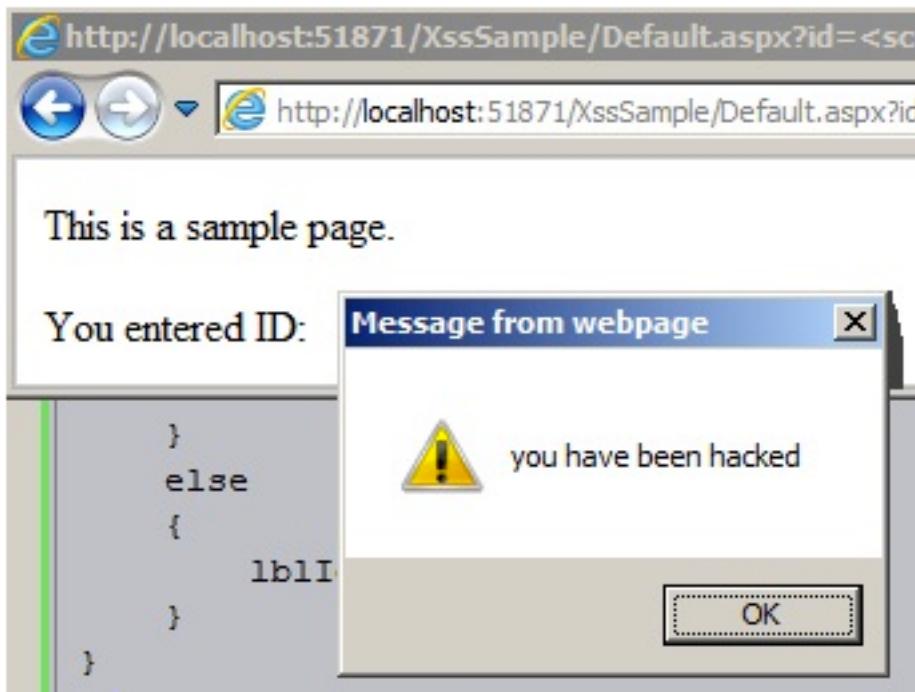


Fuente: <http://goo.gl/VbgDnv>

Una vez que el atacante es capaz de confirmar que la aplicación no valida los datos de entrada sino que los envía al navegador tal cual fueron introducidos, este podrá entonces enviar un tipo de texto potencialmente dañino, en el sentido de que contiene código JavaScript arbitrario, el cual puede redireccionar al usuario a un dominio dañino, provocar la descarga de algún virus informático, provocar la actualización involuntaria de algún estado en la aplicación y cambiar en sí toda la estructura de la página.

Tomando como base el mismo ejemplo, el atacante entonces envía código que se procesa del lado del cliente (formalmente JavaScript); el cual que será ejecutado en el navegador. En principio, la mayoría de navegadores modernos presenta soporte nativo a JavaScript como lenguaje estándar, esto facilita ampliamente la difusión del problema.

**Figura 49 Envío de código JavaScript en la aplicación**



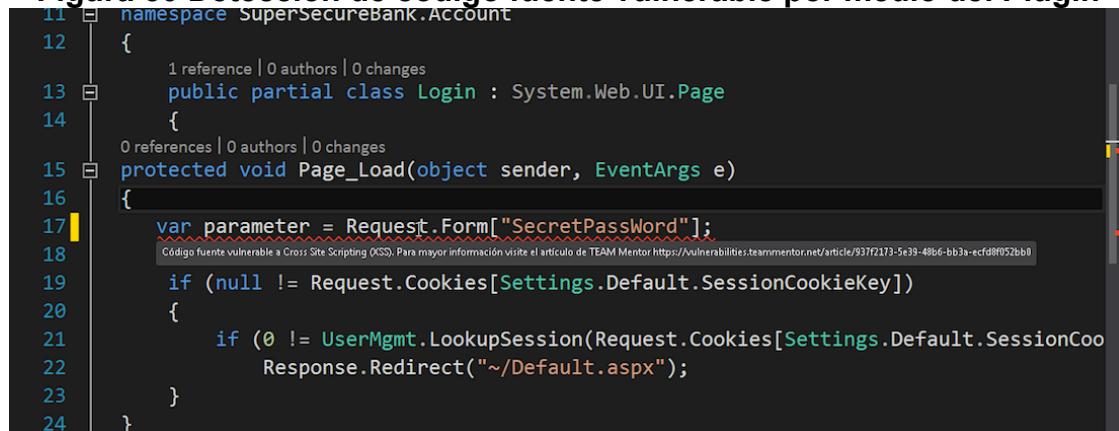
Fuente: <http://goo.gl/VbgDnv>

La recomendación emitida para disminuir el riesgo de Secuencia de Comandos entre Páginas (XSS) tiene su fundamento en la validación de los datos contra una lista de valores aceptables. Se busca de tal forma rechazar datos de entrada que no son válidos o que no se ajustan a lo esperado por la aplicación.

El prototipo propuesto busca detectar estos patrones de código donde no hay una plena validación de datos de entrada, de forma tal que el desarrollador es alertado por medio de mensajes dentro del IDE siguiendo el mantra de errores en rojo. Inmediatamente el desarrollador tendrá noción del problema así como un link informativo donde poder encontrar mayor

información al respecto. Dicho vínculo apunta a la plataforma TEAM Mentor de la empresa Security Innovation, lo que además permite que clientes potenciales accedan a los demás productos.

**Figura 50 Detección de código fuente vulnerable por medio del Plugín**



```

11  namespace SuperSecureBank.Account
12  {
13      public partial class Login : System.Web.UI.Page
14      {
15          protected void Page_Load(object sender, EventArgs e)
16          {
17              var parameter = Request.Form["SecretPassword"];
18              Código fuente vulnerable a Cross Site Scripting (XSS). Para mayor información visite el artículo de TEAM Mentor https://vulnerabilities.teammentor.net/article/937f2173-5e39-48b6-bb3a-ecfd8f052bb0
19              if (null != Request.Cookies[Settings.Default.SessionCookieKey])
20              {
21                  if (0 != UserMgmt.LookupSession(Request.Cookies[Settings.Default.SessionCoo
22                      Response.Redirect("~/Default.aspx");
23                  }
24              }

```

### Fuente: Propia

La retroalimentación dentro del IDE es un activo. En la detección de problemas de Secuencia de Comandos entre Páginas, se muestra una opción donde el desarrollador puede corregir el problema.

Nótese cómo al presionar en la linterna amarilla (el ícono con forma de bombillo) aparece una ventana modal donde se muestra en forma de vista previa como quedaría el cambio luego de aceptar la recomendación de seguridad. En este caso, se aplica la función HtmlEncode con el objetivo de hacer una limpieza de los datos de entrada y remover caracteres especiales que tiene un significado en HTML (Codificación de HTML). El hipervínculo mostrado señala otras recomendaciones más robustas en el momento de resolver el problema.

**Figura 51 Vista Previa de la corrección del problema de XSS**

```

15 0 references | 0 authors | 0 changes
16 protected void Page_Load(object sender, EventArgs e)
17 {
18     var parameter = Request.Form["SecretPassWord"];
19     Solucionar Problema de seguridad
20     {
21         if (0 != Us...
22             Respons...
23     }
24 }

```

A tooltip titled "Solucionar Problema de seguridad" is displayed over the line of code "var parameter = Request.Form["SecretPassWord"];". The tooltip contains the following text:

```

...
using SuperSecureBank.Properties;
using System.Text.RegularExpressions;
namespace SuperSecureBank.Account
...
var parameter = Server.HtmlEncode(Request.Form["SecretPassWord"]);

```

The tooltip also includes the text "SessionCoo" on the right side.

### Fuente: Propia

En el momento preciso de aceptar la recomendación brindada por el *plugin*, presionando la opción del menú que dice Solucionar Problema de Seguridad, la recomendación quedará en firme y el segmento de código será corregido de forma transparente para el desarrollador.

Justo en el momento de aceptar la recomendación que el componente de seguridad ha mostrado, el mensaje de error desaparece debido a que el cambio respectivo fue aplicado. Tal comportamiento de puede observar en la siguiente ilustración.

**Figura 52 Corrección del problema de seguridad en el código fuente**

```

15 0 references | 0 authors | 0 changes
16 protected void Page_Load(object sender, EventArgs e)
17 {
18     var parameter = Server.HtmlEncode(Request.Form["SecretPassword"]);
19     if (null != Request.Cookies[Settings.Default.SessionCookieKey])
20     {
21         if (0 != UserMgmt.LookupSession(Request.Cookies[Settings.Default.SessionCoo
22             Response.Redirect("~/Default.aspx");
23     }
24 }
25

```

### Fuente: Propia

#### 1.3.4.8 Configuración Incorrecta de seguridad

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 tal eslabón e incluso asegurar que todos los eslabones en la cadena de los 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 programación 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.

En la imagen siguiente, se detalla el modelo de riesgo y el impacto para la organización en cuanto a las configuraciones incorrectas de seguridad. Se puede notar que la incidencia de esta vulnerabilidad es alta, gran número de

aplicaciones transaccionales tienen alguna configuración incorrecta de seguridad la cual puede estar asociada a mensajes de error ricos en información técnica acerca de la tecnología subyacente.

**Figura 53 Configuración de Seguridad Incorrecta Modelo de Riesgo**



**Fuente: OWASP Top 10 Español**

La configuración incorrecta de seguridad es fácil de percibir por medio de mensajes de error y excepciones en el código fuente no manejadas. A menudo los desarrolladores hacen uso del mecanismo proporcionado por la plataforma para manejar errores en tiempo de ejecución. No obstante, al enviar el detalle del error al usuario final, se corre el riesgo de que tal error contenga información relevante para un usuario malicioso y le de así herramientas para comprometer el sistema. Tales errores involucran información acerca del motor de base de datos adyacente así como indicarle que otros ataques informáticos pueden ser perpetrados.

En la siguiente ilustración, se muestra un mensaje de error emitido por la aplicación web; la cual en el detalle del error, tal como se aprecia, brinda información acerca de la base de datos, el servidor web y la versión donde dicha aplicación está siendo hospedada.

### **Figura 54 Configuración de seguridad incorrecta por medio de mensajes de error**

---

Server Error in '/' Application.

---

*All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.*

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

**Exception Details:** System.Data.SqlClient.SqlException: All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

**Source Error:**

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

**Stack Trace:**

```
[SqlException (0x80131904): All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.]
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +392
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) +815
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataIsReady) +253
System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +61
System.Data.SqlClient.SqlDataReader.get_MetaData() +138
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString) +6738869
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task& task, BoxByRef壮大)
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, UInt32 timeoutHandle, Task`1 completionTask) +107
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +288
System.Data.SqlClient.SqlCommand.ExecuteReader() +302
SqlInjection.FrmProducts.Page_Load(Object sender, EventArgs e) +372
System.Web.UI.Control.LoadRecursive() +71
System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +3178
```

---

**Version Information:** Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.34237

### **Fuente: Propia**

En este caso particular, el sistema no muestra errores genéricos, es decir un mensaje informativo indicándole al usuario final que un error inesperado ha ocurrido y que el administrador del sitio ha sido advertido, al contrario se muestra información técnica detallada que puede ser utilizada para ganar acceso al sistema al explotar una vulnerabilidad en la plataforma específica. El detalle del error además muestra las versiones del servidor web y de la plataforma usada por el sistema.

