UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

ANÁLISIS Y SOLUCIÓN DE VULNERABILIDAD DE SEGURIDAD EN APLICACIONES WEB Y MÉTODOS DE PROTECCIÓN ANTI ROBOT Y HTTP REQUEST

PRESENTADA POR:

SANDRO GUZMÁN CANAHUIRE CHAMBI

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA

MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES

PUNO, PERÚ

2020

DEDICATORIA

A Dios, por darme la vida y sabiduría.

A mis padres, que han dedicado su vida en cuidar de mí con peculiares ejemplos, formas no ortodoxas de educación, pero en final efectivas para obtener un pensamiento crítico.

Sandro Guzmán Canahuire Chambi

AGRADECIMIENTOS

A la Universidad Nacional del Altiplano por darnos la oportunidad de formarnos bajo el entorno del conocimiento pleno de la sociedad académica en nuestras respectivas áreas.

A los docentes y personal administrativo de la escuela de Posgrado-Maestría en informática de la Universidad Nacional del Altiplano por la labor sacrificada que cumplen en beneficio de los distintos profesionales.

Sandro Guzmán Canahuire Chambi

ÍNDICE GENERAL

CAPÍTULO III

MATERIALES Y MÉTODOS

CAPÍTULO IV

RESULTADOS Y DISCUSÍON

Puno, 15 de enero de 2020

ÁREA: Informática
TEMA: Vulnerabilidad y seguridad en aplicaciones web
LÍNEA: Seguridad informática

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

La presente investigación está orientada con la finalidad de mejorar el proceso de administración, monitoreo de los posibles ataques de crackers o nuevos programadores que desean obtener información de nuestros servidores de datos sin estar autorizados, mediante la implementación de un sistema web para la administración de privilegios de seguridad sin ser estrictos en un solo servidor web. El propósito de este sistema es posibilitar el registro, validación, evaluación, publicación de resultados e información del contenido de la aplicación o aplicaciones contenidas en el servidor. El presente proyecto es de carácter descriptivo detallando las fallas de seguridad y por supuesto los mejores esquemas de protección a ser adoptados por su mayor afinidad y claridad de actividades y pone más énfasis en la adaptabilidad que en la previsibilidad, aplicando de manera dinámica durante el ciclo de vida del software y aplicaciones en la nube. En prueba de auditoria planteada como ejemplo a las paginas seleccionadas se concluye que el análisis de los casos evaluados para la verificación y documentación de los elementos a ser evaluados en las páginas web listadas se muestra resultados que permiten obtener resultados de auditoria sobre la seguridad, del mismo modo un listado de desempeño para peticiones POST y GET, asumiendo la capacidad de transferencia de datos de modo más efectivo.

Palabras Clave: Adaptabilidad, administración, infiltración, Seguridad, web crawling.

ABSTRACT

This research is aimed at improving the administration process, monitoring possible attacks by crackers or new programmers who wish to obtain information from our data servers without being authorized, through the implementation of a web system for privilege management, security without being strict on a single web server. The purpose of this system is to enable the registration, validation, evaluation, publication of results and information on the content of the application or applications contained in the server. The present project is descriptive in nature detailing the security flaws and of course the best protection schemes to be adopted due to their greater affinity and clarity of activities and places more emphasis on adaptability than on predictability, applying dynamically during the cycle of software life and applications in the cloud. In the audit test presented as an example to the selected pages, it is concluded that the analysis of the cases evaluated for the verification and documentation of the elements to be evaluated in the web pages listed shows results that allow obtaining security audit results of the Likewise, a performance list for POST and GET requests, assuming the ability to transfer data more effectively.

Keywords: Administration, adaptability, infiltración, seguridad, web crawling.

INTRODUCCIÓN

La importancia de la seguridad y la auditoria en el manejo y desarrollo de páginas web cada vez es más importante en el manejo de la información, para el desarrollo normal de las actividades comerciales y financieras de las empresas requiere una constante vigilancia y evaluación; asimismo, las empresas necesitan una opinión, preferiblemente independiente, que les ayude a medir la eficiencia y eficacia en el cumplimiento de sus objetivos.

En el presente trabajo se pretende plantear el análisis de Penetration Testing que sirven también para determinar el nivel de seguridad en: un equipo, en una red de equipos LAN (Local Área Network) o WLAN (Wireless local Área Network), aplicaciones Web entre otros, por medio de ataques informáticos simulados idénticos a los que realizaría un Cracker o Black Hat Hacker pero sin poner en riesgo la información o la disponibilidad de los servicios, esto se hace con el fin de encontrar las posibles amenazas en los sistemas IT antes de que las descubra un atacante (externo o interno). Este proceso también es conocido como Hacking Ético (Ethical Hacking). Así mismo plantear la solución a la problemática de una pequeña empresa que busca mejorar el desarrollo de su software, para diseñar páginas webs más seguras. La inseguridad de las páginas webs es un problema actual del mundo real. Poner solución a esto requiere de una labor técnica, que se condensa en una auditoría de la aplicación. El proceso de auditoría web que se describe aquí, nos servirá para detectar y resolver las vulnerabilidades del código, o de la configuración del sistema que aloja la aplicación (Pautasso *et al*., 2014)

Tengamos en cuenta también que las herramientas desarrolladas para realizar auditoria de sistemas no son usadas en su totalidad por lo que aquí planteamos el uso correcto de cada una de las herramientas como la forma de proteger un sistema web desde el desarrollo, así como la ejecución y el contante monitoreo de los elementos presentes en cada uno de los accesos del sistema web y los enlaces URL que estos suponen haber sido creados.

Capítulo I: Revisión de la literatura y marco operacional se expresó los estudios de los antecedentes internacionales, nacionales y locales, considerando el marco conceptual que sostiene la investigación y la ilustración de términos.

Capítulo II: Planteamiento del problema se definió el propósito del estudio y enunciado del estudio, así como la fundamentación del problema, la importancia y las limitaciones de la pesquisa.

Capítulo III: Enfoca sobre la materiales y métodos empleada durante la investigación, el diseño, tipo, población, tamaño de la muestra y tratamiento estadístico.

Capítulo IV: En este capítulo de resultados se demostró la confiabilidad y validez de los instrumentos empleados, análisis de los resultados obtenidos durante la investigación y la interpretación de los resultados.

Finalmente se presentó las conclusiones y las recomendaciones de la investigación, seguida por la referencia y finalizando en los apéndices.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1. Marco teórico

1.1.1. Auditoria informática

Es la revisión independiente de alguna o algunas actividades, funciones específicas, resultados u operaciones de una entidad administrativa, realizada por un profesional de la auditoría, con el propósito de evaluar su correcta realización y, con base en ese análisis, poder emitir una opinión autorizada sobre la razonabilidad de sus resultados y el cumplimiento de sus operaciones.

Es la revisión técnica, especializada y exhaustiva que se realiza a los sistemas computacionales, software e información utilizados en una empresa, sean individuales, compartidos y/o de redes, así como a sus instalaciones, telecomunicaciones, mobiliario, equipos periféricos y demás componentes. Dicha revisión se realiza de igual manera a la gestión informática, el aprovechamiento de sus recursos, las medidas de seguridad y los bienes de consumo necesarios para el funcionamiento del centro de cómputo. El propósito fundamental es evaluar el uso adecuado de los sistemas para el correcto ingreso de los datos, el procesamiento adecuado de la información y la emisión oportuna de sus resultados en la institución, incluyendo la evaluación en el cumplimiento de las funciones, actividades y operaciones de funcionarios, empleados y usuarios involucrados con los servicios que proporcionan los sistemas computacionales a la empresa (Muñoz, 2002).

La auditoría, para Ramírez y Álvarez (2003), es aquella actividad consistente en la emisión de una opinión profesional sobre si el objeto analizado presenta adecuadamente la realidad que pretende reflejar y cumple las condiciones que le han sido prescritas. En así que se deduce la importancia de establecer una opinión objetiva, fundada en las evidencias encontradas, sobre las diferencias existentes entre el planteamiento del funcionamiento de cualquier área a auditar y su ejecución real en la organización, y comunicarlas a las personas correspondientes.

Así mismo, una auditoría informática viene a ser la revisión, verificación y evaluación con un conjunto de métodos, técnicas y herramientas de los sistemas de información de una organización, de forma continua y a petición de una organización con el fin de mejorar su rentabilidad, seguridad y eficacia, cuyo propósito es el de asegurar sus sistemas de información, para cumplir sus objetivos estratégicos de la propia organización y que los sistemas prestan el apoyo adecuado a la consecución de estos objetivos, tanto en el presente como en su evolución futura (Pablos *et al*., 2006).

1.1.2. Seguridad informática

La seguridad informática es la aplicación de técnicas y métodos orientados a salvaguardar la información, la misma que se logra con la definición de procesos de seguridad interna y la implementación de herramientas dedicas y especializadas en el manejo y el trato de la información, ofreciéndonos una información fiable y confiable. Teniendo como objetivo mantener integridad, disponibilidad, privacidad, control y autenticidad de la información manejada por un computador (Aldegani, 1997).

Se puede definir seguridad informática como, la protección contra todos los daños sufridos o causados por las herramientas informáticas y originadas por el acto voluntario o de mala voluntad de una persona ajena a la institución. Para proteger al sistema informático hay que poner frenos contra todo tipo de amenazas potenciales y poder multiplicar el número de barreras sucesivas dado que ninguna protección es infalible, así en el momento en el que un usuario extraño pudiera pasar una protección inmediatamente seria bloqueada por otra. También es necesario proteger todos los medios de acceso de una organización, a menudo se ve a la

empresa sobreproteger su conexión a internet, dejando las demás vías de acceso sin protección alguna (Royer, 2004).

También, Lopez (2010), menciona que la seguridad informática es aquella disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas que se ocupan de conseguir un sistema de información seguro y confiable para una organización. Pero a pesar de tomar todas las medidas de seguridad siempre existirá un margen de riesgo por lo que es necesario tener en cuenta lo siguiente:

- ¿Cuáles son los elementos que componen el sistema?, generalmente esta información se obtendrá mediante entrevistas con los responsables de la organización para la que se hace el estudio de riesgos y mediante apreciación directa de estos.
- ¿Cuáles son los peligros que afectan al sistema, accidentales o provocados?, estos se obtienen tanto de los datos aportados por la organización como por el estudio directo del sistema que se realiza al sistema mediante la realización de pruebas y muestreos sobre el mismo.
- ¿Cuáles son las medidas que deberían adoptarse para conocer, prevenir, impedir, reducir o controlar los riesgos potenciales?, aquí se tratara de decidir cuáles son y serán los servicios y mecanismos de seguridad que se utilizaran para reducir los riesgos lo máximo posible.

A partir del estudio de riesgos y la implantación de medidas, se debe hacer un seguimiento periódico de revisión y actualización para las medidas que se adopten, puesto que todo elemento que tienen un sistema informático tiende a ser afectado por fallos de seguridad y a los datos, es por eso que el software es aun el más vulnerable (Aguilera, 2011).

1.1.1.1. Tipos de seguridad informática:

Según Lopez (2010), se tiene 2 tipos de seguridad informática las que se detalla a continuación:

a) **Activa:** Este tipo de seguridad, comprende un conjunto de defensas o medidas cuyo objetivo es evitar o reducir los riesgos que amenazan directamente al sistema.

b) **Pasiva**: Este tipo de seguridad, hace referencia a la integridad, confidencialidad y disponibilidad de los datos que genera un sistema, por lo general estas son las propiedades que debe tener un sistema para considerarlo seguro.

Los tipos de seguridad informática están formadas por las medidas que se establecen para cuando se origina el incidente de seguridad, minimizar su repercusión y facilitar la recuperación del sistema y tener siempre al día copias de seguridad de los datos.

1.1.1.2. Propiedades de un sistema de información seguro

Para Hermoso (2013), la auditoría de seguridad debe entender por qué se desarrolla una aplicación y porque dichas aplicaciones deben de ser protegidas. De esta manera, dejando a un lado los sistemas y plataformas sobre los que funcionan las aplicaciones web se debe tomar importancia al valor que representa de la información que generan dichas aplicaciones. En muchos casos puede dar que la información que genera o trate la aplicación pueda tener carácter confidencial. La información podría variar y ser desde datos introducidos directamente por los usuarios y clientes, hasta datos inferidos o calculados gracias a los hábitos y costumbres del usuario al utilizar dicha aplicación; no siempre toda la información generada por las aplicaciones es confidencial, ni la filtración de ésta tiene porque atentar contra la privacidad de nadie, sí que se podría considerar quizá igual de valiosa.

En tanto menciona el autor que el valor de la información se puede deducir que para una aplicación ésta es crítica y que por lo tanto tener las medidas correctas de seguridad para mantenerla protegida es muy importante. Por lo que indica que existen tres propiedades de la seguridad de la información tales como:

- **Confidentiality (Confidencialidad):** Para no invadir la privacidad de los usuarios comprometiendo la confidencialidad de sus datos y/o acciones, es decir, que sólo tengan acceso a dicha información las personas debidamente autorizadas.

- **Integrity (Integridad):** Se debe conservar la integridad de los datos garantizando que la información es sólo modificada por aquellos que cuentan con los apropiados permisos. apropiados.

- **Availability (Disponibilidad):** Se debe asegurar que la información esté disponible cuando se requiere.

Figura 1: Propiedades de la seguridad de la información.
Fuente: Hermoso (2013)

1.1.1.3. Seguridad de las aplicaciones web

Tenemos entendido que la seguridad es un aspecto importante para proteger la integridad y privacidad de los datos y recursos de su aplicación Web. Por lo tanto se debería designar una estrategia de seguridad para las aplicaciones Web, que usen soluciones de seguridad de eficacia, e implementar métodos de autenticación, autorización y validación de datos, para proteger la aplicación de una serie de amenazas a la que está expuesta.

Dentro de estas tenemos:

a) DIRBUSTER

DirBuster es una aplicación Java multi hilo diseñada para obtener por fuerza bruta los nombres de directorios y archivos en servidores Web/de aplicación. A menudo ocurre que lo que ahora parece un servidor Web en una fase de instalación por omisión no lo es, y tiene páginas y aplicaciones ocultas. DirBuster trata de encontrar estos y demás vulnerabilidades de las páginas web (OWASP, 2008)

Sin embargo, las herramientas de esta naturaleza a menudo son solo tan buenas como la lista de archivos y directorios con los que vienen. Un enfoque diferente fue usado para generar esto. La lista fue generada desde cero, rastreando en Internet y colectando los directorios y archivos que son realmente usados por los desarrolladores.

DirBuster viene con un total de 0 listas diferentes, esto hace a DirBuster extremadamente efectivo encontrando esos archivos y directorios ocultos. Y si eso no fuera suficiente, DirBuster también tiene la opción de realizar fuerza bruta pura, lo que no les deja lugar para esconderse a los archivos y directorios ocultos (Khan, 2018).

b) SQL

Es una herramienta desarrollada en python para realizar inyección de código sql automáticamente. Su objetivo es detectar y aprovechar las vulnerabilidades de inyección SQL en aplicaciones web. Una vez que se detecta una o más inyecciones SQL en el host de destino, el usuario puede elegir entre una variedad de opciones entre ellas, enumerar los usuarios, los hashes de contraseñas, los privilegios, las bases de datos, O todo el volcado de tablas / columnas específicas del DBMS, ejecutar su propio SQL SELECT, leer archivos específicos en el sistema de archivos y mucho más (Tovar, 2015).

Características:

- Soporte completo para MySQL, Oracle, PostgreSQL y Microsoft SQL. Además de estos cuatro sistemas de gestión de bases de datos, sqlMap

también puede identificar Microsoft Access, DB2, Informix, Sybase y Interbase.

- Amplia base de datos de sistema de gestión de huellas dactilares basadas en inband error messages, analizar el banner, las funciones de salida de comparación y características específicas tales como MySQL comment injection. También es posible forzar a la base de datos de sistema de gestión de nombre si ya lo saben.

- Soporte completo para 2 técnicas de SQL injection: blind SQL injection y inband SQL injection.

c) ISO 9126

Es un estándar internacional para la evaluación de la calidad del Software. Está reemplazado por el proyecto SQuare, ISO-25000:2005, el cual sigue los mismos conceptos. Este estándar es el más usado. El estándar está dividido en cuatro partes las cuales dirigen, realidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido. El modelo de calidad establecido en la primera parte del estándar, ISO-9126, clasifica la calidad del Software en un conjunto estructurado de características y sub-características de la siguiente manera:

- Funcionalidad
- Fiabilidad
- Usabilidad
- Eficiencia
- Mantenibilidad
- Portabilidad
- Calidad en uso

d) ISO/IEC 27032

La norma facilita la colaboración segura y fiable para proteger la privacidad de las personas en todo el mundo, ayuda a prepararse, detectar, monitorizar y responder a los ataques de seguridad cibernética, extrayendo los aspectos únicos de esa actividad y de sus dependencias en otros dominios de seguridad, específicamente información de seguridad, seguridad de las redes, seguridad en Internet e información de protección de infraestructuras críticas (CIIP).

Esta norma establece una descripción general de Seguridad Cibernética, una explicación de la relación entre la ciberseguridad y otros tipos de garantías, una definición de las partes interesadas y una descripción de su papel en la seguridad cibernética, una orientación para abordar problemas comunes de Seguridad Cibernética (ISO, 2014).

1.1.2. Vulnerabilidades

Las vulnerabilidades son las debilidades que presenta alguna aplicación que pudiera resultar de manera directa o indirecta un riesgo para la organización como también para los usuarios que la utilizan.

