

UNIVERSIDAD TÉCNICA DEL NORTE



**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN SISTEMAS COMPUTACIONALES**

TEMA:

**“ANÁLISIS DE LOS ATAQUES A APLICACIONES WEB SQL
INJECTION Y CROSS SITE SCRIPTING Y SUS MEDIDAS DE
PRECAUCIÓN Y DEFENSA”**

AUTOR:

EDGAR PATRICIO SUBIA PONCE

DIRECTOR:

ING. MAURICIO REA

IBARRA – ECUADOR

2016



**UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA**

**AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE**

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100315212-9		
APELLIDOS Y NOMBRES:	Subía Ponce Edgar Patricio		
DIRECCIÓN:	Azaya, Latacunga 7-27 e Isla San Salvador		
EMAIL:	epsubia@utn.edu.ec		
TELÉFONO FIJO:	062545331	TELÉFONO MÓVIL:	0980876375

DATOS DE LA OBRA	
TÍTULO:	Análisis de los ataques a aplicaciones web Sql Inyección y Cross Site Scripting y sus medidas de precaución y defensa
AUTOR (ES):	Subía Ponce Edgar Patricio
FECHA: AAAAMMDD	2016/03/22
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	<input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> SGRADO
TÍTULO POR EL QUE OPTA:	Ingeniero en Sistemas Computacionales
ASESOR /DIRECTOR:	Ing. Mauricio Rea

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD


Yo, Edgar Patricio Subía Ponce, con cédula de identidad Nro.1003152129, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

3. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 22 días del mes de marzo de 2016

EL AUTOR:

(Firma).....
Nombre: Edgar Patricio Subía Ponce



UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERIA EN CIENCIAS APLICADAS
CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, **SUBÍA PONCE EDGAR PATRICIO**, con cédula de identidad Nro. **100315212-9**, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor de la obra o trabajo de grado denominado: **“ANÁLISIS DE LOS ATAQUES A APLICACIONES WEB SQL INJECTION Y CROSS SITE SCRIPTING Y SUS MEDIDAS DE PRECAUCION Y DEFENSA”**, que ha sido desarrollado para optar por el título de **INGENIERO EN SISTEMAS COMPUTACIONALES** en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Firma:.....

Nombre: Edgar Patricio Subía Ponce

Cédula: 100315212-9

Ibarra, 22 de marzo de 2016

CERTIFICACIÓN DIRECTOR

Certifico que la tesis realizada por el señor Edgar Patricio Subía Ponce ha trabajado en el desarrollo del trabajo de grado **“ANÁLISIS DE LOS ATAQUES A APLICACIONES WEB SQL INJECTION Y CROSS SITE SCRIPTING Y SUS MEDIDAS DE PRECAUCION Y DEFENSA.”**, previo a la obtención del título de ingeniero en sistemas computacionales, realizándola con interés profesional y responsabilidad, lo cual certifico en honor a la verdad.



Ing. Mauricio Rea
Director de Tesis

DEDICATORIA

A Dios por permitirme darme la vida, a mis padres por inculcarme buenos valores y darme su apoyo incondicional, para poder lograr este paso muy importante en mi vida profesional.

A mi novia Aby que ha estado presente conmigo en las buenas y en las malas, dándome su apoyo para poder culminar esta etapa de mi vida.

A mis amigos con los que he compartido buenos momentos y nos hemos dado la mano a lo largo de la carrera universitaria.

Y finalmente a mi gran amigo y tutor Ing. Mauricio Rea que gracias a sus enseñanzas, apoyo moral y profesional me ha guiado a lo largo de este camino universitario, para lograr culminar esta meta.

Edgar Subía.

AGRADECIMIENTO

Gracias a la Universidad Técnica del Norte por haberme permitido ser parte de esta casona universitaria y poder formarme como persona con excelentes valores y a ser un gran profesional.

Un agradecimiento muy especial al Ing. Mauricio Rea, Director de Trabajo de Grado por haberme brindado su guía en la elaboración del presente trabajo de grado.

Y por último a todos los que supieron valorarme por quien soy, a ellos, un Dios le pague.

Edgar Subía.

INDICE DE CONTENIDOS

BIBLIOTECA UNIVERSITARIA	i
AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD	ii
CONSTANCIAS	ii
CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO	iii
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE	iii
CERTIFICACIÓN DIRECTOR	iv
DEDICATORIA	v
AGRADECIMIENTO	vi
INDICE DE CONTENIDOS	vii
INDICE FIGURAS	x
INDICE TABLAS	xiii
RESUMEN	xv
ABSTRACT	xvi
1 INTRODUCCIÓN	1
1.1 PROBLEMA	2
1.1.1 ANTECEDENTES.....	2
1.1.2 SITUACIÓN ACTUAL.....	2
1.1.3 PLANTEAMIENTO DEL PROBLEMA	3
1.2 JUSTIFICACIÓN.....	3
1.3 OBJETIVOS.....	4
1.3.1 OBJETIVO GENERAL.....	4
1.3.2 OBJETIVOS ESPECÍFICOS.....	4
1.4 ALCANCE	4
2 MARCO TEÓRICO	7
2.1 SEGURIDAD INFORMÁTICA	7
2.1.1 PRINCIPIOS DE LA SEGURIDAD INFORMÁTICA	8
2.1.2 FUNCIONES DE LA SEGURIDAD INFORMÁTICA.....	8
2.1.3 SERVICIOS DE LA SEGURIDAD INFORMÁTICA	11
2.1.4 PROBLEMAS DE LA SEGURIDAD INFORMÁTICA	11
2.1.5 AMENAZAS	13
2.2 ETHICAL HACKING	15
2.2.1 INTRODUCCIÓN AL ETHICAL HACKING	15
2.2.2 HACKER	17
2.2.3 TIPOS DE HACKER.....	17
2.2.4 ÉTICA DEL HACKER.....	18
2.2.5 MODALIDADES DE ETHICAL HACKING (TESTEO).....	19
2.2.6 ETAPAS DE ETHICAL HACKING.....	20
2.2.7 BENEFICIOS DE ETHICAL HACKING.....	21
2.2.8 HERRAMIENTAS DE ETHICAL HACKING	22
2.3 METODOLOGÍA DE DESARROLLO XP.....	23
2.3.1 ROLES XP.....	23
2.3.2 CICLO DE VIDA DE XP.....	24

2.3.3	FASES DE XP	26
2.3.4	ARTEFACTOS DE XP	29
2.3.5	PRACTICAS XP.....	30
2.4	METODOLOGÍAS DE PENTESTING.....	30
2.4.1	METODOLOGÍA OSSTMM (OPEN SOURCE SECURITY TESTING METHODOLOGY MANUAL)	30
2.4.2	METODOLOGÍA OTP (OWASP TESTING PROJECT)	37
2.4.3	COMPARATIVA DE LAS METODOLOGIAS.....	42
2.5	KALI LINUX.....	44
2.6	HERRAMIENTAS DE DESARROLLO.....	44
2.6.1	LENGUAJE DE PROGRAMACION PHP	44
2.6.2	SERVIDOR WEB APACHE.....	44
2.6.3	MVC (MODELO, VISTA, CONTROLADOR).....	45
2.6.4	FRAMEWORK SYMFONY	45
2.6.5	BASE DE DATOS POSTGRES.....	47
3	VULNERABILIDADES INFORMÁTICAS	48
3.1	INTRODUCCIÓN.....	48
3.2	TIPOS DE VULNERABILIDADES	48
3.3	SQL INJECTION	51
3.3.1	INYECCIONES MEDIANTE SALIDAS DE VARIABLES	53
3.3.2	INYECCIÓN CON DATOS NUMÉRICOS	53
3.3.3	INYECCIÓN POR MEDIO DE COMENTARIOS.....	55
3.3.4	INYECCIÓN MEDIANTE CONSULTAS CONCATENADAS Y NOMBRES DE TABLAS.....	56
3.3.5	INYECCIÓN LIMITANDO RESULTADOS	57
3.3.6	DEMOSTRACIÓN.....	58
3.3.7	MECANISMOS DE DEFENSA.....	63
3.4	CROSS SITE SCRIPTING	64
3.4.1	TIPOS DE VULNERABILIDADES	64
3.4.2	DEMOSTRACIÓN.....	66
3.5	TABLA COMPARATIVA DE LOS LENGUAJES VULNERABLES POR SQL INYECCIÓN Y CROSS SITE SCRIPTING	68
4	DESARROLLO DEL APLICATIVO	70
4.1	PLANIFICACIÓN DEL PROYECTO	70
4.1.1	ROLES	70
4.1.2	HISTORIAS DE USUARIO.....	70
4.1.3	DISEÑO DEL SISTEMA	73
4.1.4	TAREAS	75
4.1.5	ESTIMACIÓN DE ESFUERZO	80
4.2	FASE DE DISEÑO	81
4.3	FASE DE ITERACIÓN	84
4.3.1	ITERACIÓN I.....	84
4.3.2	ITERACIÓN II	84

4.4	FASE DE PRUEBAS.....	88
4.4.1	DISEÑO DE PRUEBAS DE ACEPTACIÓN.....	88
5	CONCLUSIONES Y RECOMENDACIONES.....	90
5.1	CONCLUSIONES.....	90
5.2	RECOMENDACIONES.....	90
6	BIBLIOGRAFÍA Y LINKOGRAFÍA.....	91

INDICE FIGURAS

Figura 1: Diagrama de la metodología de pentesting	5
Figura 2: Arquitectura MVC de Symfony	6
Figura 3: Método de Intrusión Cross Site Scripting	6
Figura 4: Cifrado Simétrico	9
Figura 5: Cifrado Asimétrico	9
Figura 6: Integridad de datos	10
Figura 7: Fase de footprinting.....	22
Figura 8: Ciclo de vida de un proyecto en XP	24
Figura 9: Fases XP	28
Figura 10: Esquema o Mapa de Seguridad	32
Figura 11: Plantilla de Pruebas de seguridad.....	35
Figura 12: Test de Seguridad en el ciclo de desarrollo de una aplicación.....	41
Figura 13: Inyección con dato numérico	54
Figura 14: Inyección exitosa inserción en la base de datos	55
Figura 15: Inyección mediante consultas concatenadas, nombres de tablas y limitando resultados	57
Figura 16: Inyección mediante consultas concatenadas, nombres de tablas y limitando resultados con software SQLMAP.....	58
Figura 17: Resultados de la inyección con Software SQLMAP.....	58
Figura 18: Demostración ataque SQLi	58
Figura 19: Demostración ataque inyección formularios mediante sentencias.....	59
Figura 20: Demostración ataque inyección 2 formularios mediante sentencias.....	59

Figura 21: Demostración ataque inyección formularios mediante sentencias.....	60
Figura 22: Ataque a página externa mediante consultas propias.....	60
Figura 23: Vulnerabilidad en página que indica las columnas donde se inyecta código.....	61
Figura 24: Inyección mediante consulta construida -1+union+select+1, 2, table_name, 4, 5, 6, 7, 8+from+information_schema.tables+limit+60, 1--	61
Figura 25: Aplicación que ayuda a detectar vulnerabilidades sqli	62
Figura 26: Aplicación web utilizando framework	62
Figura 27: Error al inyectar sentencia sqli en framework.....	63
Figura 28: Inyección XSS en validaciones de texto	66
Figura 29: Mensaje ante inyección XSS.....	67
Figura 30: Herramientas que ayudan a buscar fallas de validaciones ante ataques XSS	67
Figura 31: Arquitectura del Sistema	73
Figura 32: Arquitectura Funcional.....	74
Figura 33: Módulos del aplicativo	74
Figura 34: Modelo Relacional del aplicativo.....	81
Figura 35: Diagrama de Caso de Uso CU_ADMINISTRACIÓN.....	81
Figura 36: Caso de Uso CU_CATÁLOGO	83
Figura 37: Diseño de la estructura de datos para acceso al sistema	84
Figura 38: Ingreso al sistema.....	84
Figura 39: Administración de Perfiles.....	85
Figura 40: Administración de Usuarios.....	85
Figura 41: Diseño de la estructura de datos del módulo catálogos.....	86
Figura 42: Interfaz catálogo de productos	86
Figura 43: Interfaz de categorías	87

Figura 44: Interfaz producto	87
Figura 45: Interfaz Pedido	88

INDICE TABLAS

Tabla 1: Amenazas de la Seguridad Informática.....	13
Tabla 2: Las Etapas y Descripción del Hacking Ético.....	20
Tabla 3: Ciclo de vida de XP.....	25
Tabla 4 Artefactos.....	29
Tabla 5 Módulos y secciones de la Metodología OSSTMM.....	32
Tabla 6 Fases de una prueba en la metodología	37
Tabla 7 Tabla comparativa de las dos metodologías	42
Tabla 8: Componentes de PHP y Symfony	46
Tabla 9 Tipos de Vulnerabilidades	49
Tabla 10 Comandos generales DCL en una base de datos	51
Tabla 11 Comandos generales DDL en una base de datos.....	51
Tabla 12 Comandos generales DML en una base de datos	51
Tabla 13 Clausulas generales en una base de datos.....	52
Tabla 14 Operadores de comparación en una base de datos.....	52
Tabla 15 Vulnerabilidad y Solución del Ataque mediante variables.....	53
Tabla 16 Vulnerabilidad y Solución del Ataque mediante datos numéricos.....	53
Tabla 17: Roles de desarrollo	70
Tabla 18: Estimación de Trabajo	70
Tabla 19: Historia de Usuario 1	71
Tabla 20: Historia de Usuario 2.....	71
Tabla 21: Historia de Usuario 3.....	72
Tabla 22: Planificaciones de iteraciones.....	72

Tabla 23: Tarea Nro. 1.1	75
Tabla 24: Tarea Nro. 1.2.....	75
Tabla 25: Tarea Nro. 1.3.....	76
Tabla 26: Tarea Nro. 1.4.....	76
Tabla 27: Tarea Nro. 2.1	77
Tabla 28: Tarea Nro.2.2.....	77
Tabla 29: Tarea Nro. 2.3.....	78
Tabla 30: Tarea Nro. 2.4.....	78
Tabla 31: Tarea Nro. 3.1	79
Tabla 32: Tarea Nro. 3.2.....	79
Tabla 33: Estimación de Esfuerzo por tarea	80
Tabla 34: Descripción caso de uso CU_ADMINISTRACIÓN	82
Tabla 35: Descripción caso de uso CU_CATÁLOGO	83
Tabla 36: Pruebas de aceptación.....	89

RESUMEN

Desde el descubrimiento del SEQUEL en el año de 1974, dio origen al lenguaje que especifica las características de las base de datos que manejan el modelo relación, es que con este descubrimiento nace el SQL Inyección, un ataque que permite encontrar fallas de seguridad y poder modificar casi por completo la base de datos, así mismo Cross Site Scripting surge para poder inyectar código malicioso del lado del cliente como del servidor causando gran daño a empresas.

Este trabajo tiene como fin dar a conocer el funcionamiento de los ataques y como poder prevenir las aplicaciones web ante vulnerabilidades encontradas.

Capítulo 1.- Se habla sobre el problema, la situación actual y la justificación que se estudia para la realización de este proyecto de tesis, también se describen, el objetivo general y los específicos, los cuales ayudarán a la solución de dicho estudio.

Capítulo 2.- Se recopila toda la información necesaria, que se utilizará para poder comprender mejor el estudio, se describe las herramientas de software y los conceptos básicos sobre tecnología.

Capítulo 3.- Es un estudio de las vulnerabilidades más críticas y usadas a nivel mundial como lo son, el SQL Inyección y Cross Site Scripting que afectan a las aplicaciones web, también se realiza demostraciones sobre los ataques mencionados.

Capítulo 4.- Se empieza a desarrollar el aplicativo de acuerdo a las metodologías planteadas.

Capítulo 5.- Se termina el proyecto con las recomendaciones y conclusiones respectivas a cada objetivo.

ABSTRACT

Since the discovery of SEQUEL in the year 1974, gave rise to language that specifies the characteristics of the database that handled the relationship model, is that with this discovery comes the SQL injection, an attack to find security flaws and power almost completely modify the database, also arises Cross Site Scripting to inject malicious code on the client side and the causing great damage to businesses server.

This work aims to publicize the operation of the attacks and how to prevent web applications against vulnerabilities found.

Chapter 1. There is talk about the problem, the current situation and justification being studied for the realization of this thesis project, also described the general objective and specific, which will help the solution of this study.

Chapter 2. All necessary information, which will be used to better understand the study, software tools and technology basics described is collected.

Chapter 3. A study of the most critical vulnerabilities and used worldwide as they are, the SQL Injection and Cross Site Scripting affecting web applications, demonstrations about these attacks is also performed.

Chapter 4. Begins to develop the application in accordance with the methodologies proposed.

Chapter 5. Each objective the project is completed with the respective conclusions and recommendations.

1 INTRODUCCIÓN

A principios de los años 90s la Web tenía un formato muy sencilla y simple que no era nada más que documentos de texto planos que se enlazaban entre sí mediante hipervínculos, en la actualidad conforme avanza la tecnología, los simples documentos se han ido convirtiendo en grandes aplicaciones capaces de interactuar con el usuario permitiendo manejar y almacenar grandes cantidades de información de cualquier tipo, este avance ha traído en si problemas de seguridad en los contenidos que se publicaban, especialmente en aplicaciones que ofrecían servicios que requerían información muy sensible del cliente como lo eran tarjetas de crédito, datos personales, etc.

Por tal motivo se recurre a la seguridad informática que se encarga de proteger los datos de manera óptima con estándares, protocolos y reglas facilitando minimizar los posibles riesgos en la información. De igual manera con el paso del tiempo nace el Hacking Ético que ayuda a realizar pruebas de testeo en sistemas informáticos permitiendo descubrir las deficiencias y fallas que el desarrollador comete en la codificación de programas y configuración de servidores que permite al atacante comprometer la seguridad en una aplicación web, también ayuda a comprender de mejor manera las técnicas o métodos más utilizados por los hackers, tratando de poder resolver las vulnerabilidades que existen en las aplicaciones en un entorno web. A lo largo de este estudio se comprenderá y se podrá diferenciar a un pirata informático del verdadero rol de un Hacktivista¹.

¹ **Hactivista.**- Es toda actividad hacker motivada por fines políticos o sociales

1.1 PROBLEMA

1.1.1 ANTECEDENTES

De forma tradicional el desarrollo de software solamente ha estado a cargo de los programadores, quienes realizaban, el diseño, la arquitectura, la implementación y un test de la aplicación; dichos analistas o desarrolladores eran los encargados de buscar falencias o brechas de seguridad para darle una solución, en algunos casos no se toma en cuenta la seguridad informática, sólo el desarrollo. Esto ha ocasionado que los intrusos o atacantes llamados crackers², utilicen métodos de intrusión para apropiarse de la información vulnerable de las empresas o instituciones y posteriormente realizar fraudes informáticos; por tal motivo se ha tomado fuerte interés en la seguridad informática.

1.1.2 SITUACIÓN ACTUAL

Actualmente las aplicaciones informáticas han sido dirigidas a la parte web, con el afán de que la información esté al alcance de todos y tenga mayor disponibilidad, para que los clientes puedan tener acceso a servicios en cualquier lugar y momento. Esto ha ocasionado que la información este más vulnerable en la red, por lo que es importante mantener una seguridad a nivel de aplicaciones web, para que los usuarios finales no estén expuestos a fraudes informáticos. Son escasas las entidades a nivel nacional que han optado por contratar personal especializado en la seguridad informática.

² **Crackers:** Este término se utiliza para aquellos que infringen daño en algún sistema informático o roban información para hacer mal uso de la misma.

1.1.3 PLANTEAMIENTO DEL PROBLEMA

El desconocimiento de Seguridad Informática ha hecho que existan los ataques a diversos tipos de plataformas y lenguajes de programación, el más común y vulnerable PHP³. La Seguridad Informática ayuda a garantizar la privacidad, integridad y cifrado de la información de sistemas informáticos y de sus propios usuarios. Técnicamente es imposible lograr un sistema informático ciento por ciento seguros, pero buenas medidas de seguridad evitan daños y problemas que pueden ocasionar intrusos. El avance de Internet ha sido una herramienta muy peligrosa debido a que los atacantes van adquiriendo día a día más habilidades que les permiten tener mayores beneficios. Este hecho viene provocando una creciente necesidad de implantar mecanismos con la finalidad de proteger o reducir al mínimo los riesgos asociados a los incidentes de seguridad informática.

1.2 JUSTIFICACIÓN

La presente investigación se hace con el fin de poder corregir y mejorar el desarrollo de aplicaciones para prevenir futuros ataques, ya que esas deficiencias de seguridad ocasionan que la información quede vulnerable y de libre acceso, para que el intruso pueda hacer mal uso de la información. El presente proyecto también tiene como finalidad hacer conocer la importancia que tiene la seguridad dentro del desarrollo de software y ayudar a difundir conocimientos acerca de Hacking Ético.

³ **PHP**(PHP Hypertext Pre-procesador), lenguaje de programación web

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Analizar los dos métodos de ataques: SQL Injection y Cross Site Scripting, que se utilizan en aplicaciones web para prevenir y defenderse de los intrusos (Hackers⁴, Crackers).

1.3.2 OBJETIVOS ESPECÍFICOS

- Realizar un estudio de los aspectos más importantes sobre seguridad informática y Hacking Ético
- Identificar y evaluar los riesgos que se exponen ante las vulnerabilidades en aplicaciones web.
- Seleccionar una metodología de pentesting, apta para corregir errores en el desarrollo de las aplicaciones web.
- Desarrollar el aplicativo web donde se demostrará que una aplicación es vulnerable y se indicará la solución ante los ataques efectuados.

1.4 ALCANCE

Este caso de estudio tiene como función analizar las posibles vulnerabilidades que puede tener una aplicación web, utilizando métodos de ataque como el SQL Injection y Cross Site Scripting. Para este análisis se utilizarán las herramientas de software libre PHP con un framework de desarrollo symfony, con base de datos postgresql y se utilizará una arquitectura MVC⁵. Este proyecto será realizado de acuerdo a la metodología de desarrollo XP (Extreme Programming) y se buscó una metodología adecuada de testeo para dicho estudio, la cual se basa en Hacking Ético para su estudio.

⁴ **Hackers:** Personas que saben de informática que realizan entradas remotas a sistemas informáticos no autorizados con el fin de reportar fallas de seguridad.