Un manejo eficiente de mensajes de error así como evitar mostrar detalles técnicos de las excepciones en tiempo de ejecución ayudan sustancialmente a mitigar que se muestre algún dato relevante.

#### 1.3.4.9 Exposición de datos sensibles

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 se den pérdidas millonarias tanto para los clientes como para las instituciones financieras como para la empresa víctima del ataque.

Se hace importante identificar cuáles 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 cuáles 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.

**Figura 55 Exposición de datos sensibles Modelo de Riesgo**



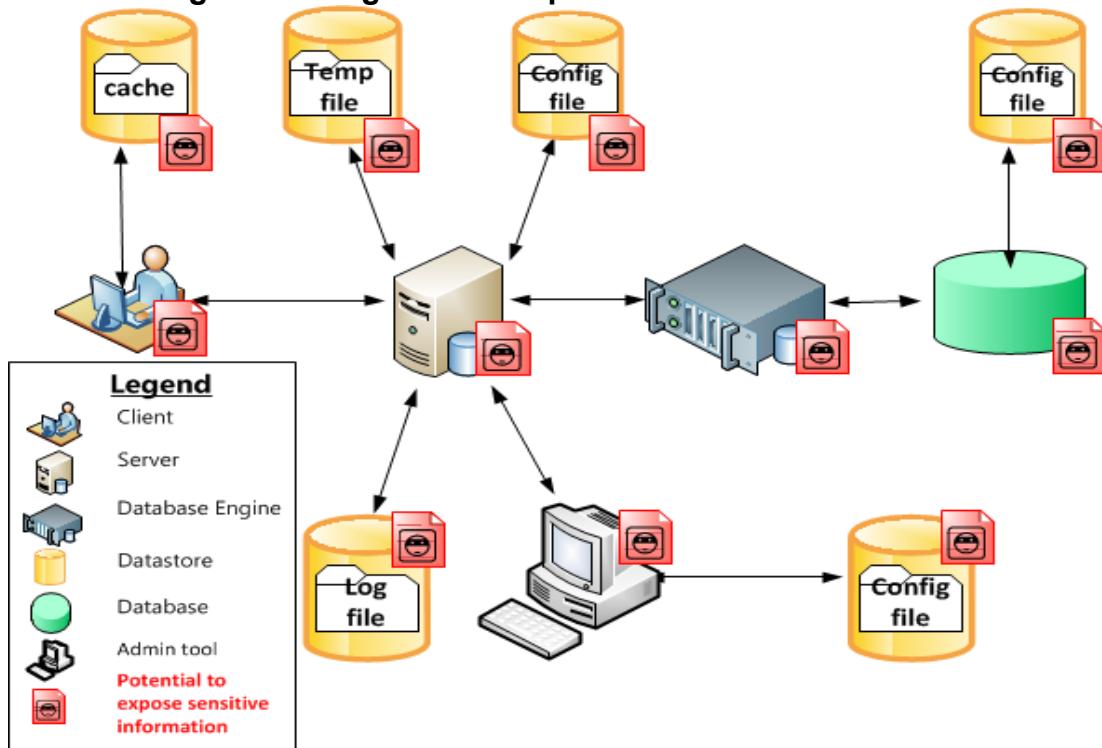
**Fuente:** OWASP Top 10 2013 Español.

Microsoft (2010) en un documento titulado Quick Security Reference: Exposure of Sensitive Information publicado el 05 de noviembre de 2010 establece que:

La falta de protección adecuada de datos sensibles puede ser difícil de reconocer en el desarrollo de aplicaciones, pero es una debilidad común utilizada por los atacantes para robar información sensible. La información sensible se define como: Información privilegiada o reservada que, si comprometida a través de la alteración, la corrupción, la pérdida, mal uso o la divulgación no autorizados, podría causar daños graves a la organización o persona ser dueño de ella .

En dicho artículo, Microsoft muestra el siguiente diagrama donde presenta los diferentes lugares donde se halla información sensible, la cual puede ser explotada por un usuario malicioso.

**Figura 56 Diagrama de exposición de datos sensibles**



Fuente: <http://goo.gl/IONvSK>

En este diagrama de arquitectura, es importante notar que en todos los elementos de arquitectura que forman parte de un sistema transaccional, se maneja o existe información confidencial y si no existe un manejo adecuado y eficiente de estos componentes entonces un atacante podrá potencialmente tener acceso a dicha información.

#### 1.4 Pruebas

Las pruebas en el *software* son un elemento fundamental porque contribuyen activamente en el mejoramiento de la calidad de las aplicaciones

que son implementadas. Realizar pruebas en el *software* permite encontrar errores en una etapa temprana antes de que el sistema sea colocado en un ambiente de producción donde será más costoso su direccionamiento.

Sommerville (2011) acota:

Las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el *software*, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del sistema.

El proceso de prueba tiene dos metas distintas:

1. Demostrar al desarrollador y al cliente que el *software* cumple con los requerimientos. En el caso del *software* personalizado, significa que en el documento de requerimientos debe haber, por lo menos, una prueba por cada requerimiento.
2. Encontrar situaciones donde el comportamiento del *software* sea incorrecto, indeseable o no esté de acuerdo con su especificación. Tales situaciones son consecuencia de defectos en el *software*. La prueba de defectos tiene la finalidad de erradicar el comportamiento indeseable del sistema, como caídas del sistema, interacciones indeseadas con otros sistemas, cálculos incorrectos y corrupción de datos. (p. 206).

Esposito & Saltarello (2009) también brindan su punto de vista acerca de las pruebas de *software* al afirmar que:

Las pruebas ocurren en varios niveles. Usted tiene pruebas de unidad para determinar que los componentes individuales del software cumplen con los requerimientos funcionales. Usted tiene pruebas de

integración para determinar que el *software* se acopla en el ambiente y en la infraestructura y cuando dos o más componentes trabajan bien juntos. Finalmente usted tiene pruebas de aceptación para determinar cuando el sistema terminado cumple con los requerimientos del cliente. Las pruebas de unidad y de integración pertenecen al equipo de desarrollo y tienen como objetivo hacer que el equipo se sienta confiado de la calidad del software. (p. 98).

#### **1.4.1 Pruebas Unitarias**

Las pruebas unitarias o pruebas de unidad tienen como finalidad detectar errores en el código fuente. Tales mecanismos permiten que el desarrollador pruebe su propio código fuente a fin de que encuentre errores de lógica.

Sommerville (2011) refiriéndose a las pruebas unitarias indica que:

Las pruebas de unidad son el proceso de probar componentes del programa como métodos o clases de objetos. Las funciones o métodos individuales son el tipo más simple de componente. Las pruebas deben llamarse para dichas rutinas con diferentes parámetros de entrada. (p.211).

De forma complementaria, los autores Esposito & Saltarello (2009) brindan un enfoque acerca de las pruebas de unidad al indicar que:

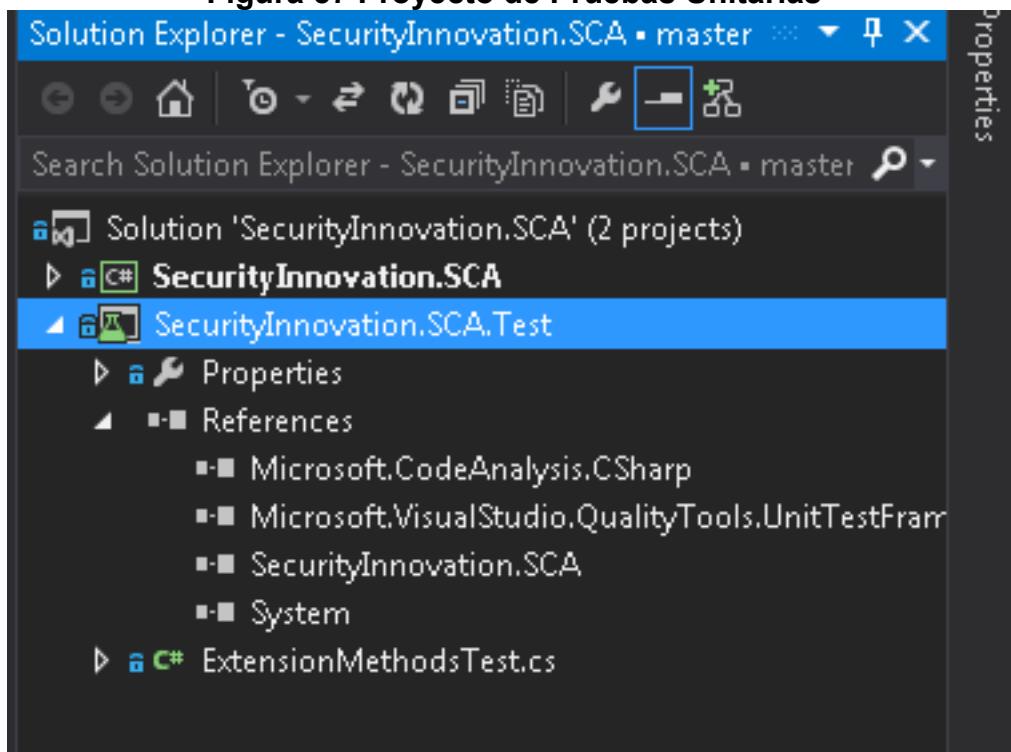
Las pruebas de unidad verifican que unidades individuales de código están funcionando adecuadamente. Una prueba de unidad es la parte más pequeña del software que se puede probar, típicamente un método. (p. 100).

#### 1.4.2 Pruebas Unitarias para el Prototipo

Para el prototipo funcional propuesto, se han definido pruebas unitarias para verificar que los métodos extendidos funcionen de forma correcta.

En la siguiente imagen, se ilustra el proyecto de pruebas unitarias creado para verificar que los métodos implementados no contienen errores de lógica.

Figura 57 Proyecto de Pruebas Unitarias



Fuente: Propia

Seguidamente, se muestra un escenario de pruebas unitarias desarrollado para realizar las pruebas respectivas a los métodos extendidos implementados y que facilitan la lectura del código fuente y la programación de este.