Cuando se habla de sistemas las vulnerabilidades se observará que en muchas ocasiones surgen por utilizar ficheros de configuración por defecto o por copiar éstos de un sistema a otro o incluso de internet cuando no van a tener los mismos requisitos ni características requeridas; como también se puede encontrar es configurar mal los permisos de los ficheros. Un fallo al configurar estos vulneraría los pilares de la seguridad de la información. Otro fallo habitual es encontrarse con el uso de contraseñas por defecto con las que el fabricante distribuye sus productos que en caso de ser cambiadas suele ser por contraseñas débiles que no siguen ningún tipo de política de contraseñas que exijan unos mínimos de complejidad.

Como también, en el software los fallos más comunes aparecen por otros motivos, como proyectos mal planificados dónde el límite de tiempo es el principal enemigo de los desarrolladores puesto que hace que estos se centren más en el desarrollo de las funcionalidades de la aplicación que en implementar controles de seguridad. Otro de los motivos es el desconocimiento de los programadores al no desarrollar de forma segura el software (Hermoso, 2013).

1.1.2.1. Vulnerabilidades de software

OWASP (2013), refiere el termino vulnerabilidad a la a la violación de una política de seguridad. Esto puede deberse a las reglas de seguridad inadecuadas o a problemas que se presentan dentro del mismo software. En supuesto, todos los sistemas de ordenadores tienen vulnerabilidades, cuya seriedad depende que sean o no usados para causar un daño al sistema. Una vulnerabilidad:

- Admite que un atacante ejecute órdenes como otro usuario
- Admite que un atacante tenga acceso a los datos de acceso restringido
- Admite que un atacante hacerse pasar por otra entidad
- Admite que un atacante conduzca una denegación de servicio.
- Admite que un atacante reúna información sobre las actividades del sistema.
- Admite que un atacante disimule sus actividades

1.1.2.2. Enfoques para realizar un análisis de vulnerabilidades

En auditoría una de las fases es el análisis o evaluación de vulnerabilidades que se podrá enfocar utilizando diversas estrategias y varían de acuerdo a los enfoques:

- **Black box (Caja negra):** Para este tipo de análisis los auditores se ponen en la piel de un atacante el cual no tienen conocimiento alguno de la aplicación o de los sistemas asociados a ésta.
- **White box (Caja blanca):** Aqui se presenta a los auditores con todo el conocimiento del aplicativo así como con una copia del código fuente para que la revisen a conciencia.
- **Grey box (Caja gris):** Este enfoque utiliza los dos enfoques anteriores.

El enfoque más apropiado para realizar la evaluación de vulnerabilidades será el de caja negra, puesto que los auditores tendrán el mismo conocimiento que el que pudiera tener un posible atacante. En tanto se deberá entender que éste podría generar alguno que otro falso positivo o no encontrar ninguna vulnerabilidad aun existiendo éstas.

El enfoque que realiza una revisión más exhaustiva y fiable sería la de caja blanca, pero ésta también será la más laboriosa y costosa económicamente. Las auditorías de caja blanca suelen incluir una revisión de código con la que los auditores tendrán total conocimiento sobre la aplicación y podrán identificar de forma sencilla los puntos más relevantes (Hermoso, 2013).

1.1.2.3. Tipos de vulnerabilidades de software

Según OWASP (2013), se tienen los siguientes:

- A1 – Inyección SQL: Corresponde a la inyección de código y es una de las más comunes.
- A2 – Pérdida de autenticación y gestión de sesión: Corresponde al mal manejo de las sesiones en diferentes aplicaciones.
- A3 – Secuencia de comandos en sitios cruzados: Ocurre cuando existe validación de la información ingresada por el atacante. 12
- A4 – Referencia directa insegura a objetos: Ocurre cuando un desarrollador expone información la cual puede ser manipulada por un atacante que puede acceder a datos que no están autorizados.
- A5 – Configuración de seguridad incorrecta: Corresponde a configuraciones no adecuadas que pueden impactar en la seguridad de la propia aplicación.
- A6 – Exposición de datos sensibles: Se refiere a la protección incorrecta de datos críticos tales como: números de tarjetas de crédito, contraseñas, entre otros las cuales pueden ser remplazadas, modificadas o robas.
- A7 – Ausencia de control de acceso a funciones: Corresponde a la falta de controles desde el servidor, para permitir un posible atacante, acceder a funciones a las que no está autorizada.
- a8 - Falsificación de peticiones en sitios cruzados (CSRF): Permite a un atacante generar peticiones sobre una aplicación vulnerable a partir de la sesión de la víctima.

- A9 - Utilización de componentes con vulnerabilidades conocidas*:* Corresponde a la explotación de librerías, framework y otros componentes vulnerables por parte de un atacante con el fin de obtener acceso o tener un impacto grave en el servidor.

- A10 – Redirecciones y reenvíos no validados: Ocurre cuando los atacantes aprovechan el uso de redirecciones de sitios web a otros sitios en el momento de utilizar información no confiable para redirigir a las víctimas a sitios de phishing o que contienen malware.

Tabla 1

Vulnerabilidades Del Software

	AGENTE DE AMENAZA	VECTORES DE ATAQUE	DEBILIDAD DE SEGURIDAD	IMPACTOS TÉCNICO	IMPACTO DE NEGOCIOS
A1: INYECCIÓN SQL	Aquellos que puedan enviar datos que son de confianza en el sistema: - Los usuarios externos - Los usuarios internos - Administrador (es)	Envíos de simples ataques de texto que explotan la sintaxis del intérprete apuntado; cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas.	Ocurre cuando una aplicación envía datos no confiables a un intérprete. A menudo se encuentran en SQL, LDAP o consultas XPath, comandos de sistema operativo, los analizadores XML, argumentos del programa, etc. Son fáciles de descubrir al examinar el código, pero más difícil a través de pruebas. Escáneres y fuzzers ayudan	Él envió de datos no autorizados puede causar la pérdida de datos o la corrupción, la denegación de acceso.	Los datos pueden ser hurtados, modificados o eliminados. Podría ser dañado el perfil de la organización
A2: PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIÓN	Atacantes externos anónimos, como los usuarios con sus propias cuentas, que pueden intentar robar cuentas de los demás.	Utilizan fugas o fallas en las funciones de gestión de autenticación o sesión para que puedan hacerse pasar por usuarios.	Los desarrolladores construyen sistemas de gestión de sesiones y autenticación personalizada, pero no siempre esta es la correcta. Por lo general tienen deficiencias en áreas tales como: - Cierre de sesión - Administración de contraseñas - Tiempos de espera - Pregunta secreta - Cuenta de actualización y otros Encontrar este tipo de defectos puede ser difícil de identificarlas, ya que cada aplicación es única.	Tales defectos pueden permitir que algunas o incluso todas las cuentas sean atacadas. Una vez conseguido esto, el atacante puede hacer nada a la víctima podía hacer. Las cuentas con privilegios son con frecuencia blanco de ataques.	El valor de negocio de los datos afectados o funciones de la aplicación.

A3: SECUENCIA DE COMANDOS EN SITIOS CRUZADOS	Aquellos que puedan enviar datos que son de confianza en el sistema: - Los usuarios externos - Los usuarios internos Administrador (es)	Se envía secuencias de comandos de ataque basados en texto que se aprovechan de la intérprete en el navegador. Casi cualquier fuente de datos puede ser un vector de ataque, incluyendo fuentes internas tales como los datos de la base de datos.	El XSS es la más frecuente falla de seguridad en las aplicaciones web, estas ocurren cuando una aplicación incluye datos proporcionados por el usuario en una página enviados al navegador sin validar correctamente el contenido. Existen tres tipos conocidos de fallas XSS: - Almacenados - Refleja - DOM XSS basado.	Se pueden ejecutar scripts en el navegador de la víctima para: - Retener sesiones de usuario - Modificar sitios web - Insertar contenido incompatible - Redirigir a los usuarios - Retener el navegador del usuario con malware	Valor de negocio del sistema afectado y todos los datos que procesa. Impacto en el negocio de la exposición pública de la vulnerabilidad.
A4: REFERENCIAS DIRECTA INSEGURA A OBJETOS	Tipos de usuarios de su sistema. ¿Alguno de los usuarios sólo tienen acceso parcial a ciertos tipos de datos del sistema?	El usuario autorizado del sistema, simplemente cambia el valor del parámetro que se refiere directamente a un objeto del sistema a otro objeto que el usuario no está autorizado para el acceso al sistema.	Las aplicaciones suelen utilizar el nombre real o la clave de un objeto al generar páginas web y no siempre verifican que el usuario está autorizado para el objeto de destino. Esto resulta en un defecto de referencia objeto directo insegura. Los testers pueden manipular fácilmente los valores de parámetros para detectar ese tipo de errores y análisis de código muestra rápidamente si la autorización se verifica correctamente.	Los defectos pueden comprometer todos los datos que pueden ser referenciados por el parámetro. A menos que el espacio de nombres es escasa ahí es fácil para un atacante acceder a todos los datos disponibles de este tipo.	El valor comercial de los datos expuestos. El impacto en el negocio de la exposición pública de la vulnerabilidad.

A5: CONFIGURACIÓN DEFECTUOSA DE SEGURIDAD	Atacantes externos anónimos Usuarios con sus propias cuentas que pueden intentar poner en peligro el sistema.	Acceso a las cuentas por defecto, las páginas no utilizadas, defectos sin parches, archivos desprotegidos y directorios, etc., para obtener acceso no autorizado o el conocimiento del sistema.	Reconfiguración de Seguridad puede ocurrir a cualquier nivel de las aplicaciones, incluyendo la plataforma, servidor web, servidor de aplicaciones, el marco y el código personalizado. Los desarrolladores y los administradores de red deben trabajar en conjunto para asegurar que toda la pila está correctamente configurado.	Los defectos suelen darse en el acceso no autorizado a algunos datos del sistema o la funcionalidad. En ocasiones, estas fallas dan como resultado un compromiso total del sistema.	El sistema puede estar comprometido sin que nadie lo sepa. Todos los datos pueden ser hurtados o modificarse lentamente con el tiempo. Los costos de recuperación podrían ser costoso.
A6: LA EXPOSICIÓN DE DATOS SENSIBLES	Atacantes externos anónimos Usuarios con sus propias cuentas que pueden intentar poner en peligro el sistema.	Pueden acceder a las cuentas por defecto: - Las páginas no utilizadas - Defectos sin parches - Archivos desprotegidos - Directorios, etc. Esto para obtener acceso no autorizado o conocer el sistema.	Reconfiguración de Seguridad puede ocurrir a cualquier nivel de un conjunto de aplicaciones, incluyendo la plataforma, servidor web, servidor de aplicaciones, el marco y el código personalizado. Los desarrolladores y los administradores de red deben trabajar en conjunto y asegurarse que toda la pila está correctamente configurado.	Los defectos suelen dar los atacantes el acceso no autorizado a algunos datos del sistema o la funcionalidad. A veces estas fallas dan como resultado un compromiso total del sistema.	El sistema puede ser completamente comprometido sin que nadie lo sepa. Todos los datos pueden ser hurtados o modificarse lentamente con el tiempo. Los costos de recuperación podrían ser altos.

A7: FALTA DE FUNCIÓN QUE CONTROLA EL NIVEL DE ACCESO	Cualquier persona que pueda cargar contenido en los navegadores de los usuarios, y así obligarlos a presentar una solicitud para su sitio web. Cualquier otro sitio web o de otros alimentos HTML que el acceso del usuario puede hacer esto.	Generalmente crean solicitudes y trucos la victima HTTP forjadas en la presentación de ellos por medio de las etiquetas de imagen, XSS, o muchas otras técnicas. Si al usuario está se le reconoce como autenticado, el ataque tiene éxito.	El CSRF se aprovecha del hecho de que la mayoría de las aplicaciones web permiten a los atacantes predecir todos los detalles de una acción en particular. Dado que los navegadores envían credenciales como cookíes de sesión automáticamente, los atacantes pueden crear páginas web maliciosas que generan peticiones forjados que son indistinguibles de los legítimos.	Los atacantes pueden causar víctimas para cambiar los datos a la víctima se le permite modificar o realizar cualquier otra función que la víctima está autorizado a utilizar o solicitar cambio de estado, como el cierre de sesión o registrarte.	Considere el valor de negocio de los datos afectados o funciones de la aplicación. Imagine que no es seguro si los usuarios de la intención de tomar estas acciones. Considere el impacto en su reputación.
A8: FALSIFICACIÓN DE PETICIONES EN SITIO CRUZADO	Considere la posibilidad de cualquier persona que pueda engañar a los usuarios en la presentación de una solicitud a su sitio web. Cualquier sitio web o de otros alimentos HTML que los usuarios pueden hacer uso de este.	Enlaces atacante redirigir a validados y trucos victimas a hacer clic en él. Las víctimas son más propensos a hacer clic sobre el mismo; ya que el enlace es a un sitio válido. El atacante se dirige inseguro hacia adelante para eludir los controles de seguridad.	Las aplicaciones redirigen a los usuarios a otras páginas y a veces, la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes para elegir la página de destino. Detectar y busque redirecciones donde puede configurar la dirección URL completa. Forwards sin comprobar son más difíciles, ya que se dirigen a las páginas internas.	Estas redirecciones pueden intentar instalar malware o engañar a las víctimas para que revelen contraseñas u otra información delicada. Forwards no seguras pueden permitir derivación de control de acceso.	El valor de negocio de conservar la confianza de sus usuarios.
A9: UTIL IZAC	Algunos de los componentes vulnerables por lo	Se envía simples ataques basados en texto que explotan la	Identifican un componente débil por barrido o un análisis manual, para	Cada aplicación tiene estos problemas porque la mayoría de los equipos de	Tener en cuenta lo que la vulnerabilidad podría significar para

	general se pueden identificar y explotar con herramientas automatizadas, la ampliación de la piscina agente de amenaza más allá de los atacantes dirigidos a incluir actores desordenados.	sintaxis del intérprete apuntado. Las fuentes de datos puede ser un vector de inyección, como las fuentes internas.	personalizar el exploit según sea necesario y ejecutar el ataque.	desarrollo no se centran en garantizar que sus componentes permanecen hasta la fecha. En muchos casos, los desarrolladores no conocen ni todos los componentes que están utilizando, no importa sus versiones.	la empresa controlada por el application.lt afectado podría ser trivial o podría significar compromiso completo.
A10: VULNERABILIDAD DE REDIRECCIONES Y DESTINOS INVÁLIDOS	Considere la posibilidad de cualquier persona que pueda engañar a los usuarios en la presentación de una solicitud a su sitio web. Cualquier sitio web o de otros alimentos HTML que los usuarios pueden hacer uso de este.	Enlaces atacante redirigir a validados y trucos victimas a hacer clic en él. Las víctimas son más propensos a hacer clic sobre el mismo; ya que el enlace es a un sitio válido. El atacante se dirige inseguro hacia adelante para eludir los controles de seguridad.	Con frecuencia las aplicaciones redirigir a los usuarios a otras páginas, o utilice internos hacia delante de una manera similar. A veces, la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes para elegir la página de destino. Detectar redirecciones sin control es fácil. Busque redirecciones donde puede configurar la dirección URL completa. Forwards sin comprobar son más difíciles, ya que se dirigen a las páginas internas.	Estas redirecciones pueden intentar instalar malware o engañar a las víctimas para que revelen contraseñas u otra información sensible. Forwards no seguras pueden permitir derivación de control de acceso.	El valor de negocio de conservar la confianza de sus usuarios.

Fuente: OWASP (2013).

1.1.3. Aplicaciones Web

Una aplicación web es una aplicación informática o tipo de software que está basada en un código de lenguaje de programación como HTML, JavaScript o CSS, la cual esta soportada para ser ejecutada en un navegador y puede ser accedida vía web en una red interna o externa. Entonces, se puede definir como un programa informático basado en un lenguaje que se ejecuta en un entorno del navegador web (Wiboo media, 2017).

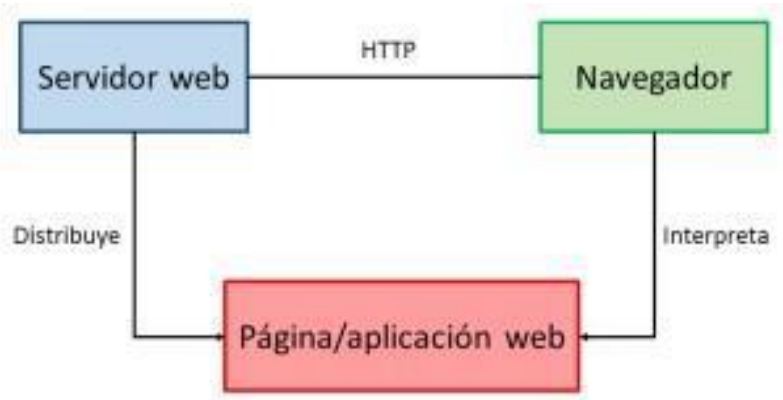


Figura 2: Modelo esquemático de una aplicación web
Fuente: Wiboo media (2017)

Este, responde a un modelo centralizado donde se publica en un lugar común para todos los dispositivos, así no es necesario descargar periódicamente nuevas actualizaciones, sino que al acceder a una URL se dispone siempre del último contenido. Además, todos los desarrollos son multiplataforma lo que permite el acceso a la aplicación o el uso del mismo desde cualquier sistema operativo o móvil.

Sin embargo, una aplicación web, en términos de lenguaje y desarrollo se podría decir que no se diferencia mucho de un sitio web puesto que trabaja de forma similar, puesto que una aplicación web tiene la funcionalidad de realizar una acción en específico por el cliente como, por ejemplo, transacciones bancarias, visualización de correo electrónico, compras online y otras (GCFAprendeLibre, 2016).