⁵ **MVC:** patrón de arquitectura de software Modelo, Vista, Controlador

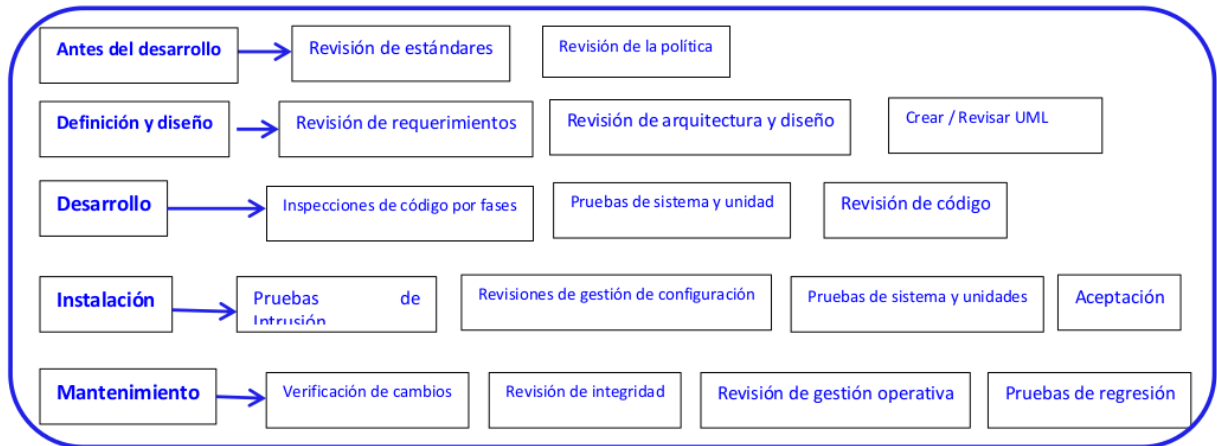


Figura 1: Diagrama de la metodología de pentesting

Fuente: OWASP Testing Project. (17th September, 2014). Recuperado de <https://www.owasp.org>

La aplicación constó de dos versiones: una con la deficiencia de seguridades y la otra ya presentada las correcciones debidas. Se construyó una aplicación sencilla basada en una lista de categorías de productos la cual consta de los módulos:

- Modulo Usuarios
- Modulo Catálogo

El modulo Usuario: Está basado a lo que corresponde a seguridades

El modulo Catálogos: Contiene toda la información del producto, sus detalles ya sean este precio, cantidad, etc.



Figura 2: Arquitectura MVC de Symfony
 Fuente: Autoría Propia

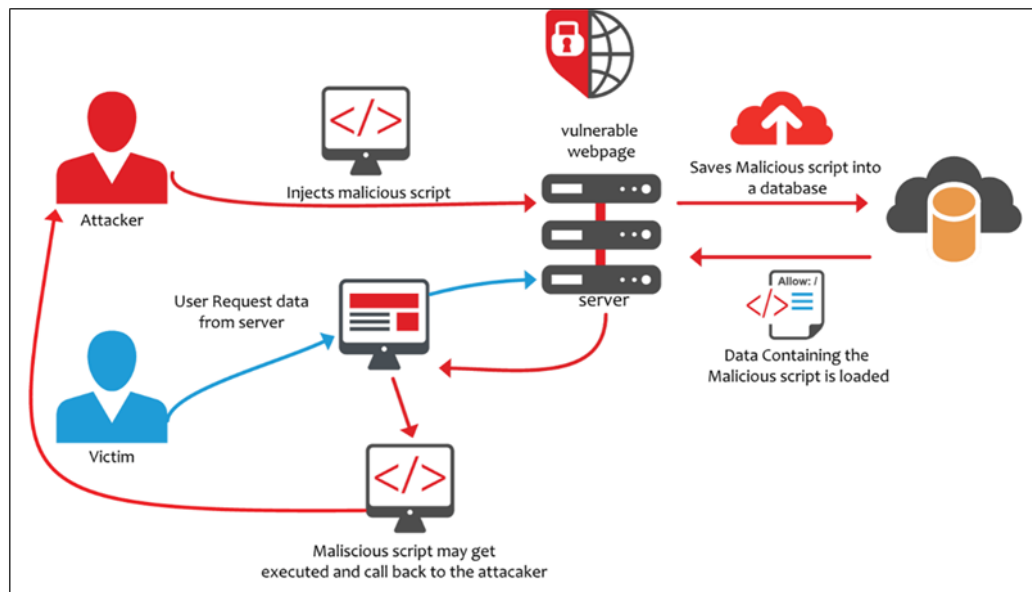


Figura 3: Método de Intrusión Cross Site Scripting
 Fuente: Securityec recuperado de <http://securityec.com/wp-content/uploads/2013/08>

2 MARCO TEÓRICO

En esta sección se analizará conceptos sobre seguridad informática y los diferentes roles que están involucrados en la Seguridad Informática.

2.1 SEGURIDAD INFORMÁTICA

Para estudiar el término de seguridad informática de mejor manera se puede definir dos conceptos, por un lado se hablaría de seguridad informática de forma general y por otro de seguridad informática aplicada a una plataforma web.

Es una rama de la informática que se encarga de la protección de toda la infraestructura tecnológica y todo lo que se relaciona con ella, para poder disminuir riesgos se usan estándares, protocolos y herramientas que ayudan a proteger toda la información confidencial de una organización.(Ramírez Parra, 2012)

La seguridad informática según el autor (Costas Santos, 2014a), consiste en asegurar que los recursos informáticos de una organización, sean utilizados de la manera que se decidió, al igual que el acceso a la información allí contenida. Además su modificación, debe ser realizada sólo por las personas que se encuentren acreditadas y dentro de los límites de su autorización.

La seguridad informática en la web, se deriva de la seguridad informática, y sus principales funciones están enfocadas a la protección de datos en aplicaciones, servicios y todo lo que se relacione a la web. Al hablar de toda web también se habla de los riesgos que existen, tanto en los ambientes internos como externos de una organización.

2.1.1 PRINCIPIOS DE LA SEGURIDAD INFORMÁTICA

Existen una serie de reglas básicas que se debe seguir para poder tener asegurada la información las cuales se detallan a continuación:

- Determinar presuntos inconvenientes y amenazas a la seguridad, priorizando y corrigiendo los riesgos.(Escrivá Gascó, Romero Serrano e Ramada, 2013)
- Garantizar la adecuada utilización de los recursos y la aplicación de los sistemas.
- Limitar las pérdidas y conseguir la adecuada recuperación del sistema en caso de un incidente de seguridad.
- Cumplir con el marco legal y con los requisitos impuestos a nivel organizativo.

Estos puntos son esenciales que se deben tomar en cuenta para tener una mejor optimización en los recursos de cualquier organización, con el objeto de prevenir cualquier tipo de infiltración y/o vulnerabilidad de los sistemas informáticos.

2.1.2 FUNCIONES DE LA SEGURIDAD INFORMÁTICA

Para que los principios de la seguridad informática tengan validez y pueda cumplirse el proceso de correcciones ante posibles brechas de inseguridad en los sistemas informáticos o aplicaciones web, se deben completar ciertas funciones que garanticen que la información se encuentre asegurada, estas son:

- **CONFIDENCIALIDAD**

Cualidad que presenta un documento, mensaje o archivo al ser enviado y que únicamente pueda ser leído por el receptor-destino; esta técnica se utiliza para evitar que otro individuo que intercepta el mensaje pueda leerlo, según el autor(Costas Santos, 2014b).

- Cifrado Simétrico: Es una técnica muy antigua y muy útil que se utiliza para encriptar⁶ una clave, esta puede ser una cadena de palabras, números o combinaciones alfanuméricas, y que ayuda a que el mensaje que recibe el receptor pueda descifrar y no sufra alteraciones en dicho mensaje.

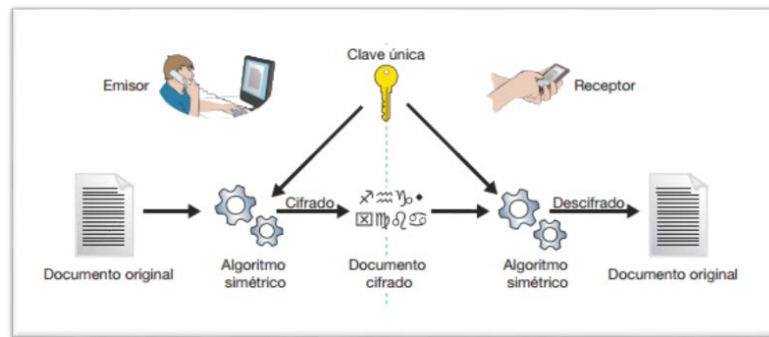


Figura 4: Cifrado Simétrico
Fuente: (Buendía, 2013)

- Cifrado Asimétrico: Este tipo de cifrado tiene un peculiar cambio ya que para poder descifrar el mensaje el destinatario debe poseer dos claves: una pública y otra privada; si no se posee las dos claves no se podrá descifrar el contenido que lleva el mensaje, este tipo de cifrado presenta un problema, es más lento que el simétrico, pero que hace que sea más seguro y confiable. (Franchi, 2013)



Figura 5: Cifrado Asimétrico
Fuente: (Buendía, 2013)

⁶ **Encriptar:** Proceso que se realiza a un documento (mensaje), para ocultar los datos mediante una clave

- **INTEGRIDAD**

La integridad es la cualidad que posee un documento o archivo que no ha sido alterado y que además permite comprobar que no se ha producido manipulación alguna en el documento original.(Hermoso Metaute, 2013)

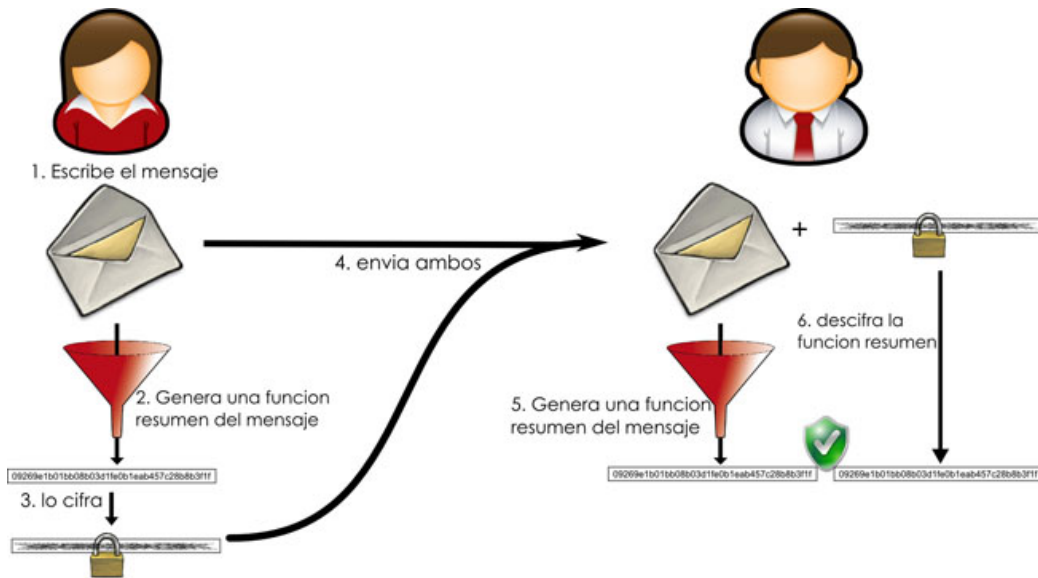


Figura 6: Integridad de datos
Fuente: (Hermoso Metaute, 2013)

- **DISPONIBILIDAD**

Capacidad de un servicio, de unos datos o de un sistema, de ser accesible y utilizable por los usuarios (o procesos) autorizados cuando estos lo requieran. Supone que la información pueda ser recuperada en el momento que se necesite, evitando su pérdida o bloqueo.(Costas Santos, 2014a)

- **FIABILIDAD**

Básicamente la fiabilidad está basado en una probabilidad, la cual se encarga de verificar si el sistema o aplicación está funcionando correctamente y pueda resistir ante cualquier circunstancia presentada.(Rafael e Manuel, 2013)

Todas estas funciones son esenciales para proteger cualquier dato, cuando la información viaje a través de redes y no pueda ser interceptada por un hacker.

2.1.3 SERVICIOS DE LA SEGURIDAD INFORMÁTICA

✓ Autenticación.- Valida la identificación del sujeto para que pueda acceder a la información que desea consultar.

✓ Autorización.- Especifica y da paso la administración a los sujetos que tienen acceso y permiso para poder modificar, guardar e insertar la información en la aplicación.

✓ Auditoria.- Permite la detección o recuperación de información, ante alguna anomalía que se encuentre en el sistema, mediante un histórico, el cual indica todos los eventos que se van realizando en la aplicación.(Samaniego, 2009)

2.1.4 PROBLEMAS DE LA SEGURIDAD INFORMÁTICA

Al hablar de seguridad informática, existe una serie de factores que impide mantener a salvo la información, esto hace que se encuentran involucrados ciertos aspectos que son perjudiciales para el funcionamiento de un sistema informático, ya que pueden causar daños tanto en hardware, software, como en los propios datos almacenados en un sistema.

A continuación se describen las amenazas que causan los problemas en la seguridad informática.

- **SEGURIDAD FÍSICA**

Consiste en el aseguramiento de la infraestructura tecnológica, tanto dentro como fuera del centro de cómputo. La seguridad física se basa en sí, a toda la infraestructura que existe en un centro de datos que pueden ser, las instalaciones eléctricas, hardware, etc. En varias ocasiones el ambiente de la infraestructura no es la adecuada debido a la humedad que existe en la misma, por falta de conexiones a tierra, las cuales influyen bastante, porque hacen que los equipos se estropeen, esto genera mucha pérdida de información y de equipos materiales.(Gómez Fernández e Fernández Rivero, 2015)

- **SEGURIDAD EN LOS RECURSOS HUMANOS**

- PERSONAL INTERNO.- Es un recurso muy importante que posee una organización, pero el más peligroso, debido a que manipula ciertas partes de los activos, que pueden ser, base de datos, aplicaciones, etc.
- PERSONAL EXTERNO.- Se debe tener mucho cuidado con las acciones que realiza una persona particular que ingresa a la empresa, porque puede tener acceso a zonas restringidas y a la información más vulnerable de la entidad.

- **SEGURIDAD ACTIVA**

Consiste en detectar y prevenir cualquier tipo de errores que existan en los sistemas, antes de que produzca cualquier tipo de riesgo.

- **SEGURIDAD PASIVA**

Se realiza un seguimiento a los procedimientos necesarios para reducir incidentes de seguridad, tomando medidas correctas como por ejemplo, las copias de seguridad que se realizan en un sistema informático.(Escrivá Gascó, Romero Serrano e Ramada, 2013)

2.1.5 AMENAZAS

Las amenazas pueden ser provocadas de múltiples maneras, entre las más conocidas están: personas, condiciones físicas-ambientales y software o lógicas, en esta sección se tratará sobre las amenazas lógicas según el autor(Gómez Vieites, 2014)

- **AMENAZA LOGICA**

Son programas que de una u otra manera ponen en riesgo a un sistema sea este operativo o software, y son creados de manera mal intencionado ya sea por malware⁷ o por bugs⁸, existen varias amenazas, tales como las siguientes:

Tabla 1: Amenazas de la Seguridad Informática

AMENAZA	DESCRIPCIÓN
Software Incorrecto	Se denomina así a toda aplicación que presenta errores de codificación, y a los programas que sirven para vulnerar a sistemas, se los conoce como exploit, un exploit, no sólo puede ser mediante un software, se puede hacer mediante contacto físico en el cual se cuestiona al individuo y se obtiene información valiosa.

⁷ **Malware:** Programa malintencionado (virus, espías, gusanos, troyanos, etc.) que afecta a los sistemas para: controlar o realizar acciones remotas.

⁸ **Bugs:** Errores de programación

Herramientas de seguridad	Las Herramientas de seguridad son muy importantes por cuanto nos ayudan a defendernos ante cualquier ataque de intruso, pero de igual forma son peligrosas ya que pueden utilizar las mismas herramientas para detectar errores y aprovechar para atacar.
Puertas traseras o backdoors	Es una falencia que evita la autenticación del sistema para poder ingresar y causar alguna mala intención o espionaje, en algunos casos las puertas traseras son construidas con el fin de tener una entrada secreta.
Bombas lógicas	Es un fragmento de código que es usado con el fin de causar daño al sistema o a toda una organización dentro de su infraestructura, este puede estar oculto hasta que se activen ciertas condiciones pre-programadas.
Virus	Programas que tienen una secuencia de código que se adhiere a una aplicación y se ejecuta al mismo causando daños en el sistema ya sean estos: borrando información de discos, bloquear redes etc.
Gusano o Worm	Programa que es capaz de ejecutarse por sí solo y atacar principalmente a componentes del sistema operativo que es invisible para el usuario.

Troyanos o Caballos de Troia	Instrucciones que realiza un programa al ser ejecutado por el usuario de tal forma que este piensa que las tareas se llevan con normalidad, sin embargo ejecuta funciones ocultas que perjudican la seguridad en una aplicación.
---	--

Fuente: Autoría Propia

2.2 ETHICAL HACKING

Se tratará sobre las técnicas y principios del Hacking Ético, donde se describe el verdadero perfil de un hacker; también se pretende comprender el uso de las herramientas que se utilizan para atacar ordenadores o cualquier tipo de aplicaciones.(Corletti Estrada, 2011)

2.2.1 INTRODUCCIÓN AL ETHICAL HACKING

Dentro del ámbito informático cuando se habla de hackers se piensa que es un individuo que se encarga de interceptar datos en una red para llevar todos los elementos tecnológicos al límite para hacerlos fallar, básicamente la tarea de un hacker no es dañar, es conocer. Los hackers tienen una serie de reglas, una de las fundamentales es mantener el anonimato y la discreción de sus actividades; es aquí donde nace la ética que a lo largo de la historia a tratado de diferenciar lo que es bueno o lo que es malo permitiendo formar principios, pero hay hackers que no son éticos y que desviaron sus principios, no por conocimiento sino por el simple hecho del lucro, esto ha ocasionado una gran confusión y ha generado el mito de que todo hacker es un delincuente. (Tori, 2008)

- **DEFINICIÓN.-** Básicamente el Ethical Hacking es una disciplina de la seguridad informática que se encarga de evaluar las posibles amenazas que pueden existir en un centro de datos conectado a internet, ante los posibles ataques que realizan los hackers, se realizan pruebas de pentesting que consiste en encontrar vulnerabilidades para luego reportar a la organización los errores encontrados para que tomen medidas de cómo proteger dicha información e impedir que exista el robo de datos importantes de una institución.

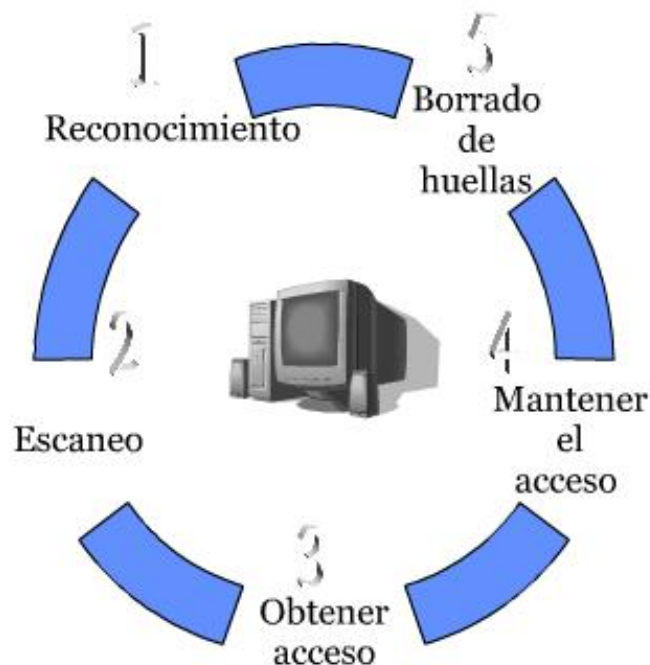


Figura 7: Ethical Hacking
Fuente: Prof. Constantino Malagón, recuperado de https://www.nebrija.es/~cmalagon/seguridad_informatica/transparencias/Modulo_0.pdf

2.2.2 HACKER

El término hacker, (es una palabra intraducible), cuyo origen es de difícil determinación, pero que muy probable esté vinculado a los llamados hacks (término del argot inglés), que son golpes secos que efectuaban los técnicos en telefonía al tratar de reparar sus aparatos. La función del hacker es encontrar fallas en diferentes sistemas ya sean estas en telecomunicaciones, hardware, software, etc., para luego reportar dichas vulnerabilidades a las organizaciones afectadas para su corrección. (Hector Jara, 2012)

2.2.3 TIPOS DE HACKER

✓ Hackers de Sombrero Negro o Crackers.- Son individuos conocedores de tecnología que utilizan sus conocimientos para corromper el funcionamiento de un sistema o robar información que contenga dicho sistema y lo utilizan a su conveniencia. En éstos está ausente la ética por no reportar los errores encontrados.(Flores Quispe, 2013)

✓ Hackers de Sombrero Blanco.- Personas con gran conocimiento de hacking, le utilizan como mecanismo de defensa, buscan vulnerabilidades e intentan implantar medidas para contrarrestar las fallas, se dedican a difundir el Ethical Hacker en empresas, para asegurar la información de los hackers de sombrero negro.

✓ Hackers de Sombrero Gris.- Es una combinación mixta entre los hackers de sombrero blanco y los de sombrero negro; utilizan sus conocimientos algunas veces para atacar y otras para defenderse, según sea la oportunidad, por ejemplo: si se encuentra un error en una aplicación, estos analizan si lo reportan o no según lo que les ofrezca la empresa.(Alonso Torres, 2015)

2.2.4 ÉTICA DEL HACKER

Ethical Hacker (hacker ético), es para referirnos a los profesionales de la seguridad de la información que utilizan sus conocimientos de hacking con fines defensivos. La función del Ethical Hacker será, por ende, determinar lo que un intruso puede hacer sobre un sistema y/o información, además de velar por su protección (Hector Jara, 2012).

Un hacker para que sea ético debe tener principio y valores, con los que pueda tomar decisiones en cuanto corresponde a los ataques. Son múltiples las motivaciones que un hacker puede tener, que van desde una simple curiosidad y las ganas de divertirse, hasta intentar chantajear a personas con algo económico a partir de lo que sabe, todo depende de sus objetivos y conveniencias.