Figura 58 Escenario de Prueba de Unidad

```

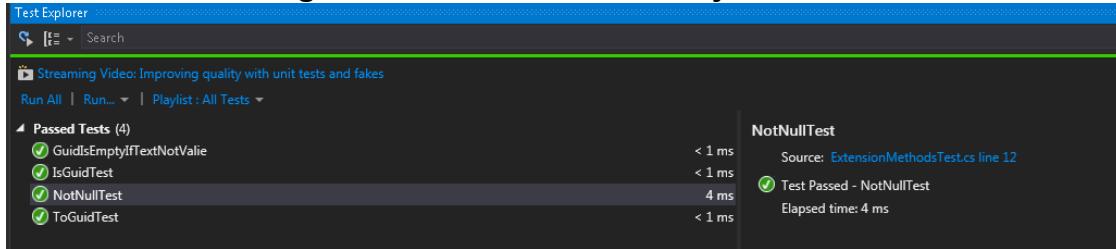
5  namespace SecurityInnovation.SCA.Test
6  {
7      [TestClass]
8      public class StringExtensionMethodsTest
9      {
10         [TestMethod]
11         public void NotNullTest()
12         {
13             var temp = "Hello,World";
14             Assert.IsTrue(temp.NotNull());
15         }
16
17         [TestMethod]
18         public void IsGuidTest()
19         {
20             var tempGuid = "FB1D7409-AFE4-44AB-ACB1-337BC13FD8C0";
21             Assert.IsTrue(tempGuid.IsGuid());
22         }
23
24         [TestMethod]
25         public void ToGuidTest()
26         {
27             var tempGuid = "FB1D7409-AFE4-44AB-ACB1-337BC13FD8C0";
28             var result = tempGuid.ToGuid();
29             Assert.IsTrue(result != Guid.Empty);
30         }
31
32         [TestMethod]
33         public void GuidIsEmptyIfTextNotValue()
34         {
35             var tempGuid = "NOQUIT";
36             var result = tempGuid.ToGuid();
37             Assert.IsTrue(result == Guid.Empty);
38         }
39     }
40 }
```

Fuente: Propia

En el momento de ejecutar las pruebas de unidad dentro del ambiente integrado se puede observar que los métodos implementados no contienen errores puesto que las pruebas de unidad previamente desarrolladas han sido

ejecutadas de forma exitosa. El color verde significa que las pruebas de unidad se han ejecutado sin errores.

**Figura 59 Pruebas unitarias ejecutadas**



Fuente: Propia

### 1.4.3 Pruebas Funcionales del Prototipo

En esta sección, se presentan las pruebas funcionales que han sido aplicadas al prototipo con el objetivo de validar que los módulos funcionan tal como fueron definidos.

#### 1.4.3.1 Instalación del componente dentro de Visual Studio.NET

La extensión de seguridad podrá ser instalada dentro de Visual Studio .NET 2012 o superior. El instalador se distribuye en forma de un componente con extensión .VSIX (formato propietario de Microsoft para las extensiones). A continuación, se muestra el escenario de pruebas propuesto.

**Cuadro 11 Instalación del componente en Visual Studio.NET**

Escenario de Pruebas	
Número de Prueba	Prueba 1
Título	Instalación del componente en Visual Studio.NET.
Descripción de la Prueba	Esta prueba verifica que el plugin se instale correctamente dentro de Visual Studio y pueda ser utilizado correctamente.
Precondiciones	Tener una instancia de Visual Studio 2012 o superior instalado en la computadora donde se utilizará el componente.
Pasos	<ol style="list-style-type: none"> <li>1. Localizar el instalador de la extensión de seguridad, el cual es un archivo con la extensión .vsix.</li> <li>2. Aceptar los términos y condiciones definidos.</li> <li>3. Dar click en el botón de aceptar una vez que el componente ha sido instalado.</li> <li>4. Dentro de Visual Studio, ir al menú de Herramientas, seleccionar el sub menú Extensiones y Actualizaciones (Extensions and Updates).</li> <li>5. Dentro de la ventana desplegada, se debe buscar el componente el cual se llama SecurityInnovation.SCA</li> </ol>
Resultados Esperados	La extensión de seguridad figura dentro de la lista de extensiones disponibles.

Fuente: Propia

#### **1.4.3.2 Deshabilitar el componente dentro de Visual Studio.NET**

En este escenario de pruebas, se busca poder deshabilitar de forma temporal el componente dentro del ambiente integrado. En el momento en que se desabilite el componente, la funcionalidad incluida en el mismo detendrá su operación. El usuario deberá reiniciar Visual Studio .NET (con ayuda del asistente que aparecerá luego de deshabilitar el componente). En el siguiente cuadro, se muestra el escenario de pruebas.

**Cuadro 12 Deshabilitar el componente dentro de Visual Studio.NET**

Escenario de Pruebas	
Número de Prueba	Prueba 2
Título	Deshabilitar el componente dentro de Visual Studio.NET
Descripción de la Prueba	En esta prueba se verifica que la extensión pueda ser deshabilitada y de esta forma que no se utilice dentro de Visual Studio.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio 2012 o posterior.
Pasos	<p>1.Dentro de Visual Studio, ir al menú de Herramientas, seleccionar el sub menú Extensiones y Actualizaciones (Extensions and Updates).</p> <p>2.Dentro de la ventana desplegada, se debe buscar el componente el cual se llama SecurityInnovation.SCA</p> <p>3.Una vez ubicada la extensión en la ventana desplegada, se podrá seleccionar la opción para deshabilitar el componente.</p> <p>4.Se debe reiniciar Visual Studio para el cambio tenga efecto</p>
Resultados Esperados	Una vez reiniciado Visual Studio .NET la extensión debe estar deshabilitada

Fuente: Propia

#### **1.4.3.3 Desinstalación del componente**

Dentro del entorno integrado, el desarrollador podrá desinstalar la extensión de seguridad y de esta forma dejar el ambiente como estaba antes de la instalación de este. La instalación y desinstalación de la extensión no tiene ningún efecto de lado, es decir, si se han aceptado las recomendaciones propuestas y se desinstala el componente no habrá ningún error en tiempo de compilación ni ejecución. En el siguiente caso de pruebas, se muestra la secuencia de pasos a seguir.

**Cuadro 13 Desinstalación del componente**

Escenario de Pruebas	
Número de Prueba	Prueba 3
Título	Desinstalación del componente dentro de Visual Studio.NET
Descripción de la Prueba	En esta prueba se verifica que la extensión pueda ser desinstalada y de esta forma que no se utilice dentro de Visual Studio.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio 2012 o posterior.
Pasos	<p>1.Dentro de Visual Studio, ir al menú de Herramientas, seleccionar el sub menú Extensiones y Actualizaciones (Extensions and Updates).</p> <p>2.Dentro de la ventana desplegada, se debe buscar el componente el cual se llama SecurityInnovation.SCA</p> <p>3.Una vez ubicada la extensión en la ventana desplegada, se podrá seleccionar la opción para desinstalar el componente.</p> <p>4.Se debe reiniciar Visual Studio para el cambio tenga efecto</p>
Resultados Esperados	Una vez reiniciado Visual Studio .NET la extensión debe estar desinstalada, es decir, no debe figurar dentro de la lista de Extensiones y Actualizaciones.

Fuente: Propia

#### **1.4.3.4 Detección de Vulnerabilidades de Secuencia de Comandos entre Páginas**

En esta prueba, se verifica que la extensión de seguridad pueda detectar problemas relacionados con Secuencia de Comandos entre Páginas (XSS). Es necesario identificar una aplicación web vulnerable a este riesgo y proceder a abrir el proyecto con Visual Studio, el cual hará uso del *plugin* para realizar la detección del problema. El componente brindará de este modo una solución del código fuente con el objetivo de prevenir la vulnerabilidad.

**Cuadro 14 Prueba de detección de Secuencias de comandos entre páginas.**

Escenario de Pruebas	
Número de Prueba	Prueba 4
Título	Detección de Vulnerabilidades de Secuencia de Comandos entre Páginas
Descripción de la Prueba	Esta prueba tiene como fundamento que la extensión de seguridad pueda detectar vulnerabilidades de Secuencia de Comandos entre Páginas, en una aplicación Web vulnerable, y poder solucionar el problema de seguridad por medio de las herramientas de reingeniería que el componente proporciona.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio .NET 2012 o posterior
Pasos	<ol style="list-style-type: none"> <li>1. Identificar una aplicación Web, desarrollada en C#, vulnerable a Secuencia de Comandos entre páginas.</li> <li>2. Explorar la vulnerabilidad de forma manual u automática para verificar que efectivamente el software es vulnerable.</li> <li>3. Abrir el proyecto vulnerable con Visual Studio.NET</li> <li>4. El componente identificará los segmentos del código fuente vulnerable por medio de mensajes de error.</li> <li>5. El desarrollador analiza el detalle del error y visita los links mostrados en el detalle del mensaje.</li> <li>6. Posteriormente el desarrollador presiona la linterna amarilla donde puede solucionar el problema del código fuente.</li> <li>7. El componente realiza la reingeniería para solucionar el problema de seguridad.</li> </ol>
Resultados Esperados	Desde la perspectiva del código fuente, la vulnerabilidad ha sido solucionada y la aplicación Web no es vulnerable.

Fuente: Propia

#### **1.4.3.5 Detección de Vulnerabilidades de Inyección de SQL**

Esta prueba tiene como objetivo la detección de vulnerabilidades de inyección de SQL en una aplicación web. El escenario de pruebas propone una aplicación donde se puedan manipular los datos de entrada de forma tal que se puedan introducir sintaxis SQL para provocar un comportamiento distinto al definido por la aplicación. La extensión de seguridad, utilizando

técnicas de análisis estático de código, detecta los problemas de inyección y muestra un mensaje informativo en conjunto con la solución del problema.

En el siguiente cuadro, se especifica el escenario de prueba:

**Cuadro 15 Detección de vulnerabilidades de Inyección de SQL**

Escenario de Pruebas	
Número de Prueba	Prueba 5
Título	Detección de Vulnerabilidades de Inyección de SQL
Descripción de la Prueba	Esta prueba tiene como fundamento que la extensión de seguridad pueda detectar vulnerabilidades de Inyección de SQL, en una aplicación Web vulnerable, y poder solucionar el problema de seguridad por medio de las herramientas de reingeniería que el componente proporciona.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio .NET 2012 o posterior.
Pasos	<ol style="list-style-type: none"> <li>1. Identificar una aplicación Web, desarrollada en C#, vulnerable a Inyección de SQL.</li> <li>2. Explorar la vulnerabilidad de forma manual u automática para verificar que efectivamente el software es vulnerable. Manipular datos de entrada a fin de detectar la vulnerabilidad.</li> <li>3. Abrir el proyecto vulnerable con Visual Studio.NET</li> <li>4. El componente identificará los segmentos del código fuente vulnerable por medio de mensajes de error.</li> <li>5. El desarrollador analiza el detalle del error y visita los links mostrados en el detalle del mensaje.</li> <li>6. Posteriormente el desarrollador presiona la linterna amarilla donde puede solucionar el problema del código fuente.</li> <li>7. El componente realiza la reingeniería para solucionar el problema</li> </ol>
Resultados Esperados	Desde la perspectiva del código fuente, la vulnerabilidad de Inyección de SQL ha sido solucionada y la aplicación Web no es vulnerable. Al intentar perpetrar el vector de ataque éste debe fallar.

Fuente: Propia

#### **1.4.3.6 Detección de Vulnerabilidades de Autenticación Rota y Manejo de Sesiones**

El manejo incorrecto de sesiones tiene lugar cuando, por medio del lenguaje JavaScript u otro lenguaje del lado del cliente, se pueden modificar variables almacenadas en sesión. Tales variables dependiendo del negocio pueden tener un contexto sensible pues pueden almacenar información acerca del monto total a cancelar, si el usuario es o no administrador de la plataforma, algún estado, el precio de algún artículo, entre otros.