1.1.3.1. Evolución de las aplicaciones Web

Desde los años 90, los sitios web eran utilizados específicamente como folletos digitales donde el principal uso era el de publicar información. Hoy en día son sitios más grandes y complejos; puesto que, con el desarrollo de las tecnologías,

internet, la aparición de las aplicaciones de comercio electrónico y las páginas dinámicas, estos han dejado de ser solamente folletos digitales y han pasado a ser aplicaciones de software.

La Web 1.0, surge en los años 90, fue la forma más básica de desarrollo web puesto que solo se tenía navegadores que solamente procesan texto, siendo bastante rápidos pues solo ofrecían la lectura de información. En cuando al usuario, no podía interactuar con el contenido de la página, siendo totalmente limitado a lo que los administradores publiquen en ésta (Fuchs *et al.*, 2010).

A continuación, veremos algunos elementos que permitieron identificar los diseños típicos de un sitio Web 1.0:

- Páginas estáticas en vez de dinámicas por el usuario que la visita.
- El uso de framesets o Marcos.
- Libros de visitas online o guestbooks.
- Formularios HTML enviados vía email.
- No se pueden introducir comentarios ni otro tipo de contenido.
- Todas sus páginas se crean de forma estática y muy pocas veces se actualizan.
- Cuando una página es actualizada no se trata de una nueva versión, sino de una nueva forma de ver y organizar las cosas.

La Web 2.0 presento sitios web que facilitan el hecho de compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración web; también permitió a los usuarios interactuar y colaborar entre sí como los creadores de contenido, es decir por usuarios en una comunidad virtual, a diferencia de la web 1.0 (Areitio y Areitio, 2008).

La Web 3.0 se utiliza para describir la evolución del uso y la interacción de las personas en internet de diferentes formas entre los que se incluyen la transformación de la red en una gran base de datos, donde se crean contenidos accesibles por múltiples aplicaciones, se aplican tecnologías de inteligencia artificial, la web semántica, la web 3D, etc. Se basa fundamentalmente en la idea

de añadir metadatos semánticos y ontológicos a la World Wide Web. Su objetivo fue mejorar Internet, ampliando la interoperabilidad entre los sistemas informáticos usando agentes inteligentes. Con la web 3.0 se busca que los usuarios puedan conectarse desde cualquier lugar, cualquier dispositivo y en cualquier momento (Hendler, 2009).

En la web 4.0, las aplicaciones ya no estarán en las PC‟s, estarán en el internet y serán accesibles desde cualquier lugar. Se utilizara una red inteligente cuyo objetivo será el de unir las inteligencias donde tanto las personas como las cosas se comuniquen entre sí para generar la toma de decisiones. En el futuro se espera que haya agentes en la Web que conozcan, aprendan y razonen como lo hacen las personas (Aghaei *et al.*, 2012).

Ya hoy se está hablando de un nuevo concepto que está aún en desarrollo y es conocido como la Web Ubicua, pues la idea que se persigue es que se vayan complementando algunas tecnologías y lo demás dependerá de la imaginación de los desarrolladores y de las principales empresas del sector, pues es elevadísima la inversión que se está dedicando a I+D en esta dirección, donde ya se observan tecnologías futuristas como Google Glass, Microsoft Surface, Siri de Apple, etc. Y que sin duda alguna está cambiando la manera de explotar la información empresarial y su inversión económica. (López, *et al.*, 2012).

1.1.3.2. Tipo de aplicaciones web

Según Alarcón (2016), las aplicaciones web se usan generalmente en smartphones y tablets, sin dejar de lado los computadores. Existen aplicaciones nativas como un tipo de software que se adapta a una plataforma determinada, o también aplicaciones web que es otro tipo de software que tiene lugar en el navegador web y guarda ciertas diferencias con respecto a las denominadas aplicaciones nativas. Las aplicaciones web se aplican en entornos como:

a) **Web mail:** Sistemas de acceso al correo electrónico por medio de un navegador web.

b) **Wikis:** Sitios y aplicaciones web cuyas páginas y contenidos pueden ser editados directamente por los usuarios finales.

c) **Weblogs:** Sitios y aplicaciones web cuyas páginas y contenidos son de fácil actualización, de tal manera que permite a sus autores publicar contenidos con presionar un solo botón.

d) **Tiendas online:** Tipo de comercio que se usa como medio principal para realizar transacciones desde un sitio web o una aplicación conectada a internet en la que los usuarios y clientes pueden realizar sus compras.

1.1.3.3. Arquitectura de capas

Una aplicación web puede ser desarrollada de distintas formas puesto que existen muchas variaciones posibles; las aplicaciones web son aplicaciones que generalmente se desarrollan distribuidas en 3 capas o niveles y se utiliza el navegador web como la primera capa, la lógica de negocio sería la 2da capa la que se puede desarrollar en diferentes lenguajes de programación como PHP, Java, Python, Perl, ASP.NET, etc y la base de datos sería la última capa (Salgado, 2014).

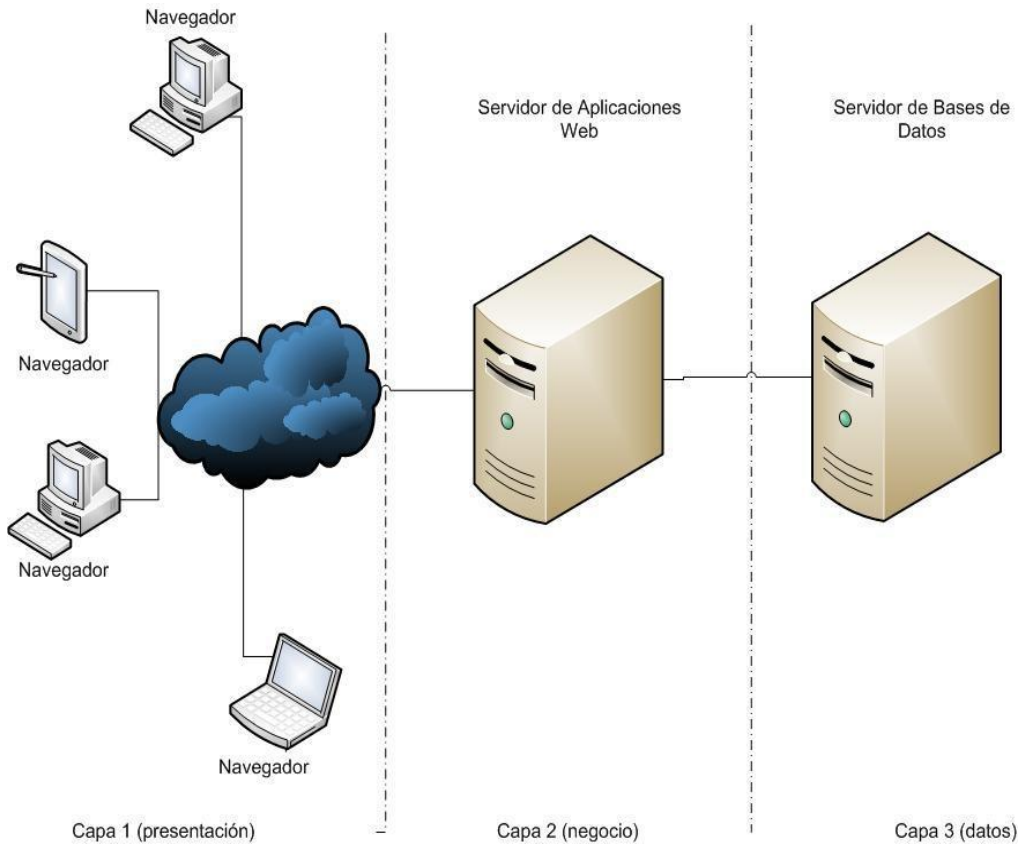


Figura 3: Arquitectura Web en 3 Capas
Fuente: Salgado (2014)

El navegador web, al cual toma el nombre de cliente envía peticiones a la capa media, la cual las procesa y accede a la base de datos para realizar las transacciones solicitadas. Las peticiones realizadas entre el cliente y el servidor se realizan a través del protocolo HTTP y se emplea el lenguaje HTML para mostrar el contenido de respuesta a la petición inicial entre cliente y servidor. (Salgado, 2014)

1.1.3.4. Lenguajes de programación

a) HTML5

Es un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red.

HTML5 es, de hecho, una mejora de esta combinación, el pegamento que une todo. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. A partir de ahora, HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales. Las barreras entre sitios webs y aplicaciones finalmente han desaparecido. Las tecnologías requeridas para el proceso de integración están listas. El futuro de la web es prometedor y la evolución y combinación de estas tres tecnologías (HTML, CSS y Javascript) en una poderosa especificación está volviendo a Internet la plataforma líder de desarrollo. HTML5 indica claramente el camino (Gauchat, 2012).

Está reemplazando previos complementos o plugins, como Flash o Java applets, por nuevas y mejores tecnologías ahora incluso WebGL y Canvas para representación gráfica de datos. El desarrollo web contribuye a la simplificación de la interfaz de desarrollo sobre el problema real de transportabilidad del programa final, como el caso particular de un correo electrónico o página web que únicamente requiere ejecutarse sobre un navegador web, dejando al desarrollador concentrarse en el algoritmo y el objetivo planteado. Las aplicaciones web están diseñadas para apoyar el desarrollo de sitios web dinámicos, aplicaciones y servicios web. Las aplicaciones Web intentan aliviar

el exceso de carga asociado con actividades comunes usadas en desarrollos web. Por ejemplo, muchos de ellos proporcionan bibliotecas para acceder a bases de datos, administración general de archivos y carpetas, mantenimiento de ellas, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código (Pautasso *et al.*, 2014).

b) JAVA SCRIPT

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos (Upton, 2015).

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262 (Upton, 2015).

c) PHP

El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (FormInterpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI. PHP es uno de los lenguajes más populares dentro del software libre en la programación para Web, su diversidad y soporte han sido fundamentales en lograr esta popularidad. Principales características de PHP:

Acceso a gran número de gestores de bases de datos (Adabas D, dbm, dBase, filePro, Hyperwave, Informix, Internase, LDAP, Microsoft SQL server, mSQL, MySQL, MariaDB, ODBC, Oracle, PostgreSQL, Solid y Sybase).

- Envío de correo con SMTP.

- Acceso a servidores de FTP.
- Acceso a SNMP para gestión de redes y equipos.
- Generación dinámica de gráficos y documentos PDF.
- Análisis de documentos XML.
- Generación de datos en WDDX (Intercambio Web de Datos distribuidos).
- Soporte de hilos de ejecución a partir de PHP 4

1.1.3.5. Protocolo HTTP

El protocolo de transferencia de hipertexto, conocido también como HTTP, actúa como un protocolo cliente – servidor mediante el cual se transfiere información entre los clientes Web y los servidores HTTP, es el más utilizado en internet. Esta se basa en operaciones sencillas de solicitud/repuesta de un mensaje con los datos de la solicitud. El servidor responderá con un mensaje similar, que contiene el estado de la operación y su resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan identificado mediante un localizador uniforme de recursos (URL) (Donate, 2005).

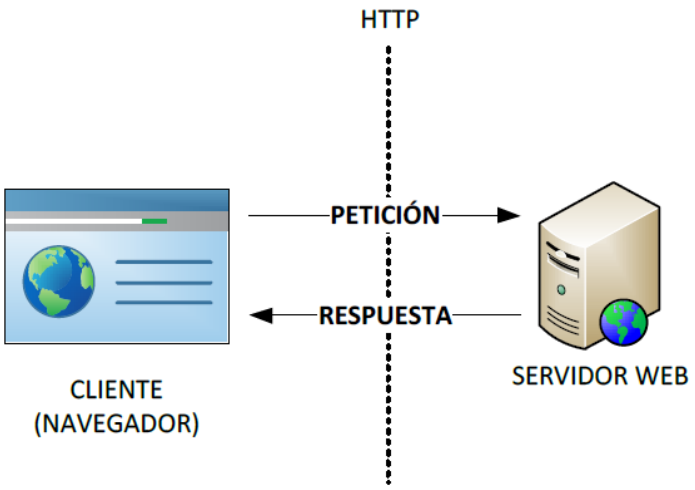


Figura 4: Protocolo HTTP
Fuente: Donate (2005).

El protocolo seguro de transferencia de hipertexto, conocido como HTTPS, cifra los datos al momento de transmitirlos de una forma más segura a través de internet, está basado en el protocolo HTTP. HTTPS es utilizado generalmente

en instituciones financieras por solicitar datos personales y contraseñas o cualquier sitio web que necesite información personal para completar una transacción en línea.

El sistema HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible (usuario y claves de paso normalmente) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar (Mateu, 2004).

1.1.3.6. Pruebas de Software

El objetivo de la prueba del software es asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación apropiada de datos, y la implementación apropiada delas reglas de negocio. Este tipo de pruebas se basan en técnicas de caja negra, esto es, verificar el sistema (y sus procesos internos), la interacción con las aplicaciones que lo usan vía GUI y analizar las salidas o resultados. En esta prueba se determina que pruebas de sistema (usabilidad, volumen, desempeño, etc.) aseguraran que la aplicación alcanzara sus objetivos (Sommerville, 2005).

La prueba del sistema incluye:

- Prueba funcionalidad
- Prueba usabilidad
- Prueba de performance
- Prueba de documentación y procedimientos
- Prueba de seguridad y controles
- Prueba de volumen
- Prueba de esfuerzo (Stress)

- Prueba de recuperación
- Prueba de múltiples sitios

Para sistema web se recomienda especialmente realizar mínimo el siguiente grupo de pruebas de sistema:

- Humo
- Usabilidad
- Performance
- Funcionalidad

1.1.4. Seguridad en las aplicaciones web

Hoy en días las organizaciones realizan el esfuerzo significativo y una elevada inversión para asegurar que la información y recursos se encuentren protegidos cuando estos se encuentren expuestos a sus servicios de la red. El Internet se ha convertido un factor primordial en las comunicaciones y también un evidente riesgo de acceso y de mal uso de los servicios e información que se tiene disponibles. Se clasifican como sistemas más críticos aquellos donde la seguridad debe de ser muy significativa, pero en general todas las aplicaciones web deben de estar protegidos y asegurados ante los principales ataques.

Tabla 2

Seguridad en aplicaciones Web

Disponibilidad	Propiedad o característica de los activos consistentes en que las entidades o procesos autorizados tienen acceso a los mismos cuando lo requieren.
Autenticidad	Propiedad o característica consistente en que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos
Integridad	Propiedad o característica consistente en que el activo de información no ha sido alterado de manera no autorizada.
Confidencialidad	Propiedad o característica consistente en que la información ni se pone a disposición, ni se revela a individuos, entidades o procesos no autorizados.
Trazabilidad	Propiedad o característica consistente en que las actuaciones de una entidad pueden ser imputadas exclusivamente a dicha entidad.

Fuente: UNAM-CERT (2009).

Las aplicaciones web son el punto más común para los ataques informáticos por su fácil acceso a través de internet, muchas de ellas contienen información sensible de instituciones que mueven todo su negocio mediante una aplicación web. Una institución u organización mientras más va automatizando sus procesos mediante aplicaciones web, se vuelve más importante la necesidad de implementar seguridad en sus procedimientos e información (UNAM-CERT, 2009).

Los problemas de seguridad en los sitios web se encuentran a nivel aplicación y son el resultado de la escritura de código con problemas. Programar aplicaciones web seguras no es una tarea fácil, pues se requiere por parte del programador, una concepción general de los riesgos que puede correr la información contenida, solicitada y recibida por el sistema. En la actualidad, aunque existen muchas publicaciones que permiten formar un criterio sobre el tema, no existen acuerdos básicos sobre lo que se debe o no se debe hacer (Areitio, 2008).

1.2. Antecedentes

Bacudio y Yuan (2011), en su investigación afirman que las pruebas de penetración son una serie de actividades realizadas para identificar y explotar vulnerabilidades de seguridad. Ayuda a confirmar la efectividad o ineficacia de las medidas de seguridad que se han implementado. Este documento proporciona una descripción general de las pruebas de penetración. Discute los beneficios, las estrategias y la metodología para realizar pruebas de penetración. La metodología de las pruebas de penetración incluye tres fases: preparación de la prueba, prueba y análisis de prueba. La fase de prueba implica los siguientes pasos: recopilación de información, análisis de vulnerabilidad y explotación de vulnerabilidad

Codina (2006), en tu trabajo, menciona que la evaluación de recursos digitales es una disciplina de las Ciencias de la Documentación en que la Web, por primera vez en un recurso creíble y valioso para académicos y profesionales. Sin embargo, esos recursos valiosos han compartido siempre su lugar en la Web con sitios (a) de interés ridículo, (b) fraudulentos, (c) plagados de errores o (d) las tres cosas a la vez. Por tal motivo, como hemos señalado, en algún momento de la década de los noventa, a la vez que fue quedando clara el gran interés cultural, intelectual y científico de la Web como fuente de

información, emergió la necesidad de desarrollar métodos que permitieran separar, por decirlo en modo metafórico, el grano de la paja o, en términos más técnicos, métodos que permitieran determinar qué sitios web merecían formar parte de directorios o bases de datos de recursos digitales de calidad ofrecidos a usuarios del mundo académico o profesional.