✓ VALORES FUNDAMENTALES.- Un hacker para que tenga ética debe cumplir con ciertos valores fundamentales que son:

- Pasión
- Libertad
- Conciencia Social
- Verdad
- Anti-corrupción
- Curiosidad
- Interés
- Igual social
- Libre Acceso a la información
- Accesibilidad
- Actividad
- Precaución Responsable
- Creatividad

✓ PRINCIPIOS ÉTICOS

- **Principio de accesibilidad.-** Todo sujeto tiene derecho al libre acceso de la información, sin privar ni restringir su uso.
- **Principio de privacidad y disposición de la información.-** Todas las personas tienen derecho de tener su espacio, tener información del almacenamiento y de su manipulación. **Principio de Transparencia.-** Todos los datos que se obtienen al realizar el ataque deben ser recolectados, almacenados y deben estar a disposición del sujeto.
- **Principio de Seguridad.-** La información que proporciona un sujeto o es recolectada para una empresa, debe ser protegida para evitar pérdidas, destrucción, manipulación o modificación no autorizada por el usuario.
- **Principio de Responsabilidad.-** Cualquier interferencia con los datos de una persona o de un derecho de tener un control sobre la información, debe ser justificada a tiempo y de una manera apropiada ante la persona afectada.

2.2.5 MODALIDADES DE ETHICAL HACKING (TESTEO)

- **Red Teaming.-** Es una prueba encubierta, la cual solo un grupo de ejecutivos principales le conoce. En esta modalidad se pueden realizar técnicas de intrusión como es la ingeniería social donde el atacante puede obtener información de la víctima.
- **Blue Teaming.-** Solo el personal del centro de cómputo sabe de las pruebas, esto se lo aplica cuando el encargado de las seguridad de la organización ve algún incidente que afecta a las comunicaciones internas de la organización.(Benchimol, 2011)

- **Hacking Ético Interno (Caja Blanca).**- Este control se lo hace cuando el hacker conoce la infraestructura interna tanto de la red como de los activos tecnológicos, este control toma poco tiempo a evaluar la aplicación logrando obtener mejor resultados sobre las vulnerabilidades.(Shakeel, 2011)
- **Hacking Ético Extremo (Caja Negra).**- En esta modalidad no se facilita información acerca del sistema a ser evaluado, este test se lo realiza a toda la infraestructura de la red desde un punto remoto generalmente desde el internet.
- **Hacking de Aplicaciones Web.**- Se simulan ataques reales a las vulnerabilidades de una o varias aplicaciones
- **Hacking Ético de Sistemas de Comunicaciones.**- Se analiza las vulnerabilidades en las comunicaciones como puede ser redes de datos comunicaciones de voz etc.

2.2.6 ETAPAS DE ETHICAL HACKING

Existen cuatro etapas que ayudan a tener un mejor proceso ante cualquier ataque que se presente y estos son los siguientes:

Tabla 2: Las Etapas y Descripción del Hacking Ético

ETAPA	DESCRIPCION
Reconocimiento	Es una fase de preparación en donde el atacante recopila toda la información del individuo y objetivo
Escaneo	Consiste en analizar puertos TCP/IP, UDP, que utiliza la víctima en sus aplicaciones y ver qué servicio ofrece ese software. Aquí se pueden utilizar herramientas para poder ver que puertos están disponibles.(Graves, 2010)
Explotación	Básicamente en esta fase el ataque aplica sus conocimientos y busca vulnerabilidades ya sea mediante herramientas o metodologías, para que el hacker pueda tener una penetración exitosa depende de cómo este

	configurado la arquitectura y como este configurado el sistema
Mantener el Acceso	Se trata de poder conseguir el mayor tiempo posible al objetivo ya sea que bloqueen la sesión o cualquier mecanismo de defensa que se aplica al ataque.
Cubrir las Huellas	Todo atacante debe destruir las evidencias que ha dejado para así poder mantener el acceso al sistema hackeado, existen varias herramientas que se puede usar para borrar sus huellas ya sea troyanos, rootkits ⁹ , o borrando los logs ¹⁰ del sistema

Fuente: Autoría Propia

2.2.7 BENEFICIOS DE ETHICAL HACKING

- Ofrecer un panorama acerca de las vulnerabilidades halladas en los sistemas de información, lo cual es de gran ayuda al momento de aplicar medidas correctivas.
- Identificar sistemas que son vulnerables a causa de la falta de actualizaciones.
- Disminuir tiempo y esfuerzos requeridos para afrontar situaciones adversas en la organización.

⁹ **RootKits.**- Código malicioso que se apodera completamente del sistema, puede borrar archivos de sistema y detener programas virus, gusanos

¹⁰ **Logs.**- Son archivos donde se almacena todo los eventos ocurridos y permite conocer los hábitos de los usuarios en un sistema.

2.2.8 HERRAMIENTAS DE ETHICAL HACKING

- ✓ **Footprinting (Huella).**- Es el primer paso que realiza el hacker para poder recolectar toda la información pública posible y para poder obtener toda esa información se usan herramientas que se encuentran disponibles para cualquier persona que tiene acceso a la internet son los buscadores web como son Google, Yahoo! etc. Estos proporcionan cualquier tipo de información muy vital para el atacante ya que se puede encontrar datos como pueden ser el organigrama con áreas internas, los nombres de los ejecutivos, otras fuentes son las redes sociales. Los medios que se utilizan para el Footprinting a través de la red son: whois, traceroute y nslookup que son comandos que se utilizan mediante un terminal ya sea mediante Linux, Windows o Mac.

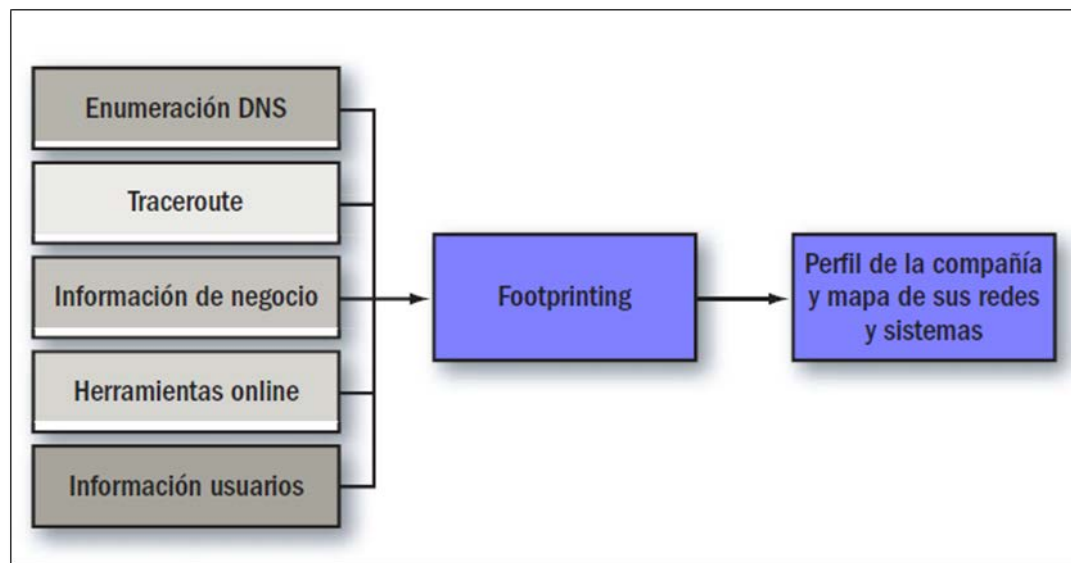


Figura 8: Fase de footprinting
Fuente: (Hector Jara, 2012)

- ✓ **Fingerprinting.-** Básicamente lo que se trata de hacer es conocer qué tipo de sistema operativo usa la víctima, ya que es de suma importancia que el atacante conozca la versión del sistema porque ayuda a conocer que vulnerabilidades se pueden aplicar al objeto en cuestión.(Baloch, 2014)
- ✓ **Ingeniería Social.-** Es considerada como un arte ya que consiste en manipular o engañar a la víctima ya sea por descornamiento de algún tema, la víctima brinda información ya sea personal o pública, mediante mensajes de texto, por correo electrónico etc.

2.3 METODOLOGÍA DE DESARROLLO XP

Se tomó XP como metodología de desarrollo por la rapidez en la que se puede desarrollar proyectos a corto plazo y porque permite realizar cambios de acuerdo a las necesidades en el transcurso del aplicativo, está orientado fuertemente hacia la codificación, en esta metodología se trabaja con equipos máximo de 10 personas.

2.3.1 ROLES XP

- ✓ **Programador.-** Básicamente este personaje es quien elabora y produce código en el sistema.
- ✓ **Cliente.-** Es el encargado de construir las historias de usuario para dar una validez al funcionamiento del aplicativo, el asigna prioridades a las historias de usuario y decide cuales se deben instalar primero.
- ✓ **Encargado de pruebas (tester).-** Tiene la obligación de interactuar con el cliente para ayudarlo a escribir las pruebas funcionales y es el responsable de las pruebas de soporte.

- ✓ **Encargado de seguimiento (tracker).**- Verifica el grado de dificultad en las estimaciones realizadas, es el encargado de realizar un seguimiento al proceso de iteraciones.(Gómez, Duarte e Güevara, 2014)
- ✓ **Entrenador (Coach).**- Un requisito para que puedan alcanzar el rol de entrenador es el estar familiarizado con los procesos que realiza XP, ya que ayuda a los miembros de un grupo para asegurar que se cumplan las prácticas.
- ✓ **Consultor.**- Miembro que posee conocimientos específicos en algún área del tema del proyecto y con el grupo tiene el poder para resolver el problema.(Castillo, 2014)
- ✓ **Gestor.**- El encargado de crear un vínculo con el programador ya que su función es el de coordinar.

2.3.2 CICLO DE VIDA DE XP

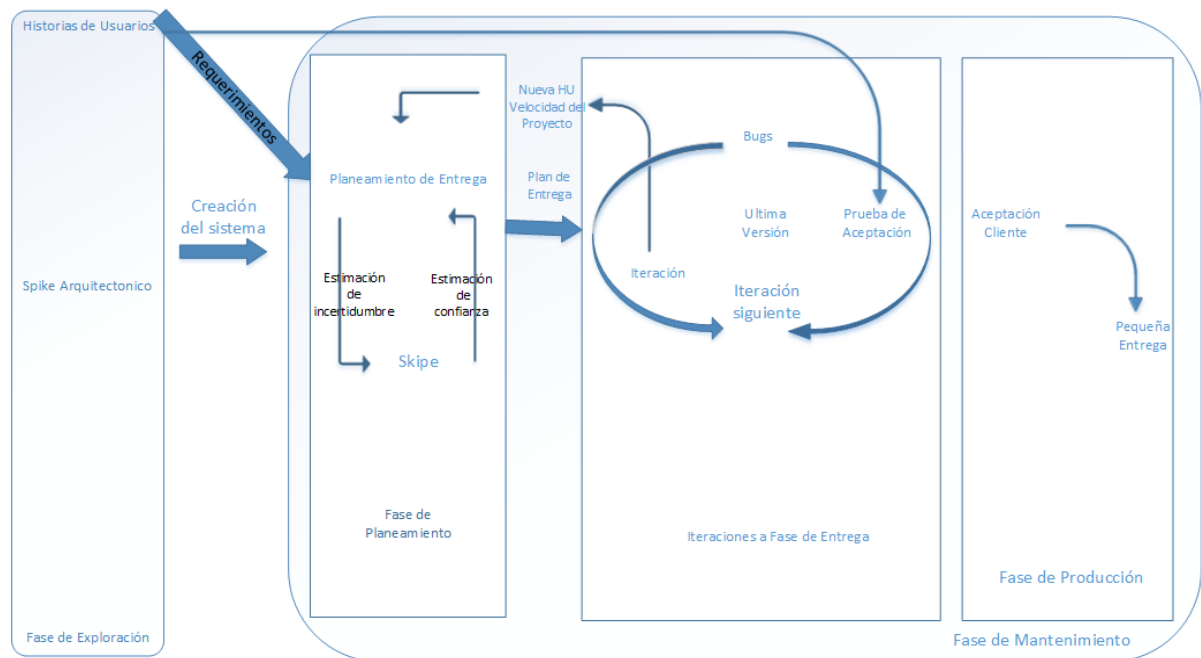


Figura 9: Ciclo de vida de un proyecto en XP
Fuente: Autoría propia

El ciclo de vida de esta metodología consiste en 6 fases que se describen a continuación

Tabla 3: Ciclo de vida de XP

CICLO	DESCRIPCIÓN
Exploración	Los clientes dan la mayor información en forma de historias de usuario, al mismo instante los desarrolladores que trabajan en equipo se familiarizan con las herramientas que se utilizarán en el proyecto, se realiza una prueba a la tecnología y se explora las posibles arquitecturas del sistema mediante un prototipo, esta fase no sobrepasa el tiempo de semanas o pocos meses.
Planificación de Entrega	El cliente establece la prioridad de las historias de usuario, y los codificadores establecen el tiempo de estimación del esfuerzo y se elabora un cronograma y se toma decisiones de la próxima cita que no debe sobrepasar los 3 meses para entregar un avance. La planificación es basada en cuanto al tiempo o al alcance del proyecto.
Interacciones	Las iteraciones están realizadas con el plan de entrega, en donde no deben sobrepasar las 3 semanas, el clientes es quien decide que historias se implementan en cada iteración, una recomendación que dan varios autores es que en la primera atracción se realice la arquitectura del sistema y esto se logra escogiendo las historias que más fuercen al proyecto.
Producción	En esta fase se deben realizar pruebas de rendimiento ante de que se entregue el sistema al cliente, al mismo tiempo se debe toma decisiones sobre las opciones de nuevos cambios, estos

	cambios son documentadas
Mantenimiento	Es una fase muy complicada debido a que mientras la primera versión del proyecto está en producción al mismo tiempo que se realizan nuevas iteraciones, para poder cumplir esta fase se debe realizar tareas de soporte para el cliente.
Muerte del proyecto	Se le llama así porque el cliente no tiene más historias de usuario que agregarle en el proyecto, otra situación que causa la muerte del proyecto es que el proyecto no cumple con las expectativas del cliente o no existe el presupuesto necesario para mantenerlo.

Fuente: Autoría propia

2.3.3 FASES DE XP

- ✓ Fase 1: Planificación del Proyecto.
 - Historias de Usuario.- Se recolecta todas las requerimientos del cliente pero en lenguaje PNL¹¹ y sin detalles, se usa para poder estimar el esfuerzo y el tiempo que tomara desarrollar la aplicación.(Gonzalez, 2012). Son usadas en la fase de pruebas para observar y verificar el cumplimiento del desarrollo.
 - Roles.- Los roles en este trabajo se los ha definido según (Beck, 1999) que ya fueron tratados en temas anteriores.

¹¹ **PNL.-** Programación Neurolingüística es una técnica usada para la recolección de información por medio de la comunicación en lenguaje normal para hacer entender lo que se requiere al cliente

- Plan de Entrega.- Se exponen las historias definidas que se usaran para cada versión del programa, en esta fase toman un papel importante el cliente y el programador ya que ambos deciden el tiempo para la implantación del sistema
 - Iteraciones.- Se debe definir iteraciones de 3 semanas de duración del aplicativo aproximadamente, al inicio de cada iteración los clientes seleccionan las historias de usuario de acuerdo al plan de entrega, estas historias son divididas en tareas que por lo general dura de 1 a 3 días.
 - Velocidad del Proyecto.- Se emplea la velocidad para poder contralar que todas las tareas se ejecuten en el tiempo indicado por cada iteración.
 - Rotaciones.- La metodología estudiada aconseja a trabajar conjuntamente ya que ayuda a mejorar la calidad y productividad del sistema desarrollado.
 - Reuniones Diarias.- Es aconsejable realizar reuniones diarias para intercambiar dudas e ideas para que la aplicación no sufra cambios al ya finalizar el trabajo.
- ✓ Fase 2: Diseño.-
- Diseño simple.- Elaborar un diseño simple y sencillo, ayuda a entender y reducir tiempo, tratando de evitar mucho esfuerzo en el desarrollo.
 - Riesgos.- Para evitar que existan riesgos, la metodología propone trabajar en pareja.
 - Re factorizar.- revisar una y otra vez el código para optimizar su rendimiento.

- ✓ Fase 3: Codificación.- En esta fase el cliente juega un papel importante ya que la mayor parte del tiempo debe pasar y estar pendiente del proyecto, el cliente es quien agrega el tiempo máximo en el que se presenten avances.
- ✓ Fase 4: Pruebas.- se realizan dos tipos de pruebas, la primera que corresponde al funcionamiento de cada versión validando y verificando el cumplimiento de las historias de usuario, y el segundo llamado test de aceptación en donde el cliente o usuario verifica que su funcionamiento.

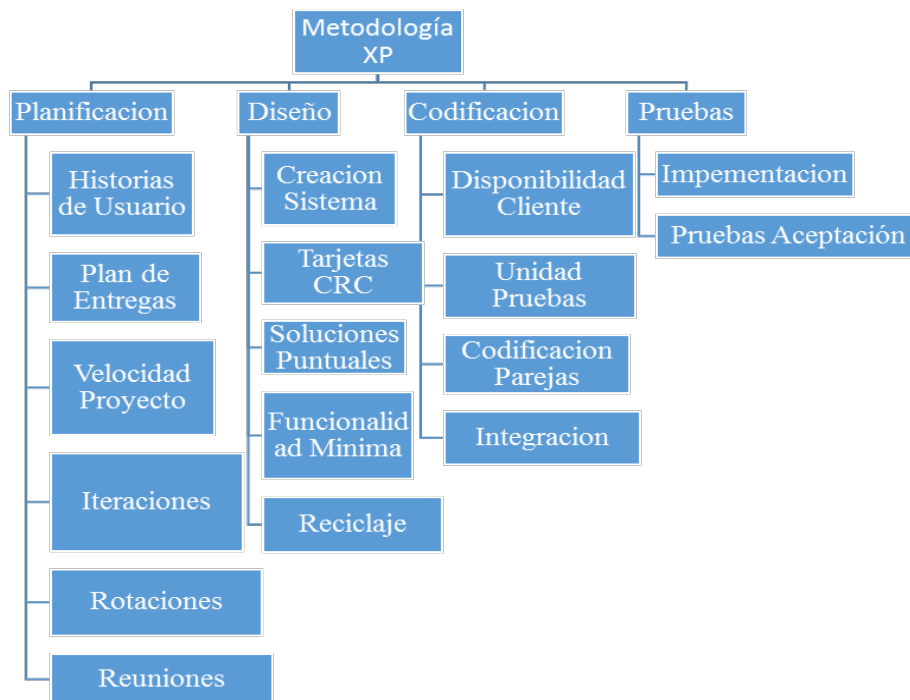


Figura 10: Fases XP
Fuente: Autoría Propia

2.3.4 ARTEFACTOS DE XP

Tabla 4: Artefactos

Historias de Usuario	<p>Establecen los requisitos del cliente.</p> <p>Las establece el cliente.</p> <p>Son la base para las pruebas de aceptación.</p> <p>Se caracterizan por prioridad, riesgo y esfuerzo.</p>
Tareas	<p>Se usan para describir la obligación que se tiene en cada historia de usuario.</p> <p>Se clasifican de acuerdo a su objetivo puede ser: desarrollo, diseño, instalación, etc.</p>
Tarjetas CRC(Clase-Responsable-Colaborador)	<p>Se divide en tres secciones para mejor el desempeño a lo largo del proyecto:</p> <p>Clase: Cualquier persona, cosa, evento, concepto, pantalla o reporte.</p> <p>Responsabilidad: Son las cosas que conocen y las que realizan, sus atributos y métodos.</p> <p>Colaboradores: Son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.</p> <p>Nota: Esta clase de artefactos no se tomaran en el transcurso del desarrollo del aplicativo.</p>

Fuente: Autoría propia

2.3.5 PRACTICAS XP

En esta metodología cabe resaltar la importancia del cliente, las pruebas, la refactorización, la simplicidad, la propiedad colectiva del código que se ven reflejadas en las cuatro prácticas esenciales de XP.(Gómez, Duarte e Güevara, 2014)

- ✓ **Entregas limitadas o pequeñas.-** Se entrega avances de módulos del sistema, pero no significa que las tareas queden sin terminar al contrario lo que garantiza estas entregas es la satisfacción del cliente
- ✓ **Semanas de trabajo de 40 Horas.-** En esta metodología se trabaja intensamente, no se labora horas extras ya que se busca utilizar al máximo las energías del programador,
- ✓ **Cliente en el Sitio.-** Se mantiene reuniones directas con el cliente ya que es vital que este activo durante todo el proceso desarrollo.
- ✓ **Programación en pareja.-** Lo que se trata de hacer es optimizar el código, ahorrar tiempo y tener un aplicativo de calidad para el cliente.

2.4 METODOLOGÍAS DE PENTESTING

2.4.1 METODOLOGÍA OSSTMM (OPEN SOURCE SECURITY TESTING METHODOLOGY MANUAL)

Es un manual de metodología abierta que sirve para realizar pruebas y analizar la seguridad en un ambiente operativo, se basa y utiliza OML¹², que está publicado y liberado bajo la licencia de bienes comunes creativos (Licencia Creative Commons).

¹² OML(Open Methodology License).- Es una metodología de licencia abierta

Pete Herzog director de la ISECOM ¹³ (Hezrog, 2015) exige que un test de seguridad sea considerado un test OSSTMM¹⁴ si y solo si cumple los siguientes puntos:

- ✓ Cuantificable.
- ✓ Consistente y que se pueda repetir.
- ✓ Valido más allá del período tiempo actual.
- ✓ Basado en el mérito del testeador y analista, y no en marcas comerciales.
- ✓ Exhaustivo.
- ✓ Concordante con leyes individuales y locales y el derecho humano a la privacidad.

OSSTMM es el proyecto más destacado de la ISECOM, pero existen otras metodologías que son patrocinadas por la misma institución.

- ✓ **Scare.-** Se encarga de analizar códigos fuentes y evalúa riesgos
- ✓ **Child Safety and Security Methodology.** - Es una metodología que enseña seguridad a los niños a través de juegos e historias.
- ✓ **Home Security Methodology.-** El objetivo de esta metodología es mantener seguros a los hogares y protegerlos de cualquier amenaza.
- ✓ **National Security Methodology.-** Se define políticas y estándares para mejorar la seguridad nacional.

¹³ **ISECOM.-** Instituto para la seguridad y metodologías abiertas

¹⁴ **OSSTM (OPEN SOURCE SECURITY TESTING METHODOLOGY MANUAL).-** Manual de la metodología que sirve para corregir riesgos a nivel de infraestructura.

2.4.1.1 MÓDULOS DE SEGURIDAD

Esta metodología propone un esquema o mapa de seguridad que consta de seis secciones tal como indica la figura.