El escenario de pruebas descrito permite identificar cuándo las variables en sesión pueden ser accedidas por estos programas y alertar al usuario. En el siguiente cuadro, se muestra el caso de pruebas propuesto:

**Cuadro 16 Detección de vulnerabilidades de autenticación rota**

Escenario de Pruebas	
Número de Prueba	Prueba 6
Título	Detección de Vulnerabilidades de Autenticación Rota y Manejo de Sesiones.
Descripción de la Prueba	Esta prueba tiene como fundamento que la extensión de seguridad pueda detectar vulnerabilidades de Manejo incorrecto de sesiones, en una aplicación Web vulnerable,y poder solucionar el problema de seguridad por medio de las herramientas de reingeniería que el componente proporciona.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio .NET 2012 o posterior.
Pasos	<ol style="list-style-type: none"> <li>1. Identificar una aplicación Web,desarrollada en C#, que presente un manejo incorrecto de variables de sesión.</li> <li>2.Explotar la vulnerabilidad de forma manual u automática para verificar que efectivamente el software es vulnerable.Manipular las variables en sesión por medio de código JavaScript.</li> <li>3.Abrir el proyecto vulnerable con Visual Studio.NET</li> <li>4.El componente identificará los segmentos del código fuente vulnerable por medio de mensajes de error.</li> <li>5.El desarrollador analiza el detalle del error y visita los links mostrados en el detalle del mensaje.</li> <li>6.Posteriormente el desarrollador presiona la linterna amarilla donde puede solucionar el problema del código fuente.</li> <li>7. El componente realiza la reingeniería para solucionar el problema de seguridad.</li> <li>8. El desarrollador publica la aplicación Web nuevamente en el servidor y trata de explotar la vulnerabilidad nuevamente.</li> <li>9.La vulnerabilidad ha sido solucionada.</li> </ol>
Resultados Esperados	Las variables en sesión son protegidas correctamente de forma tal que no puedan ser manipuladas por código JavaScript o cualquier otro lenguaje del lado del cliente.

Fuente: Propia

#### 1.4.3.7 Detección de Vulnerabilidades de Exposición de datos Sensibles

En este escenario de prueba, se pretende que la extensión de seguridad pueda identificar cuando se exponen datos sensibles en la aplicación. Una vez aceptada la solución propuesta por la extensión, el problema de seguridad debe haberse mitigado.

**Cuadro 17 Prueba Exposición de datos sensibles**

Escenario de Pruebas	
Número de Prueba	Prueba 7
Título	Detección de vulnerabilidades de Exposición de datos sensibles.
Descripción de la Prueba	Esta prueba tiene como fundamento que la extensión de seguridad pueda detectar vulnerabilidades de Exposición de datos sensibles, en una aplicación Web vulnerable,y poder solucionar el problema de seguridad por medio de las herramientas de reingeniería que el componente proporciona.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio .NET 2012 o posterior.
Pasos	<ol style="list-style-type: none"> <li>1. Identificar una aplicación Web,desarrollada en C#, que presente un manejo incorrecto de errores, que puedan revelar información sensible.</li> <li>2.Explotar la vulnerabilidad de forma manual u automática para verificar que efectivamente el software es vulnerable.Manipular los datos de entrada para inducir al error.</li> <li>3.Abrir el proyecto vulnerable con Visual Studio.NET</li> <li>4.El componente identificará los segmentos del código fuente vulnerable por medio de mensajes de error.</li> <li>5.El desarrollador analiza el detalle del error y visita los links mostrados en el detalle del mensaje.</li> <li>6.Posteriormente el desarrollador presiona la linterna amarilla donde puede solucionar el problema del código fuente.</li> <li>7. El componente realiza la reingeniería para solucionar el problema de seguridad.</li> <li>8. El desarrollador publica la aplicación Web nuevamente en el servidor y trata de explotar la vulnerabilidad nuevamente.</li> </ol>
Resultados Esperados	Los errores desplegados por la aplicación deben contener poca o nula información sensible y debe ser genérico, indicando que un error ocurrió y que el administrador del sistema fué alertado.

Fuente: Propia

#### **1.4.3.8 Detección de Vulnerabilidades de Configuración Incorrecta de Seguridad.**

El objetivo de esta prueba es garantizar que la extensión de seguridad pueda encontrar configuraciones incorrectas de seguridad y reportarlas al desarrollador dentro del mismo entorno de desarrollo integrado, es decir dentro de Visual Studio.NET 2013. Se tiene un sitio web con tales configuraciones incorrectas, las herramientas de diagnóstico del componente detectan el problema el cual es presentado al desarrollador. Este último atiende la recomendación y soluciona el problema.

**Cuadro 18 Detección de configuraciones incorrectas de seguridad**

Escenario de Pruebas	
Número de Prueba	Prueba 8
Título	Detección de vulnerabilidades de Configuración Incorrecta de Seguridad.
Descripción de la Prueba	Esta prueba tiene como fundamento que la extensión de seguridad pueda detectar vulnerabilidades de Configuración Incorrecta de Seguridad tal es el caso de la falta de manejo de excepciones controladas, en una aplicación Web vulnerable,y poder soluciona el problema de seguridad por medio de las herramientas de reingeniería que el componente proporciona.
Precondiciones	Haber instalado la extensión de seguridad en una instancia de Visual Studio .NET 2012 o posterior.
Pasos	<ol style="list-style-type: none"> <li>1. Identificar una aplicación Web,desarrollada en C#, que contenga una configuración incorrecta de seguridad</li> <li>2.Explotar la vulnerabilidad de forma manual u automática para verificar que efectivamente el software es vulnerable.Manipular los datos de entrada para inducir al error.</li> <li>3.Abrir el proyecto vulnerable con Visual Studio.NET</li> <li>4.El componente identificará los segmentos del código fuente vulnerable por medio de mensajes informativos al usuario.</li> <li>5.El desarrollador analiza el detalle del error y visita los links mostrados en el detalle del mensaje.</li> <li>6.Posteriormente el desarrollador presiona la linterna amarilla donde puede solucionar el problema del código fuente.</li> <li>7. El componente realiza la reingeniería para solucionar el problema de seguridad.</li> <li>8. El desarrollador publica la aplicación Web nuevamente en el servidor y trata de explotar la vulnerabilidad nuevamente.</li> <li>9.La vulnerabilidad ha sido solucionada.</li> </ol>
Resultados Esperados	Las excepciones en tiempo de ejecución son manejadas correctamente

Fuente: Propia

## 2. INTERPRETACIÓN DE RESULTADOS

Tal como se indicó en la sección correspondiente al marco metodológico, se han definido la observación y las encuestas realizadas a los miembros del equipo de ingeniería de la empresa Security Innovation como instrumentos para la recolección de datos, los cuales brindan un curso de acción en la elaboración del prototipo funcional.

En este capítulo, se analizan los resultados recopilados por medio de la utilización de estos instrumentos.

### 2.1 Observación como instrumento de análisis

La empresa Security Innovation se encuentra bien posicionada en un mercado emergente: seguridad de aplicaciones. A lo largo de los años, la empresa se ha enfocado en el tema de educación, proporcionando herramientas y mejores prácticas en la industria a fin de solventar esa necesidad recurrente de cambiar de enfoques para evitar ser víctima de algún incidente en seguridad informática.

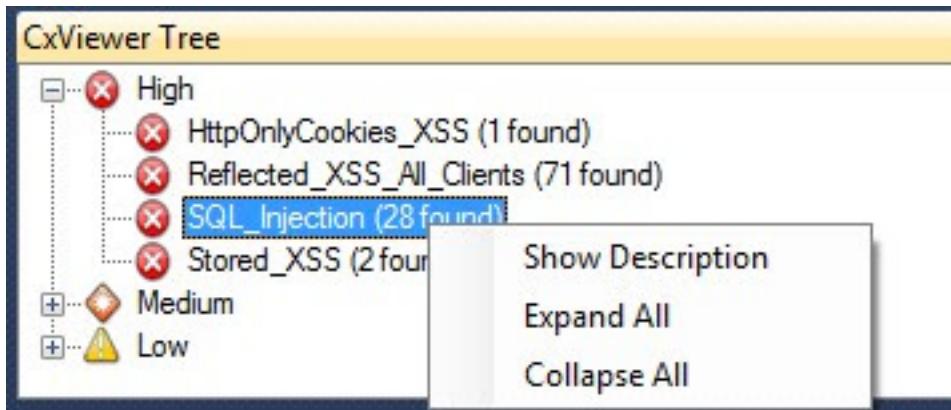
El tiempo no transcurre en vano, la diversidad de clientes de la compañía en términos de sectores constata que no hay ningún sector en la industria que se encuentre fuera del alcance de los delitos informáticos. De esta forma, se ha observado los diferentes productos que la empresa tiene para ofrecer y cómo estos productos son comercializados de manera conjunta con otras soluciones en la industria.

Desde la perspectiva del desarrollo del *software*, las empresas que buscan las soluciones de la compañía tienen un denominador común: ellos tienen dentro de su organización alguna herramienta, de terceras partes, que hace análisis estático o dinámico de código y les gustaría que los productos que ofrece la empresa se integren con esas herramientas existentes. Cabe mencionar que tales herramientas de terceros, distribuidas a nivel corporativo, tienen un costo económico considerable.

Así mismo una empresa pequeña que no puede optar por las herramientas de terceros, no podrán utilizar o consumir el contenido ni las mejores prácticas de forma integrada. Es aquí donde nace la necesidad de que la empresa cuente con su propia herramienta que hace análisis estático de código.

En la siguiente imagen, se ilustra como el contenido de TEAM Mentor, uno de los productos de la empresa Security Innovation, son mostrados dentro de Visual Studio .NET; pero es necesario que la herramienta de terceros denominada Checkmarx, que es la que hace el análisis estático de código, esté debidamente instalada y con la licencia respectiva.

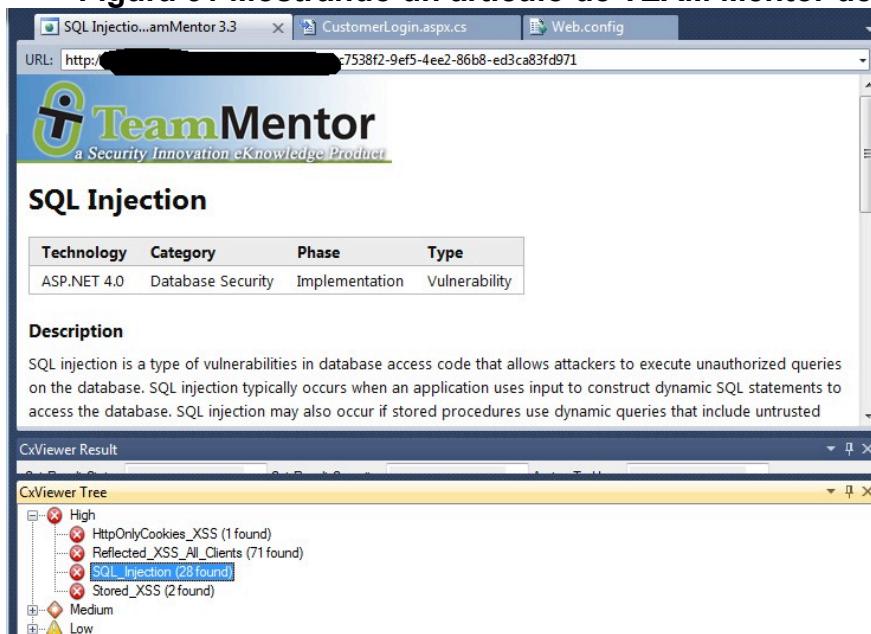
**Figura 60 Checkmarx mostrando vulnerabilidades dentro de Visual Studio**



Fuente: <http://goo.gl/zdZg9u>

Como parte de la integración con la herramienta de terceros, una vez que se seleccione la opción Show Description, una mejor práctica desarrollada por la empresa es mostrada dentro del ambiente de desarrollo.