Franco y Guerrero (2013), en su investigación concluyen que el sistema de Administración de Controles de Seguridad Informática basado en ISO/IEC 27002. La gestión de la seguridad de la información es un factor cada vez más determinante en la competitividad de las organizaciones. La gestión del riesgo y el aseguramiento de la información se apoyan en la aplicación de normas internacionales como el estándar ISO/IEC 27002. El proceso de implementación de la norma y su gestión permanente se facilita a través del uso de software que en la actualidad es mayoritariamente comercial. La restricción de idioma, poca documentación y limitada disponibilidad de software abierto que se ajuste a requerimientos particulares de organizaciones en el contexto latinoamericano, limita la aplicación de la norma y la efectividad de su uso. Este artículo presenta una plataforma web abierta denominada SGCSI que permite apoyar la gestión de controles de seguridad de la información de acuerdo con el estándar ISO 27002. Tras la evaluación comparativa del uso de la plataforma por parte de expertos y aprendices en seguridad, se pudo evidenciar su efectividad en la auditoria sobre el cumplimiento de los 32 objetivos de control establecidos por la norma.

Romaniz (2008), en su artículo menciona que las complejas y sensibles funcionalidades de las actuales aplicaciones web ha movido el perímetro de seguridad de las organizaciones, y una parte significativa del mismo ahora reside en las propias aplicaciones web. Y los privilegios de acceso a funcionalidades y datos ya no son uniformes y abiertos, sino que requieren de complejos esquemas., resultando esencial la fortaleza de los mecanismos de control de acceso. Las debilidades en los controles de acceso pueden surgir de diferentes fuentes: un diseño pobre de la aplicación hace muy difícil y hasta imposible el chequeo por accesos no autorizados, un simple descuido puede dejar desprotegidas funcionalidades críticas, o suposiciones erróneas acerca del comportamiento de los usuarios dejan a una aplicación web sin protección y pasible de un quiebre de seguridad. En muchos casos, detectar una brecha en los controles de acceso puede resultar trivial, pero en otros casos, puede ser muy difícil, quedando ocultos defectos sutiles dentro de la lógica de la aplicación, especialmente en aplicaciones

complejas y de alta seguridad. La lección más importante es que cuando se chequea la robustez de los controles de acceso se debe mirar en todas direcciones, debiendo ser paciente y testear cada paso particular de todas las funcionalidades de la aplicación.

Bermúdez y Bailón (2015), en su investigación tuvieron como objetivo analizar los procesos críticos de Credigestión respecto a las gestiones de seguridad adecuadas para garantizar la confidencialidad, integridad y disponibilidad de la información, mediante la formulación recomendaciones de seguridad y controles basados en la Norma ISO/IEC 27001, llegando a las conclusiones que se debe elaborar un manual de políticas de seguridad de la información donde se detallen controles de seguridad acorde a la realidad y necesidades actuales de la empresa, la constante concienciación a los funcionarios, así como el monitoreo continuo, encamina a la empresa a la correcta gestión de la seguridad de la información; como también la seguridad total no existe, pero gestionar controles de seguridad en el proceso y manejo de la información, se vuelve un complemento esencial, pues le permite asegurar información valiosa no sólo de la empresa sino también de los clientes.

Hernández y Mejia (2015), en su estudio concluyen que existen muchas herramientas que proporcionan la detección para diferentes propósitos, es decir, algunas herramientas cubren desde escaneo de vulnerabilidades en aplicaciones web, hasta escaneo de vulnerabilidades en dispositivos móviles; como también indican que existen herramientas muy específicas para la detección de problemas de seguridad muy específica, como en el que solamente se enfoca en el escaneo de sitios web. Sin embargo, aún con la existencia de las herramientas antes mencionadas las organizaciones continúan con un desconocimiento de cuándo deben ser utilizadas, por lo tanto, la matriz de trazabilidad propuesta se considera una aportación muy importante para las organizaciones ya que proporciona una guía para la utilización de técnicas y herramientas sobre una vulnerabilidad en específico, que permitirá prevenir un tipo de ataque.

Hermoso (2013), en su estudio se tiene en cuenta la complejidad que presenta el protocolo HTTP debido a los diferentes métodos de transferencia y codificaciones soportadas, se observa que el desarrollo se convierte en una tediosa tarea dónde se tiene que poner especial atención y dónde un pequeño error en el manejo y procesado de estos datos puede resultar en la aplicación inutilizada; en tal sentido indico que sería necesario trabajar el

proyecto teniendo en cuenta los requisitos de modularidad y portabilidad, puesto que estos eran las elementos diferenciadores de esta herramienta.

Tituaña (2017), en su investigación, llego a la conclusión que el Municipio del Distrito Metropolitano de Quito cuenta con una dependencia la cual es la responsable de manejar la infraestructura informática del MDMQ, esta cuenta con los mecanismos y dispositivos de seguridad que permitan detectar intrusiones y ataques, también ha establecido controles, políticas y buenas prácticas para mejorar la Seguridad de la Información. Sin embargo, el MDMQ ha sufrido incidentes de seguridad, es por eso que se realizaron las pruebas de penetración con el propósito de identificar si existen o no vulnerabilidades en las aplicaciones web que presta el MDMQ; por otro lado indica que existen varias metodologías para realizar pruebas de seguridad, pruebas de penetración y evaluaciones de seguridad en redes, como OSSTMM, ISSAF, NIST800-115, PTES Sin embargo, varios de los servicios y aplicaciones que presta el MDMQ son a través de Internet, es por esta razón que utilizó la Guía Open Web Application Security Project (OWASP) V4.

Almanza y Cano (2010), en su estudio observaron un gran esfuerzo considerando la panorámica latinoamericana. Es importante resaltar que dentro de este laborioso esfuerzo son significativos los cambios en la forma cómo la seguridad informática se ha transformado y juega un papel importante dentro de la realidad nacional. Este año la participación en la X Encuesta Nacional de Seguridad Informática fue de 194 personas de los diferentes sectores productivos del país. Como en el año anterior, otros países adelantaron el mismo ejercicio, entre ellos México, a través de la Universidad del Valle de Atemajac (UNIVA), cuyos resultados están disponibles en el sitio web de dicha institución. Así mismo, Uruguay, Argentina y Paraguay. El análisis presentado a continuación está basado en una muestra aleatoria de la encuesta interactiva, realizada a través de una página web dispuesta por la Asociación Colombiana de Ingenieros de Sistemas (ACIS) para tal fin. Dadas las limitaciones de tiempo y recursos disponibles en la Asociación, se ha realizado un conjunto de análisis básicos, los cuales pretenden ofrecer los elementos más sobresalientes de los resultados obtenidos para orientar al lector sobre las tendencias identificadas en el estudio. Con esto en mente y considerando otros estudios internacionales como el 2010 Global State of Information.

Aristides (2019), en su artículo tuvo como objetivo construir un modelo que permita, a quien adopte la Web Security Testing Guide a la vez conocer el grado de cumplimiento

de los lineamientos de dicha guía así como decidir sobre qué aspectos de la misma se pondrá mayor o menor énfasis y esto será posible con el método de evaluación adoptado que va más allá de un simple método de evaluación aditivo. Por lo tanto concluyo que el método de evaluación elegido para la construcción del modelo de evaluación, el método Logic Score of Preference (LSP), permite que puedan darse no solo mayor o menor peso a distintos ítems, sino que algunos de estos ítems puedan ser considerados opcionales y otros mandatorios, pudiendo cada uno contar con un grado determinado de esa cualidad, permitiendo de esta manera adaptar el modelo a las necesidades particulares de la organización.

Salgado (2014), en su investigación, tuvo como objetivo analizar las aplicaciones web de la SBS para identificar los riesgos de seguridad más comunes mediante el top ten de OWASP y definir buenas prácticas para el aseguramiento de las aplicaciones. Llegando a las siguientes conclusiones como son: Las acciones que realizan el manejo de información es elevado, por ser sistemas que constantemente actualizan sus datos y son esenciales para el correcto funcionamiento de SBS, se seleccionaron algunas de las aplicaciones de la SBS que se encuentran desarrolladas bajo la plataforma JEE; también se identificó las vulnerabilidades, amenazas y riesgos más comunes presentes en una aplicación Web, a partir del estudio realizado sobre las tecnologías web, las metodologías aplicadas para el desarrollo sobre plataformas JEE, las tecnologías utilizadas y las arquitecturas. Por otro lado se realizó una evaluación de los riesgos basados en el Top Ten de OWASP, de las vulnerabilidades, amenazas, el impacto, detectándose que en las aplicaciones de la SBS hay dos riesgos bien claros que presentan una vulnerabilidad media-alta, al igual que la amenaza, pudiendo llegar el impacto a ser grande en caso de ser explotada; estos riesgos son: Almacenamiento Criptográfico Inseguro y Protección Insuficiente en la Capa de Transporte. Por último se realizó una lista de buenas prácticas para asegurar las aplicaciones, corregir los riesgos detectados y los que estén por surgir en el proceso de desarrollo de nuevas funcionalidades en las aplicaciones estudiadas, partiendo de los resultados obtenidos y siguiendo los resultados encontrados en la investigación.

Briones y Hernandez (2018), en su investigación, tuvieron como objetivo Auditar la seguridad en el servidor web de la empresa PUBLINEXT S.A. utilizando mecanismos basados en OWASP. Llegando así a las conclusiones que en la aplicación web se encontraron varias vulnerabilidades provenientes desde cómo se ha desarrollado la

aplicación y la forma como sed tiene configurado el servidor web, las vulnerabilidades identificadas fueron consideradas con un nivel de riesgos entre medio y bajo debido al impacto que estas pueden causar en caso de que un atacante aproveche dicha vulnerabilidad; como también los análisis del checklist reflejan que la empresa se defiende adecuadamente contra la mayoría de los riesgos asociados con el software de hoy en día, la empresa se encuentra en un nivel ajustado para aplicaciones que manejan transacciones business-to-business, implementan funciones sensibles o críticas o incluyen el proceso de otros activos sensibles. Por ultimo se planteó una solución que garantice el aseguramiento de los servicios web y de los datos, mediante el uso de un componente de seguridad como medida de autenticación (Autenticación JAAS) que trabaja en conjunto con una aplicación (REST), que ofrecería al negocio visibilidad, escalabilidad y rendimiento.

Valle y Gavidia (2015), en su investigación tuvieron como objetivo Analizar las vulnerabilidades de Software para mejorar la seguridad en los Sistemas Informáticos. Llegando a las siguientes conclusiones que el análisis de vulnerabilidades ha permitido identificar las 3 debilidades más comunes y que más afectan a los sistemas informáticos a nivel de software que son inyección sql, secuencia de comandos de sitios cruzados y falsificación de peticiones de sitios cruzados a través de técnicas y herramientas que ha permitido poder evaluar los factores de riesgo de explotación, detección prevalencia e impacto que estas tienen cuando se produce un ataque a dichos sistemas. Como también la implementación de mecanismos de protección se debe considerar los requerimientos institucionales, el entorno de funcionamiento y lenguajes de programación, esto determinó utilizar las tecnologías de desarrollo de php y mysql por ser open source y tener funciones que permiten evitar las vulnerabilidades más comunes a nivel de software.

Sánchez (2017), en su estudio tuvo como objetivo analizar las vulnerabilidades existentes y diseñar procesos correctivos para la página web de la Dirección de Educación a Distancia y Virtual de la Universidad Técnica de Ambato. Llegando a la conclusión que el análisis de vulnerabilidades es un proceso necesario para toda entidad. La Universidad Técnica de Ambato debe poner énfasis en la misma para garantizar la seguridad informática de todos sus dominios y aplicaciones web; como también mediante el uso de herramientas de fácil acceso y de software libre fueron los resultados acerca de las vulnerabilidades tanto a la página web realizada en Joomla como a la aplicación web manejada por Moodle. Por último el uso de la metodología OWASP de acuerdo a sus

distintos pasos, se llevó a cabo mediante la obtención de información crucial de la entidad, consiguiendo así focalizar el estudio y realizar los distintos análisis a varios sectores del sitio web y de la aplicación que es manejados por la Dirección de Educación a Distancia y Virtuales.

Mejía (2018), en su estudio tuvo como objetivo de desarrollar servicios web REST "inseguros" en lenguaje de programación PHP para auto-aprendizaje en la explotación de vulnerabilidades y su mitigación. Llegando a la conclusión que si bien las herramientas de análisis de código y vulnerabilidades empleadas para este proyecto pueden ser usadas en la recolección de información acerca de problemas de seguridad en nuestros servicios web REST, no se debe confiar al totalmente en sus resultados puesto que al haber falsos positivos y falsos negativos, se deben realizar las respectivas verificaciones manuales. Como también no es suficiente con emplear una sola herramienta para el análisis de vulnerabilidades, dado que se pueden obviar algunas debilidades entre una y otra; al igual pueden presentarse reportes en los que no se muestren vulnerabilidades de día cero que no constarán en la base de datos de firmas de las herramientas; se debe considerar el uso de pasarelas XML como mecanismo de seguridad en profundidad complementario al aseguramiento de los servicios web REST a nivel de código y configuración del servidor. Las pasarelas estarán encaminadas a proteger la seguridad en el entorno de producción; de igual manera se debe tomar en cuenta firewalls de aplicación que analizarán los paquetes antes de atender cualquier petición. Luego del análisis de los servicios web REST con herramientas de análisis de código, se han detectado un total de 14 vulnerabilidades, habiendo sido mitigadas satisfactoriamente doce, las dos vulnerabilidades restantes fueron analizadas, concluyendo que se tratan de falsos positivos.

Montes (2016), en su investigación tuvo como objetivo realizar un análisis de seguridad de la Aplicación Web desarrollada en PHP como también de los Aplicativos móviles en IOS, Android y Windows Phone que se han creado en función del sistema para contrarrestar las vulnerabilidades y brindar un producto final confiable y seguro. Llegando a las siguientes conclusiones como son: análisis de la aplicación web para determinar las vulnerabilidades que existían y se procedió a corregirlas, esto permite tener un producto final robusto y eficaz que a la vez permite brindar mayor seguridad a la aplicación web que se encuentra implementada; tambien se analizó cada una de las Aplicaciones creadas para dispositivos móviles en IOS, Android y Windows Phone y se

detectó vulnerabilidades y posteriormente se procedió a mejorar la seguridad de la información, y funcionamiento de los mismos a nivel general. Y por último se observó la disminución de las amenazas sobre los posibles ataques por vulnerabilidades de los sistemas tanto de la aplicación web y aplicaciones móviles, con lo cual se obtuvo un producto sólido y confiable tanto para el propietario del producto como también para los usuarios que lo van a utilizar.

Aguilar y Dávila (2013), en su investigación tuvieron como objetivo diseñar, desarrollar e implementar una aplicación web que permita la gestión del distributivo de la Facultad de Ingeniería de la Universidad de Cuenca. Llegando a las conclusiones que por medio de esta aplicación web, es posible visualizar, almacenar e imprimir el distributivo, tanto general de cada Escuela, como particular de cada docente, además se pueden realizar reportes personalizados filtrando la información por Escuela, Año y Ciclo; como también durante el transcurso de desarrollo del presente proyecto, hemos tenido la oportunidad de conocer a profundidad las herramientas orientadas a la Web como GWT e Hibernate, las cuales han sido de vital importancia para llegar al producto final entregado. Y por último se concluye que las herramientas empleadas, han sido seleccionadas por ser herramientas libres, profesionales y de escasa documentación, lo que ha hecho que la investigación sea robusta.

Quirola (2019), en su tesis tuvo como objetivo analizar las vulnerabilidades informáticas para optimizar la seguridad de la información del sistema de gestión médica SISMEDICALEC, propiedad de la empresa Incomsis. Llegando a las conclusiones que el análisis de vulnerabilidades informáticas es una acción necesaria para toda empresa que desarrolle y comercialice sistemas informáticos. Para ello la empresa INCOMSIS, debe tener como prioridad el poder garantizar la aplicación de normas de seguridad en todos sus dominios y aplicativos web; la protección a las Cookies de sesión de un usuario es ineficaz lo que provoca un alto riesgo de secuestro de sesiones. Y actualmente no poseen un mecanismo de cierre automático al ingresar un usuario credenciales erróneas, lo que puede provocar riesgos y permitir ataques de fuerza bruta.

Bermúdez y Bailón (2015), en su investigación se plantearon como objetivo de analizar los procesos críticos de Credigestión respecto a las gestiones de seguridad adecuadas para garantizar la confidencialidad, integridad y disponibilidad de la información, mediante la formulación recomendaciones de seguridad y controles basados en la Norma ISO/IEC

27001. Llegando a la conclusión que el análisis realizado demuestra que los activos de información de las áreas, refleja potenciales índices de riesgos, los cuales exponen a la información a daños, robo o modificaciones que pueden causar un impacto negativo dentro de las actividades del negocio. Como también se pretende que la empresa tome acciones que le permitan prevenir y detectar oportunamente vulnerabilidades a las que están expuestos los sistemas de procesamiento de información, así como la información que es manejada y generada por los funcionarios. La implementación de controles de seguridad basados en la norma ISO/IEC 27001, les permite mejorar tres características importantes como son: la confidencialidad, integridad y disponibilidad de la información. Y por último la seguridad total no existe, pero gestionar controles de seguridad en el proceso y manejo de la información, se vuelve un complemento esencial, pues le permite asegurar información valiosa no sólo de la empresa sino también de los clientes.