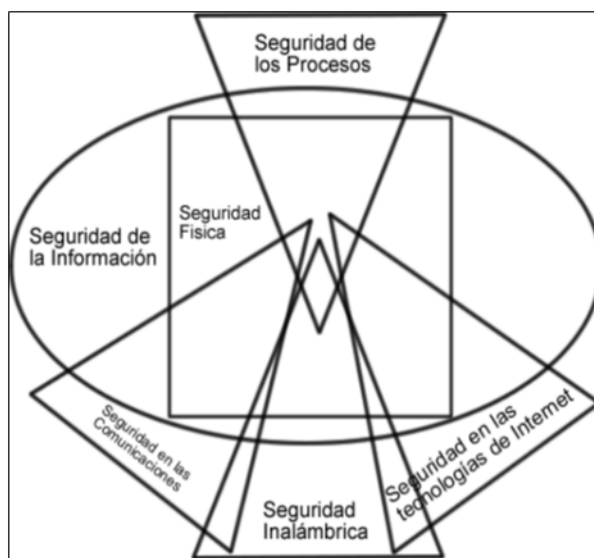


Figura 11: Esquema o Mapa de Seguridad

Fuente: OSSTMM recuperado de <http://isecom.securenetsltd.com/OSSTMM.es.2.1.pdf>, pag.23

Existen varios módulos con los que cuenta este esquema los cuales son integradas con procesos y tareas a desarrollarse todas las secciones dentro de cada módulo deben cumplirse para tener un mayor seguridad al nivel de los sistemas, los procesos que no cumplan con lo que dice la metodología se deben definir como no aplicable.

Tabla 5: Módulos y secciones de la Metodología OSSTMM

Módulos	Secciones
Seguridad de la Información	Revisión de la Inteligencia Competitiva Revisión de Privacidad Recolección de Documentos
Seguridad de los Procesos	Testeo de Solicitud Testeo de Sugerencia Dirigida

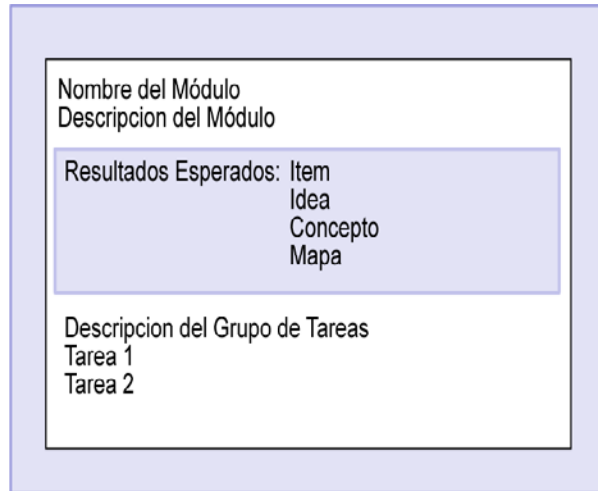
	Testeo de las Personas Confiables
Seguridad en las tecnologías de Internet	<p>Logística y Controles</p> <p>Sondeo de Red</p> <p>Identificación de los Servicios de Sistemas</p> <p>Búsqueda de Información Competitiva</p> <p>Revisión de Privacidad</p> <p>Obtención de Documentos</p> <p>Búsqueda y Verificación de Vulnerabilidades</p> <p>Testeo de Aplicaciones de Internet</p> <p>Enrutamiento</p> <p>Testeo de Sistemas Confiados</p> <p>Testeo de Control de Acceso</p> <p>Testeo de Sistema de Detección de Intrusos</p> <p>Testeo de Medidas de Contingencia</p> <p>Descifrado de Contraseña</p> <p>Testeo de Denegación de Servicios</p> <p>Evaluación de Políticas de Seguridad</p>
Seguridad en las Comunicaciones	<p>Testeo de PBX</p> <p>Testeo del Correo de Voz</p> <p>Revisión del FAX</p> <p>Testeo del Modem</p>
Seguridad Inalámbrica	Verificación de Radiación Electromagnética

	<p>(EMR)</p> <p>Verificación de Redes Inalámbricas [802.11]</p> <p>Verificación de Redes Bluetooth</p> <p>Verificación de Dispositivos de Entrada Inalámbricos</p> <p>Verificación de Dispositivos de Mano Inalámbricos</p> <p>Verificación de Comunicaciones sin Cable</p> <p>Verificación de Dispositivos de Vigilancia Inalámbricos</p> <p>Verificación de Dispositivos de Transacción Inalámbricos</p> <p>Verificación de RFID</p> <p>Verificación de Sistemas Infrarrojos</p> <p>Revisión de Privacidad</p>
Seguridad Física	<p>Revisión de Perímetro</p> <p>Revisión de monitoreo</p> <p>Evaluación de Controles de Acceso</p> <p>Revisión de Respuesta de Alarmas</p> <p>Revisión de Ubicación</p> <p>Revisión de Entorno</p>

Fuente: OSSTMM recuperado de <http://isecom.securenetsltd.com/OSSTMM.es.2.1.pdf>, pag.25

2.4.1.2 PLANTILLA DE PRUEBAS DE DATOS DE LA SEGURIDAD

La metodología utiliza una serie de reglas, lineamientos y métodos para realizar y garantizar el testeo de seguridad que se aplica a cada sección, se utiliza un artefacto el cual nos ayuda para mitigar las tareas y evaluar el riesgo que existe.



Nombre del Módulo
Descripción del Módulo

Resultados Esperados: Item
Idea
Concepto
Mapa

Descripción del Grupo de Tareas
Tarea 1
Tarea 2

Figura 12: Plantilla de Pruebas de seguridad
Fuente: <http://isecom.securenetltd.com/OSSTMM.es.2.1.pdf>

2.4.1.3 EVALUACIÓN DEL RIESGO

Esta evaluación está a cargo del analista, el comprueba que todos los datos obtenidos sirvan de guía para obtener una evaluación válida, dicha evaluación depende de la información proporcionada por la empresa. Existen cuatro aspectos básicos pero fundamentales para minimizar cualquier estado de riesgo en el ambiente.

- ✓ **Seguridad.**- Todos los test se deben realizar con la mayor precaución posible para evitar grandes pérdidas. El analista debe dar prioridad al respeto por seguridad humana ya sea salud física, emocional y ocupacional.
- ✓ **Privacidad.**- Se debe cumplir el derecho a la privacidad personal sin importar las leyes y legislaciones regionales.

- ✓ **Practicidad.-** Las pruebas deben estar bien diseñadas capaz de encontrar el mínimo error.
- ✓ **Usabilidad.-** Los test deben ser entendibles y capaces de encontrar una seguridad útil.

2.4.1.4 TIPOS DE RIESGOS

Son fallas que se presentan en cualquier tipo de controles que separan los activos de las amenazas de una institución.

- ✓ **Vulnerabilidad.-** Se basa en analizar el error que puede producir un acceso no autorizado o a un acceso legítimo.(Hezrog, 2015)
- ✓ **Debilidad.-** Es cuando no existe un control de interacciones de autenticación con el usuario ya que no posee limites en la cantidad de intentos.
- ✓ **Preocupación.-** Existe este tipo de riesgo cuando hay un fallas en el control de procesos ya sea en cuanto a la confidencialidad, privacidad y la integridad.

Un ejemplo de preocupación es un proceso que genere archivos de log con los datos de los participantes involucrados pero no almacena correctamente la fecha y la hora de transacción.

- ✓ **Exposición.-** Se encarga de contabilizar las acciones no justificadas que dejan al descubierto los activos ya sea de forma directa o indirecta. Un ejemplo sobre exposición son los banners que brindan información de la aplicación que se encuentran ejecutándose detrás de un puerto específico.
- ✓ **Anomalía.-** Se produce cuando un elemento desconocido no puede identificarse en las operaciones normales y puede generar problemas de seguridad en un futuro.

Un ejemplo de anomalía se presenta cuando en una red de datos en una respuesta ICMP proviene de una dirección IP no existente.

2.4.2 METODOLOGÍA OTP (OWASP TESTING PROJECT)

Es un proyecto que está diseñado para prevenir y testear aplicaciones web consta de una metodología que está pensada para ser puesta en práctica desde el punto de vista del codificador, básicamente OWASP se basa en un test de aplicación web de caja negra, aquí el Hacker Ético no conoce nada o tiene poca información sobre la aplicación a ser evaluada.(Salgado Yáñez, 2014)

Este tipo de pruebas consiste de los siguientes aspectos:

- ✓ El Evaluador o Testeador.- quien está encargado de todas las actividades de pruebas que se realizan en la aplicación.
- ✓ Herramientas y Metodología.- las herramientas que se utilizan en este tipo de pruebas son de gran ayuda ya que ayudan a descubrir puertas abiertas y tener éxito al realizar los ataques, al igual que la metodología que ayuda a seguir un proceso para impedir vulnerabilidades en la aplicación.
- ✓ La aplicación.- que es la caja negra a evaluar.

La prueba realizada en cualquier aplicación de tipo web consta de dos fases:

Tabla 6: Fases de una prueba en la metodología

FASES	DETALLE
Pasiva	<p>Se realiza pruebas hasta conocer la lógica de la aplicación que está en modo de test y verificar si existe algún punto de acceso abierto.</p> <p>Se puede utilizar herramientas para poder capturar o recopilar información ya sean (cookies, parámetros, cabeceras HTTP).</p>

Activa	<p>El evaluador realiza los procesos recomendados que demanda la metodología.</p> <p>Esta fase consta de 9 categorías de 66 procesos en total las cuales son</p> <p>Pruebas de Manejo de la Configuración(Información de recolección + Gestión de la configuración)</p> <p>Pruebas de la Lógica del Negocio</p> <p>Pruebas de Autenticación</p> <p>Pruebas de Autorización</p> <p>Pruebas de Manejo de la Sesión</p> <p>Pruebas de Validación de los Datos</p> <p>Pruebas de Negación de Servicio</p> <p>Pruebas de Servicios Web</p> <p>Pruebas Ajax.</p>
--------	---

Fuente: Autoría propia

2.4.2.1 ETAPAS DE LA SEGURIDAD OTP

- ✓ **Recolección de Información.-** Es una etapa que demanda mucha dedicación y tiempo ya que mayor sea la cantidad y la calidad de la información recauda, existirá la posibilidad de encontrar vulnerabilidades o backdoors, esta etapa es la integral dentro de cualquier metodología de testing de caja negra donde se asume que el atacante no cuenta con información técnica detallada que se utilizado para la construcción del sistema. Para lograr conseguir toda la información y su plataforma utilizada existen una variedad de recursos como pueden ser: inspección automatizada o manual de robots, uso de motores de búsqueda (google, Firefox), reconocimiento de parámetros GET/POST, descubrimiento detallado de la aplicación plataforma utilizada y lenguajes de programación.(Owasp, 2015)

- ✓ **Gestión de la configuración.-** Permite realizar un análisis de la arquitectura y topología de la aplicación, el objetivo principal en esta etapa es el servidor en que se encuentra alojado la aplicación, algunas pruebas que se realizan en esta etapa son: verificación sobre los certificados tanto SSL/TLS, Verificación sobre el servicio de Base Datos (listener de conexiones) con el fin de buscar cualquier tipo de mala configuración o “exposición” de datos sensibles, Verificación de extensiones de las paginas (*.php, *.asp, *.jsp, etc.) además de verificación sobre los tipos MIME¹⁵ soportados por el servidor web
- ✓ **Pruebas de Autenticación.-** Se realiza este tipo de pruebas en el caso de que la aplicación contenga formularios de accesos privilegiado para cada usuario se pueden realizar pruebas como por ejemplo: Verificación de credenciales viajando sobre canales cifrados y no cifrados, Si la aplicación permite la opción de “recordar password”, analizar el comportamiento de dicha funcionalidad así como también analizar el comportamiento de otras funcionalidades relacionadas, Prueba de Cierre de Sesión y Gestión de Cache de Navegación.(Owasp, 2015)
- ✓ **Gestión de Sesión.-** En esta etapa interviene un campo muy importante que es el protocolo HTTP ya que sirve para que interactúe el usuario con el servidor y se almacene toda la información que el usuario proporcione, es aquí donde el tester realiza pruebas para romper y obtener las sesiones de los usuario, las pruebas relacionadas a esta etapa son: Pruebas sobre los atributos de las cookies generadas para mantener el estado de las sesiones en el lado del cliente, pruebas de XSS¹⁶

¹⁵ **MIME (Multipurpose Internet Mail Extensions).** - Son especificaciones dirigidas al intercambio por medio de internet cualquier tipo de archivos sean estos texto, audio, video.

¹⁶ **XSS (Cross-Site Scripting).**- Es un tipo de vulnerabilidad donde se ejecuta código de JavaScript para poder obtener información, inicio de sesiones etc.

- ✓ **Pruebas de Autorización.-** Es donde se consultan los roles, permisos y privilegios que tiene un usuario para poder acceder a diferentes recursos de la aplicación, las pruebas que se pueden realizar son: Prueba para evitar esquema de autorización, Pruebas de elevación de privilegios para determinar si es posible que un usuario pueda elevar su rol y acceder a recursos para los que inicialmente no debería acceder.
- ✓ **Pruebas de Lógica de Negocio.-** Se emplea mucho tiempo y creatividad ya que es donde el tester o intruso verifica la funcionalidad del sistema realizándose preguntas como el que pasa si, es una de las pruebas más difíciles ya que no existen ayudas como herramientas automatizadas que ayuden a verificar las función del aplicativo, el propio testeador debe hacer uso de sus conocimientos y habilidades para realizar este tipo de test.
- ✓ **Pruebas de Validación de datos.-** Una debilidad muy común en las aplicaciones son las validaciones de las entradas que proporciona el cliente o a veces de la interfaz de la aplicación, es por esto que se deben tomar medidas de control sobre el tipo de dato que se ingresa en una aplicación, las pruebas más comunes que realizan son: inyecciones de código SQL e inyecciones de código JavaScript.
- ✓ **Pruebas de Denegación de Servicios.-** Básicamente consiste en saturar con peticiones a la aplicación que se encuentra del lado del servidor. Las pruebas más usadas para este tipo de pruebas son el bloqueo de cuentas de usuarios, almacenamiento de demasiados datos en Sesión.

- ✓ **Pruebas de Servicios Web.-** Las vulnerabilidades que existen en este tipo de pruebas van orientadas más a los archivos XML que son intercambiados con el cliente y el servidor, las pruebas que usan en este tipo de pruebas es: Prueba en la estructura del XML.
- ✓ **Pruebas Ajax.-** Es una técnica que se usa mucho en páginas web para dinamizar las respuestas del lado del servidor, una técnica que se usa para este tipo de pruebas es la prueba de vulnerabilidades de Ajax, en concreto, las relacionadas con el objeto XMLHttpRequest.

Cabe mencionar que en esta sección de la metodología se utilizara las pruebas de validación a datos Inyección SQL y XSS.

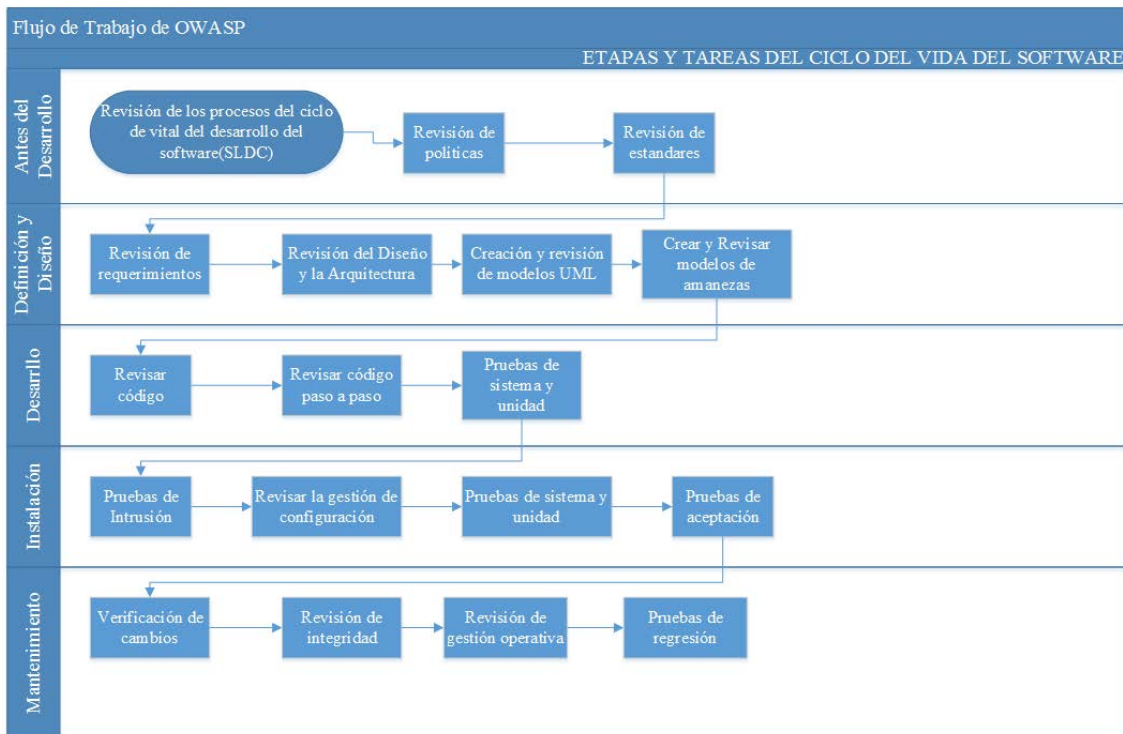


Figura 13: Test de Seguridad en el ciclo de desarrollo de una aplicación
Fuente: Autoría propia

2.4.3 COMPARATIVA DE LAS METODOLOGÍAS

A continuación se detalla un cuadro comparativo de las metodologías más utilizadas como son OSSTMM¹⁷ y OWASP¹⁸.

Tabla 7: Tabla comparativa de las dos metodologías

METODOLOGÍA ITEMS	OSSTMM	OWASP
Rigor de la metodología(meticulosidad)	Muy Alta. Está en constante revisión y actualización.	Muy Alta. Se centra solo en la parte web, es muy didáctica e instructiva.
Niveles de detalle	Menos detallada debido a los constantes actualizaciones	Muy detallada. Se orienta perfectamente en el trabajo del auditor
Facilidad de Uso	Media ya que hace falta entrenamiento y práctica.	Alta, muy técnico aunque hace enfoque a las herramientas, sugiere usos y muestra ejemplos
Ámbitos de aplicación	General. Todo tipo de organizaciones y pymes, instituciones educativas.	Todo tipo de organizaciones orientadas a la web.
Entornos de aplicabilidad	Todos. Incluso los que no se han implementado, es dinámico y potente en el diseño.	Solo los entornos web y aplicaciones enfocadas a la web.
Uso por los auditores	Es el más usado en el mercado, sugiere un experiencia muy alta	No es muy usado, pero cuando se aplica es muy buena debido a su madurez y detalle.

¹⁷ OSSTMM.- Manual de metodologías Abiertas para el testeado de seguridades de infraestructura.

¹⁸ OWASP.- Proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro

Ventajas	Ampliamente documentada, utiliza plantillas para prevenir riesgos innecesarios ya que se integra y cumple todos los estándares que emite la seguridad de la información.	Facilidad de uso de los controles más conocidos en el Top Ten. Está muy estructurado, hace hincapié en las auditorías web que se asemeja en el marketing y en el negocio, Sirve para conocer los fundamentos de las auditorías y como funciona los pentesting.
Inconvenientes	No recomienda herramientas, se basa en la creatividad del auditor, no permite adaptarse con otras metodologías.	Habría que combinar con otras herramientas y metodologías para que la auditoría fuese más completa. La estructura o anatomía de la web y los servidores no son auditadas.

Fuente: Maximiliano Pérez, Elche, Marzo 2012

NOTA: De acuerdo al estudio de ambas metodologías, se escoge a OWASP porque está enfocado a la parte web.

2.5 KALI LINUX

Es una distribución Linux gratuita basada en Debian y desarrollado por una organización llamada Offensive Security expertos en auditoría y seguridad informática, es un proyecto que se creó a través de otro proyecto llamado Backtrack que actualmente está fuera de servicio debido a que se unió con Kali. Kali, actualmente consta con más de 300 programas destinados a la realización y verificación de ataques, básicamente a lo que corresponde con seguridad informática y no para delitos informáticos, siendo algunas de las más conocidas Nmap (un escáner de puertos), Wireshark (un sniffer), John the Ripper (Un crackeador de passwords) y la suite Aircrack-ng (Software para pruebas de seguridad en redes inalámbricas), además del framework Metasploit, que sirve para encontrar vulnerabilidades en sistemas informáticos especialmente enfocados a la web.

2.6 HERRAMIENTAS DE DESARROLLO

2.6.1 LENGUAJE DE PROGRAMACION PHP

Es un lenguaje de alto nivel y se ejecuta del lado del servidor y fue creado para la implementación de sitios web estáticos, interactivos y dinámicos se puede crear una variedad de páginas web debido a que posee una gran cantidad de librerías, entre su funcionalidad permite conectarse con base de datos relacionales. (Anabell, 2004)

2.6.2 SERVIDOR WEB APACHE

Es un programa que se ejecuta al lado del servidor, es de código abierto y se lo puede ejecutar en diversas plataformas como por ejemplo Linux, Microsoft y Mac. Cabe mencionar que este servidor es el más utilizado al nivel mundial debido a su gran estabilidad, versatilidad y confiabilidad. (Asensio Hildago, 2014)

2.6.3 MVC (MODELO, VISTA, CONTROLADOR)

Es un patrón de diseño creacional, que ayuda a construir el esqueleto para dar solución a problemas comunes en el desarrollo de software, este patrón plantea la solución en tres capas que son:

- ✓ **El modelo.-** que representa la realidad o la lógica del negocio.
- ✓ **El controlador.-** aquí se conoce todos los métodos y atributos del modelo, procesa las interacciones con el usuario.
- ✓ **La vista.-** convierte a las anteriores capas en una página web para que pueda el usuario interactuar con dichas capas.

2.6.4 FRAMEWORK SYMFONY

Es un framework PHP que utiliza un patrón de diseño MVC, symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.(Fabien Potencier, 2013).