**Figura 61 Mostrando un artículo de TEAM Mentor dentro del IDE**



Fuente: <http://goo.gl/zdZg9u>

Por medio del proceso de observación, se puede notar que la integración con herramientas de terceros juega un papel fundamental en la comercialización de los productos y se crean alianzas estratégicas. Se puede entonces indicar que el modelo funciona muy bien para aquellas empresas que ya tienen dentro de sus actividades diarias, herramientas de terceras partes para revisar la seguridad del código fuente.

Se comprende además por medio del proceso de observación que tener una herramienta propietaria de la empresa que realice estas tareas sería beneficioso pues los productos se comercializan de forma más unificada. Esto quiere decir que el mismo componente que hace análisis estático de código muestra *links* a mejores prácticas.

### **2.1.1 Observación del estado de la seguridad.**

Afortunadamente la empresa Security Innovation crea de forma conjunta con el Instituto Ponemon, investigaciones para determinar la situación actual de seguridad en las organizaciones. Tales reportes brindan un panorama muy claro acerca de las verdaderas necesidades en las organizaciones, se puede comprender cómo ataques informáticos comunes siguen afectando la infraestructura tecnológica actual.

Al igual que estos reportes también se ha tomado como base otras investigaciones donde se ilustra como vulnerabilidades comunes (mismas que se identifican en el prototipo propuesto) inciden a que se cometan otros delitos informáticos. Una de estas investigaciones consultadas es la creada por el

argentino Cristian Borghello, publicada en el sitio SeguInfo.com y que lleva por nombre “viagra.gob.ar: un asunto de seguridad nacional”<sup>13</sup>. En esta investigación, se muestra cómo se ha podido incluir, en sitios gubernamentales, publicidad engañosa la cual puede pasar inadvertida para el usuario final. Esta modalidad de ataques aprovechan una debilidad existente en un sitio web, el cual perfectamente puede ser gubernamental, con el objetivo de promocionar algún producto específico.

Esta investigación, interesante por su naturaleza, ayuda a comprender que es necesario un enfoque para mejorar y poner en práctica la seguridad de las aplicaciones.

De forma complementaria también se han consultado las investigaciones realizadas por el periodista norteamericano Brian Krebs, el cual mantiene un sitio web muy respetable en asuntos de seguridad informática que lleva por nombre KrebsonSecurity.<sup>14</sup>

Las diversas investigaciones fidedignas proporcionadas por múltiples empresas ayudan considerablemente a comprender el fenómeno actual de la seguridad en el *software*.

---

<sup>13</sup> <http://www.segu-info.com.ar/articulos/107-viagra-gob-ar.htm>

<sup>14</sup> <http://krebsonsecurity.com/>

## 2.2 Resultados de la encuesta

Los resultados que a continuación se presentan han sido recopilados a partir de la retroalimentación brindada por Roman Garber, líder del equipo de desarrollo y Dinis Cruz, arquitecto de seguridad de la empresa. La retroalimentación brindada por estas dos personas es esencial, ya que su experiencia tanto en el conocimiento de los productos de la empresa como en la industria genera mucho valor en el momento de recopilar información. La encuesta ha sido aplicada en el idioma inglés debido a que las personas encuestadas no hablan castellano.

### 2.2.1 Pregunta 1

¿Cuáles considera usted que son los beneficios para la empresa al contar con un plugin para Visual Studio.NET que realice análisis estático de código y que brinde una solución en el código fuente?

Con esta pregunta se busca conocer, desde el punto de vista de los interesados, cuáles son los beneficios que, con base en su criterio, tendría la empresa al contar con este *software*. Principalmente se trata de conocer las opiniones de los miembros de la organización.

Para la recolección de esta respuesta, se ha presentado una caja de texto donde los encuestados brindan sus opiniones. Dentro de los datos recolectados, se identifica que algunos de los beneficios para la organización mostrados en el siguiente cuadro.

### Cuadro 19 Resultados de la pregunta 1

Resultados Recolectados
1.Mejor integración con los productos existentes de la empresa.
2.Disminuir la dependencia de herramientas de terceros.
3.Mejor aceptación por parte de los desarrolladores.
4.Personalización de los productos de la empresa.

**Fuente: Encuesta realizada a empleados de la empresa Security Innovation.**

En este caso, se percibe que habría beneficios significativos para la empresa al contar con el *plugin* propuesto.

#### 2.2.2 Pregunta 2

¿Considera usted que el hecho de no contar con una herramienta propietaria que realice análisis estático de código fuente y provea soluciones de código, es una limitante para incrementar las ventas y crear nuevos productos?

Esta pregunta ha sido elaborada para determinar si la empresa no ha podido aprovechar una ventaja competitiva e incrementar las ventas de los productos a raíz de no poder comercializar productos integrados. A continuación, se presentan los datos recolectados:

**Cuadro 20 Resultados de la Pregunta 2**

Encuestados		
Opciones	Respuestas Obtenidas	Cantidad
Si	50%	1
No	50%	1

**Fuente:** Encuesta realizada a empleados de la empresa Security Innovation.

**Gráfico 3 Resultados de la pregunta número 2**

**¿Considera usted que el hecho de no contar con una herramienta propietaria que realice análisis estático de código fuente y provea soluciones de código, es una limitante para incrementar las ventas y crear nuevos productos?**



**Fuente:** Encuesta realizada a empleados de la empresa Security Innovation.

Tal como se deduce en el gráfico anterior, el 50% de los entrevistados considera que el área de ventas de la empresa no ha podido explotar el potencial de los demás productos que se podrían integrar y comercializar con

la extensión de seguridad propuesta. Para esta pregunta, se habilitó un campo de texto con comentarios donde se puede inferir que las personas encuestadas concuerdan en que una extensión de seguridad, que pueda ser integrada con las demás soluciones de la empresa, ayudaría y facilitaría la comercialización de los productos.

### **2.2.3 Pregunta 3**

¿Cómo considera usted que será el impacto en el área de ventas de los productos de la empresa cuando se disponga de un componente que realice análisis estático de código?

El propósito de esta pregunta es poder determinar el punto de vista de los entrevistados en cuanto al impacto que tendría la extensión de seguridad en las ventas de los demás productos. Puesto que la idea principal del componente es que se integre con los diversos productos que tiene la empresa, conocer la percepción de los entrevistados es muy importante.

A continuación, se presentan los datos recolectados:

**Cuadro 21 Resultado de la pregunta 3**

<b>Encuestados</b>		
<b>Opciones</b>	<b>Respuestas Obtenidas</b>	<b>Cantidad</b>
Bajo	50%	1
Medio	50%	1
Alto	0%	0
No hay impacto	0%	0

**Fuente:** Encuesta realizada a empleados de la empresa Security Innovation.

A continuación, se muestra la representación gráfica de estos resultados:

**Gráfico 4 Resultado de la pregunta 3**

**¿Cómo considera usted que será el impacto en el área de ventas de los productos de la empresa cuando se disponga de un componente que realice análisis estático de código?**



**Fuente:** Encuesta realizada a empleados de la empresa Security Innovation.

Del gráfico y de los datos anteriores, se comprende que las personas encuestadas consideran que el *plugin* propuesto para Visual Studio.NET tiene un impacto en la comercialización de los demás productos, pese a que el 50% de los encuestados considera que el impacto sería bajo, son conscientes de que existe tal impacto positivo.

#### 2.2.4 Pregunta 4

En términos de integración, ¿considera usted que sería fácil integrar el *plugin* de Visual Studio .NET con productos existentes tales como la plataforma TEAM Mentor?

En esta pregunta, se pretende determinar si el componente propuesto sería fácil de integrar con los productos que la empresa ofrece. Esta pregunta tiene relevancia puesto que la complejidad e incapacidad de integrarse fácilmente con las demás soluciones sería un problema en lugar de un valor agregado como se espera.

Esta pregunta consiste de dos opciones (Sí/No) y en vista de que el 100% de los encuestados considera que el *plugin* propuesto sería fácil de integrar con las soluciones que la empresa ofrece (es decir la respuesta fue afirmativa) por tal razón no se grafican los resultados.

#### 2.2.5 Pregunta 5

¿Cree usted que teniendo un *plugin* que realiza análisis estático de código y brindar soluciones en el código fuente puede ayudar a reducir el número de vulnerabilidades en aplicaciones escritas en C#?

La pregunta anterior tiene su fundamento analizar si la seguridad en el código fuente de las aplicaciones web desarrolladas en el lenguaje de programación C#, se vería mejorada, es decir si el *plugin* verdaderamente aporta una oportunidad de mejora en la mitigación de las vulnerabilidades en el código fuente.

En esta pregunta, se han formulado dos opciones de respuesta (Sí/No) y en vista de que el 100% de los entrevistados considera que si habría una mejora en la reducción de vulnerabilidades en el código fuente no se ha presentado ningún gráfico que lo represente.

### **2.2.6 Pregunta 6**

¿Cree usted que tener un *plugin* que realiza análisis estático de código y brinda soluciones en el código fuente puede ayudar a que los desarrolladores entiendan los riesgos de seguridad en el código?

Un enfoque educativo donde los desarrolladores de *software* comprendan las causas que hacen que el programa sea inseguro es vital. Esta pregunta pretende determinar si el análisis estático de código fuente y la retroalimentación que el *plugin* le proporciona al desarrollador dentro del ambiente integrado de desarrollo constituye el pilar para que ellos entiendan los riesgos en el código fuente.

Por tal razón, esta pregunta consta de dos respuestas (Sí/No). En este caso, el 100% de las personas encuestadas indican que el *plugin* sí les ayudaría a los desarrolladores a comprender los riesgos que constituyen el *software* inseguro y a entender las vulnerabilidades. En vista de que la respuesta fue afirmativa en un 100% de los resultados, no se considera necesario mostrar una representación gráfica de tal comportamiento.

### 2.2.7 Pregunta 7

¿Considera usted que el hecho de que las plataformas de desarrollo no presenten por defecto un mecanismo donde brindar retroalimentación en términos de seguridad tiene un impacto negativo en la seguridad del software?

### 2.2.8 Pregunta 8

¿Cuáles considera usted que son los beneficios de una extensión para Visual Studio .NET que brinde diagnóstico a problemas de seguridad y solución a esos problema?

Esta es una pregunta abierta donde se ha pretendido conocer, por parte de los encuestados, cuáles consideran que son los beneficios o el valor agregado que del *plugin* que permita realizar un diagnóstico del código fuente y brindar seguidamente una solución al código fuente, es decir por medio de la reingeniería del código fuente.

A continuación, se muestran los resultados recolectados:

**Cuadro 22 Respuestas de la pregunta 8**

Resultados Recolectados
1.Proporcionar seguridad dentro del ambiente integrado
2.Resolver problemas de seguridad en tiempo real.