González y Montesino (2018), en su artículo tuvieron como objetivo determinar hasta qué punto son válidos los procedimientos, herramientas y pruebas de seguridad propuestas en las metodologías ISSAF, OSSTMM, OWASP, PTES y NIST SP 800-115 para abordar los retos actuales de ciberseguridad en el campo del desarrollo y mantenimiento de las aplicaciones web. Teniendo como conclusiones que la Guía de Pruebas de OWASP resultó la más completa, seguida de la metodología de ISSAF. No obstante, ninguna metodología demostró ser capaz de brindar métodos, herramientas o pruebas de seguridad para detectar todas las vulnerabilidades comparadas. Los resultados alcanzados demuestran la necesidad de un proceso de adaptación y completamiento de las metodologías existentes ya que ninguna, por sí sola, contiene todos los elementos requeridos para realizar una evaluación de seguridad actual en aplicaciones web. Como también el incremento del uso de las aplicaciones web como base para la informatización de servicios en la sociedad, así como las continuas noticias de incidentes de seguridad que involucran este tipo de aplicaciones, hace necesario seguir profundizando en formas de evaluación basadas en pruebas de penetración y otras que permitan minimizar la ocurrencia e impacto de estos incidentes.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1. Identificación del problema

En este contexto cada vez más competitivo y complejo tecnológicamente, el éxito de cada empresa va depender de la administración de sus recursos, e incluso las de tecnología de información y comunicación (Tarouco y Graeml, 2011), puesto que a medida que las empresas iban evolucionando, la información iba incrementando en grandes volúmenes, los cuales debían ser soportados por activos tecnológicos como servidores, redes, software y hardware especializado (Espinoza, 2016). Es así que la auditoria tiende a tomar importancia para que los sistemas de información se desempeñen correctamente ya que estos otorgan los controles necesarios (Pinilla, 2012).

En tanto los ataques más peligrosos en la actualidad que atentan contra la seguridad de un sitio web son dos: La Inyección de código SQL y XSS, abreviaturas de Cross-site scripting. Inyección SQL es la infiltración de código intruso dentro del código SQL de las bases de datos generalmente usando las entradas de los formularios. El objetivo es alterar el funcionamiento del programa y lograr que se ejecute el código "invasor" incrustado.

Es una técnica poderosa con la que se puede manipular direcciones URL de cualquier forma, en entradas de búsquedas, formularios o registros de email, para inyectar el código. XSS es la inyección de código JavaScript en aplicaciones web, páginas web y hasta en el navegador web ya sea en código insertado (script, iframes) en HTML, aprovechando las variables que se pasan entre páginas usando una URL o robando las cookies. Ataca principalmente las aplicaciones que procesan datos obtenidos de una entrada sin ningún

tipo de chequeo o validación, estos esquemas son bastante difundidos por lo que mejoran cada año así como mejoran los métodos de protección.

2.2. Enunciado del problema

¿De qué manera podemos detectar las fallas de seguridad en servidores web y plantear la solución teórica planteada contra esquemas de escaneo de recursos web, rutinas anti XSS y los estándares de protección web?

2.3. Justificación

La seguridad de aplicaciones web es una rama de la Seguridad Informática que se encarga específicamente de la seguridad de sitios web, aplicaciones web y servicios de auditoria web, la seguridad de aplicaciones web se basa en los principios de la seguridad de aplicaciones, pero aplicadas específicamente a la World Wide Web. Las aplicaciones, comúnmente son desarrolladas usando lenguajes de programación tales como PHP, JavaScript, Python, Ruby, ASP.NET, JSP, entre otros.

La motivación principal de esta investigación fue la de detallar las principales formas de ataque y como proteger un sitio web, la que finalmente aplica sobre las aplicaciones web que estén alojadas en el hosting a probarse, también notar las falencias de los nuevos desarrolladores y dar a conocer un listado de buenas proactivas, así como herramientas

Este proyecto aportará con información útil y relevante sobre cada proceso, permitiendo mantener la comunicación estrecha, entre evaluador y evaluado, haciendo en primer lugar que la seguridad e integridad de los datos sea mantenida y crear mejores formas de desarrollo en aplicaciones web.

Finalmente, el sistema aportará información útil y relevante sobre cada proceso, permitiendo mantener la comunicación estrecha, entre evaluador y evaluado, haciendo en primer lugar que la distancia no sea un obstáculo que impida el avance progresivo.

2.4. Objetivos

2.4.1. Objetivo general

Describir las fallas de seguridad en servidores web y plantear la solución teórica planteada contra esquemas de escaneo de recursos web, rutinas anti XSS y los estándares de protección web.

2.4.2. Objetivos específicos

- Realizar una recopilación de los sistemas web a probar los fallos de seguridad, basándose en los requerimientos básicos de seguridad.
- Registrar y validar y garantizar la calidad de la información obtenida de los sitios observados.
- Evaluar los resultados y plantear esquemas de seguridad e integridad de datos para un futuro mejorado en la seguridad.

2.5. Hipótesis

2.5.1. Hipótesis general

Los esquemas de escaneo de recursos web, rutinas anti XSS y los estándares de protección web. Permiten identificar las fallas de seguridad en servidores web.

2.5.2. Hipótesis específicas

- los esquemas de escaneo basándose en los requerimientos básicos de seguridad permiten evaluar las fallas los sistemas web
- la información obtenida de los sitios observados permiten evaluar para garantizar la calidad de la información
- la implementación de la información obtenida de los sitios observados permiten evaluar para garantizar la calidad de la información para un futuro mejorado en la seguridad.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. Lugar de estudio

La presente investigación se realizó en la ciudad de Puno en la Universidad Nacional del Altiplano, ubicado en la ciudad de Puno, departamento de Puno, provincia Puno y distrito Puno. En la siguiente figura se muestra la ubicación del lugar de estudio:

Figura 5: Ubicación de la Universidad Nacional del Altiplano
Fuente: Google Maps

3.2. Población

La población para efectos de ejecución del presente trabajo de investigación se eligió 04 sitios web a criterio del investigador para poder llevar a cabo la presente investigación, las pruebas de penetración mediante dos aplicaciones web, DirBuster y SQLMAP, efectuadas a los sitios web elegidos, estos permitieron observar con objetividad el proceso de administración, monitoreo de los posibles ataques de crackers o nuevos programadores que desean obtener información de nuestros servidores de datos sin estar autorizados

3.3. Muestra

Dado que el presente estudio utiliza pruebas de penetración la muestra será la misma población ya que solo se trabajó con la totalidad de la población, es decir los 04 sitios web elegidos a criterio del investigador.

3.4. Método de investigación

La naturaleza del presente estudio implica la aplicación del conocimiento en la solución de problemas prácticos, se ubica dentro de la investigación aplicada, debido a que no habrá manipulación de variables dependientes y dado que el origen de los datos proviene de pruebas/técnicas computacionales, el presente estudio es: no experimental-documental (Arias, 2006).

3.5. Descripción detallada de métodos por objetivos específicos

La metodología optada es de prueba de penetración mediante dos aplicaciones web, DirBuster y SQLMAP.

3.5.1. Para el objetivo específico 1

Realizar una recopilación de los sistemas web a probar los fallos de seguridad, basándose en los requerimientos básicos de seguridad.

Pruebas de penetración

Las pruebas de penetración no son simplemente la ejecución en serie de herramientas automatizadas y la generación de informes técnicos, como se ve con frecuencia. Debe proporcionar una dirección clara y concisa sobre cómo proteger

la información y los sistemas de información de una organización de los ataques del mundo real.

Un factor crítico en el éxito de las pruebas de penetración es su metodología subyacente. Se debe utilizar un enfoque sistemático y científico para documentar con éxito una prueba y crear informes dirigidos a diferentes niveles de gestión dentro de una organización. No debería ser restrictivo permitir que el probador explore completamente sus intuiciones (Bacudio y Yuan, 2011).

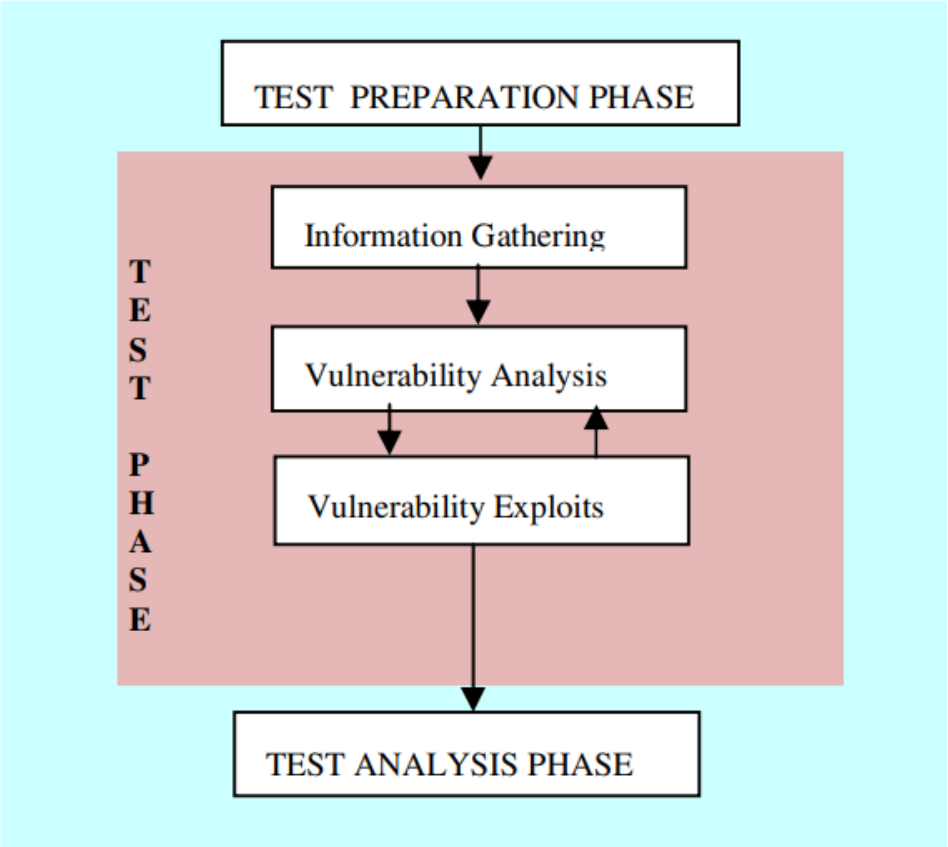


Figura 6. Metodología de prueba de penetración
Fuente: Adaptado de Bacudio y Yuan (2011).

La mayor parte del proceso de prueba de penetración se realiza durante la fase de prueba. Se puede utilizar una variedad de herramientas automatizadas en esta fase. La figura 6, enumera algunas de estas fases que implica los siguientes pasos: recopilación de información, análisis de vulnerabilidad y exploits de vulnerabilidad. Todos los documentos necesarios para el examen se organizan y finalizan durante la fase de preparación del examen.

3.5.2. Para el objetivo específico 2

Registrar y validar y garantizar la calidad de la información obtenida de los sitios observados.

La prueba de penetración es el sondeo sistemático de un sistema que podría ser cualquier combinación de aplicaciones, hosts o redes. Se describe la metodología general de las pruebas de penetración.

Esta sección es un proceso útil para descubrir y resolver las debilidades de seguridad de las aplicaciones, especialmente las aplicaciones web.

Esta sección ilustra el proceso de prueba de penetración utilizando dos aplicaciones web, DirBuster y SQLMAP. Cabe señalar que las áreas probadas aquí consisten en solo un pequeño subconjunto de las áreas que deben probarse. Por lo tanto, el proceso de prueba de penetración puede ser más completo y puede llevar más tiempo y esfuerzo cuando se realiza en aplicaciones web más complicadas. Se supone que la fase de preparación de la prueba se ha completado antes de la fase de prueba.

Hay tres pasos involucrados en la fase de prueba: recopilación de información, análisis de vulnerabilidad y explotación de vulnerabilidad. Estas fases se denominan fase de descubrimiento, fase de vulnerabilidad y fase de simulación de ataque. La fase de prueba es seguida por la fase de análisis de prueba.

3.5.3. Para el objetivo específico 3

Evaluar los resultados y plantear esquemas de seguridad e integridad de datos para un futuro mejorado en la seguridad.

Después del paso de análisis de vulnerabilidad, las evaluaciones obtenidas deben tener una buena idea de las áreas que serán objeto de ataques. Con la lista de vulnerabilidades a la mano de las dos aplicaciones que fueron explotadas.

La explotación de vulnerabilidades en la aplicación se descubrió que la aplicación tiene vulnerabilidades en sus mecanismos de autenticación y es vulnerable a la inyección SQL y XSS.

La ausencia de mecanismos de validación rigurosos permitía a cualquier usuario ingresar al sistema. Una vulnerabilidad a esta vulnerabilidad es inundar el sitio con contenido que comprometa la seguridad del sistema, como agregar blogs que dañen la reputación de otros usuarios, enlaces al sitio del estafador o cualquier sitio potencialmente peligroso.

Sabiendo que la aplicación es vulnerable a la inyección SQL respaldada con la información recopilada de los mensajes de error, se exploraron más a fondo las diferentes funciones de la aplicación con el objetivo de obtener más ideas para explotar el sistema.

CAPÍTULO IV

RESULTADOS Y DISCUSÍON

Una vez realizado la implementación del software se presentan los siguientes resultados de acuerdo a los objetivos planteados y se presentan de la siguiente manera:

4.1. Contexto en que se desarrolla

Hasta hace pocos años no era posible plantearse la necesidad de auditorías sobre las aplicaciones informáticas con el fin de evitar ataques externos por red. La razón es simple: porque hasta los años 80 no se desarrolló internet tal como lo conocemos hoy día. En aquellos años, el uso de ordenadores personales apenas estaba extendido y los pocos que había, (fuera de ciertos ámbitos como el de defensa, o el universitario), no estaban conectados a ninguna red. Sus funciones y capacidades eran muy limitadas.

La única forma de acceder ilegítimamente a un ordenador personal era, o bien ganar acceso físico a la habitación donde se encontrara la máquina, o bien, que su dueño insertara en él un dispositivo externo como un disquete, previamente infectado. Así pues, el papel de la seguridad de las aplicaciones no era tan importante como lo es hoy.

A partir de los años 80 se crea el estándar Ethernet, y se formalizan protocolos como

TCP/IP, o DNS. Hasta los 90 no aparece el WWW. Es en ésta década cuando se abre la red al comercio y se dispara el número de ordenadores conectados.

Con el avance tan veloz de las tecnologías TIC, en pocos años se pasa de conexiones no permanentes, a través de llamadas por líneas telefónicas, a conexiones por cable o fibra óptica permanentes, que no inhabilitan la línea de teléfono. Las tarifas se reducen para aumentar la demanda, y se pasa de pagar por tiempo de conexión, o cantidad de megas

descargados, a una tarifa plana mensual, que hace internet más asequible para todo el mundo. Las páginas web evolucionan rápidamente y pasan de ser sitios estáticos, que ofrecen texto informativo y poco más, a sitios con contenido dinámico que ejecutan aplicaciones y devuelven sus resultados.

4.2. Realizar una recopilación de los sistemas web a probar los fallos de seguridad, basándose en los requerimientos básicos de seguridad.

En este proceso de recopilación de sistemas web para evaluar la información e identificar el nivel de vulnerabilidades, se tomó en cuenta las siguientes páginas web:

Pronabec : https://www.pronabec.gob.pe/

Sunat : http://www.sunat.gob.pe/

Quesito.pe : https://www.quesito.pe/

Los Software y herramientas de auditoria a utilizar fueron los siguientes:

- **Lighthouse:** Es una herramienta automatizada de código abierto diseñada para mejorar la calidad de tus apps web. Puedes ejecutarla como una extensión de Chrome o desde la línea de comandos. Le proporcionas a Lighthouse una URL que quieres auditar, Lighthouse ejecuta una serie de pruebas contra la página, y luego genera un informe sobre el rendimiento de la página. A partir de aquí, puedes usar las pruebas desaprobadas como indicadores de lo que puedes hacer para mejorar tu app.

- **DirBuster**: Es una aplicación Java multi hilo diseñada para obtener por fuerza bruta los nombres de directorios y archivos en servidores Web/de aplicación. A menudo ocurre que lo que ahora parece un servidor Web en una fase de instalación por omisión no lo es, y tiene páginas y aplicaciones ocultas. DirBuster trata de encontrar estos.

Sin embargo, las herramientas de esta naturaleza a menudo son solo tan buenas como la lista de archivos y directorios con los que vienen. Un enfoque diferente fue usado para generar esto. La lista fue generada desde cero, rastreando en Internet y colectando los directorios y archivos que son realmente usados por los desarrolladores DirBuster viene con un total de 0 listas diferentes (Más información puede ser encontrada más adelante), esto hace a DirBuster extremadamente efectivo encontrando esos archivos y directorios

ocultos. Y si eso no fuera suficiente, DirBuster también tiene la opción de realizar fuerza bruta pura, lo que no les deja lugar para esconderse a los archivos y directorios ocultos.

Método de empleo

El software de evaluación de vulnerabilidades denominado “Lighthousese”, se instala como un plugin/extensión sobre el navegador Chrome, descargable en la siguiente dirección.

https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=es

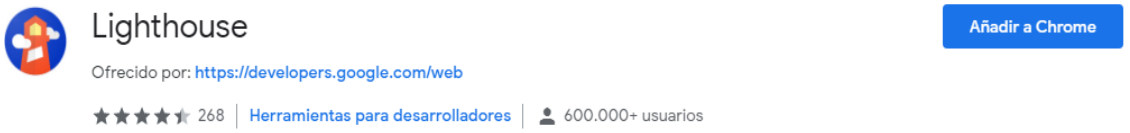


Figura 7. Extensión Lighthousese
Fuente: Panel de complemento o extensiones de Google Chrome.

Se agrega como complemento sobre el navegador se carga el sitio web para su verificación y auditoria de vulnerabilidades, el software complemento inicia la evaluación de vulnerabilidades, como se muestra en la siguiente figura.