Este framework utiliza algunos componentes muy útiles para mejorar y optimizar el desarrollo y son los siguientes:

Tabla 8: Componentes de PHP y Symfony

COMPONENTES	DESCRIPCIÓN
ORM	Es un objeto relacional mapeador que se utiliza en la programación para transformar en clases, atributos y datos, todas las tablas columnas y relaciones de una base de datos.
DOCTRINE	Es una librería imitada de hibernate, que ayuda a crear una capa de persistencia para trabajar con objetos en php.
DQL	Es un lenguaje de consulta basado en doctrine, es semejante a las sentencias de SQL y se emplea para obtener objetos en lugar de tablas.
YAML	Formato de serialización de archivos que trabaja con datos nativos de lenguajes de programación

Fuente: Autoría Propia

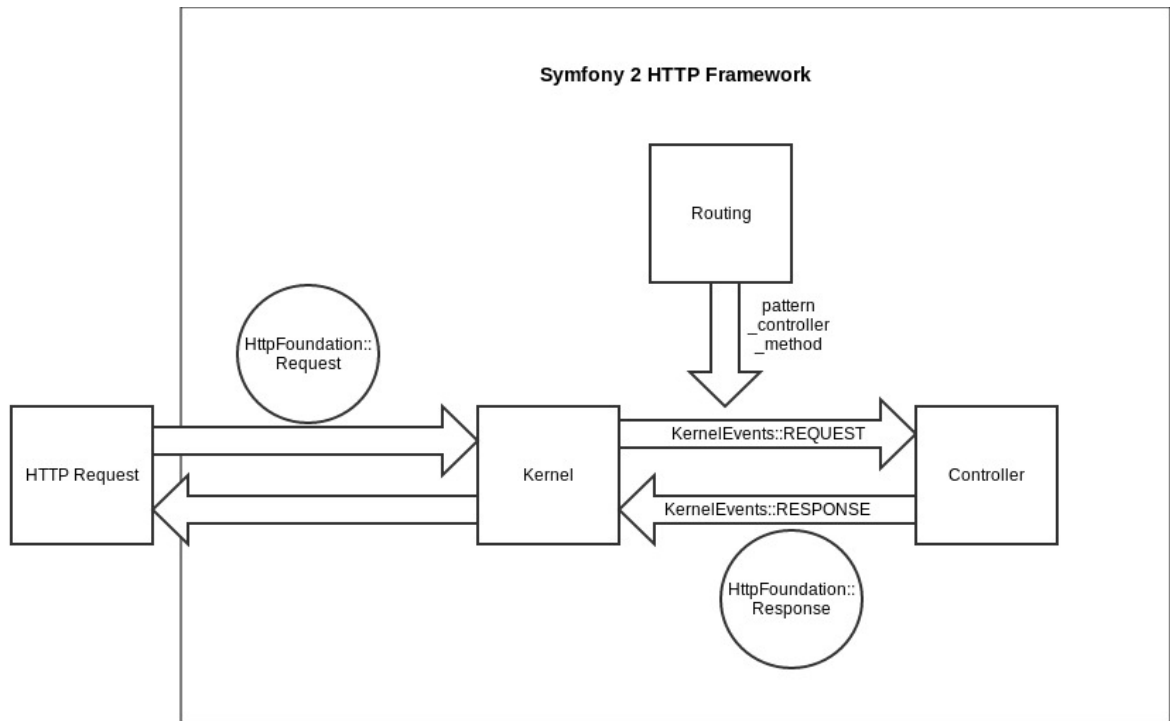


Figura 14: Arquitectura Interna de Symfony

Fuente: Carles Climent Granell 2013, recuperado de <https://github.com/carlescliment/cursos-symfony2/blob/master/1-introduccion/que-es-symfony.md>

2.6.5 BASE DE DATOS POSTGRES

Es un motor de base de datos dbms¹⁹ que sirve para la creación y acceso a los datos, se compone de un lenguaje ddl²⁰, de un dml²¹ y un sql²², y necesita de la ayuda de un intérprete como lo es pgadmin que sirve para interactuar y con la base de datos.

Postgres fue pionera en muchos conceptos que sólo estuvo disponible en algunos sistemas de bases de datos comerciales mucho más tarde. (Momjian, 2014)



Figura 15: Logotipo de la base de datos Postgresql

Fuente: yesbev 2014, recuperado de <http://yevbes.es/instalacion-de-postgresql>

¹⁹ **DBMS.-** Sistema de administración de base de datos.

²⁰ **DDL(Data Definition Language).-** describe las estructuras de información y programas que usan para construir y actualizar información que contiene la base de datos

²¹ **DML(Data Manipulation Language).-**es utilizado para escribir programas que creen, extraigan y actualizasen información.

²² **SQL(Structured Query Language).-** sirve para extraer información de la base

3 VULNERABILIDADES INFORMÁTICAS

3.1 INTRODUCCIÓN

Una vulnerabilidad es una debilidad de un sistema ya sea software, hardware, datos, estos últimos son los más importantes dentro de una organización porque estos no se los puede reparar y a los otros componentes si se los puede reparar, cuando existe debilidades un atacante puede quebrantar la integridad y cometer fraudes, alterar aplicaciones, una vulnerabilidad ocurre por un bug en la aplicación o algún error en la codificación o a veces por descuidos del usuario al recordar sus contraseñas en navegadores.

3.2 TIPOS DE VULNERABILIDADES

Existen muchos riesgos en la seguridad informática, en esta sección se hablará de las vulnerabilidades más conocidas y el impacto que ocasionan en aplicaciones web.

Según la guía de pruebas de OWASP(Ten, 2013) ,clasifican los riesgos en una escala del 1 al 10 según al impacto de mayor peligro y de acuerdo a los métodos más usados para atacar aplicaciones web.

Tabla 9: Tipos de Vulnerabilidades

RIESGO	DESCRIPCIÓN
1. Inyección	Estas fallas ocurren cuando datos no confiables son enviados en forma de parámetros que se enlazan con la basa de datos.
2. Pérdida de Autenticación y Gestión de Sesiones	Las autenticaciones y sesiones en la codificación son mal implementadas, permitiendo a los intrusos robar cookies, contraseñas y poder usurpar la identidad del usuario.(Hermoso Metaute, 2013)
3. Secuencia de comandos en sitios cruzados (XSS)	Este riesgo se presenta cuando no existen controles y permite introducir o enviar código sin ningún tipo de validación, generalmente se usan scripts que actúan sobre HTTP.
4. Referencia directa insegura a objetos	Ocurre cuando el desarrollador apunta hacia objetos internos que pueden ser archivos, directorios, base de datos etc., sin un debido control el atacante puede manipular las referencias para acceder a datos sensibles.
5. Configuración de seguridad incorrecta	Se debe asegurar y tener implementados políticas para tener una buena configuración ya sea en plataformas, base de datos, servidores, aplicaciones etc. Debido a que no son seguras por defecto.
6. Exposición a datos sensibles	Varias aplicaciones web no protegen sus datos sean estos tarjetas de crédito, contraseñas, lo que hace que el sistema sea vulnerable, estos datos requieren de mayor protección tratando de cifrar los datos en cuestión.

7. Ausencia de control de acceso a funciones	Este riesgo es muy común en el desarrollo de aplicaciones debido a que no existe un debido control de acceso en las funciones tanto del servidor como en la interfaz que se hace al usuario.
8. Falsificaciones de peticiones en sitios cruzados (CSRF).	Este ataque se presenta cuando se obliga al usuario autenticado a enviar peticiones HTTP falsas sean estos inicios de sesión, información de la autenticación, esto permite que el intruso pueda generar pedidos que el sistema piensa que son ordenes generadas por la víctima.
9. Utilización de componentes con vulnerabilidades conocidas	La mayoría de las aplicaciones utilizan librerías, framework el cual si se conoce la vulnerabilidad del componente facilitaría al atacante a cometer sus fraudes llegando a comprometer datos sensibles incluso borrando todo el contenido del servidor.
10. Redirecciones y envíos no validos	Frecuentemente las páginas web redirigen a otras páginas a los usuarios sin validaciones apropiadas, los atacantes pueden reenviar a la víctima a sitios de phishing ²³ y obtener información valiosa de la víctima.

Fuente: Autoría propia

²³ **Phishing.-** Técnica que se usa para usurpar la identidad de la víctima para poder obtener información.

3.3 SQL INJECTION

Es un método muy utilizado a nivel mundial para detectar anomalías en aplicaciones web, este método consiste en la comunicación con la base de datos mediante sentencias SQL a través de un navegador web, cuyo objetivo es extraer información posible que se encuentra almacenado en una base de datos.(De La Quintanaillanes, 2013).

Existen varios comandos genéricos que utilizan la gran mayoría de base de datos, los que se listaran a continuación:

Tabla 10: Comandos generales DCL en una base de datos

COMANDOS DCL(Data Control Language Statements)	
GRANT	Se utiliza para otorgar permisos.
REVOKE	Se utiliza para quitar permisos.
DENY	Se utiliza para denegar acceso.

Fuente: Hernán Marcelo Racciatti

Tabla 11: Comandos generales DDL en una base de datos

COMANDOS DDL(Data Definition language Statements)	
CREATE	Se utiliza para crear nuevas tablas, campos e índices.
DROP	Se utiliza para borrar tablas e índices.
ALTER	Se emplea cuando se modifica tablas agregando campos o cambiando definiciones en los campos a alterar.

Fuente: Hernán Marcelo Racciatti

Tabla 12: Comandos generales DML en una base de datos

COMANDOS DML (Data Manipulation Language Statemnets)	
SELECT	Sirve para realizar consultas a registros de la base de datos.
INSERT	Se emplea para cargar gran cantidad de datos.
UPDATE	Empleado para modificar los valores en los registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de la base de datos.

Fuente: Hernán Marcelo Racciatti

Tabla 13: Clausulas generales en una base de datos

CLAUSULAS	
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.

Fuente: Hernán Marcelo Racciatti

Tabla 14: Operadores de comparación en una base de datos

Operadores de Comparación	
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en comparación de un modelo
IN	Utilizado para especificar registros de una base de datos

Fuente: Hernán Marcelo Racciatti

3.3.1 INYECCIONES MEDIANTE SALIDAS DE VARIABLES

Este tipo de inyecciones son muy comunes y básicas al momento de realizar pruebas para saber si existe vulnerabilidad, se llama salida de variable porque lo que se hace es inyectar algo diferente en los cuadros de texto que no se lo común a lo que el usuario escribe, por ejemplo tenemos la siguiente consulta.

```
Select * from usuarios where usuario ='$usuario' and clave='$clave';
```

Tabla 15: Vulnerabilidad y Solución del Ataque mediante variables

Vulnerabilidad	Solución
comillas dobles (“), las comillas simples (‘)	Escapar los caracteres especiales utilizados en las consultas SQL, utilizando funciones de los lenguajes como por ejemplo en php htmlspecialchars , o a nivel de base de datos como en postgres pg_escape_string

Fuente: Autoría Propia

3.3.2 INYECCIÓN CON DATOS NUMÉRICOS

Es aconsejable delimitar siempre entre comillas a los campos enteros dentro de las sentencias SQL porque resulta más difícil inyectar como ejemplo:

Tabla 16: Vulnerabilidad y Solución del Ataque mediante datos numéricos.

Vulnerabilidad	Solución
SELECT nombre FROM usuarios WHERE id_user = \$id	SELECT nombre FROM usuarios WHERE id_user = ‘\$id’

Fuente: Autoría Propia

Los id o datos numéricos por lo general se muestran en las urls de los navegadores, si al final del id=12, se le añade un punto y coma (;) y una consulta adicional sin el punto y coma (;) se puede hacer mucho daño a la base de datos, un ejemplo de ello podría ser:

```
Select * from usuarios where id=12; insert into usuarios values(x,x,x,x,x)
```

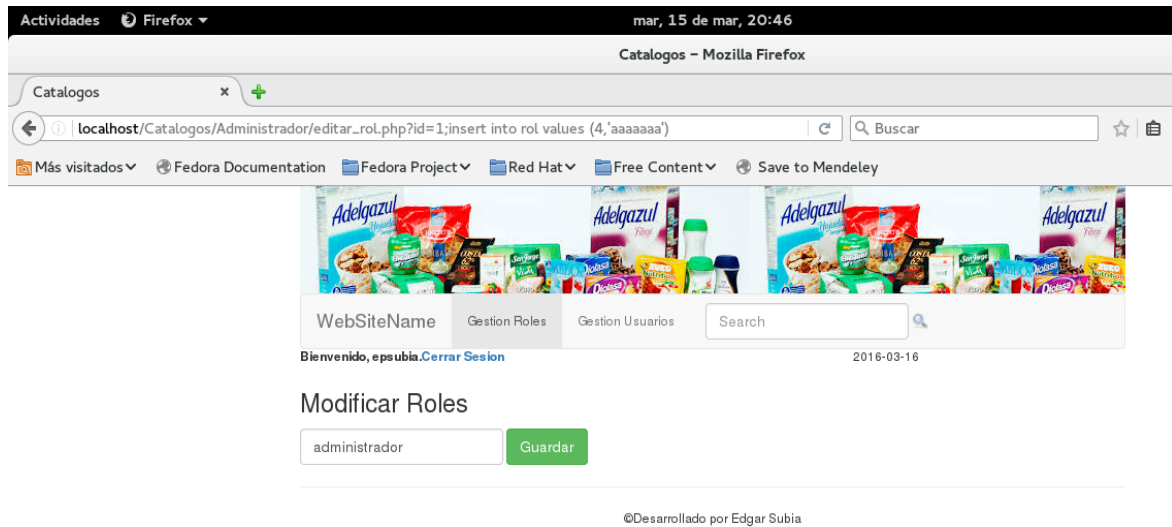


Figura 16: Inyección con dato numérico
Fuente: Autoría Propia

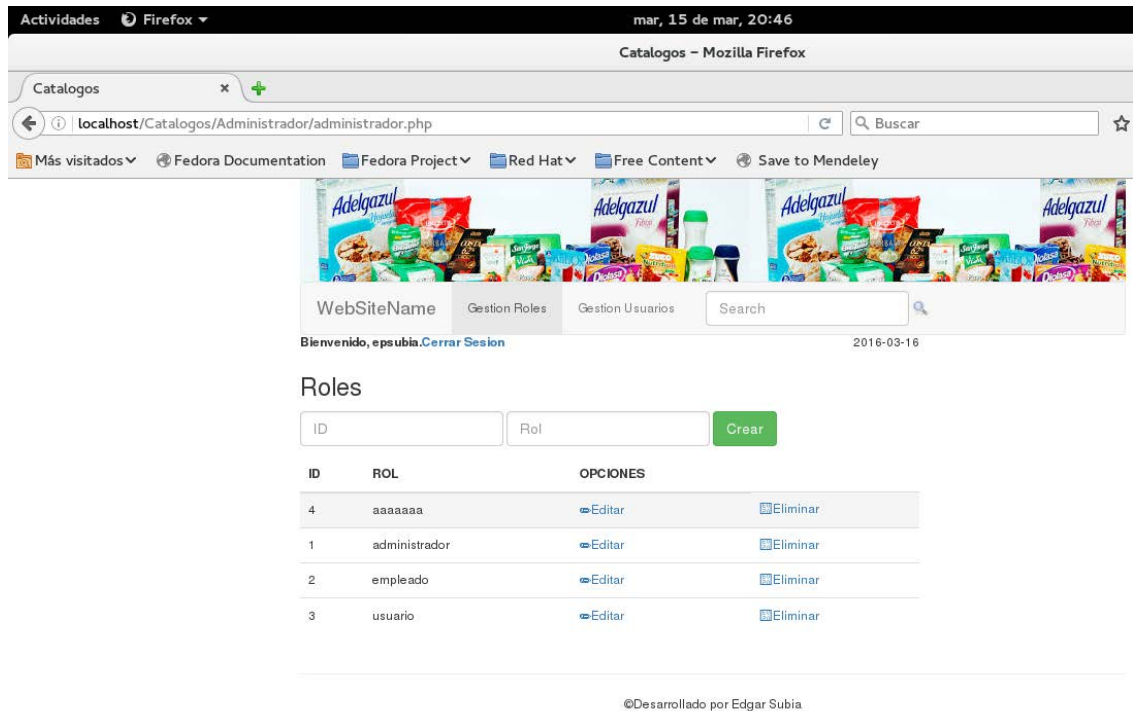


Figura 17: Inyección exitosa inserción en la base de datos
Fuente: Autoría Propia

3.3.3 INYECCIÓN POR MEDIO DE COMENTARIOS

Existen dos formas de realizar este ataque y son por medio comentarios a través de una sola línea (--), y comentarios multilinea (/ * y */).

- Una sola línea.- Depende mucho el motor de base que se está usando, en este caso se está usando postgres por lo que se utiliza -- con un espacio, aplicando a la consulta siguiente:

```
Select * from usuarios where usuario ='usuario' and clave='Sclave';
```

Al final del nombre de usuario se añade un (;) y después se añade el comentario quedaría así:

```
Select * from usuarios where usuario ='usuario'; -- and clave='Sclave';
```


- Comentario Multilínea.- Nos ayuda a excluir comandos que van en las consultas SQL, un ejemplo puede ser el siguiente:

```
Select * from bd_tuto where usuario = 'ejemplo' /* ' and password = 'blablabla'  
and ID=65535 */;
```

3.3.4 INYECCIÓN MEDIANTE CONSULTAS CONCATENADAS Y NOMBRES DE TABLAS

Hasta el momento se ha analizado técnicas, el cual impide realizar consultas propias, en esta inyección, se hará uso de una tabla general que se llama information_schema.tables, el cual tiene una columna llamada tabla_name, en donde se conocerá todos los nombres de las tablas que existe en dicho esquema.

Para lo cual se procede a unir dos o más consultas, pero el número de tablas que devuelve la consulta original e inyectada deben ser iguales, para controlar el número de tablas de la consulta inyectada solo se debe añadir columnas falsas por ejemplo si la consulta inyectada posee 4 columnas, la sentencia quedaría de la siguiente manera:

```
Select table_name, 2, 3, and 4 from information_schema.tables;
```

En donde las columnas 2,3,4 son números inútiles, pero que cumplen con la función de regular con el número de columnas existentes en la consulta inyectada, ahora solo queda conocer la cantidad de columnas que existe en la consulta original, y esto se logra con hacer utilizando la cláusula order by que va después de un where, esto permite conocer su cantidad de columnas ya que mientras se ordena por columna existente la sentencia es verdadera y procede a seguir realizando el análisis y si no cumple se genera un error. Por ejemplo:

order by 1 El resultado será correcto

order by 2 El resultado será correcto

order by 3 El resultado será correcto

order by 4 Habrá un error (esa columna no existe)

Por ultimo solo queda en unir las consultas con un **select**, la primera sentencia termina sin punto y coma (;), y sigue con un **unión** seguido de la segunda sentencia **select**, el ejemplo terminado quedaría así:

```
Select * from usuarios where id=12 union select table_name,2,3 from information_schema.tables;
```

3.3.5 INYECCIÓN LIMITANDO RESULTADOS

Existirá consultas que se deban realizar limitaciones y se debe usar la instrucción **limit**, esta instrucción debe ir después de un **select** para decidir qué resultados mostrar. Un ejemplo podría ser:

```
Select * from usuarios where usuario = 'admin' limit 0,1 ; -- ' and password = '$password';
```

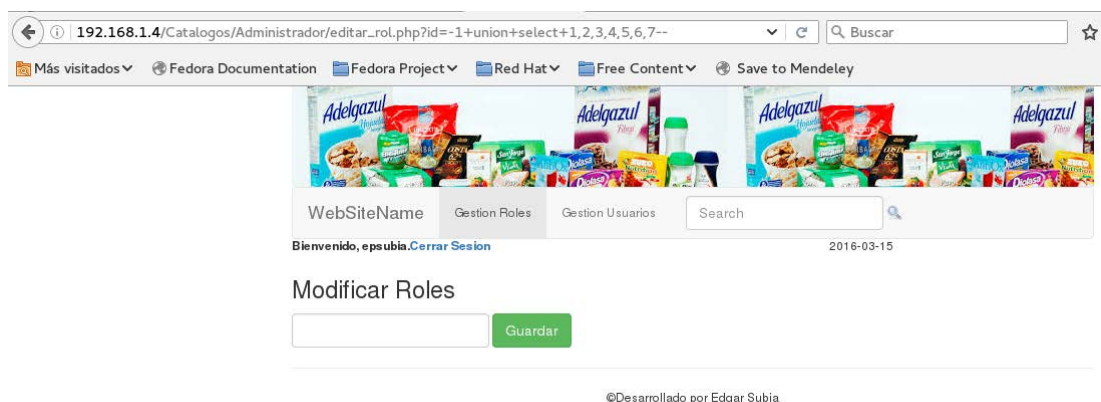


Figura 18: Inyección mediante consultas concatenadas, nombres de tablas y limitando resultados
Fuente: Autoría Propia

```

Administrator: C:\Windows\system32\cmd.exe
C:\sqlmap>sqlmap.py -u http://192.168.1.4/Catalogos/Administrador/editar_rol.php?id=1 --dbs

```

Figura 19: Inyección mediante consultas concatenadas, nombres de tablas y limitando resultados con software SQLMAP
Fuente: Autoría Propia

```

Administrator: C:\Windows\system32\cmd.exe
[14:14:39] [INFO] resumed: "telefono", "int4"
[14:14:39] [INFO] resumed: "usuario", "varchar"
[14:14:39] [INFO] resumed: "clave", "varchar"
[14:14:39] [INFO] resumed: "id_rol", "int4"
Database: public
Table: usuario
18 columns
+-----+-----+
| Column      | Type      |
+-----+-----+
| apellido_usu | varchar   |
| clave        | varchar   |
| email        | varchar   |
| id_rol       | int4      |
| id_usuario   | int4      |
| nombre_usu   | varchar   |
| telefono     | int4      |
| usuario      | varchar   |
+-----+-----+
[14:14:39] [INFO] fetched data logged to text files under 'C:\Users\Edguitar\sqlmap\output\192.168.1.4'
C:\sqlmap>sqlmap.py -u http://192.168.1.4/Catalogos/Administrador/editar_rol.php?id=1 -D public -I usuario -C usuario,clave --dump

```

Figura 20: Resultados de la inyección con Software SQLMAP
Fuente: Autoría Propia

3.3.6 DEMOSTRACIÓN

1.- Se ingresa en los cuadros de texto de login caracteres especiales como(‘,”,@,#),etc. Para ver si se produce un error.

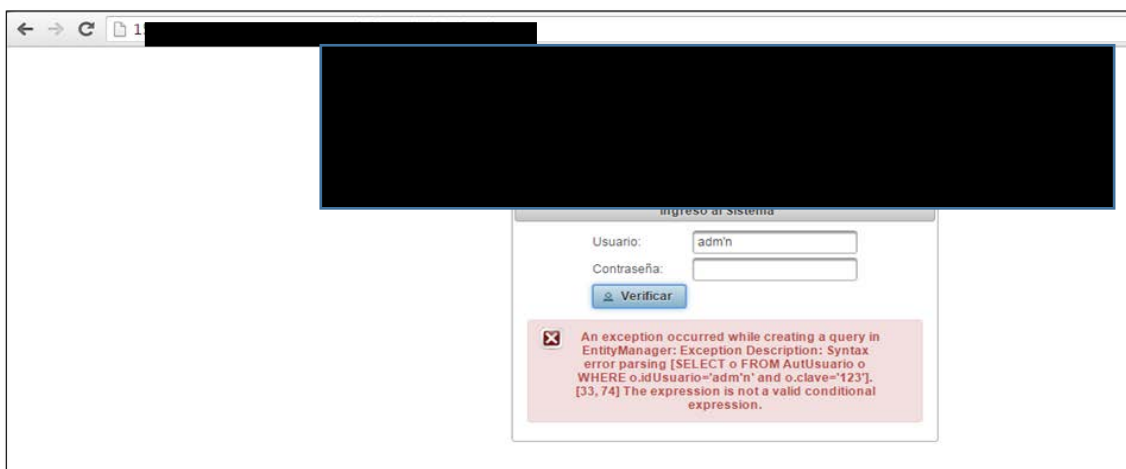


Figura 21: Demostración ataque SQLi
Fuente: Autoría Propia

2.- Una vez detectado que existe una vulnerabilidad de sqli, se procede a saltar el acceso de login con el siguiente código en ambos cuadros de texto.