**Fuente: Encuesta realizada a empleados de la empresa Security Innovation.**

Tal como se puede apreciar, los encuestados reconocen que habría beneficios sustanciales al implementar un componente con estas características.

### **2.2.9 Pregunta 9**

¿Qué atributos le gustaría a usted tener en un *plugin* de Visual Studio.NET en el momento de brindar retroalimentación dentro del IDE?

El interés fundamental de esta pregunta radica en tomar ventaja del criterio experto de los encuestados a fin de identificar las características u atributos más importantes que debe tener una extensión de seguridad para que genere interés por parte de los desarrolladores.

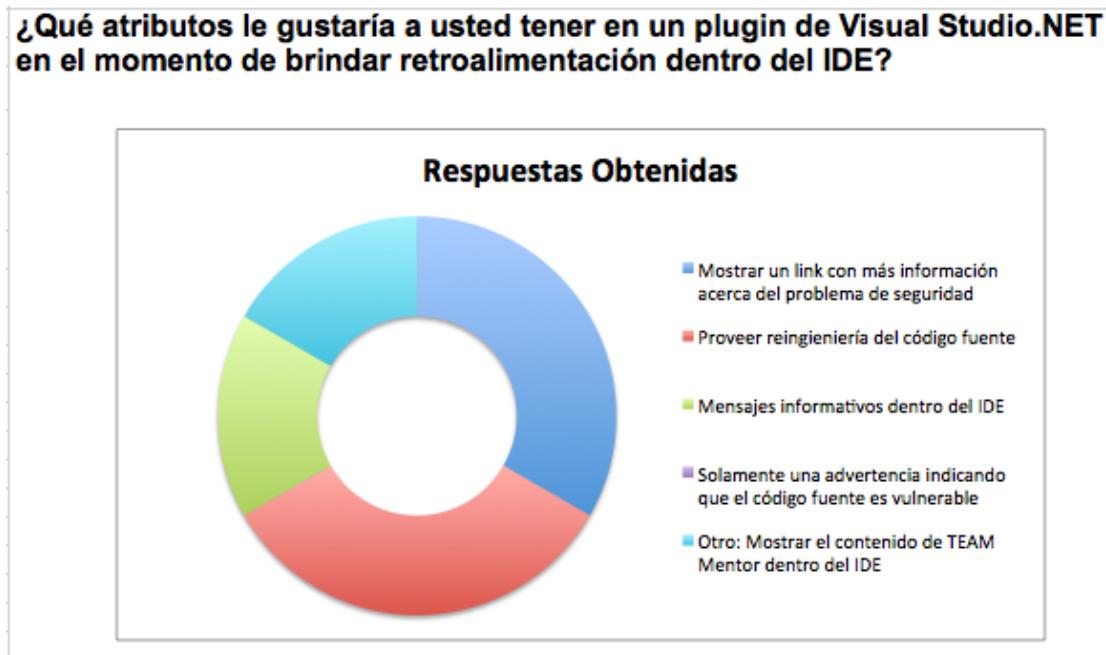
Así mismo se han enumerado algunas de las características más importantes con el fin de determinar cuáles consideran los encuestados que son pertinentes. De igual forma, se ha agregado la opción donde el encuestado pueda indicar si existe otra característica que considere atractiva.

**Cuadro 23 Respuestas de la pregunta 9**

Resultados		
Opciones	Respuestas Obtenidas	Cantidad
Mostrar un link con más información acerca del problema de seguridad	100%	2
Proveer reingeniería del código fuente	100%	2
Mensajes informativos dentro del IDE	50%	1
Solamente una advertencia indicando que el código fuente es vulnerable	0%	0
Otro: Mostrar el contenido de TEAM Mentor dentro del IDE	50%	1

**Fuente: Encuesta realizada a empleados de la empresa Security Innovation.**

### Gráfico 5 Respuestas de la pregunta 5



**Fuente:** Encuesta realizada a empleados de la empresa Security Innovation.

En el gráfico anterior, se muestra que el 100% considera que tanto mostrar un *link* con más información acerca del problema de seguridad como proveer reingeniería en el código fuente son bastante importantes.

De igual forma, se puede observar que el hecho de mostrar mensajes informativos en el ambiente integrado de desarrollo es importante. El 50% de los encuestados indica que poder mostrar dentro del ambiente integrado de desarrollo el contenido de la plataforma en línea TEAM Mentor sería un candidato importante.

### 2.2.10 Pregunta 10

En general, ¿considera usted que creando un *plugin* para Visual Studio.NET con diagnóstico y solución de código enfocado en seguridad tendrá buena aceptación en la comunidad del desarrollo del *software*?

Es importante considerar la aceptación de la comunidad desarrolladora de software, especialmente cuando se trata de herramientas que se instalan dentro del ambiente integrado de desarrollo. Esta pregunta trata de conocer el criterio de los interesados sobre si habría o no una buena aceptación por parte de la comunidad desarrolladora de *software* en la utilización del *plugin*.

La pregunta ha sido formulada de forma tal que se presentan dos opciones mutuamente excluyentes con los valores de Sí y No. Como resultado, el 100% de los entrevistados considera que sí habría una buena aceptación por parte de la industria en la adopción y utilización de la extensión de seguridad.

Debido a que el 100% de los encuestados ha brindado una respuesta positiva no se ha ilustrado tal comportamiento.

## CONCLUSIONES

Lewis Mumford, historiador, sociólogo, filósofo de tecnología y crítico de literatura expresó que “Detrás de todos los grandes inventos materiales del último siglo y medio no había sólo un largo desarrollo de la técnica; había también un cambio de mentalidad.”. En esta frase, se expresa cuán importante es el cambio de mentalidad por adoptar y comprender necesidades emergentes, ideas innovadoras y cambios de paradigmas, especialmente en una rama de las ciencias como es la computación.

En su libro *Trillions, Thriving In the Emerging Information Ecology*, los autores Lucas, Ballay y McManus indican que:

Hay un punto de vista - generalmente llamado “determinismo tecnológico” – que esencialmente dice que cada ruptura tecnológica inexorablemente guía a la próxima. Una vez que tuvimos las bombillas, inevitablemente tuvimos que tropezar con los tubos al vacío. Cuando nos dimos cuenta de lo que podían hacer, rápidamente fuimos guiados a los transistores y los circuitos integrados y los microprocesadores no se quedaron atrás. Este proceso – sigue el argumento- es esencialmente automático, con cada pieza del dominó inevitablemente golpeando a la próxima, mientras somos guiados hacia un futuro desconocido pero predeterminado.

Nosotros no estamos seguros si llegaremos tan lejos, pero ciertamente es el caso que cada era tecnológica establece las estaciones para la próxima. (p. 2).

Fundamentalmente la investigación presentada en este documento en conjunto con el prototipo funcional está fuertemente ligada con el punto de vista de los autores mencionado anteriormente.

En el campo de la seguridad informática también se ve un patrón similar, inicialmente la preocupación para los negocios consistía en adquirir *software* antivirus, luego el enfoque proponía invertir en proteger la capa de red hasta que se ha llegado a una era de la seguridad desde la perspectiva de las aplicaciones.

El *software* inseguro está debilitando la infraestructura tecnológica, generando pérdidas millonarias a empresas y perjudicando a usuarios finales.

El prototipo funcional de un *plugin* para Visual Studio .NET que permite realizar análisis estático de código fuente busca fundamentalmente poder facultar a los ingenieros a desarrollar *software* seguro. Por medio del enfoque disruptivo de los modelos convencionales, se adopta de forma temprana la plataforma de compilación Roslyn en el marco de la compilación del código fuente como servicio.

La plataforma de compilación Roslyn, como se ha mencionado anteriormente, abre la puerta a investigaciones modernas en la creación de herramientas para la productividad y la eficiencia del código fuente y sin lugar a dudas para proponer nuevos modelos que permitan desarrollar aplicaciones que sean resistentes a ataques informáticos comunes.

En principio, la elaboración del análisis FODA donde se han podido identificar las fortalezas, oportunidades, debilidades y amenazas para el proyecto ha sido fundamental, pues constituyen los pilares del problema a resolver. La identificación de estos elementos permite que haya una mejor visibilidad del problema en cuestión, tomando ventaja de las oportunidades competitivas que la empresa tiene en el mercado y permitiendo identificar aquellas áreas de mejora continua que posibilitan que se creen mejores aplicaciones para solventar alguna necesidad de negocio.

En la elaboración del prototipo funcional propuesto, se han tomado en consideración todos los aspectos identificados dentro del análisis FODA; de forma tal que esta herramienta ha sido fundamental en esta investigación, pues permite fijar una hoja de ruta sobre el curso de acción en la elaboración del prototipo funcional.

Es importante recalcar que en la investigación que se ha realizado, la innovación tecnológica ha jugado un papel crucial. No solamente el tema de seguridad informática desde la perspectiva del desarrollo de *software*, la cual ha tomado gran fuerza en los últimos años, pero también el enfoque que se ha propuesto para mitigar riesgos de seguridad en el código fuente desde el momento en que estos son creados.

Los objetivos planteados se han podido cumplir de forma satisfactoria. En el prototipo funcional se han implementado las diferentes estaciones del ciclo del desarrollo de *software*, teniendo un prototipo funcional que se puede instalar en un ambiente que dependa que contenga Visual Studio .NET 2012 o

posterior. En cada una de las etapas del proceso del desarrollo de *software*, se han elaborado actividades indispensables, siguiendo las mejores prácticas, a fin de elaborar un prototipo funcional de una extensión de seguridad que cumple con los objetivos establecidos y que genera valor agregado en la ardua tarea del desarrollo seguro del *software*.

Se ha realizado por lo tanto, un estudio profundo de algunas de las vulnerabilidades más comunes que se encuentran en el *software*, las cuales día a día son explotadas por usuarios maliciosos a fin de perpetrar alguna actividad ilícita. Los encabezados de los diarios internacionales, blogs, foros, revistas digitales y diversos recursos en línea son un testigo confiable de este esquema, donde miles de organizaciones se ven afectadas periódicamente.

Comprender los riesgos de seguridad propuestos en este documento y la elaboración de una extensión de seguridad para el entorno integrado de desarrollo Visual Studio .NET permite brindar ese valor agregado en el momento de desarrollar aplicaciones. Se comprende por lo tanto que a pesar de que las plataformas de desarrollo de *software*, como lo es Visual Studio .NET en este caso, no presentan un modelo para ayudar al desarrollador a evitar escribir código fuente vulnerable de forma intrínseca, sí permiten un modelo extensible donde la innovación e investigación en áreas diversas posibilita que se desarrollos componentes a la medida para solucionar un problema específicos.

Una de las ventajas competitivas que ofrece el *plugin* de seguridad es que las vulnerabilidades establecidas son identificadas en el preciso momento

en que el desarrollador las introduce como parte de las actividades de programación. Existe por lo tanto una retroalimentación en tiempo real dentro del ambiente donde principalmente el desarrollador es alertado, por medio de mensajes concretos, que existe código inseguro. Luego, se le brinda la opción de que permita que el *plugin* haga una refactorización del código fuente y le provea una solución al problema.