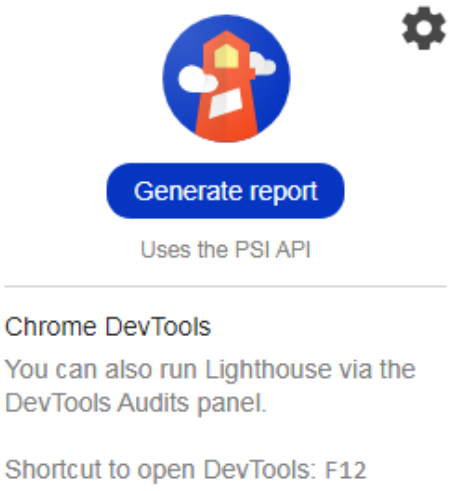


Figura 8. Ventana Lighthousese para evaluar vulnerabilidades.
Fuente: Panel de complemento o extensiones de Google Chrome.

4.3. Registrar, validar y garantizar la calidad de la información obtenida de los sitios observados

El Proyecto Abierto de Seguridad de Aplicaciones Web (OWASP), es una comunidad de nivel mundial abierto y libre, enfocado en mejorar la seguridad en las aplicaciones de software. Nuestra misión es hacer la seguridad en aplicaciones "visible", de manera que las organizaciones pueden hacer decisiones informadas sobre los riesgos en la seguridad de aplicaciones (extraído de: http://owasp.org).

En tanto, Lighthouse es una herramienta automatizada de código abierto diseñada para mejorar la calidad de tus apps web. Puedes ejecutarla como una extensión de Chrome o desde la línea de comandos. Le proporcionas a Lighthouse una URL que quieres auditar, Lighthouse ejecuta una serie de pruebas contra la página, y luego genera un informe sobre el rendimiento de la página. A partir de aquí, puedes usar las pruebas desaprobadas como indicadores de lo que puedes hacer para mejorar tu app.

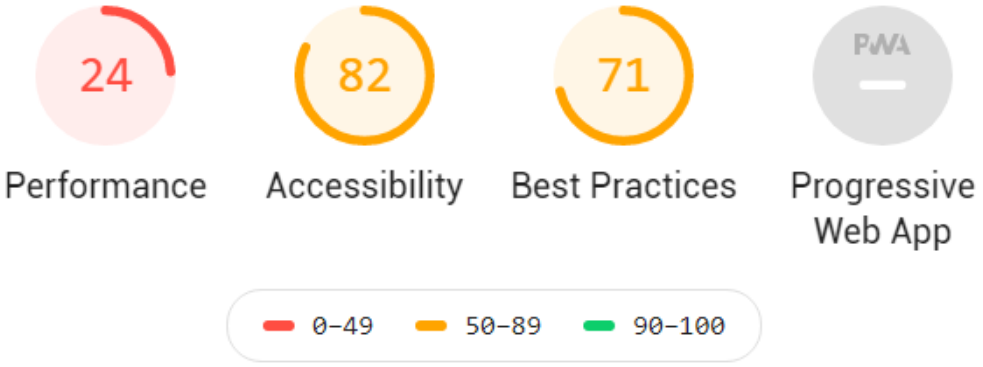
Nota: Lighthouse actualmente tiene un gran enfoque sobre las funciones de las Progressive Web Apps, como Add to homescreen y soporte sin conexión. Sin embargo, el objetivo general del proyecto es ofrecer una auditoría de extremo a extremo de todos los aspectos de la calidad de la app web. (extraído de: https://developers.google.com/web/tools/lighthouse/?hl=es)

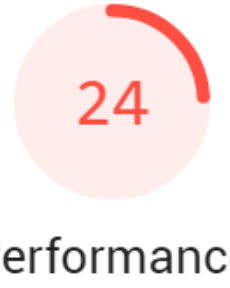
4.4. Evaluar los resultados y plantear esquemas de seguridad e integridad de datos para un futuro mejorado en la seguridad

En este proceso se ejecutó la herramienta “Lighthousese” para evaluar vulnerabilidades en la página web 1: “pronabec.gob.pe”

Figura 9. Página web para evaluación de vulnerabilidades “Pronabec”
Fuente: www.pronabec.gob.pe

Una vez aplicado la herramienta de evaluación de vulnerabilidades a la primera página web en evaluación de vulnerabilidades del ***Pronabec***, se obtuvo los siguientes resultados





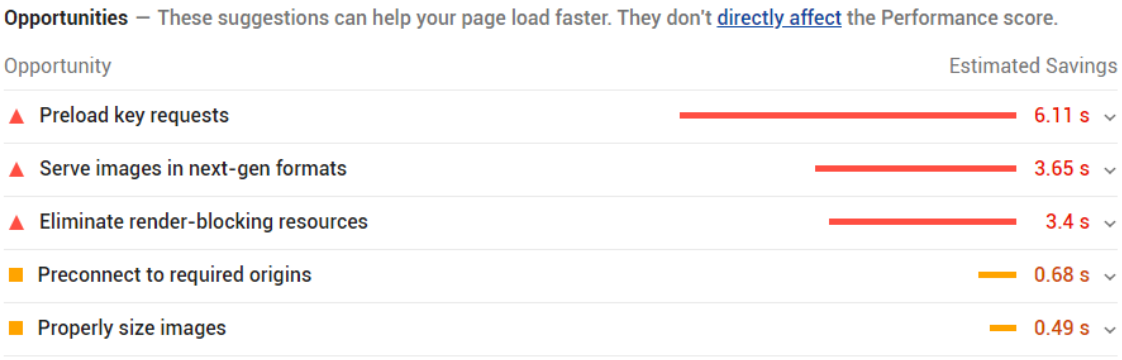


Figura 10: Resultados de evaluación del sitio web: pronabec.gob.pe

Observando los resultados, se verifica tiene un 24% de vulnerabilidad en la performancia, encontrándose entre los rangos de 0 a 49, es decir que la página web tiene deficiencias en cuanto al desempeño, en tanto dicha página web puede estar sujeta a ataques de hackers y por ende a la perdida de información de dicha institución.

Así mismo se hizo la evaluación de vulnerabilidades a la página web de la Superintendencia nacional de aduanas y de administración tributaria ***(SUNAT),*** institución que genera y registra abundante información concerniente a los tributos.

Figura 11. Página web para evaluación de vulnerabilidades "***SUNAT***"
Fuente: www.sunat.gob.pe

Una vez aplicado la herramienta de evaluación de vulnerabilidades a la primera página web en evaluación de vulnerabilidades de la ***SUNAT***, se obtuvo los siguientes resultados

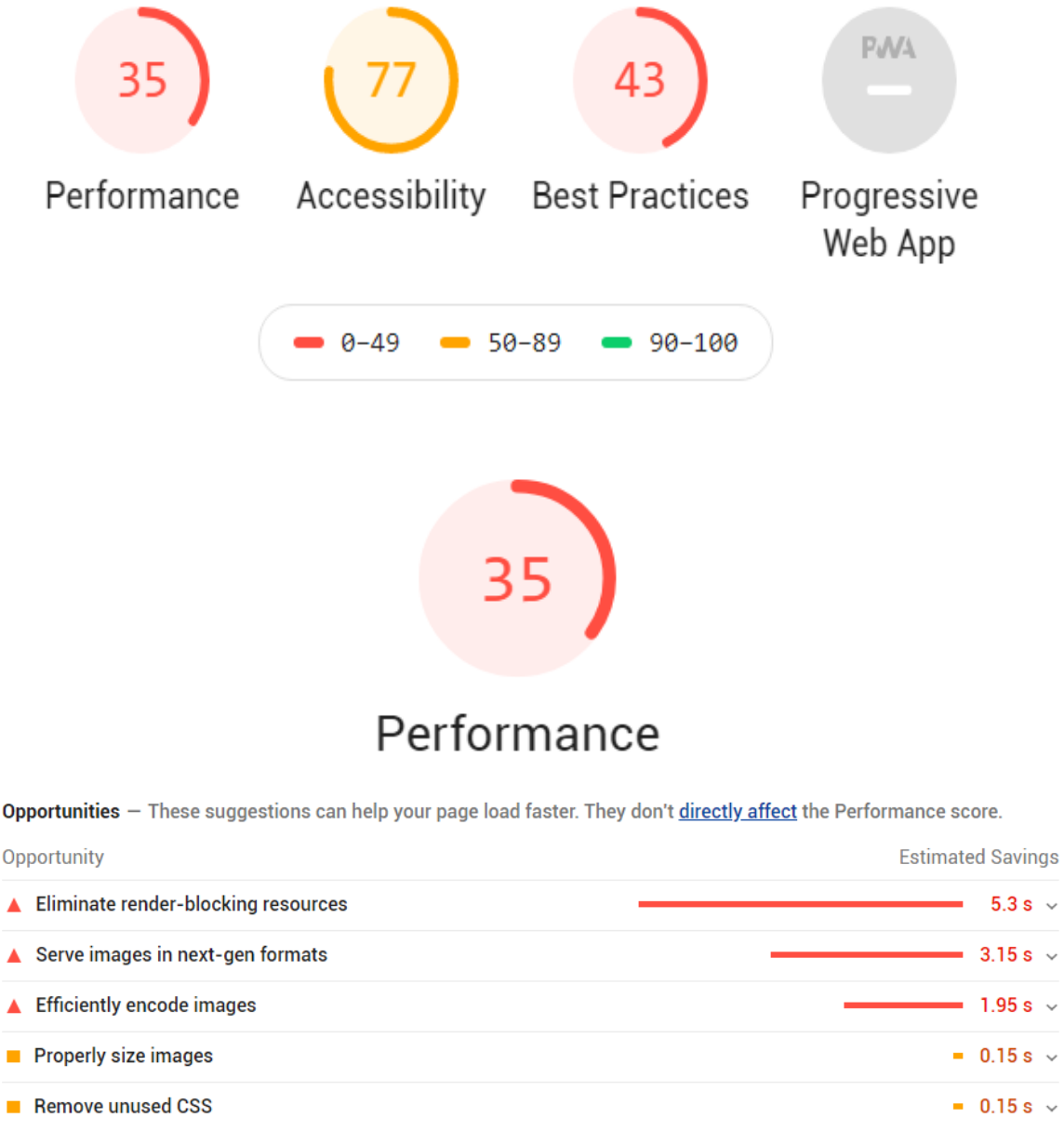


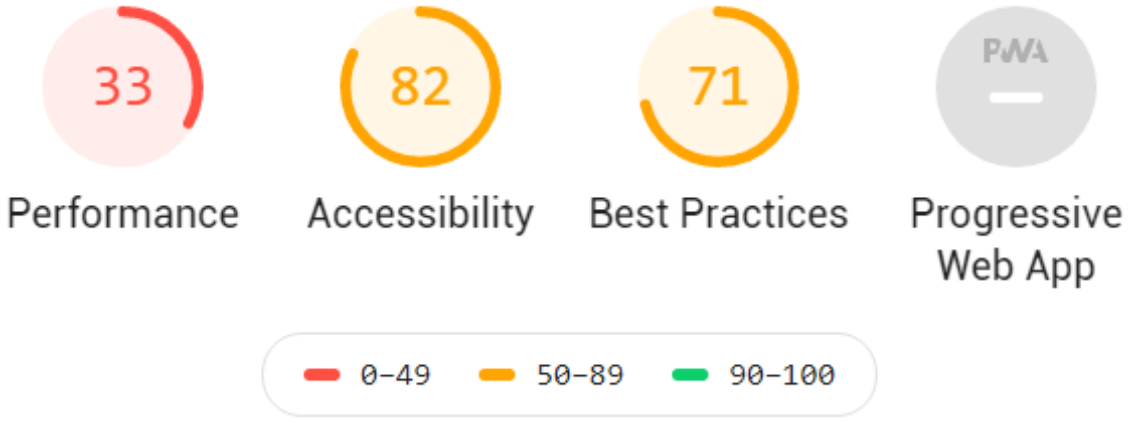
Figura 12: Resultados de evaluación del sitio web: sunat.gob.pe

Observando los resultados, se verifica tiene un 35% de vulnerabilidad en la performancia, encontrándose entre los rangos de 0 a 49, es decir que la página web tiene deficiencias en cuanto al desempeño, en tanto dicha página web puede estar sujeta a ataques de hackers y por ende a la perdida de información, puesto que es una página que está disponible y es accesible en cualquier momento del día o de la noche para los usuarios públicos.

Por último, se hizo la evaluación de vulnerabilidades a la página web ***quesito.pe***, esta es una página no perteneciente al estado peruano, puesto que esta página web pertenece a una empresa privada de distribución de alimentos, en tanto fue desarrollada con recursos propios de la empresa, así mismo como las anteriores paginas esta genera y registra información importante concerniente al servicio que brindan.

Figura 13. Página web para evaluación de vulnerabilidades "quesito.pe"
Fuente: www.quesito.pe

Una vez aplicado la herramienta de evaluación de vulnerabilidades a la primera página web en evaluación de vulnerabilidades de ***quesito.pe***, se obtuvo los siguientes resultados



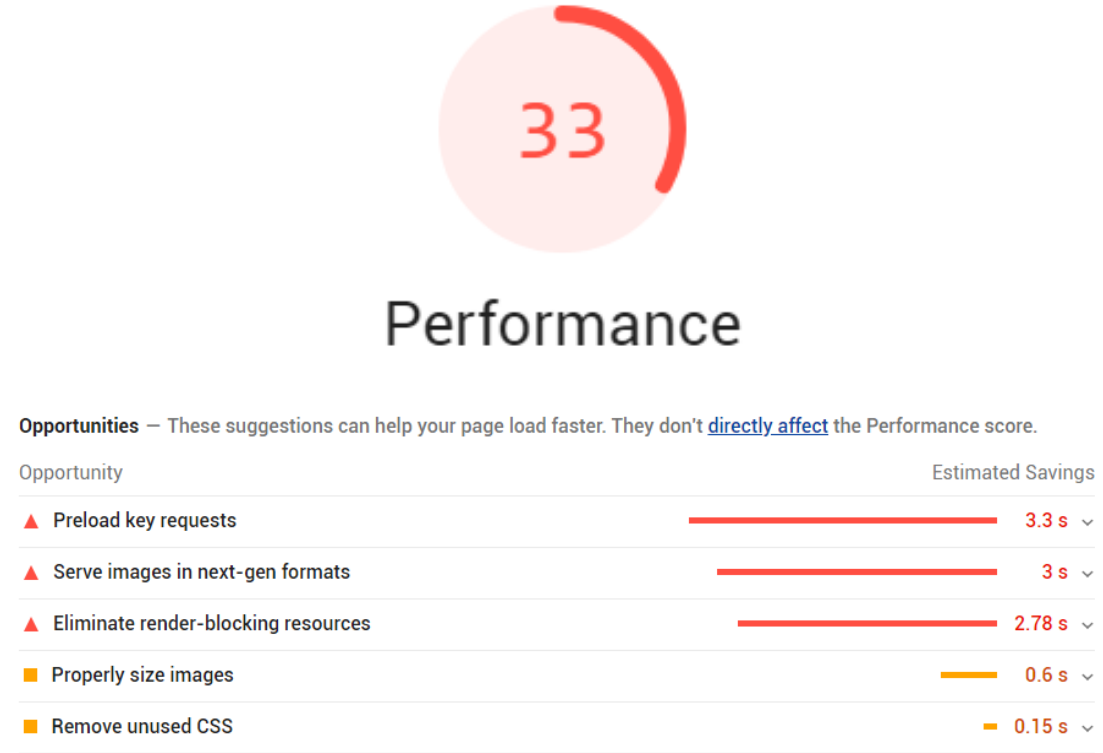


Figura 14: Evaluación del sitio web: quesito.pe

Observando los resultados, en dicha página se verifica tiene un 33% de vulnerabilidad en la performance, encontrándose entre los rangos de 0 a 49, es decir que la página web tiene deficiencias en cuanto al desempeño, en tanto puede estar sujeta a ataques de hackers y en consecuencia a la perdida de información.

CONCLUSIONES

- Las herramientas de auditoria permitieron determinar las vulnerabilidades de las páginas seleccionadas y mostrar que los riesgos reales, que incluso alguien con poca experiencia, pero mucho interés y tiempo, puede llegar a hacer estragos en una aplicación mal construida.

- Mostrar estos riesgos, y lo que se puede llegar a obtener de la explotación de un fallo, ayudará a concienciar al lector al respecto de tomarse en serio el desarrollo de software seguro, así como el uso de funciones seguras. Se concluye que las herramientas Lighthouse y DirBuster permiten validar y auditar directorios y archivos en servidores Web/de aplicación.

- La calidad de datos JSON para el reporte permitirá poder formatear y desplegar sobre cualquier lenguaje de programación a diferencia de reportes antiguos basados en XML (lenguaje de marcas extendido) que, aunque escalable no se constituyó como estándar de intercambio de datos.

- Se concluye con detallar los mejores esquemas de seguridad para proteger la información vertida en los sitios web, así como en las aplicaciones web que dependan del alojamiento sobre el que se hace la auditoria.

RECOMENDACIONES

- Se recomienda agregar el uso de software auditores más especializados durante la tarea de verificación del contenido web así mismo agregar la verificación por sesiones almacenadas y validadas como esquema de protección extra.

- Evitar el uso de direcciones físicas en lo posible son los primeros lugares que se buscan para ataques por la posibilidad de que estén con permisos root ósea en modo rwx-rwx-rwx 777, lo que podría alojar cualquier script que pueda replicarse o desproteger el sitio completo.