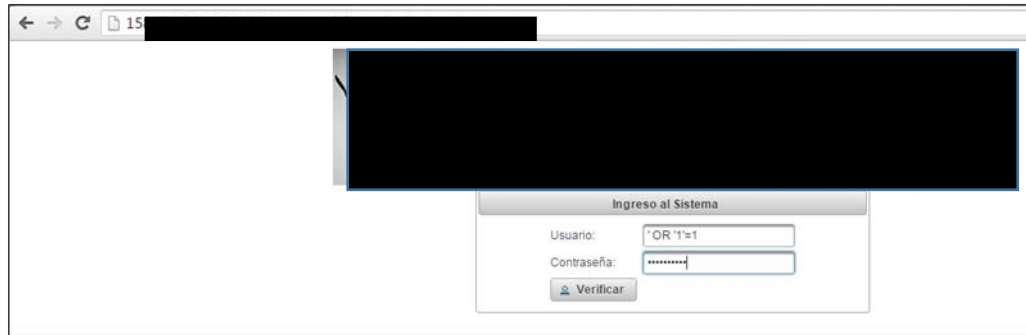


Figura 22: Demostración ataque inyección formularios mediante sentencias
Fuente: Autoría Propia

3.- Una vez que salta el proceso de autenticación, depende la aplicación no envía a la página establecida en este caso no muestra el tipo de perfil que se desee.

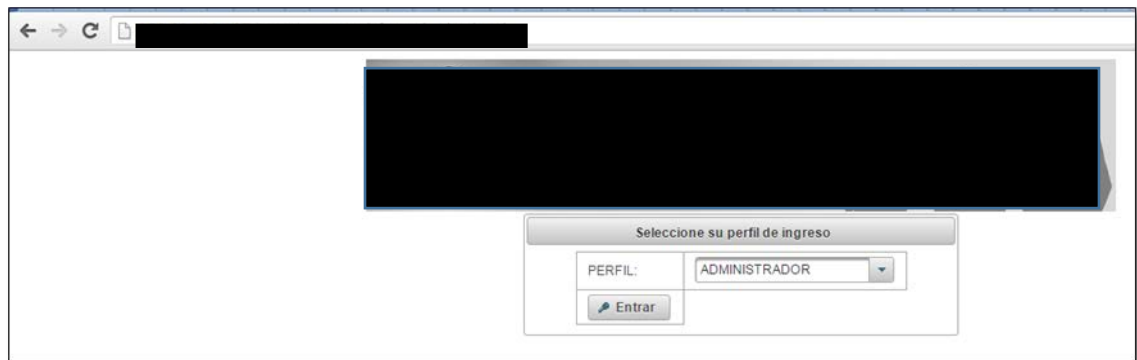


Figura 23: Demostración ataque inyección 2 formularios mediante sentencias
Fuente: Autoría Propia

4.- Una vez ingresado al sistema, se puede realizar cualquier tipo de fraude o se puede reportar la vulnerabilidad.

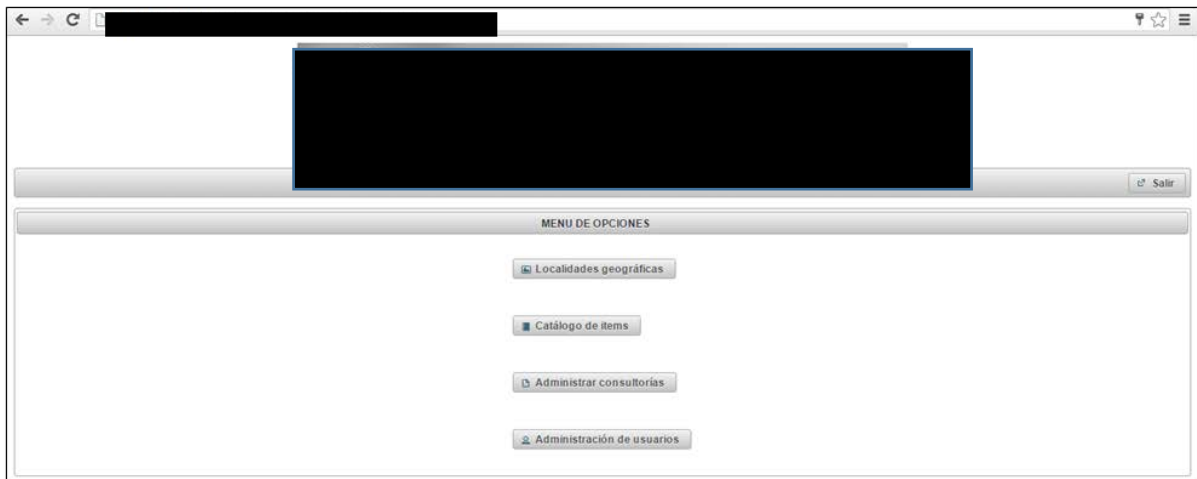


Figura 24: Demostración ataque inyección formularios mediante sentencias
Fuente: Autoría Propia

5.- Se puede explotar de varias formas cómo se inyecta una pequeña pero poderosa orden el cual nos indica cuantas columnas existe

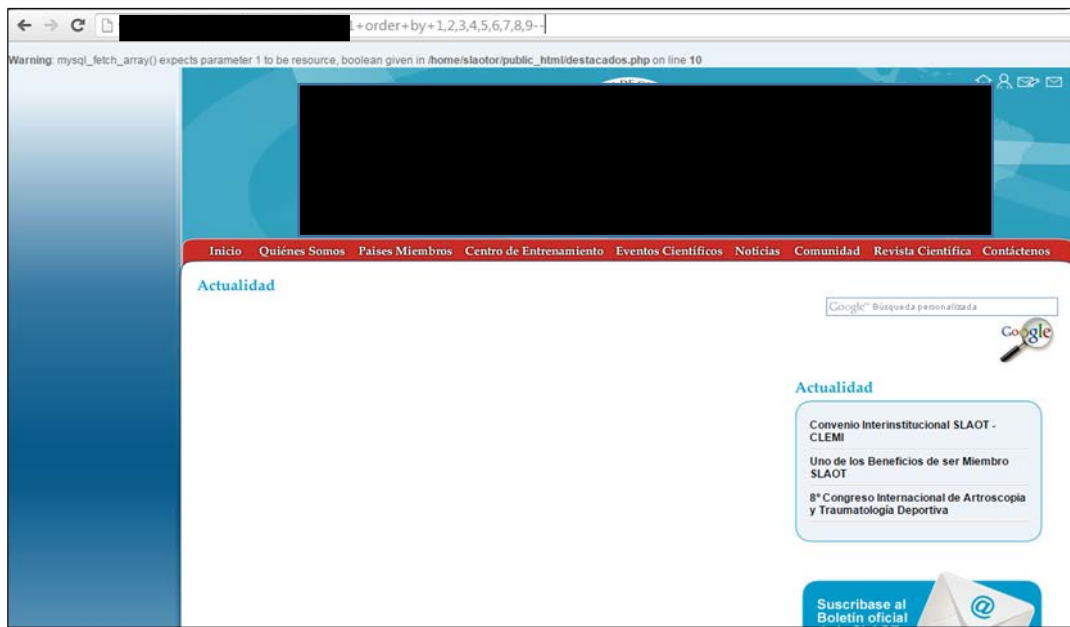


Figura 25: Ataque a página externa mediante consultas propias
Fuente: Autoría Propia

6.- Si hay vulnerabilidad muestra números posibles en donde existe datos para extraer información.

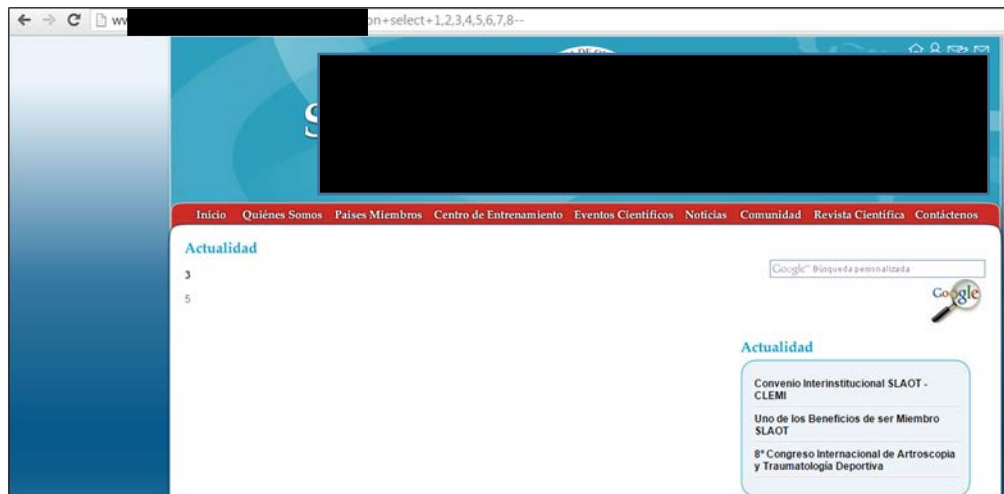


Figura 26: Vulnerabilidad en página que indica las columnas donde se inyecta código
Fuente: Autoría Propia

7.- Se formula código para seguir buscando información y obteniendo el nombre de las tablas que interese para obtener datos de dicha tabla. El parámetro inyectado es el siguiente:

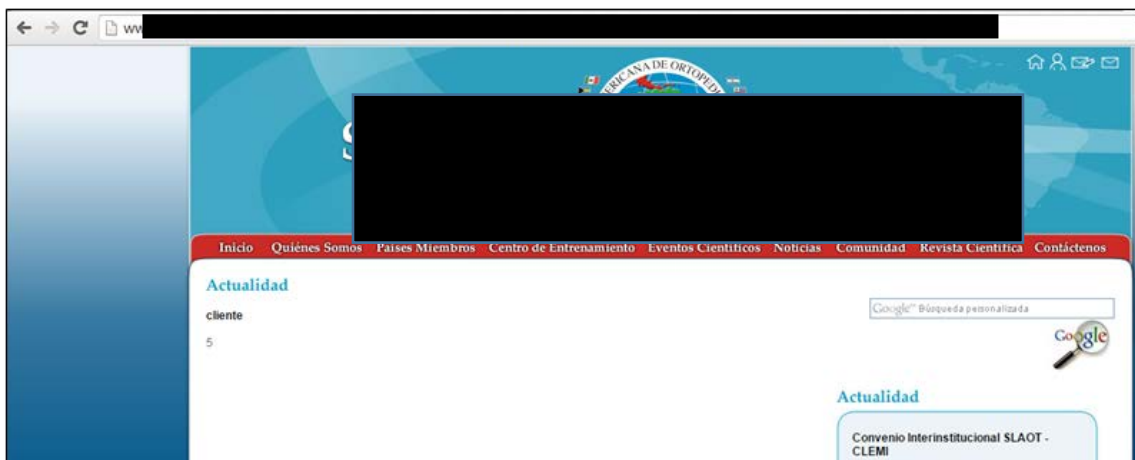


Figura 27: Inyección mediante consulta construida -1+union+select+1, 2, table_name, 4, 5, 6, 7, 8+from+information_schema.tables+limit+60, 1--
Fuente: Autoría Propia

8.- También se puede utilizar aplicaciones que cumplen la misma función de insertar código manualmente.

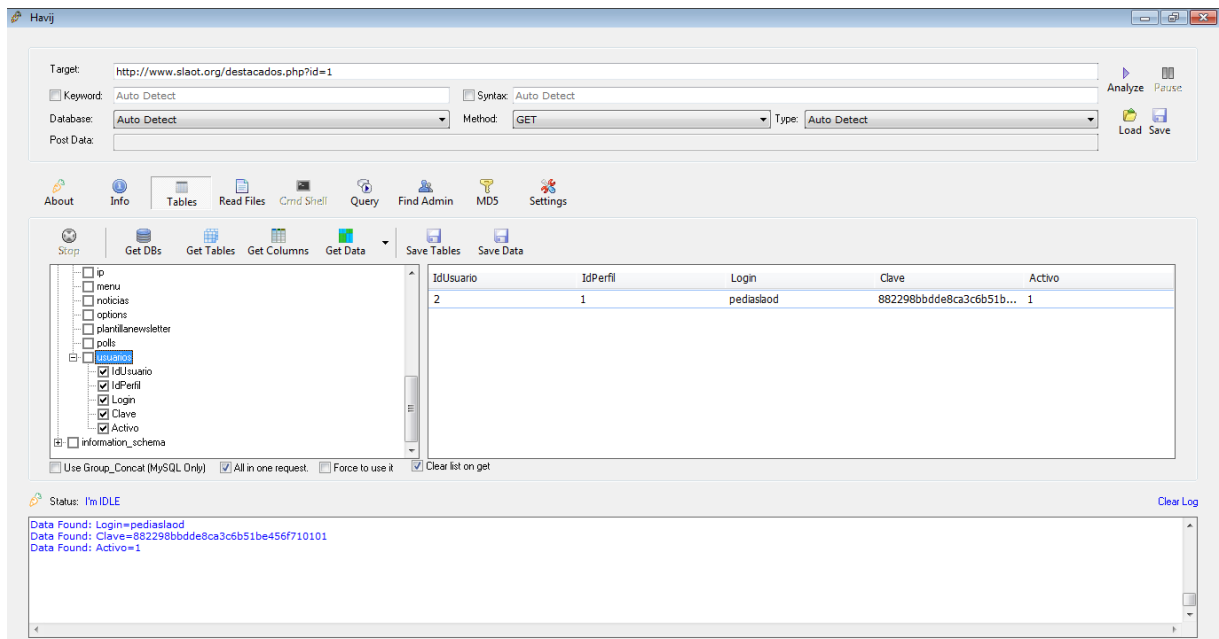


Figura 28: Aplicación que ayuda a detectar vulnerabilidades sql
Fuente: Autoría Propia

9.- Al inyectar código en aplicaciones que utilizan framework, resulta más difícil atacar la página mediante los códigos que se han demostrado.

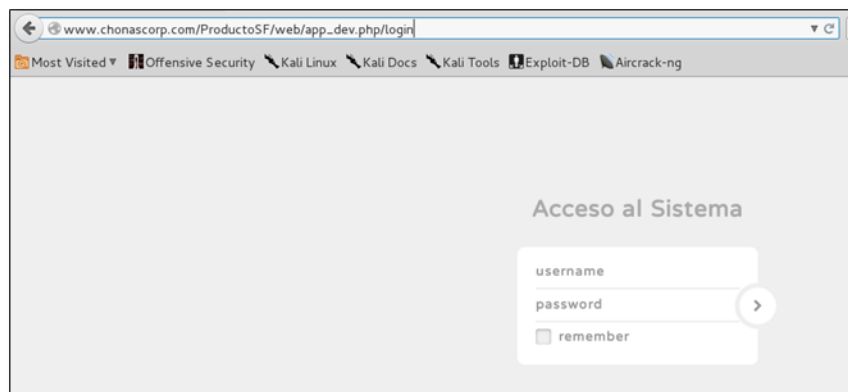


Figura 29: Aplicación web utilizando framework
Fuente: Autoría Propia

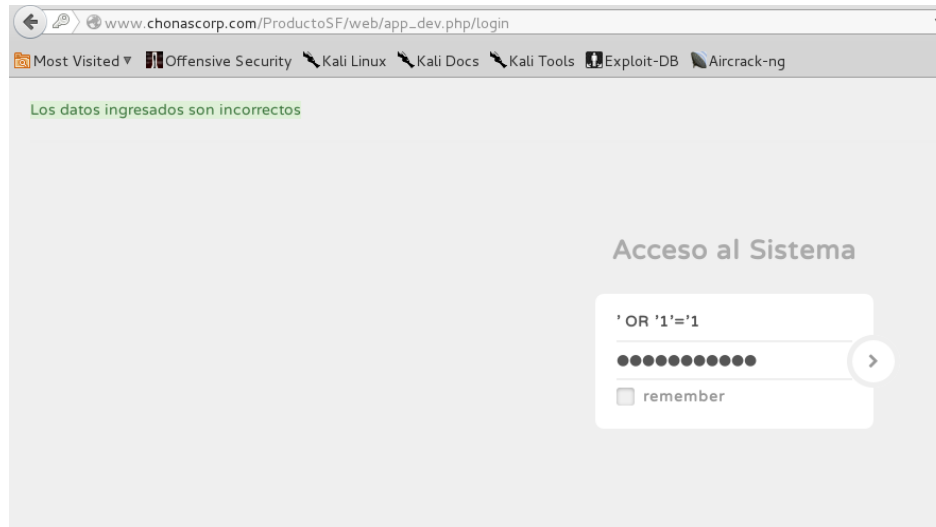


Figura 30: Error al inyectar sentencia sql en framework
Fuente: Autoría Propia

3.3.7 MECANISMOS DE DEFENSA

Consiste en asegurar y validar bien los datos de entrada que son proporcionados por el cliente, también depende mucho del lenguaje con el que se está trabajando, como el estudio se lo realiza en PHP la forma para filtrar los campos varía de acuerdo al tipo de datos.

Por ejemplo si existe un campo para ingresar números tales como “1235 or 1=1”, se lo puedo asegurar `$var_segura=intval($var_insegura);`

Si el tipo de dato es alfanumérico, se utiliza una función `pg_escape_string`, quedando de la siguiente manera `$var_segura=pg_escape_string($var_insegura);`

3.4 CROSS SITE SCRIPTING

Este ataque como el anterior estudiado es muy potente y peligroso ya que es explotado del lado del cliente y no del servidor, por lo que la seguridad de usuario queda muy vulnerable y expuesta a varios fraudes. Este método consiste en inyectar código HTML o JavaScript en una aplicación web, cuyo objetivo es consiste en que el navegador del usuario ejecute el código inyectado al momento de ver la página alterada. Comúnmente el XSS se utiliza para causa una acción indebida en el navegador de un usuario, pero depende del ataque de XSS que se realice, se puede explotar el error de un servidor o de la aplicación en sí.

El XSS se puede utilizar para hacer phishing,²⁴ robo de credenciales, troyanizar navegadores, o simplemente para hacer un deface²⁵, todo depende de la página.

3.4.1 TIPOS DE VULNERABILIDADES

- Tipo-0.- Se utiliza cuando se ejecuta código remotamente con los permisos de otro usuario.

Este tipo de ataque es conocido como DOM-Base ya que se trabaja en el navegador, y se inyecta el código en la URL; es decir el código nunca va a enviar a la página web, ejemplo de esta vulnerabilidad seria de la siguiente manera

²⁴ **Phishing.**- Es un ataque que suplanta la identidad de alguna persona con el fin de cometer un delito ya sea este robando contraseñas, tarjetas bancarias etc.

²⁵ **Deface.**- Es cuando se produce algún cambio en la interfaz de una aplicación por un atacante.

Imaginemos que la página **www.finanzas.com/index.html**

tiene el siguiente código:

```
<HTML>
<TITLE>Bienvenido!</TITLE>
Hola
<SCRIPT>
var pos=document.URL.indexOf("nombre=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR>
Bienvenido a nuestra página
</HTML>
```

Lo que da como resultado en la barra de direcciones <http://www.pagina.com/index.html?nombre=xxx>

Al final de la dirección url se puede agregar un fragmento de código JavaScript `<script>alert();</script>`, y si muestra un mensaje de alerta en un cuadro flotante, quiere decir que es vulnerable a un ataque XSS.

- ✓ Tipo-1.- Conocido como ataque no-persistente o no reflejado se utiliza en páginas no estáticas.

Este tipo de ataques se dan cuando existen huecos de seguridad, el servidor genera una página instantánea de resultados de acuerdo a información proporcionada, un ejemplo puede ser en los campos de búsqueda. La diferencia entre el ataque de tipo-0, es que este ataque ingresa al servidor y procesa la información enviado por el usuario e incluye el código en la página.

- ✓ Tipo-2.- Es un ataque muy persistente y que permite ejecutar código en páginas estáticas.

La información que proporciona el usuario es almacenada en la base de datos, en archivos de sistema o cualquier otro lugar que cause mucho daño, luego esa información es mostrada a otros usuarios para que visiten la página por eso se le conoce a este ataque como persistente. Con este tipo de ataques lo que se puede realizar es robar cookies, un ejemplo en donde se puede encontrar estas vulnerabilidades son los foros de discusión.

3.4.2 DEMOSTRACIÓN

- 1- Como vemos en este ejemplo para demostrar que la página es vulnerable se escribe código JavaScript y si este nos muestra la alerta, pues es un punto a favor para determinar su vulnerabilidad.