Este proceso de concientización ayuda considerablemente a que si existe desconocimiento de parte del desarrollador en cuanto a las vulnerabilidades en el código fuente, este pueda comprender el verdadero riesgo y así evitar introducir estos errores en futuras aplicaciones que el desarrolle.

La utilización de la plataforma de compilación Roslyn, la cual ha permitido extender las actividades que se realizan cuando el código fuente es compilado juega un papel fundamental en el momento de extender un ambiente integrado. A pesar de ser un proyecto relativamente nuevo introducido por Microsoft, es claro observar que existe un camino muy prometedor en la investigación y elaboración de herramientas para mejorar la calidad del *software*.

Esta extensión de seguridad propone un modelo simple de instalación lo cual facilita a que la empresa Security Innovation lo comercialice fácilmente. Además la integración con los demás productos de la empresa es de suma importancia pues permite que los demás productos sean conocidos por los desarrolladores. Cada vez que la extensión de seguridad encuentra un patrón

de código fuente vulnerable, le provee al usuario un vínculo a la base de datos de conocimiento TEAM Mentor, la cual brinda más información por medio de artículos concretos para comprender mejor estas vulnerabilidades y conocer de antemano el impacto que tienen para la industria.

El proceso de pruebas unitarias permitió detectar que no existen problemas de lógica en los diferentes módulos ya que todas las posibles rutas del código en su mayoría han sido verificadas. En la elaboración de las pruebas funcionales, se han utilizado aplicaciones de *software* hechas vulnerables a propósito (con fines educativos) y donde se ha podido validar que el *plugin* para Visual Studio ha detectado las vulnerabilidades establecidas en este documento.

Apoyando el pensamiento del físico danés Niels Bohr que a propósito del futuro indicaba que “Es difícil hacer predicciones, especialmente si se trata del futuro”<sup>15</sup> se concluye en que esta tendencia también se aplica a la disciplina de la computación. No obstante, la proliferación de redes bajo el protocolo Wi-Fi, el aumento desmedido de dispositivos móviles y una plena confianza del usuario en realizar transacciones por medio del servicio de Internet, exige que haya una mejora en temas de seguridad donde ellos se sientan seguros al realizar transacciones desde la comodidad de su hogar.

Afortunadamente, la personalización y extensión de los ambientes de desarrollo, característica innata de estos entornos, permite que la industria elabore componentes como el presentado en esta investigación para mejorar

---

<sup>15</sup> [http://www.economist.com/blogs/theinbox/2007/07/the\\_perils\\_of\\_prediction\\_june](http://www.economist.com/blogs/theinbox/2007/07/the_perils_of_prediction_june)

la calidad de las aplicaciones que se desarrollan. Así mismo permiten que exista investigación y desarrollo de herramientas que ayuden al ser humano a tener una mejor calidad de vida.

## RECOMENDACIONES

Aun cuando esta investigación y el prototipo funcional propuesto cumplen con los objetivos pactados a fin de brindar una solución a los problemas de la seguridad en el *software*, se hacen algunas recomendaciones con el propósito de buscar oportunidad de mejora en el desarrollo actual y fijar una hoja de ruta en el futuro del análisis estático de código.

Es importante considerar que el *software* evoluciona y es necesario proporcionar algunas observaciones para continuar mejorando la extensión de seguridad y de esta forma considerar una adaptación sencilla a futuros requerimientos.

El profesor Hiroshi Ishii del Instituto Tecnológico de Massachusetts (MIT por sus siglas en inglés) afirma que: “Las tecnologías se vuelven obsoletas en un año, las aplicaciones se reemplazan en 10 años, pero una visión fuerte sobrevivirá por más de 100 años.”. Esa visión fuerte es la que se evoca en esta investigación con el objetivo de proporcionar una herramienta a desarrolladores que les permita realizar aplicaciones de *software* más seguras.

1. Implementar la detección todas las vulnerabilidades establecidas en el estándar OWASP Top 10 2013: Existen múltiples vulnerabilidades en el código fuente que permiten que un atacante comprometa la aplicación Web. El estándar internacional OWASP Top 10 2013, ampliamente utilizado en esta investigación, brinda un detalle muy completo de los

principales riesgos a los cuales están expuestas las aplicaciones web.

Mediante una metodología basada principalmente en el riesgo que existe para una organización, este documento muestra las principales vulnerabilidades en el *software* que afectan a la industria.

Se considera importante y beneficioso que el *plugin* para Visual Studio pueda diagnosticar cuando la mayoría de los riesgos descritos en el estándar son encontrados en el código fuente. Si bien es cierto, algunos de estos riesgos no están asociados directamente con el código fuente, es decir que depende de otro elemento de infraestructura propiamente, se recomienda implementar y desarrollar las herramientas de diagnóstico para alertar al desarrollador cuando está ante una posible vulnerabilidad claramente detallada en el documento.

Se recomienda que se continúe con el diagnóstico de las vulnerabilidades especificadas en el estándar de OWASP, se recomienda así mismo que en un plazo de 5 meses, la extensión de seguridad pueda realizar el análisis de todos estos riesgos.

Pero aparte de los diez principales riesgos descritos en el OWASP Top 10- existe un número elevado de otros problemas de seguridad que también sería ideal poder dar soporte. Organizaciones como la corporación MITRE tienen disponible y de forma gratuita el catálogo de vulnerabilidades en el *software* denominado Common Weakness Enumeration (Enumeración de Vulnerabilidades Comunes), el cual es

un catálogo más extenso donde se han documentado otro tipo de vulnerabilidades. Se recomienda a medida que el *plugin* se mejora, poder categorizar, identificar y priorizar estas vulnerabilidades y facultar al *plugin* para que realice tareas de diagnóstico a fin de poder alertar al desarrollador ante estos problemas.

Es importante destacar que cada una de las vulnerabilidades definidas por MITRE tiene un identificador único y el catálogo puede ser descargado, teniendo la posibilidad de contar con la información relacionada a la vulnerabilidad en formato HTML. La extensión de seguridad dentro de Visual Studio podría eventualmente mostrar el detalle del problema en el momento de detectar patrones de código propensos a permitir que las vulnerabilidades se exploten.

Por lo tanto se recomienda que se haga uso tanto de MITRE y de OWASP como marco de referencia para comprender los riesgos existentes a nivel de programación. Los modelos de riesgo y las guías de desarrollo seguro de *software*, guías de revisión de código fuente y guía de pruebas proporcionadas por la fundación OWASP de manera gratuita, son un pilar fundamental para el desarrollo de herramientas de seguridad.

2. Aumentar el espectro del diagnóstico por cada vulnerabilidad: Los usuarios maliciosos manipulan los datos de entrada a fin de explotar alguna vulnerabilidad existente en el *software*. Desafortunadamente son muchas las fuentes de entrada de datos en una aplicación. Estas

fuentes de entrada de datos se le denominan fuentes no confiables ya que por defecto la aplicación y la plataforma de desarrollo no pueden garantizar que los datos ingresados son confiables puesto que provienen de muchos lugares. Dentro de estas fuentes figuran las cajas de texto para realizar búsquedas donde el usuario introduce el texto a encontrar, los formularios HTML para completar el registro, los formularios HTML para autenticarse en el sistema, los parámetros de la aplicación que viajan en forma de HTTP GET o HTTP POST e incluso los datos que provienen de otras aplicaciones en la forma de servicios Web. Todos estos datos deben ser propiamente validados antes de que sean procesados por la aplicación. Se recomienda que en un plazo de tres meses se extiendan las herramientas de diagnóstico de la extensión de seguridad para poder cubrir la mayor cantidad de estas fuentes, es decir, que el *plugin* pueda hacer un diagnóstico exhaustivo que permita, en una aplicación web, determinar si los datos ingresados están siendo validados correctamente. Teniendo un rango de análisis más completo para identificar las fuentes de datos no confiables, permitirá que el *plugin* brinde mayor retroalimentación al desarrollador y de esta forma mitigar gran parte de los problemas de seguridad atribuidos a la falta de validación de la recolección de los datos.

Se recomienda consultar libros de seguridad informática como lo son “**Writting Secure Code**” de Michael Howard y David LeBlanc, “**Software Security: Building Security In**” de Gary McGraw y

consultar a su vez la plataforma TEAM Mentor, la cual brinda un claro panorama de seguridad en el código fuente. Estos recursos mencionados ayudarán de forma sustancial a comprender e identificar las principales fuentes de datos no confiables, las cuales son utilizadas por usuarios maliciosos para comprometer a las aplicaciones web.

3. Implementar el uso de la librería de Microsoft denominada Microsoft Web Protection Library (WPL): Como un proyecto independiente de la plataforma de Visual Studio .NET, el equipo de desarrollo de Microsoft trabaja en una librería con rutinas y funciones previamente desarrolladas para reducir el número de vulnerabilidades en el código fuente. Dicha librería consta a su vez de dos componentes: AntiXSS que contiene una amplia gama de funciones para realizar codificación (encoding) de datos de entrada ya sean HTML, XML (Lenguaje de Marcas Extensible), CSS (Hoja de Estilos en Cascada) y JavaScript y el Motor de Seguridad en tiempo de Ejecución (SER) el cual brinda un nivel de protección que permite evitar ataques de Inyección de SQL y de Secuencias de Comandos entre Páginas.

Esta librería de Microsoft se distribuye como una aplicación independiente que puede ser instalada por medio del manejo de paquetes de Visual Studio denominado NuGet. Debido a que este componente se debe descargar de Internet, puede darse el caso de que en un determinado ambiente, por políticas de seguridad, no se le

permite al desarrollador a acceder a sitios que no estén debidamente autorizados y aprobados por el departamento de seguridad.

En este caso, se recomienda que el *plugin* para Visual Studio contenga la versión más actualizada de la librería y que permita hacer verificaciones periódicas para determinar si Microsoft ha emitido una versión más reciente de dicha librería.

Es importante dar soporte a este componente de Microsoft puesto que los expertos en seguridad ya han desarrollado rutinas predeterminadas para hacer una limpieza correcta de los datos de entrada. Incluso las mejores prácticas de seguridad abogan por la implementación y uso de esta librería cuando la tecnología adyacente sea Microsoft.

Se recomienda que en un periodo de 3 meses se pueda implementar el uso de la librería para que de forma más robusta se puedan resolver los problemas de seguridad en el código.

4. Dar soporte a múltiples idiomas: Aun cuando el idioma Inglés se considera como un idioma universal, y ampliamente utilizado en computación como el estándar de facto, es recomendable que el *plugin* propuesto provea soporte a múltiples idiomas. Esto significa que tanto las recomendaciones como la retroalimentación para el desarrollador que se muestran dentro del ambiente integrado de Visual Studio .NET sea en el desarrollador haya seleccionado como predeterminado. En términos técnicos, se recomienda que se implemente recursos basados en los distintos idiomas o al menos en una etapa inicial en los idiomas

más difundidos, de forma tal que no se tenga que hacer cambios a la lógica de las reglas de diagnóstico si no más bien cambiar el comportamiento del mensaje que es mostrado al desarrollador. El soporte a múltiples idiomas es un activo, en el sentido que hace que el desarrollador se sienta más cómodo e incluso que tenga una mejor comprensión del problema de seguridad al que se enfrenta. Se recomienda registrar esta característica como una opción de mejora en el *plugin* de manera que pueda ser atendida y desarrollada en una etapa posterior del proyecto una vez que se han establecido prioridades. Afortunadamente, la plataforma .NET ofrece un sólido paradigma para dar soporte a múltiples idiomas por medio de archivos de recurso cuya extensión es .resx, se recomienda que se implemente este tipo de archivos para la localización del idioma pues son muy fáciles de mantener y de actualizar. Se propone que en un plazo estipulado de seis meses se pueda acceder a la extensión de seguridad en múltiples idiomas, principalmente inglés y español.