BIBLIOGRAFÍA

Adelgani, G. (1997). *Seguridad Informática MP*. Argentina: Ediciones Argentina.

Aghaei S., Nematbakhsh M.A. y Khosravi H. (2012). Evolution of de World Wide Web: from Web to Web 4.0. *International Journal of Web & Semantic Technology (IJWesT)* 3(1), 1-10. Recuperado de: http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf

Aguilar, E y Dávila, D. (2013). *Análisis, diseño e implementación de la aplicación web para el manejo del distributivo de la Facultad de Ingeniería* (tesis de grado). Universidad de Cuenca, Ecuador. Recuperado de: https://dspace.ucuenca.edu.ec/bitstream/123456789/4303/1/tesis.pdf

Aguilera, P. (2011). *Redes seguras (Seguridad informática)*. Madrid, España: Editex.

Alarcón, J. (2016). *Qué son las aplicaciones web progresivas.* Recuperado de: https://www.campusmvp.es/recursos/post/Que-son-las-Aplicaciones-Web-Progresivas-o-Progressive-Web-Apps.aspx

Almanza, A. y Cano, J. (2010). Seguridad de la información. *NASCO Nacional de computadores S.A*, 1(186), 4-72. Recuperado de: https://acis.org.co/archivos/Revista/115/Revista%20Sistemas%20Edici%C3%B3n%20115.pdf

Areitio, G. y Areitio, A. (2008). *Un modelo de entorno de aprendizaje personal que integra la web 2.0*. México: McGraw-Hill, Pearson Ed. y U. de Vigo.

Aristides, A. (2019). *Evaluación de seguridad de aplicaciones web* (tesis de grado). Universidad Nacional de San Luis, Argentina.

Bacudio, A. y Yuan, X. (2011). *An Overview of Penetration Testing.* Journal of Network Security & Its Applications (IJNSA), 3(6).

Bermúdez, K. y Bailón, E. (2015). *Análisis En Seguridad Informática Y Seguridad De La Información Basado En La Norma ISO/IEC 27001- Sistemas De Gestión De Seguridad De La Información Dirigido a Una Empresa De Servicios Financieros* (tesis de grado). Universidad Politécnica Salesiana, Guayaquil. Recuperado de: https://dspace.ups.edu.ec/bitstream/123456789/10372/1/UPS-GT001514.pdf

Briones, G. y Hernandez, E. (2018). *Auditoria de seguridad del servidor web de la empresa PUBLYNEXT S.A. utilizando mecanismos basados en OWASP* (tesis de grado). Universidad de Guayaquil, Ecuador. Recuperado de: http://repositorio.ug.edu.ec/handle/redug/26837

Codina, Ll. (2006). Evaluación de calidad en sitios web: Metodología de proyectos de análisis sectoriales y de realización de auditorías. *Revista Española de Documentación Científica,* 23(1), 30-32. Recuperado de: http://eprints.rclis.org/8854/1/procedimientos2006.pdf

Daza, G. (2015). *Fortalecimiento de seguridad del centro de datos de la carrera de ingeniería en sistemas computacionales, análisis de políticas de seguridad informática. Propuesta de protección ENDPOINT y control de acceso a la red* (tesis de grado). Universidad de Guayaquil, Ecuador. Recuperado de: http://repositorio.ug.edu.ec/handle/redug/10281

Donate, F. P. (2005). *Transmisión de imágenes de video mediante Servicios Web XML sobre J2ME.* Universidad de Sevilla. Recuperado de: http://bibing.us.es/proyectos/abreproy/11372/fichero/

Espinoza, W. (2016). La tecnología de la información como herramienta construccionista para el auditor financiero hibrido. *Fides Et Ratio*, 11(11), 17-35. Recuperado de: http://www.scielo.org.bo/pdf/rfer/v11n11/v11n11_a02.pdf

Franco, D. y Guerrero, C. (2013), Sistema de Administración de Controles de Seguridad Informática basado en ISO/IEC 27002. *"Innovation in Engineering, Technology and Education for Competitiveness and Prosperity",* 1 (2), 14-16. Recuperado de: http://www.laccei.org/LACCEI2013-Cancun/RefereedPapers/RP239.pdf

Fuchs, C. y otros, (2010). Theoretical Foundations of the Web: Cognition, Communication, and Co-Operation. *Towards an Understanding of Web 1.0, 2.0, 3.0.* 1(2), 41-59.

Gauchat, J. (2012). *El gran libro de HTML5, CSS3 y Javascript*

GCFAprendeLibre. (2016). *Aplicaciones móviles.* Recuperado de: https://www.gcfaprendelibre.org/tecnologia/curso/informatica_basica/todo_acerca_de_las_aplicaciones_o_programas/2.do

González, H. R., y Montesino, R. (2018). Capacidades de las metodologías de pruebas de penetración para detectar vulnerabilidades frecuentes en aplicaciones web. *Revista Cubana de Ciencias Informáticas*, 12(4), 52–65. Recuperado de: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992018000400005&lng=pt&nrm=iso

Hendler, J. (2009). Web 3.0 Emerging. *Computer*, 42(1).

Hermoso, A. (2013). *Seguridad en aplicativos Web* (tesis de grado). Universidad de Barcelona, España. Recuperado de: http://diposit.ub.edu/dspace/bitstream/2445/49106/1/AdrianHermoso_memoria.pdf

Hernández, A. y Mejia, J. (2015). Guía de ataques, vulnerabilidades, técnicas y herramientas para aplicaciones Web. *ReCIBE. Revista Electrónica de Computación, Informática, Biomédica y Electrónica*, 4(1), V. Recuperado de: http://www.revistascientificas.udg.mx/index.php/REC/article/view/5208

ISO. (2014). *ISO 27000 Directory*. Recuperado de: http://www.27000.org/

Khan, M. (2018). EncCD: A Framework for Efficient Detection of Code Clones. *The International Arab Journal of Information Technology*, 16(5). Recuperado de: http://iajit.org/PDF/September%202019,%20No.%205/16127.pdf

Lopez, A. (2010) *Seguridad Informática*. Ecuador: Editex S.A.

López, M., Marulanda, C. y Vega, O. (2012). *Servicios de Gestión de Conocimiento Utilizando La Computación en Nube.* Cataluña: Eureca Media S.L.

Mateu, C. (2004). *Desarrollo de Aplicaciones Web. F. p. a. l. U. O. d.* Catalunya: Ed.Eureca Media, SL.

Mejía, C. (2018). *Desarrollo de servicios web REST "inseguros" para auto-aprendizaje en la explotación de vulnerabilidades* (tesis de maestría), Universidad Internacional de La Rioja, Quito, Ecuador. Recuperado de: https://reunir.unir.net/handle/123456789/7435

Montes, J. (2016). *Diseño e implementación de un portal web para el consejo nacional electoral (CNE) a fin de ayudar en la capacitación a los ciudadanos del ecuador*

acerca del código de la democracia, enfocado en la gestión de proyectos ágiles aplicando metodología SCRUM en la ingeniería de software. Enfocado al análisis de seguridad de la aplicación web y análisis de seguridad de las aplicaciones de los dispositivos móviles (IOS, ANDROID Y WINDOWS PHONE) (tesis de grado). Universidad de Guayaquil, Ecuador. Recuperado de: http://repositorio.ug.edu.ec/handle/redug/15665

Muñoz Carlos, (2002). *Auditoria de Sistemas Computacionales.* Prentice Hall. México.

OWASP (2008). *DirBuster & Beyond.* Recuperado el 2019 desde https://www.owasp.org/images/f/f4/DirBuster_OWASP-London_September-2008.pdf

OWASP. (2013). *OWASP Top Ten 2013.* Proyect. Recuperado de: https://www.owasp.org/index.php/OWASP_Top_Ten_Project

Pablos, de, C.; López, J.; Martín, S.; Medina, S.; Montero, A. y Najera, J. (2006). *Dirección y gestión de los sistemas de información en la empresa: una visión integradora, 2da edición*. Madrid: ESIC.

Pautasso, C.; Wilde, E. y Alarcon, R. (2014). REST: *Advanced Research Topics and Practical Applications.*

Pinilla (2012). *Auditoria informática*. Universidad Autónoma del Estado de Hidalgo. Recuperado de: https://www.uaeh.edu.mx/docencia/P_Presentaciones/tlahuelilpan/sistemas/auditoria_informatica/auditoria_informatica.pdf

Quirola, L. (2019). *Análisis de Vulnerabilidades de Seguridad Informática, del Sistema de Gestión Médica SISMEDICALEC, de la empresa Incomsis* (tesis de grado). Universidad Técnica De Ambato, Ecuador. Recuperado de: https://repositorio.uta.edu.ec/handle/123456789/30108

Ramírez, G. y Álvarez, E. (2003). Auditoría a la Gestión de las Tecnologías y Sistemas de Información. *Industrial Data.* 6(1), 99-102. Recuperado de: https://www.redalyc.org/pdf/816/81606114.pdf

Romaniz, S. (2008). Seguridad de aplicaciones web: vulnerabilidades en los controles de acceso. *Grupo de Investigación en Seguridad de las Tecnologías de Información y Comunicaciones.* Recuperado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/21581/1927+-+Seguridad+de+aplicaciones+web+vulnerabilidades+en+los+controles+de+acceso.pdf?sequence=1

Royer, J. (2004). *Seguridad en la informática de empresa: riesgos, amenazas, prevención y solución*. Barcelona: Eni.

Salgado, A. (2014). *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* (tesis de grado). Universidad de las Fuerzas Armadas, Ecuador. Recuperado de: http://repositorio.espe.edu.ec/xmlui/handle/21000/8245

Sánchez, J. (2017). *Análisis de vulnerabilidades y diseño de procesos correctivos de la página web de la Dirección de Educación a Distancia y Virtual de la Universidad Técnica de Ambato* (tesis de grado). Universidad Técnica de Ambato, Ecuador. Recuperado de: https://repositorio.uta.edu.ec/jspui/handle/123456789/25531

Sommerville, I. (2005).*Ingeniería del software*. Pearson educación.

Tarouco, H. y Graeml, A. (2011). Governança de tecnologia da informação: um panorama da adoção de modelos de melhores práticas por empresas brasileiras usuárias. *Revista de Administração*. 46(1), 7-18. https://doi.org/10.5700/rausp0994

Tituaña, M. (2017). *Análisis de la seguridad de aplicaciones web críticas del Municipio del distrito metropolitano de Quito* (tesis de grado). Escuela Politécnica Nacional, Quito, Ecuador. Recuperado de: https://bibdigital.epn.edu.ec/handle/15000/17543

Tovar, O. (2015). *Inyección de SQL, tipos de ataques y prevención en ASP.NET – C#.* Universidad Piloto de Colombia. Bogota, Colombia. Recuperado de: http://polux.unipiloto.edu.co:8080/00002026.pdf

UNAM-CERT. (2009). *Aspectos Básicos de la Seguridad en Aplicaciones Web.*

Upton, D. (2015).*CodeIgniter for Rapid PHP Application Development*. Packt Publishing Ltd.

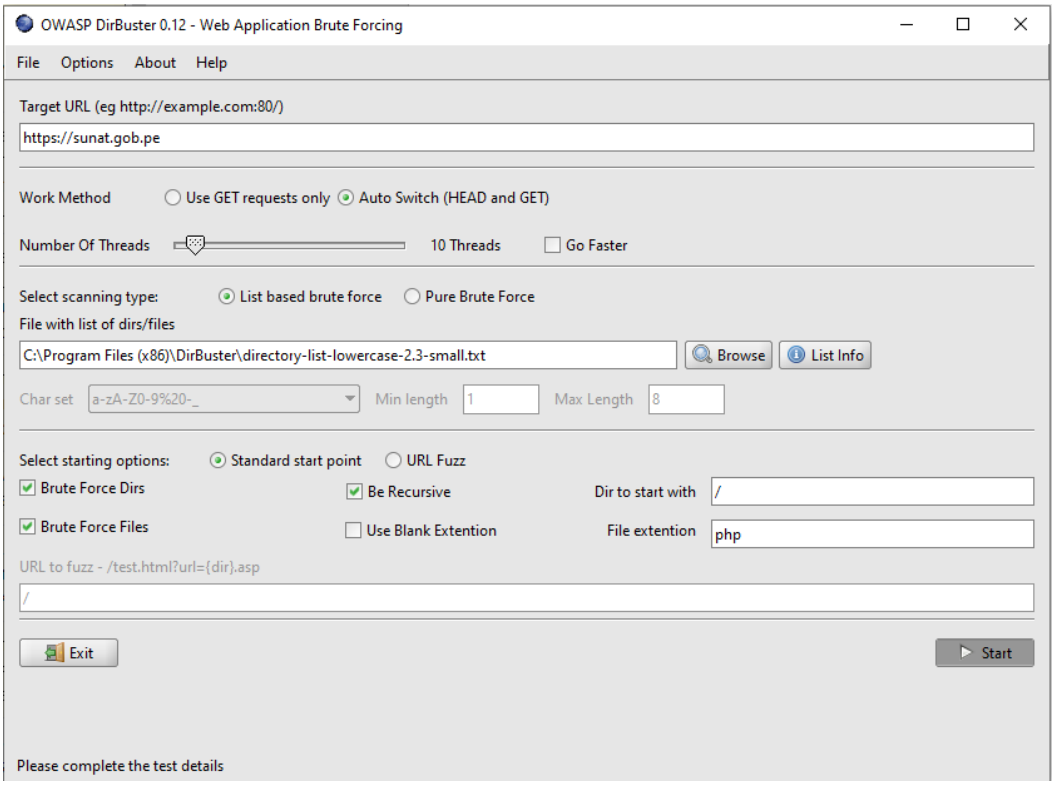
Valle, J. y Gavidia, M. (2015). *Análisis de vulnerabilidades de software para mejorar la seguridad en los sistemas informáticos* (tesis de grado). Universidad Nacional de Chimborazo, Ecuador. Recuperado de: http://dspace.unach.edu.ec/handle/51000/731

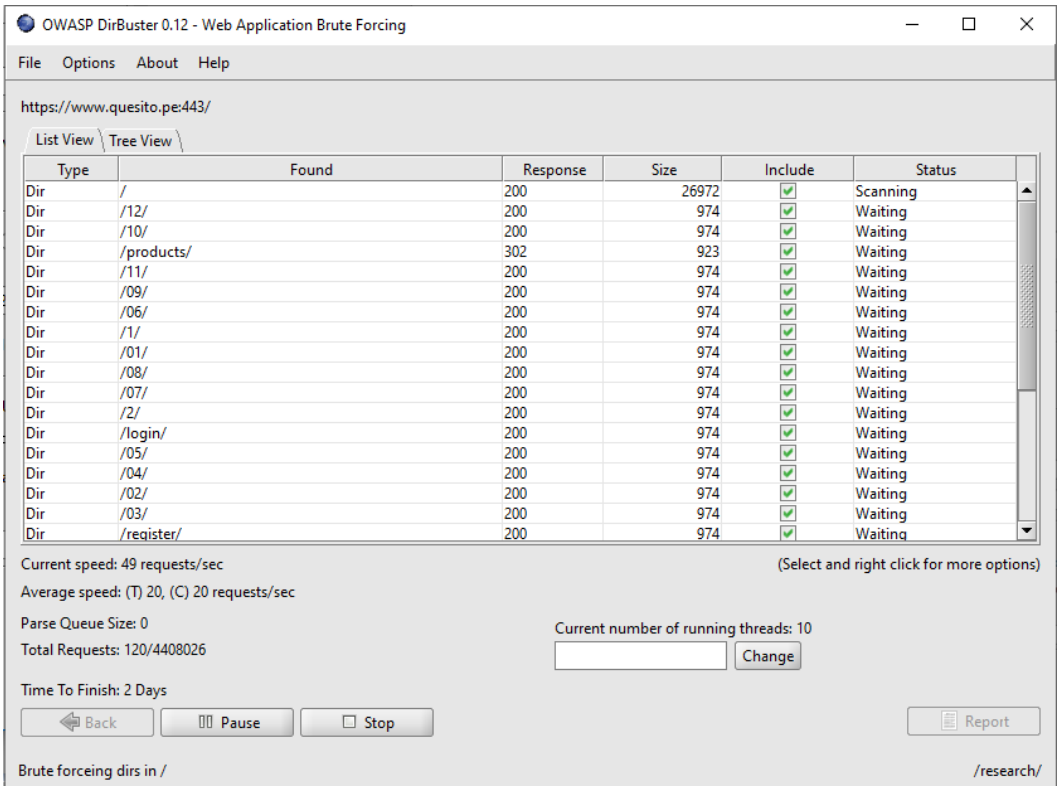
Wiboo media. (2017) *¿Qué son las aplicaciones web? Ventajas y tipos de desarrollo web*. Madrid, España. Recuperado de: https://wiboomedia.com/que-son-las-aplicaciones-web-ventajas-y-tiposde-desarrollo-web/