Figura 31: Inyección XSS en validaciones de texto

Fuente: Autoría Propia

2.- El ataque fue exitoso ya que muestra el mensaje que se inyectó en el cuadro de texto

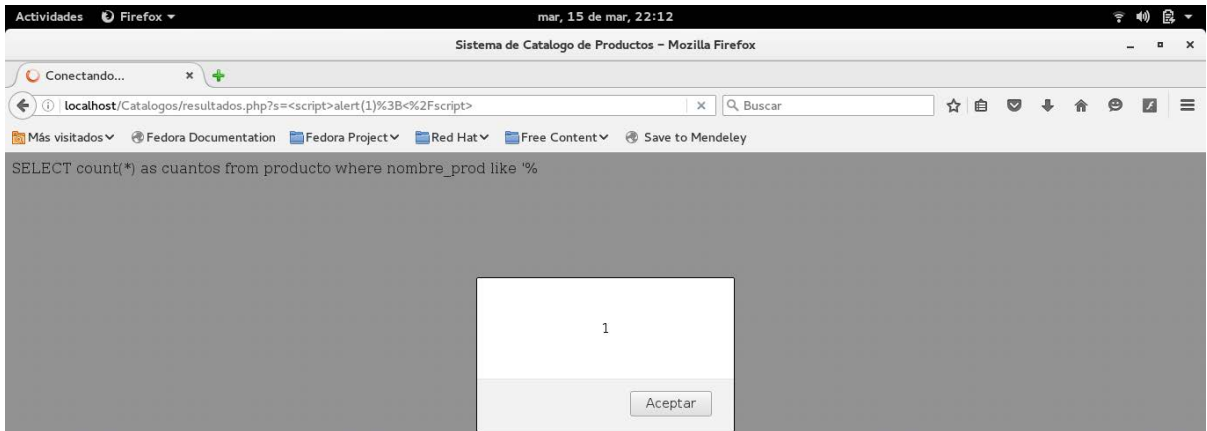


Figura 32: Mensaje ante inyección XSS
Fuente: Autoría Propia

3.- Existen plugin que sirven y ayudan a detectar los posibles ataques que se pueden realizar ante las malas validaciones

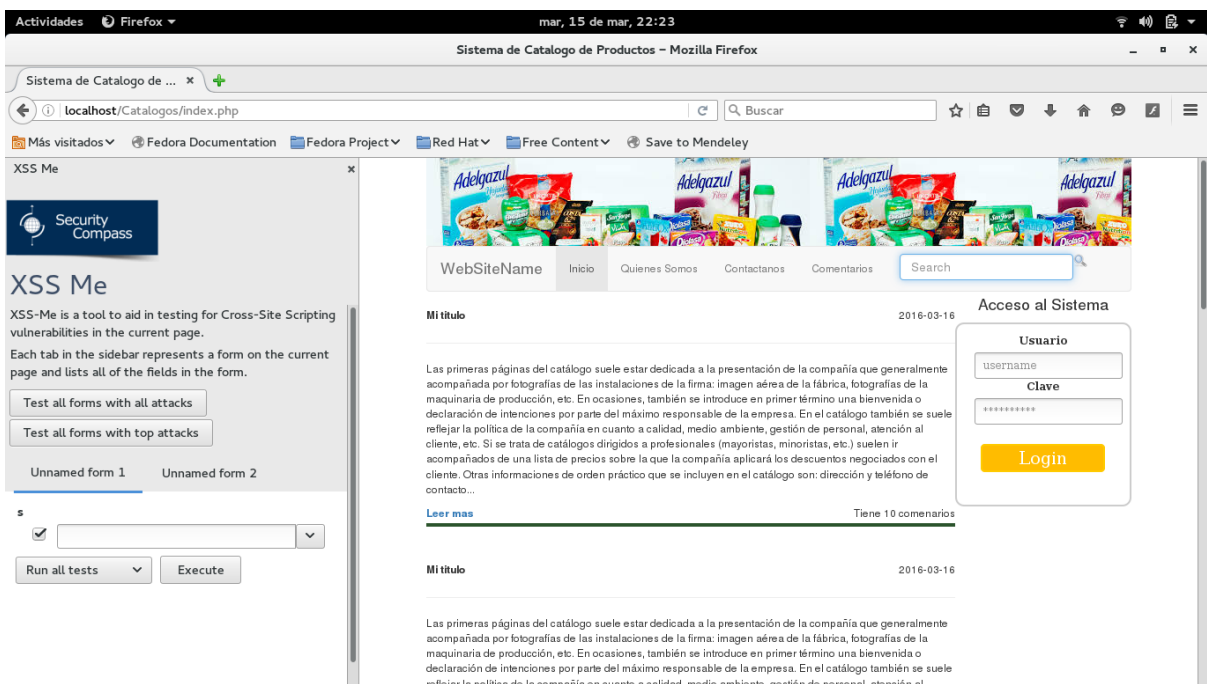


Figura 33: Herramientas que ayudan a buscar fallas de validaciones ante ataques XSS
Fuente: Autoría Propia

3.5 TABLA COMPARATIVA DE LOS LENGUAJES VULNERABLES POR SQL INYECCIÓN Y CROSS SITE SCRIPTING

Actualmente existen muchos lenguajes de programación y framework que se utiliza para el desarrollo de aplicaciones web, ya sean estos comerciales y de open source que se encuentran disponibles, sin embargo hay que considerar varios aspectos fundamentales antes de trabajar con alguno de ellos, como por ejemplo el nivel de seguridad de cada uno, y la vulnerabilidad que tienen al utilizar algunos lenguajes de programación.

A continuación se muestra en la tabla 17 una comparativa de los principales lenguajes de programación utilizados actualmente para el desarrollo de aplicaciones web y móviles, el mismo que permitirá determinar el nivel de ataques que se han registrado en cada uno. Valorado en porcentaje de 0% al 100%.

Para el cuadro comparativo se ha considerado utilizar como referente al The Hacker News. Que permite evaluar mediante un estudio realizado las vulnerabilidades de seguridad en aplicaciones web.

Existen algunos factores que se ha tomado en cuenta para obtener los porcentajes como son:

Diseño del lenguaje

Algunos lenguajes están diseñados desde el principio para evitar ciertas clases de vulnerabilidad. Mediante la eliminación de la necesidad y la capacidad de los desarrolladores para asignar directamente a la memoria, lenguajes como Java y .NET para evitar casi por completo las vulnerabilidades frente a la asignación de memoria, sobre todo el desbordamiento de búfer. Del mismo modo, los comportamientos de secuencias de comandos predeterminados entre sitios de algunos controles de ASP.NET para evitar problemas endémicos en otros entornos de programación de aplicaciones web.

Entorno operativo

Algunas vulnerabilidades sólo son relevantes en ciertos entornos de ejecución. Por ejemplo, algunas categorías de fuga de información son más agudos en el entorno móvil, que combina grandes volúmenes de los datos personales con una gran cantidad de capacidades siempre en la red.

Tabla 17: Tabla comparativa de los lenguajes más vulnerables

Tabla comparativa de vulnerabilidad		
Lenguajes de Programación	Porcentaje de aplicaciones afectadas.	
	SQL injection	Cross-Site Scripting (XSS)
.NET (C# Vb)	29%	48%
JAVA	21%	54%
PHP	56%	86%
ANDROID	27%	-
iOS	-	-
C++	-	-

Fuente: Autoría Propia

Como se puede observar en la comparativa PHP es el lenguaje con más vulnerabilidad tanto en SQL injection como en Cross – Site Scripting (XSS), lo cual es un riesgo ya que la gran cantidad de sitios web está basado su contenido en PHP como son WordPress²⁶ y Drupal.²⁷

²⁶ **WordPress.-** Gestor Documental que sirve para gestionar contenidos en un blog, pagina web.

²⁷ **Drupal.-** Gestor Documental que sirve para gestionar contenidos en un blog, pagina web.

4 DESARROLLO DEL APLICATIVO

En este capítulo se tratará el desarrollo del aplicativo utilizando la metodología de desarrollo XP. Basándose a las fases tanto de inicio, en donde se realizará el levantamiento de requerimientos o llamado historias de usuario; y terminado con la fase de pruebas de aceptación.

4.1 PLANIFICACIÓN DEL PROYECTO

4.1.1 ROLES

Cabe indicar que el aplicativo desarrollado es un pequeño demo para la demostración del análisis de los métodos de ataques web, por lo tanto este aplicativo no tiene todos los roles que la metodología XP utiliza.

Tabla 18: Roles de desarrollo

Nombre	Descripción	Rol XP
Edgar Subía	Tesista	Programador
Ing. Mauricio Rea	Consultor	Coach

Fuente: Autoría Propia

4.1.2 HISTORIAS DE USUARIO

La metodología de desarrollo XP maneja puntos de estimación para calificar a las historias de usuario de acuerdo al trabajo que implica cada una de estas.

Tabla 19: Estimación de Trabajo

Estimación		Descripción
1	=	Una semana
1/5	=	Un día
3	=	Tiempo alto
2/5	=	Dos días

Fuente: Autoría Propia

Se determinan las peticiones de usuario tomando en cuenta estimaciones cualitativas por ser preguntas cerradas. También se determinó el tipo de valoración en los criterios de prioridad, riesgo, con los valores alto, medio y bajo.

A continuación se detalla las historias de usuario.

Tabla 20: Historia de Usuario 1

Historia de Usuario	
Número: 1	Nombre Historia de Usuario: Instalación y Configuración
Usuario: Programador	Puntos Estimados: 1
Prioridad en el Negocio: Alta (Alto / Medio / Bajo)	Riesgo en el Desarrollo: Bajo (Alto / Medio / Bajo)
Descripción: Instalar y configurar el entorno de desarrollo para el aplicativo.	
Observaciones: Instalar versiones estables de las herramientas a utilizar.	

Fuente: Autoría Propia

Tabla 21: Historia de Usuario 2

Historia de Usuario	
Número: 2	Nombre Historia de Usuario: Seguridad y control de acceso
Usuario: Administrador de Sistema	Puntos Estimados: 1
Prioridad en el Negocio: Alta (Alto / Medio / Bajo)	Riesgo en el Desarrollo: Bajo (Alto / Medio / Bajo)
Descripción: Que permita ingresar al sistema con un usuario y una clave permitiendo la autorización y autenticación a la aplicación de acuerdo a su rol.	
Observaciones: El usuario logeado tendrá acceso a todos los datos y funciones del aplicativo de acuerdo a su rol.	

Fuente: Autoría Propia

Tabla 22: Historia de Usuario 3

Historia de Usuario	
Número: 3	Nombre Historia de Usuario: Gestión de catálogo
Usuario: Cliente	Puntos Estimados: 3
Prioridad en el Negocio: Alta (Alto / Medio / Bajo)	Riesgo en el Desarrollo: Medio (Alto / Medio / Bajo)
Descripción: Como cliente debe ingresar, editar y borrar la información del catálogo de productos.	
Observaciones: Los datos ingresados será información sensible del aplicativo que servirá para la demostración del análisis de los métodos de ataque.	

Fuente: Autoría Propia

Planificación de Iteraciones de acuerdo a las historias de usuarios.

Tabla 23: Planificaciones de iteraciones

Nro.	Nombre	Estimación	Prioridad	Riesgo	Iteración asignada
H1	Instalación y Configuración	1	Alta	Bajo	I1
H2	Seguridad y control de acceso.	1	Alta	Bajo	I2
H3	Gestión de catálogo	3	Alta	Media	I2

Fuente: Autoría Propia

4.1.3 DISEÑO DEL SISTEMA

Partiendo de las historias de usuario se diseña la arquitectura del sistema, la arquitectura funcional y los módulos que tendrá el aplicativo.

- **ARQUITECTURA DEL SISTEMA**

La arquitectura del sistema está basado en MVC (Modelo, Vista y Controlador) como se describe a continuación.

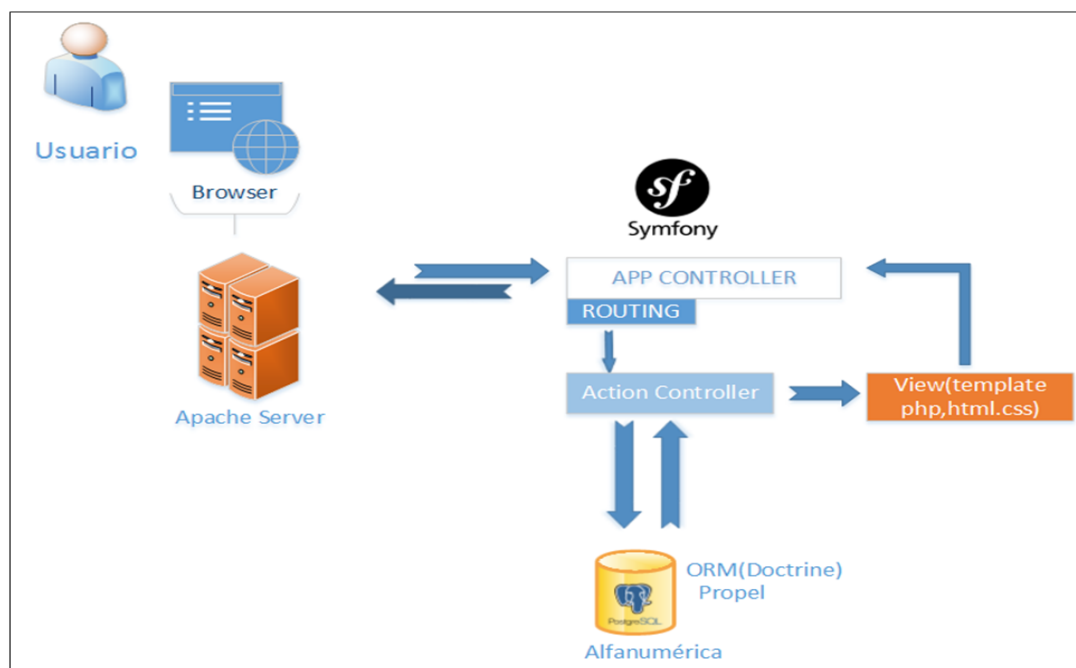


Figura 34: Arquitectura del Sistema
Fuente: Autoría Propia

- **ARQUITECTURA FUNCIONAL**

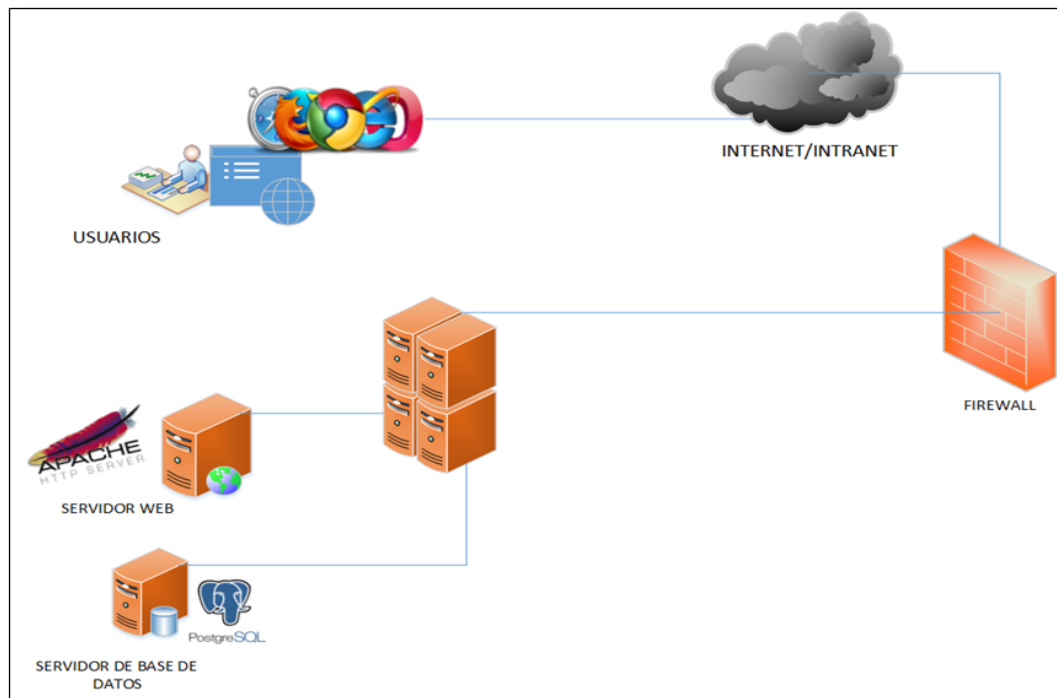


Figura 35: Arquitectura Funcional
Fuente: Autoría Propia

- **MÓDULOS DEL SISTEMA**

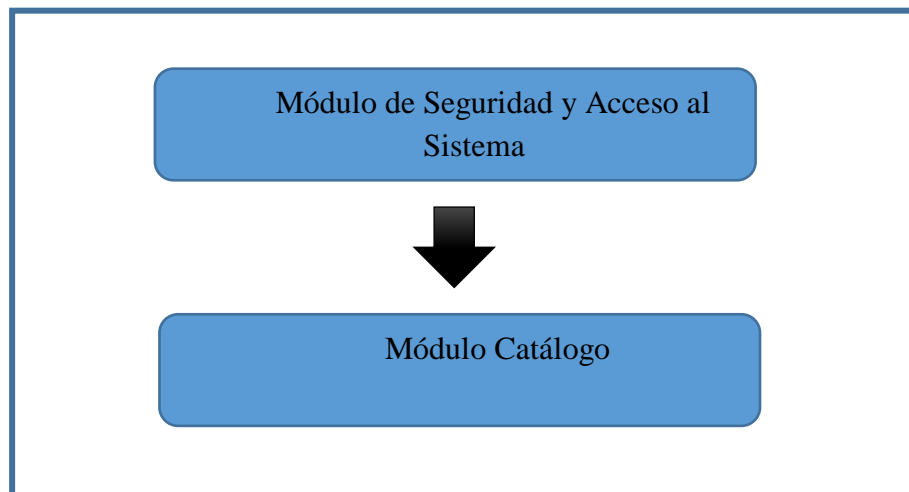


Figura 36: Módulos del aplicativo
Fuente: Autoría Propia

Descripción de Módulos:

Módulo de Seguridad y Acceso al Sistema: En módulo de seguridad se encargan de ingresar nuevos usuarios, asignarles un acceso al sistema para la navegación por el catálogo de productos.

Módulo de Catálogo: En este módulo permite realizar la gestión del catálogo de productos, con los respectivos detalles de información.

4.1.4 TAREAS

Las tareas se las realiza de acuerdo a cada historia de usuario y al módulo que pertenece, donde el programador toma las tareas detalladas para su desarrollo.

- **Módulo de Seguridad y Acceso al Sistema**
- ✓ **Historia de Usuario Nro.1:** Instalación y Configuración

Tabla 24: Tarea Nro. 1.1

Tarea de Usuario	
Nro. Tarea: 1.1	Nro. de Historia de Usuario: 1
Nombre Tarea: Instalación y configuración de Kali Linux	
Tipo de Tarea: Instalación (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1/5
Programador Responsable: Edgar Subía	
Descripción: Instalación y configuración de la distribución Kali Linux	

Fuentes: Autoría Propia

Tabla 25: Tarea Nro. 1.2

Tarea de Usuario	
Nro. Tarea: 1.2	Nro. de Historia de Usuario: 1
Nombre Tarea: Instalación y configuración de la base de datos Postgres	

Tipo de Tarea: Instalación (Desarrollo / Corrección / Instalación)	Puntos Estimados: 2/5
Programador Responsable: Edgar Subía	
Descripción: Instalación y configuración de la base de datos postgres 9.4 y su gestor de datos pgAdmin 1.0 sobre una distribución Linux.	

Fuentes: Autoría Propia

Tabla 26: Tarea Nro. 1.3

Tarea de Usuario	
Nro. Tarea: 1.3	Nro. de Historia de Usuario: 1
Nombre Tarea: Instalación del IDE de desarrollo.	
Tipo de Tarea: Instalación (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1/5
Programador Responsable: Edgar Subía	
Descripción: Instalación del IDE de desarrollo NetBeans 8.0.2.	

Fuentes: Autoría Propia

Tabla 27: Tarea Nro. 1.4

Tarea de Usuario	
Nro. Tarea: 1.4	Nro. de Historia de Usuario: 1
Nombre Tarea: Instalación y configuración del framework de desarrollo.	
Tipo de Tarea: Instalación (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1/5
Programador Responsable: Edgar Subía	
Descripción: Instalación y configuración del framework de desarrollo Symfony 2.3.38.	

Fuentes: Autoría Propia

✓ **Historia de Usuario Nro.2:** Seguridad y control de acceso

Tabla 28: Tarea Nro. 2.1

Tarea de Usuario	
Nro. Tarea: 2.1	Nro. de Historia de Usuario: 2
Nombre Tarea: Diseñar la estructura de datos para el control de acceso al sistema.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Realizar el diseño de la base de datos para el registro de los usuarios con sus respectivas claves para el acceso al sistema, y el control y asignación de permisos.	

Fuentes: Autoría Propia

Tabla 29: Tarea Nro.2.2

Tarea de Usuario	
Nro. Tarea: 2.2	Nro. de Historia de Usuario: 2
Nombre Tarea: Crear la interfaz para el ingreso al sistema	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Se creará una pantalla para que el cliente ingrese el usuario y clave para el acceso al sistema.	

Fuentes: Autoría Propia

Tabla 30: Tarea Nro. 2.3

Tarea de Usuario	
Nro. Tarea: 2.3	Nro. de Historia de Usuario: 2
Nombre Tarea: Crear la interfaz para la administración de perfiles.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Se creará una pantalla para que el administrador gestione los perfiles.	

Fuentes: Autoría Propia

Tabla 31: Tarea Nro. 2.4

Tarea de Usuario	
Nro. Tarea: 2.4	Nro. de Historia de Usuario: 2
Nombre Tarea: Crear la interfaz para la administración de usuarios.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Se creará una pantalla para que el administrador gestione los usuarios.	

Fuentes: Autoría Propia

- **Módulo de Catálogo**

- ✓ **Historia de Usuario Nro. 3: Gestión de catálogo**

Tabla 32: Tarea Nro. 3.1

Tarea de Usuario	
Nro. Tarea: 3.1	Nro. de Historia de Usuario: 3
Nombre Tarea: Diseñar la estructura de datos para el catálogo de productos.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Realizar el diseño de la base de datos para el registro del catálogo de productos.	

Fuentes: Autoría Propia

Tabla 33: Tarea Nro. 3.2

Tarea de Usuario	
Nro. Tarea: 3.2	Nro. de Historia de Usuario: 3
Nombre Tarea: Crear la interfaz del catálogo de productos.	
Tipo de Tarea: Desarrollo (Desarrollo / Corrección / Instalación)	Puntos Estimados: 1
Programador Responsable: Edgar Subía	
Descripción: Crear la interfaz para la gestión del catálogo de productos.	

Fuentes: Autoría Propia

4.1.5 ESTIMACIÓN DE ESFUERZO

- **Módulo de Seguridad y Acceso al Sistema**

Tabla 34: Estimación de Esfuerzo por tarea

Nro.	Tareas	Puntos
	Historia de Usuario Nro.1: Instalación y Configuración	
1	Instalación y configuración de Kali Linux	1/5
2	Instalación y configuración de la base de datos Postgres	2/5
3	Instalación del IDE de desarrollo Netbeans 8.0.2	1/5
4	Instalación y configuración del framework de desarrollo Symfony 2.3.38.	1/5
	Historia de Usuario Nro.2: Seguridad y control de acceso	
5	Diseñar la estructura de datos para el control de acceso al sistema.	1
6	Crear la interfaz para el ingreso al sistema	1
	Historia de Usuario Nro. 3: Gestión de catálogo	
7	Diseñar la estructura de datos para el catálogo de productos.	1
8	Crear la interfaz del catálogo de productos	1
TOTAL ESFUERZO ESTIMADO		5 SEMANAS

Fuentes: Autoría Propia

4.2 FASE DE DISEÑO

La fase de diseño en la construcción de un software es muy importante, ya que es la base fundamental de la aplicación, en ella podemos darnos cuenta el funcionamiento.

En la figura 37 se indica todas las tablas que se utilizan para cada módulo.