5. Extender el diagnóstico y el análisis estático de código para analizar código fuente desarrollado en Visual Basic.NET: Actualmente la plataforma de compilación Roslyn permite extender los compiladores del lenguaje de programación C# (léase C#) y Visual Basic .NET a través de las interfaces de programación. En esta investigación, se ha limitado el alcance a realizar análisis estático de código fuente en el lenguaje de programación C#. No obstante, las lecciones aprendidas y

el conocimiento adquirido abre las puertas para que el *plugin* pueda también realizar este análisis de seguridad en el lenguaje Visual Basic.NET.

Pese a que ambos lenguajes de programación se ubican bajo la misma sombrilla de Microsoft .NET, las diferencias sustanciales en sintaxis y semántica son un factor determinante para que se traten como dos tecnologías que de por si son diferentes. En términos de análisis estático de código para encontrar vulnerabilidades por medio de la plataforma Roslyn, habría que implementar las reglas de diagnóstico para Visual Basic, al igual que ya se hace con C# y determinar si se distribuye como un *plugin* separado o como un componente unificado que permita realizar en análisis de ambos lenguajes de programación.

Aun cuando ambos lenguajes son diferentes en múltiples aspectos, la creación de herramientas de diagnóstico para Visual Basic .NET es trivial y se puede tomar como referencia mucho del código desarrollado en C# a fin de diagnosticar código fuente vulnerable.

En la medida en que exista mayor cobertura sobre la plataforma de Microsoft .NET, mayor será el mercado para la comercialización de la extensión. Se recomienda que un plazo de ocho meses se pueda empezar a dar soporte al lenguaje de programación Visual Basic.NET con el objetivo de ampliar el rango de aplicaciones y empresas que adquieran la extensión de seguridad.

6. Permitir la personalización del plugin dentro de Visual Studio .NET: Se recomienda facultar al usuario para que pueda personalizar el componente dentro de Visual Studio .NET. Dentro de las características que se podrían personalizar, figuran el idioma en el cual se mostrarán los resultados así como dejar a preferencia del usuario la forma en que la retroalimentación es presentada. El *plugin* propuesto sigue fielmente el mantra adjudicado a los mensajes de error (es decir información, advertencia y error). Por lo tanto uno de los beneficios sería que el usuario pueda elegir qué tipo de mensaje desea ver dentro de Visual Studio .NET cuando una vulnerabilidad es encontrada. En principio por ejemplo, un desarrollador preferiría que exista un mensaje de error indicándole que existe código fuente vulnerable, así como otro desarrollador prefiere que solamente se muestre un mensaje informativo o una simple advertencia que le indique el problema. Desde la perspectiva del diagnóstico para cada vulnerabilidad definida, se puede establecer el nivel de mensaje que será mostrado.  
Se recomienda que exista una opción de personalización dentro de las propiedades de la extensión donde el desarrollador indicará qué nivel de mensajes será mostrado dentro del ambiente integrado de desarrollo. Se sugiere que un plazo de diez meses se pueda personalizar la extensión dentro del ambiente de desarrollo, permitiendo que el usuario final predetermine la mejor configuración basado en sus necesidades.

## BIBLIOGRAFÍA Y OTRAS FUENTES

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 viernes 17 de octubre de 2014 de <http://goo.gl/fjjymp>

Cáceres, I (2010). *Las variables*. Recuperado 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>.

Duntemann's, J (2004). *Wi-Fi Guide 2nd Edition*. Paraglyph Press.

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

Flanders, J (2009). *RESTful .NET*. 1005 Gravenstein Highway North, Sebastopol, California: O'Reilly Media.Inc.

Fowler, M (1999). *Refactoring: Improving the design of Existing Code*. Reading, Massachusetts: Addison-Wesley

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.

Fundación OWASP (2014). *OWASP Top 10 2013. The Ten Most Critical Web Application Security Risk.*

Gourley, D., Totty, B. (2002). *HTTP: The Definitive Guide.* Sebastopol, California: O'Reilly Media

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

Heilsberg, A., Torgersse, 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>

Lucas, P., Ballay, J., McManus, M (2012). *Trillions: Thriving in the Emerging Information Ecology*. John Wiley & Sons, Inc.

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

Microsoft (2014). *HTTP Cookies*. Recuperado el lunes 03 de noviembre de 2014 de <http://goo.gl/pxVVGG>

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](http://www.caatec.org/CAATEC/publicaciones/crdigital/CR_Digital_3.pdf)

Muthuchamy, N (2014). *Preventing SQL Injection attacks*. Recuperado el viernes 07 de noviembre de 2014 de <http://goo.gl/v1Cx5c>

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.

Pathak, P (2011). *Research in Education and Psychology*. Pearson India.

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

Rajat, R (2013). *An absolute Beginner's Tutorial on Cross Site Scripting (XSS) Prevention in ASP.NET*. Recuperado viernes 07 de noviembre de 2014 de <http://goo.gl/VbgDnv>

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>.

Universidad de Murcia (2014). *Ciencia y método científico*. Recuperado el viernes 07 de noviembre de 2014 de <http://goo.gl/B3uCbH>

# ANEXOS

## Anexo 1:

Encuesta aplicada a los ingenieros de la empresa Security Innovarion

**Static Code Analysis Survey**

**General information**

**1. What do you consider are the benefits for the company when offering a Visual Studio .NET plugin that performs static analysis with diagnostics and code fix approach?**

**\* 2. Do you think that the fact of not having a proprietary source code analysis tool to report application security issues and code fixes is a blocking issue to increase sales and come up with new products?**

Yes  
 No

Comments

**General Questions**

**3. What do you consider it would be the impact in sales of company products when having a plugin that reports security issues and provides code fixes?**

Low  
 Medium  
 High  
 No Impact at all

Comments

**4. In terms of integration, Do you consider it will be easy to integrate a Visual Studio .NET plugin that performs diagnostics and code fix with existing products like TEAM Mentor?**

Yes, It will be easy.  
 No, It won't be easy.

Do you have any comments?

## Anexo 2:

Detección de errores, dentro de Visual Studio, de manejo incorrecto de sesiones

The screenshot shows a code editor window and an 'Error List' window in Visual Studio.

In the code editor (top half), there is C# code for a button click event:

```
23     0 references | 0 authors | 0 changes
24     protected void Button1_Click(object sender, EventArgs e)
25     {
26         if (!Request.Cookies.AllKeys.Contains("Amount"))
27         {
28             System.Web.HttpCookie tennisShoesPrice = new System.Web.HttpCookie("Amount", "75.00");
29             Response.Cookies.Add(tennisShoesPrice);
30         }
31         this.Label1.Text = "Total a pagar $" + Request.Cookies["Amount"].Value;
32     }

```

The line `System.Web.HttpCookie tennisShoesPrice = new System.Web.HttpCookie("Amount", "75.00");` has a yellow lightbulb icon, indicating a warning or suggestion.

In the 'Error List' window (bottom half), there is one error entry:

Description	File	Line	Column	Project
1 Se ha detectado un manejo incorrecto de las variables de sesión. Para mayor información visite el artículo de TEAM Mentor <a href="https://vulnerabilities.teammentor.net/article/f3aa0389-da45-47d7-8ace-4a71d006958d">https://vulnerabilities.teammentor.net/article/f3aa0389-da45-47d7-8ace-4a71d006958d</a>	Cookies.aspx.cs	27	17	VulnerableApp

### Anexo 3:

Detección de errores, dentro de Visual Studio, de exposición de datos sensibles

The screenshot shows the Visual Studio IDE interface. The top navigation bar includes tabs for SecurityMisconfiguration.aspx, style.css, Cookies.aspx.cs, Global.asax.cs, Cookies.aspx, Index.html, SecurityMisconfiguration.aspx.cs, SensitiveDataExposure.aspx.cs, and SensitiveDataExposure.aspx. Below the tabs, the code editor displays the following C# code:

```
1 reference | 0 authors | 0 changes
public partial class SensitiveDataExposure : System.Web.UI.Page
{
    References | Authors | Changes
    protected void Page_Load(object sender, EventArgs e)
    {
        string cs = @"Data Source=POSEIDON\sqlexpress;Initial Catalog=NORTHMSIND;Integrated Security=True";
        SqlConnection conn = new SqlConnection(cs);
        try
        {
            SqlCommand cmd = new SqlCommand();
            cmd.CommandText = "SELECT USERNAME, PASSWORD, NAME, LASTNAME, CREDIT_CART_NUMBER FROM dbo.CUSTOMER_INFORMATION";
            conn.Open();
            cmd.Connection = conn;
            cmd.ExecuteNonQuery();
        }
        catch (Exception )
        {
            throw;
        }
    }
}
```

A tooltip box is overlaid on the code editor, containing the text: "Solucionar Problema de seguridad" and "throw new Exception("An error occurred and the administrator has been alerted");".

At the bottom of the screen, the "Error List" window is open, showing the following details:

Description	File	Line	Column	Project
4 Información sensible expuesta por medio de mensajes de error. Para mayor información visite el artículo de TEAM Mentor https://vulnerabilities.teammentor.net/article/6724620c-b128-403f-b1a0-e9261c6537566	SensitiveDataExposure..cs	29	17	VulnerableApp

## Anexo 4:

Detección de errores, dentro de Visual Studio, de secuencias de comandos entre páginas

The screenshot shows the Visual Studio IDE interface. The code editor displays C# code for a page named XSS.aspx.cs. A tooltip is open over line 18, highlighting a potential security vulnerability. The tooltip contains two options: "Solucionar Problema de seguridad" and "Solucionar Problema de seguridad". The code itself includes several using statements and a Page\_Load event handler. The error list at the bottom shows one error related to a Cross-Site Scripting (XSS) vulnerability.

```
2  using System.Text.RegularExpressions;
3  using System.Text.RegularExpressions;using System.Text.RegularExpressions;namespace VulnerableApp
4  {
5      reference | authors | changes
6      public partial class XSS : System.Web.UI.Page
7      {
8          reference | authors | changes
9          protected void Page_Load(object sender, EventArgs e)
10         {
11             var_email = Request.QueryString["email"];
12             ...
13             using System.Text.RegularExpressions;
14             using System.Text.RegularExpressions;using System.Text.RegularExpressions;namespace VulnerableApp
15             {
16                 ...
17                 ...
18             }
19         }
20     }
```

Error List:  
3 Errors | 2 Warnings | 0 Messages  
Description ▾  
☒ 5 Código fuente vulnerable a Cross Site Scripting (XSS). Para mayor información visite el artículo de TEAM Mentor <https://vulnerabilities.teammentor.net/article/937f2173-5e39-48b6-bb3a-edfd8f052bb0>