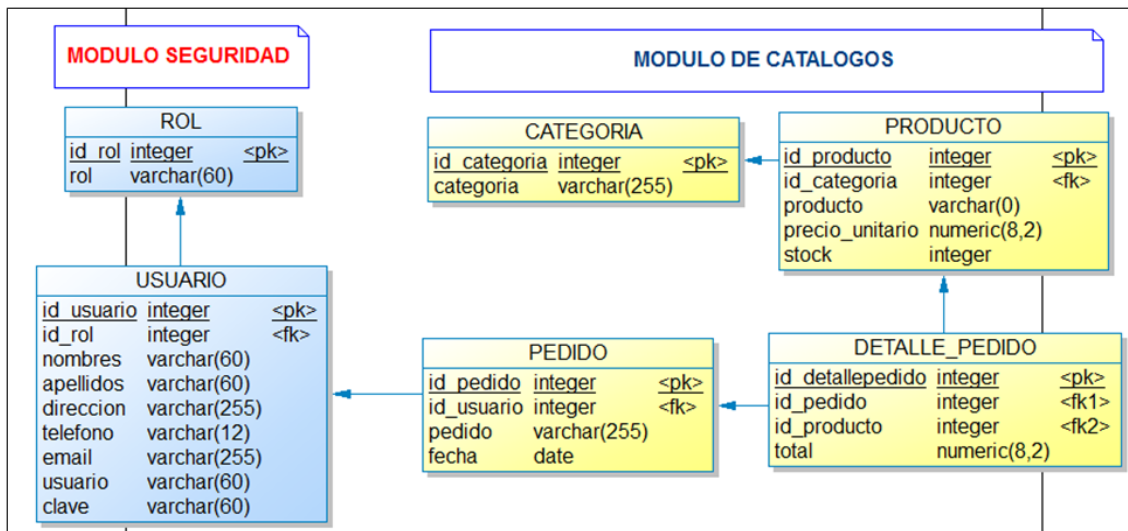


Figura 37: Modelo Relacional del aplicativo
Fuente: Autoría Propio

La figura 38 muestra los diagramas de casos de uso que ayudará a explicar las funciones de los usuarios del aplicativo además de definir el alcance de la aplicación.



Figura 38: Diagrama de Caso de Uso CU_ADMINISTRACIÓN
Fuente: Autoría Propia

Tabla 35: Descripción caso de uso CU_ADMINISTRACIÓN

Identificador de caso de uso	CU_ADMINISTRACIÓN
Nombre caso de uso	ADMINISTRACIÓN
Actores	Administrador
Propósito	Ingreso al módulo de seguridad y acceso al sistema.
Visión general	El administrador gestiona información de los perfiles y gestiona la información de usuarios.
Tipo	Primario, esencial
Curso de eventos	
Acciones del actor	Respuesta del sistema
Administrador ingresa a la administración.	El administrador ingresa sus credenciales e ingresa al módulo administrativo de la aplicación.
Administrador de perfiles.	El administrador gestiona los perfiles de la aplicación.
Administrador de usuarios.	El administrador gestiona los usuarios nuevos y existentes de la aplicación.

Fuente: Autoría Propia

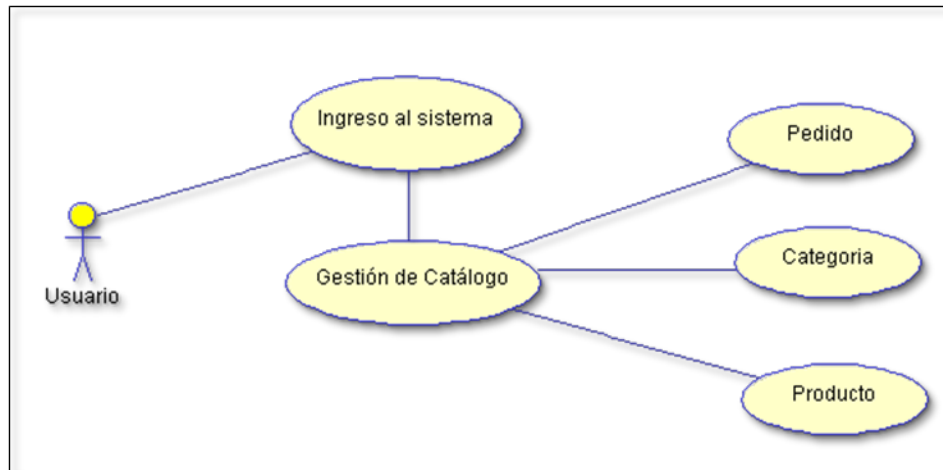


Figura 39: Caso de Uso CU_CATÁLOGO
Fuente: Autoría Propia

Tabla 36: Descripción caso de uso CU_CATÁLOGO

Identificador de caso de uso	CU_CATÁLOGO
Nombre caso de uso	CATÁLOGO
Actores	Empleado
Propósito	Ingreso al módulo de administración de catálogo.
Visión general	El usuario gestiona información de pedidos, categoría del producto, los productos y los detalles que tiene el producto.
Tipo	Primario, esencial
Curso de eventos	
Acciones del actor	Respuesta del sistema
Usuario ingresa al sistema.	El administrador ingresa sus credenciales e ingresa al módulo de gestión de catálogo.
Usuario ingresa a la gestión de catálogo.	El usuario tiene acceso a la administración de pedidos de características y detalles del producto.

Fuente: Autoría Propia

4.3 FASE DE ITERACIÓN

4.3.1 ITERACIÓN I

En la iteración 1 se realiza la instalación y configuración del ambiente de trabajo para el desarrollo del aplicativo, y a su vez para la demostración de las técnicas de vulnerabilidad a páginas web.

4.3.2 ITERACIÓN II

- **Historia de Usuario 2 Seguridad y control de acceso.**
- ✓ **Tarea 2.1 Diseñar la estructura de datos para el control de acceso al sistema.**

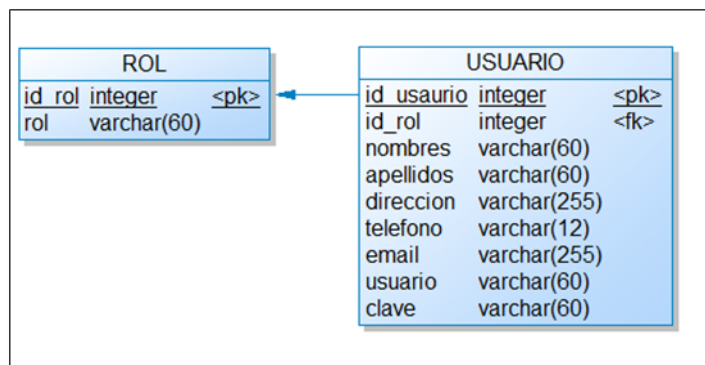


Figura 40: Diseño de la estructura de datos para acceso al sistema

Fuente: Autoría Propia

- ✓ **Tarea 2.2 Crear la interfaz para el ingreso al sistema**

Interfaz de ingreso al sistema. Muestra un formulario con los campos Usuario: y Clave: y un botón Ingresar. Hay un icono de un personaje con una llave y un bolso azul.

Figura 41: Ingreso al sistema

Fuente: Autoría Propia

✓ **Tarea 2.3 Crear la interfaz de administración de perfiles.**



Figura 42: Administración de Perfiles
Fuente: Autoría Propia

✓ **Tarea 2.4. Crear la interfaz de administración de usuarios.**

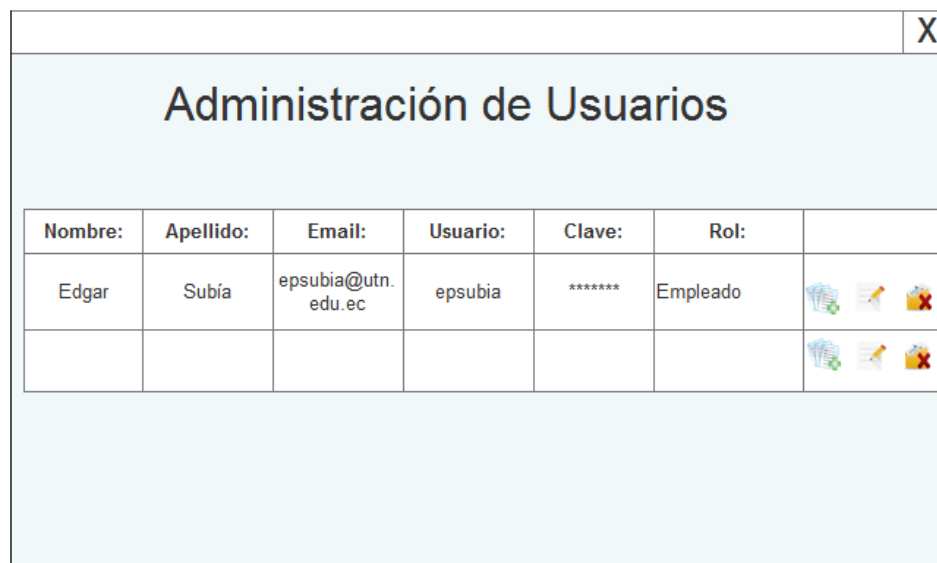


Figura 43: Administración de Usuarios
Fuente: Autoría Propia

- **Historia de Usuario 3 Gestión de catálogo.**
- ✓ **Tarea 3.1 Diseñar la estructura de datos para el catálogo de productos**

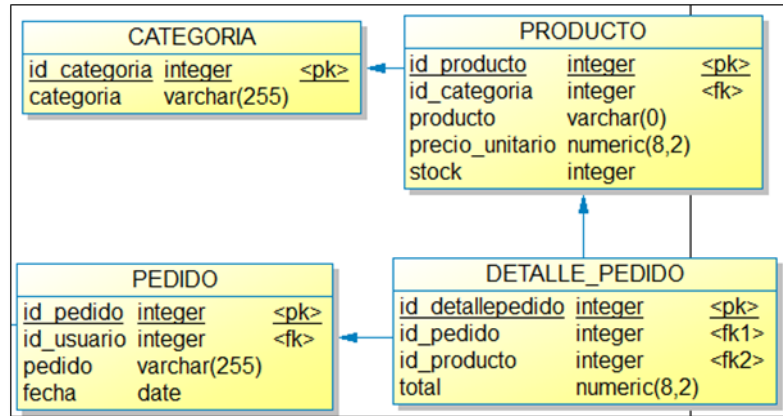


Figura 44: Diseño de la estructura de datos del módulo catálogos
Fuente: Autoría Propia

- ✓ **Tarea 3.2 Crear la interfaz del catálogo de productos**

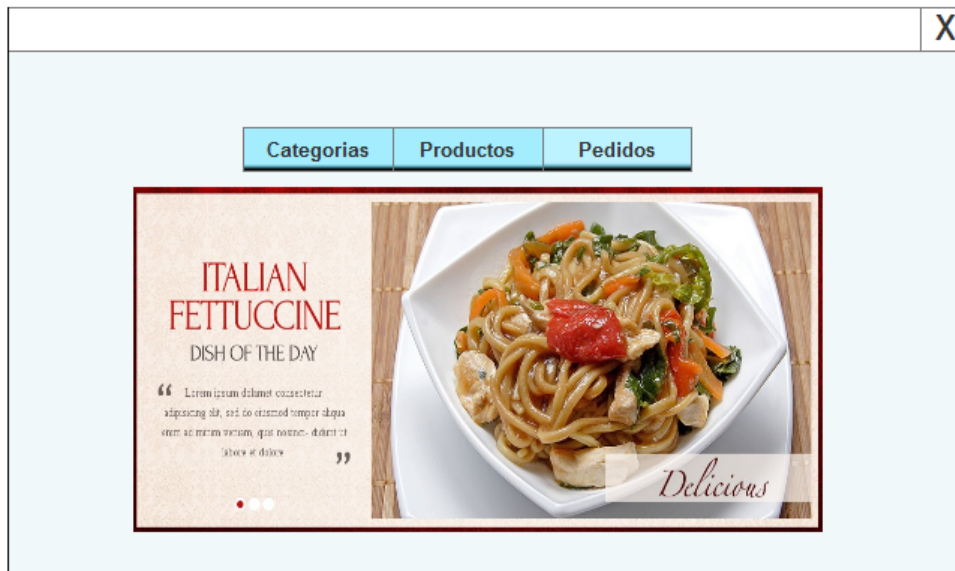


Figura 45: Interfaz catálogo de productos
Fuente: Autoría Propia

Interfaz de categorías



The screenshot shows a window titled "Categoría:" with a close button (X) in the top right corner. The window contains a table with three columns: "Categoría:", "Descripción:", and an empty column. The first row has "Licor" in the first column and three icons (a folder, a pencil, and a trash can) in the third column. The second row is empty.

Categoría:	Descripción:	
Licor		  
		  

Figura 46: Interfaz de categorías
Fuente: Autoría Propia

Interfaz Producto



The screenshot shows a window titled "Producto" with a close button (X) in the top right corner. The window contains a table with four columns: "Producto:", "Precio Unitario:", "Stock:", and an empty column. The first row has "Vino Blanco" in the first column, "23.50" in the second, "5" in the third, and three icons (a folder, a pencil, and a trash can) in the fourth. The second row is empty.

Producto:	Precio Unitario:	Stock:	
Vino Blanco	23.50	5	  
			  

Figura 47: Interfaz producto
Fuente: Autoría Propia

Interfaz Pedido



Pedido

Usuario: Edgar Subia

Fecha: 2016/01/23

Producto	Cantidad
Licor	2
Espagueti	3
Total	5

Figura 48: Interfaz Pedido
Fuente: Autoría Propia

4.4 FASE DE PRUEBAS

4.4.1 DISEÑO DE PRUEBAS DE ACEPTACIÓN

La metodología XP ofrece una utilidad al instante de realizar pruebas de aceptación ya que mediante esto se puede realizar una regresión al proceso de las iteraciones para su corrección, haciendo que sea bastante dinámica y flexible. En esta fase se realiza una descripción de las pruebas de aceptación, y corresponden a la comprobación del correcto funcionamiento de la aplicación.

Tabla 37: Pruebas de aceptación

Nombre	Evento	Descripción	Resultado
Login	Ingresar	Verifica si los datos ingresados son correctos.	Ok
Administrador	Ingresar	Verifica que se registre los roles y los usuarios correctamente.	Ok
	Eliminar	Verificar que se de baja a los usuarios que ya no correspondan a sistema.	Ok
	Editar	Verifica que la información editada sea correcta.	Ok
Catálogo	Ingresar	Verifica que la información de categoría y producto sean ingresados correctamente.	Ok
	Ingresar	Verifica que la información de los pedidos sea correctos.	Ok
	Editar	Verifica que la información editada sea correcta.	Ok

Fuente: Autoría Propia

5 CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Aplicar una buena metodología de pentest que ayude a verificar los riesgos y vulnerabilidades, y corregir los errores más comunes que se comente al momento de programar.
- Las vulnerabilidades estudiadas causan mucho daño y afectan a toda la capa de la aplicación causando pérdidas de información vital para las empresas.
- Utilizar herramientas de testeo ya sean libres o de pago sirven de mucha ayuda para determinar vulnerabilidades.
- Tener una lista o bitácora bien argumentada de la situación actual de la empresa sirve para darse cuenta en qué estado se encuentra y en qué estado de vulnerabilidad se encuentra.

5.2 RECOMENDACIONES

- Utilizar la metodología OWASP ya que este Proyecto se encuentra enfocado y trabaja en las seguridades de aplicaciones web.
- Manejar framework actuales que ya aplican seguridades para evitar ataques.
- Se recomienda usar el Sistema Operativo Kali Linux que es específico para delitos informáticos.
- Realizar reportes técnicos y ejecutivos que informen a los técnicos y a los administrativos qué hacer en caso se sufrir ataques delictivos.

6 BIBLIOGRAFÍA Y LINKOGRAFÍA

ALONSO TORRES, D. H. Evaluación de seguridad a sistemas de información en cuanto a ataques maliciosos con base en normatividad, tendencias, impacto y técnicas vigentes para ambientes empresariales a nivel nacional. 2015.

ANABELL, C. Java o PHP. **Revista Digital Universitaria**, v. Volumen, 2004. ISSN 1067-6079. Disponible em: < http://www.revista.unam.mx/vol.7/num12/art104/dic_art104.pdf >.

ASENSIO HILDAGO, L. Seguridad en aplicaciones web: una visión práctica. 2014.

BALOCH, R. **Ethical Hacking and Penetration Testing Guide**. CRC Press, 2014. ISBN 148223162X.

BECK, K. Embracing Change with Extreme Programming. **Computer**, v. 32, n. 10, p. 70-77, 1999. ISSN 0018-9162.

BENCHIMOL, D. **Hacking desde Cero**. M.P. Ediciones, 2011. ISBN 9789871773039. Disponible em: < <https://upload.wikimedia.org/wikipedia/commons/b/b9/HakingCero.pdf> >.

BUENDÍA, J. F. R. **Seguridad Informática**. Aravaca (Madrid): McGraw-Hill/Interamericana de España, S. L., 2013. ISBN 978-84-481-8569-5.

CASTILLO, E. V. Q. **Desarrollo e implantación del portal web para la fundación libres para Cristo**. 2014. Tesis Sistemas Informáticos y de Computación (ISIS), Universidad Politécnica Nacional, Quito.

CORLETTI ESTRADA, A. **Seguridad por Niveles**. Madrid: DarFE., 2011.

COSTAS SANTOS, J. **Mantenimiento de la seguridad en sistemas informáticos**. España: RA-MA Editorial, 2014a. ISBN 9788499643366. Disponible em: < <http://site.ebrary.com/lib/utnortesp/docDetail.action?docID=11046692> >.

COSTAS SANTOS, J. **Seguridad informática**. España: RA-MA Editorial, 2014b. ISBN 9788499643137. Disponible em: < <http://site.ebrary.com/lib/utnortesp/docDetail.action?docID=11038505> >.

DE LA QUINTANA ILLANES, M. M. SQL INYECTION. **Revista de Información, Tecnología y Sociedad**, p. 38-40, 2013. ISSN 1997-4044. Disponible em: < http://www.revistasbolivianas.org.bo/scielo.php?script=sci_arttext&pid=S1997-40442013000100017&nrm=iso >.

ESCRIVÁ GASCÓ, G.; ROMERO SERRANO, R. M.; RAMADA, D. J. **Seguridad informática**. España: Macmillan Iberia, S.A., 2013. ISBN 9788415991410. Disponible em: < <http://site.ebrary.com/lib/utnortesp/docDetail.action?docID=10820963> >.

FABIEN POTENCIER, F. Z. **Symfony 1.4, la guía definitiva**. 2013. Disponible em: < http://librosweb.es/libro/symfony_1_4/ >.

FLORES QUISPE, C. A. TIPOS DE HACKERS. **Revista de Información, Tecnología y Sociedad**, p. 16-18, 2013. ISSN 1997-4044. Disponible em: < http://www.revistasbolivianas.org.bo/scielo.php?script=sci_arttext&pid=S1997-40442013000100008&nrm=iso >.

FRANCHI, M. R. **Algoritmos de encriptación de clave asimétrica**. 2013. Facultad de Informática

GONZALEZ, C. **Análisis, Diseño, Desarrollo e Implementación de una Aplicación Web para la Automatización de Clientes, Vehículos, Facturación, Inventario y Campañas para Autoservicios RBS**. 2012. Departamento de Ciencias de la Computación, Escuela Politécnica del Ejército, Sangolquí.

GRAVES, K. **Certified Ethical Hacker STUDY GUIDE**. Canada: Wiley Publishing Inc 2010.

GÓMEZ, A. R.; DUARTE, A. Q.; GÜEVARA, C. D. M. Desarrollo ágil de software aplicando programación extrema. **Revista Ingenio UFPSO**, v. 5, n. 1, p. 24-29, 2014. ISSN 2389-864X.

GÓMEZ FERNÁNDEZ, L.; FERNÁNDEZ RIVERO, P. P. **Cómo implantar un SGSI según UNE-ISO/IEC 27001:2014 y su aplicación en el Esquema Nacional de Seguridad**. España: AENOR - Asociación Española de Normalización y Certificación, 2015. ISBN 9788481439014. Disponible em: < <http://site.ebrary.com/lib/utnortesp/docDetail.action?docID=11087537> >.

GÓMEZ VIEITES, Á. **Auditoría de seguridad informática**. España: RA-MA Editorial, 2014. ISBN 9788499643281. Disponible em: < <http://site.ebrary.com/lib/utnortesp/docDetail.action?docID=11046196> >.

HECTOR JARA, F. G. P. **Ethical Hacking 2.0. Red Users**. Buenos Aires: Fox Andina. 1.era Edición 2012.

HERMOSO METAUTE, A. **Seguridad en aplicativos web**. 2013.

HEZROG, P. **Metodologías de Pentesting**. 2015. Disponible em: < <http://www.isecom.org/research/osstmm.html> >. Disponible em: < <http://isecom.securenetltd.com/OSSTMM.es.2.1.pdf> >.

MOMJIAN, B. Postgresql. California, p. Documentation, 2014. Disponible em: < <http://www.postgresql.org/docs/9.3/static/release-9-3-5.html> >.

OWASP. OWASP Testing Guide v4. 2015. Disponible em: < https://www.owasp.org/index.php/Category:OWASP_Testing_Project#tab=New_OWASP_Testing_Guide >.

RAFAEL, M.; MANUEL, J. Seguridad en redes. 2013.

RAMÍREZ PARRA, J. Modelo de estándares de desarrollo seguro en aplicativos web. 2012.

SALGADO YÁNEZ, Á. L. Artículo Científico.-Análisis de las aplicaciones web de la Superintendencia de Bancos y Seguros, utilizando las recomendaciones Top Ten de OWASP para determinar los riesgos más críticos de seguridad e implementar buenas prácticas de seguridad para el desarrollo de sus aplicativos. 2014.

SAMANIEGO, W. A. C. **Estudio del Arte de los sistemas de identificación de intrusos.** 2009. Universidad Técnica Particular de Loja, Loja.

SHAKEEL, A. T., H. **BackTrack 4: Assuring Security by Penetration Testing.** 32 Lincoln Road

Olton

Birmingham, B27 6PA, UK: Packt Publishing Ltd., 2011. ISBN 978-1-849513-94-4. Disponible em: < <http://index-of.es/eBooks/BackTrack-4-Assuring-Security-by-Penetration-Testing.pdf> >.

TEN, O. **OWASP TOP 10 Los diez riesgos mas críticos en aplicaciones web.** 2013. Disponible em: < https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013 >.

TORI, C. **Hacking Etico, Seapro activo descubra las vulnerabilidades antes de que otro lo haga.** Red Users. Rosario-Argentina 2008.