ANEXOS

ANEXO 1: DirBuster

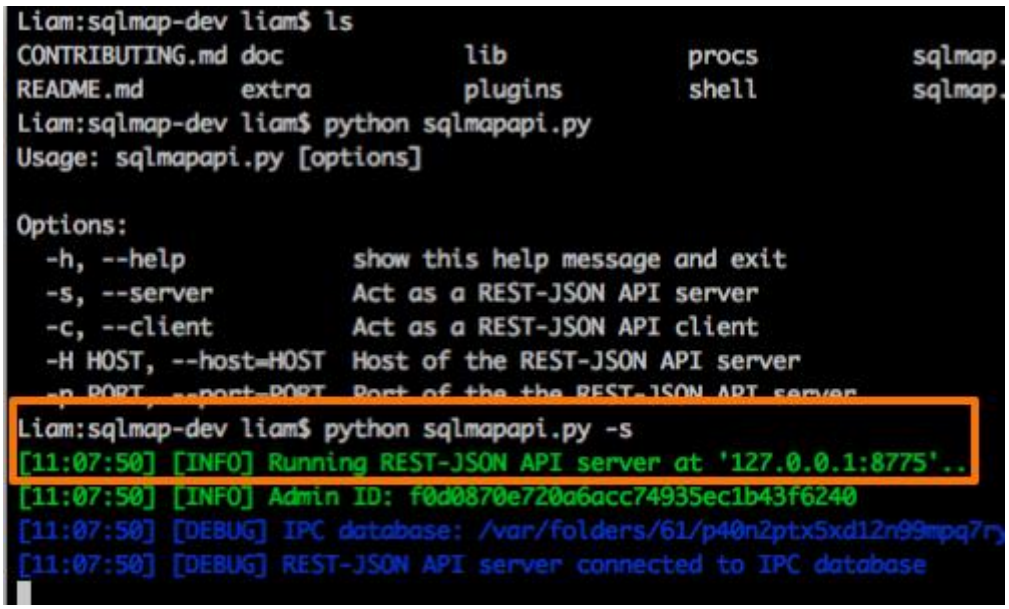
MÓDULO DE ADMINISTRACION





ANEXO 2: CÓDIGO FUENTE DE AUDITOR BÁSICO

Ejecutando sobre localhost



Código fuente de SQLmap auditor de inyección SQL

```
#!/usr/bin/env python

"""
Copyright (c) 2006-2019 sqlmap developers (http://sqlmap.org/)
See the file 'LICENSE' for copying permission
"""

from __future__ import print_function

try:
    import sys

    sys.dont_write_bytecode = True

    try:
        __import__("lib.utils.versioncheck")  # this has to be the first non-standard
import
    except ImportError:
        sys.exit("[!] wrong installation detected (missing modules). Visit
'https://github.com/sqlmapproject/sqlmap/#installation' for further details")

    import bdb
    import distutils
    import glob
    import inspect
```

```
import json
import logging
import os
import re
import shutil
import sys
import tempfile
import threading
import time
import traceback
import warnings

warnings.filterwarnings(action="ignore", message=".*was already imported",
category=UserWarning)
warnings.filterwarnings(action="ignore", category=DeprecationWarning)

from lib.core.data import logger

from lib.core.common import banner
from lib.core.common import checkIntegrity
from lib.core.common import checkPipedInput
from lib.core.common import createGithubIssue
from lib.core.common import dataToStdout
from lib.core.common import filterNone
from lib.core.common import getDaysFromLastUpdate
from lib.core.common import getSafeExString
from lib.core.common import maskSensitiveData
from lib.core.common import openFile
from lib.core.common import setPaths
from lib.core.common import weAreFrozen
from lib.core.convert import getUnicode
from lib.core.data import cmdLineOptions
from lib.core.data import conf
from lib.core.data import kb
from lib.core.common import MKSTEMP_PREFIX
from lib.core.common import setColor
from lib.core.common import unhandledExceptionMessage
from lib.core.exception import SqlmapBaseException
from lib.core.exception import SqlmapShellQuitException
from lib.core.exception import SqlmapSilentQuitException
from lib.core.exception import SqlmapUserQuitException
from lib.core.option import init
from lib.core.option import initOptions
from lib.core.patch import dirtyPatches
from lib.core.patch import resolveCrossReferences
from lib.core.settings import GIT_PAGE
from lib.core.settings import IS_WIN
```

```
    from lib.core.settings import LAST_UPDATE_NAGGING_DAYS
    from lib.core.settings import LEGAL_DISCLAIMER
    from lib.core.settings import THREAD_FINALIZATION_TIMEOUT
    from lib.core.settings import UNICODE_ENCODING
    from lib.core.settings import VERSION
    from lib.parse.cmdline import cmdLineParser
    from thirdparty import six
except KeyboardInterrupt:
    errMsg = "user aborted"

    if "logger" in globals():
        logger.critical(errMsg)
        raise SystemExit
    else:
        import time
        sys.exit("\r[%s] [CRITICAL] %s" % (time.strftime("%X"), errMsg))

def modulePath():
    """
    This will get us the program's directory, even if we are frozen
    using py2exe
    """

    try:
        _ = sys.executable if weAreFrozen() else __file__
    except NameError:
        _ = inspect.getsourcefile(modulePath)

    return getUnicode(os.path.dirname(os.path.realpath(_)),
encoding=sys.getfilesystemencoding() or UNICODE_ENCODING)

def checkEnvironment():
    try:
        os.path.isdir(modulePath())
    except UnicodeEncodeError:
        errMsg = "your system does not properly handle non-ASCII paths. "
        errMsg += "Please move the sqlmap's directory to the other location"
        logger.critical(errMsg)
        raise SystemExit

    if distutils.version.LooseVersion(VERSION) <
distutils.version.LooseVersion("1.0"):
        errMsg = "your runtime environment (e.g. PYTHONPATH) is "
        errMsg += "broken. Please make sure that you are not running "
        errMsg += "newer versions of sqlmap with runtime scripts for older "
        errMsg += "versions"
        logger.critical(errMsg)
```

```
        raise SystemExit

    # Patch for pip (import) environment
    if "sqlmap.sqlmap" in sys.modules:
        for _ in ("cmdLineOptions", "conf", "kb"):
            globals()[_] = getattr(sys.modules["lib.core.data"], _)

        for _ in ("SqlmapBaseException", "SqlmapShellQuitException",
"SqlmapSilentQuitException", "SqlmapUserQuitException"):
            globals()[_] = getattr(sys.modules["lib.core.exception"], _)

def main():
    """
    Main function of sqlmap when running from command line.
    """

    try:
        dirtyPatches()
        resolveCrossReferences()
        checkEnvironment()
        setPaths(modulePath())
        banner()

        # Store original command line options for possible later restoration
        cmdLineOptions.update(cmdLineParser().__dict__)
        initOptions(cmdLineOptions)

        if checkPipedInput():
            conf.batch = True

        if conf.get("api"):
            # heavy imports
            from lib.utils.api import StdDbOut
            from lib.utils.api import setRestAPILog

            # Overwrite system standard output and standard error to write
            # to an IPC database
            sys.stdout = StdDbOut(conf.taskid, messagetype="stdout")
            sys.stderr = StdDbOut(conf.taskid, messagetype="stderr")
            setRestAPILog()

        conf.showTime = True
        dataToStdout("[!] legal disclaimer: %s\n\n" % LEGAL_DISCLAIMER,
forceOutput=True)
        dataToStdout("[*] starting @ %s\n\n" % time.strftime("%X /%Y-%m-%d/"),
forceOutput=True)
```

```
    init()

    if not conf.updateAll:
        # Postponed imports (faster start)
        if conf.smokeTest:
            from lib.core.testing import smokeTest
            os._exitcode = 1 - (smokeTest() or 0)
        elif conf.vulnTest:
            from lib.core.testing import vulnTest
            os._exitcode = 1 - (vulnTest() or 0)
        elif conf.liveTest:
            from lib.core.testing import liveTest
            os._exitcode = 1 - (liveTest() or 0)
        else:
            from lib.controller.controller import start
            if conf.profile and six.PY2:
                from lib.core.profiling import profile
                globals()["start"] = start
                profile()
            else:
                try:
                    start()
                except Exception as ex:
                    os._exitcode = 1

                    if "can't start new thread" in getSafeExString(ex):
                        errMsg = "unable to start new threads. Please check OS (u)limits"
                        logger.critical(errMsg)
                        raise SystemExit
                    else:
                        raise

except SqlmapUserQuitException:
    if not conf.batch:
        errMsg = "user quit"
        logger.error(errMsg)

except (SqlmapSilentQuitException, bdb.BdbQuit):
    pass

except SqlmapShellQuitException:
    cmdLineOptions.sqlmapShell = False

except SqlmapBaseException as ex:
    errMsg = getSafeExString(ex)
    logger.critical(errMsg)
```

```
        raise SystemExit

    except KeyboardInterrupt:
        print()

    except EOFError:
        print()

        errMsg = "exit"
        logger.error(errMsg)

    except SystemExit:
        pass

    except:
        print()
        errMsg = unhandledExceptionMessage()
        excMsg = traceback.format_exc()
        valid = checkIntegrity()

        if valid is False:
            errMsg = "code integrity check failed (turning off automatic issue creation).
"
            errMsg += "You should retrieve the latest development version from official
GitHub "
            errMsg += "repository at '%s'" % GIT_PAGE
            logger.critical(errMsg)
            print()
            dataToStdout(excMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("tamper/", "waf/")):
            logger.critical(errMsg)
            print()
            dataToStdout(excMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("ImportError", "ModuleNotFoundError", "Can't
find file for module")):
            errMsg = "invalid runtime environment ('%s')" % excMsg.split("Error: ")[-
1].strip()
            logger.critical(errMsg)
            raise SystemExit

        elif all(_ in excMsg for _ in ("SyntaxError: Non-ASCII character", ".py on
line", "but no encoding declared")) or any(_ in excMsg for _ in ("source code string
cannot contain null bytes", "No module named")):
```

```
            errMsg = "invalid runtime environment ('%s')" % excMsg.split("Error: ")[-
1].strip()
            logger.critical(errMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("MemoryError", "Cannot allocate memory")):
            errMsg = "memory exhaustion detected"
            logger.critical(errMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("No space left", "Disk quota exceeded", "Disk
full while accessing")):
            errMsg = "no space left on output device"
            logger.critical(errMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("The paging file is too small",)):
            errMsg = "no space left for paging file"
            logger.critical(errMsg)
            raise SystemExit

        elif all(_ in excMsg for _ in ("Access is denied", "subprocess", "metasploit")):
            errMsg = "permission error occurred while running Metasploit"
            logger.critical(errMsg)
            raise SystemExit

        elif all(_ in excMsg for _ in ("No such file", "_")):
            errMsg = "corrupted installation detected ('%s'). " %
excMsg.strip().split('\n')[-1]
            errMsg += "You should retrieve the latest development version from official
GitHub "
            errMsg += "repository at '%s'" % GIT_PAGE
            logger.critical(errMsg)
            raise SystemExit

        elif "Read-only file system" in excMsg:
            errMsg = "output device is mounted as read-only"
            logger.critical(errMsg)
            raise SystemExit

        elif "OperationalError: disk I/O error" in excMsg:
            errMsg = "I/O error on output device"
            logger.critical(errMsg)
            raise SystemExit

        elif "Violation of BIDI" in excMsg:
            errMsg = "invalid URL (violation of Bidi IDNA rule - RFC 5893)"
```

```
            logger.critical(errMsg)
            raise SystemExit

        elif "Invalid IPv6 URL" in excMsg:
            errMsg = "invalid URL ('%s')" % excMsg.strip().split('\n')[-1]
            logger.critical(errMsg)
            raise SystemExit

        elif "_mkstemp_inner" in excMsg:
            errMsg = "there has been a problem while accessing temporary files"
            logger.critical(errMsg)
            raise SystemExit

        elif any(_ in excMsg for _ in ("tempfile.mkdtemp", "tempfile.mkstemp")):
            errMsg = "unable to write to the temporary directory '%s'. " %
tempfile.gettempdir()
            errMsg += "Please make sure that your disk is not full and "
            errMsg += "that you have sufficient write permissions to "
            errMsg += "create temporary files and/or directories"
            logger.critical(errMsg)
            raise SystemExit

        elif all(_ in excMsg for _ in ("twophase", "sqlalchemy")):
            errMsg = "please update the 'sqlalchemy' package (>= 1.1.11) "
            errMsg += "(Reference:
https://qiita.com/tkprof/items/7d7b2d00df9c5f16fffe)"
            logger.critical(errMsg)
            raise SystemExit

        elif all(_ in excMsg for _ in ("scramble_caching_sha2", "TypeError")):
            errMsg = "please downgrade the 'PyMySQL' package (=< 0.8.1) "
            errMsg += "(Reference:
https://github.com/PyMySQL/PyMySQL/issues/700)"
            logger.critical(errMsg)
            raise SystemExit

        elif "must be pinned buffer, not bytearray" in excMsg:
            errMsg = "error occurred at Python interpreter which "
            errMsg += "is fixed in 2.7. Please update accordingly "
            errMsg += "(Reference: https://bugs.python.org/issue8104)"
            logger.critical(errMsg)
            raise SystemExit

        elif "can't start new thread" in excMsg:
            errMsg = "there has been a problem while creating new thread instance. "
            errMsg += "Please make sure that you are not running too many processes"
            if not IS_WIN:
```

```
            errMsg += " (or increase the 'ulimit -u' value)"
        logger.critical(errMsg)
        raise SystemExit

    elif "can't allocate read lock" in excMsg:
        errMsg = "there has been a problem in regular socket operation "
        errMsg += "('%s')" % excMsg.strip().split('\n')[-1]
        logger.critical(errMsg)
        raise SystemExit

    elif "'DictObject' object has no attribute '" in excMsg and all(_ in errMsg for _
in ("(fingerprinted)", "(identified)")):
        errMsg = "there has been a problem in enumeration. "
        errMsg += "Because of a considerable chance of false-positive case "
        errMsg += "you are advised to rerun with switch '--flush-session'"
        logger.critical(errMsg)
        raise SystemExit

    elif all(_ in excMsg for _ in ("pymysql", "configparser")):
        errMsg = "wrong initialization of pymsql detected (using Python3
dependencies)"
        logger.critical(errMsg)
        raise SystemExit

    elif "bad marshal data (unknown type code)" in excMsg:
        match = re.search(r"\s*(.+)\s+ValueError", excMsg)
        errMsg = "one of your .pyc files are corrupted%s" % (" ('%s')" %
match.group(1) if match else "")
        errMsg += ". Please delete .pyc files on your system to fix the problem"
        logger.critical(errMsg)
        raise SystemExit

    elif kb.get("dumpKeyboardInterrupt"):
        raise SystemExit

    elif any(_ in excMsg for _ in ("Broken pipe",)):
        raise SystemExit

    for match in re.finditer(r'File "(.+?)", line', excMsg):
        file_ = match.group(1)
        try:
            file_ = os.path.relpath(file_, os.path.dirname(__file__))
        except ValueError:
            pass
        file_ = file_.replace("\\", '/')
        if "../" in file_:
            file_ = re.sub(r"(\.\./)+", '/', file_)
```

```
            else:
                file_ = file_.lstrip('/')
            file_ = re.sub(r"/{2,}", '/', file_)
            excMsg = excMsg.replace(match.group(1), file_)

        errMsg = maskSensitiveData(errMsg)
        excMsg = maskSensitiveData(excMsg)

        if conf.get("api") or not valid:
            logger.critical("%s\n%s" % (errMsg, excMsg))
        else:
            logger.critical(errMsg)
            dataToStdout("%s\n" % setColor(excMsg.strip(),
level=logging.CRITICAL))
            createGithubIssue(errMsg, excMsg)

    finally:
        kb.threadContinue = False

        _ = getDaysFromLastUpdate()
        if _ > LAST_UPDATE_NAGGING_DAYS:
            warnMsg = "you haven't updated sqlmap for more than %d days!!!" % _
            logger.warn(warnMsg)

        if conf.get("showTime"):
            dataToStdout("\n[*] ending @ %s\n\n" % time.strftime("%X /%Y-%m-
%d/"), forceOutput=True)

        kb.threadException = True

        if kb.get("tempDir"):
            for prefix in (MKSTEMP_PREFIX.IPC, MKSTEMP_PREFIX.TESTING,
MKSTEMP_PREFIX.COOKIE_JAR, MKSTEMP_PREFIX.BIG_ARRAY):
                for filepath in glob.glob(os.path.join(kb.tempDir, "%s*" % prefix)):
                    try:
                        os.remove(filepath)
                    except OSError:
                        pass

            if not filterNone(filepath for filepath in glob.glob(os.path.join(kb.tempDir,
'*')) if not any(filepath.endswith(_) for _ in (".lock", ".exe", ".so", '_'))):  # ignore
junk files
                try:
                    shutil.rmtree(kb.tempDir, ignore_errors=True)
                except OSError:
                    pass
```

```
        if conf.get("hashDB"):
            conf.hashDB.flush(True)

        if conf.get("harFile"):
            try:
                with openFile(conf.harFile, "w+b") as f:
                    json.dump(conf.httpCollector.obtain(), fp=f, indent=4, separators=(',', ':
'))
            except SqlmapBaseException as ex:
                errMsg = getSafeExString(ex)
                logger.critical(errMsg)

        if conf.get("api"):
            conf.databaseCursor.disconnect()

        if conf.get("dumper"):
            conf.dumper.flush()

        # short delay for thread finalization
        _ = time.time()
        while threading.activeCount() > 1 and (time.time() - _) >
THREAD_FINALIZATION_TIMEOUT:
            time.sleep(0.01)

        if cmdLineOptions.get("sqlmapShell"):
            cmdLineOptions.clear()
            conf.clear()
            kb.clear()
            conf.disableBanner = True
            main()

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
    except SystemExit:
        raise
    except:
        traceback.print_exc()
    finally:
        # Reference: http://stackoverflow.com/questions/1635080/terminate-a-multi-
thread-python-program
        if threading.activeCount() > 1:
            os._exit(getattr(os, "_exitcode", 0))
        else:
            sys.exit(getattr(os, "_exitcode", 0))
```

```
else:
    # cancelling postponed imports (because of Travis CI checks)
    from lib.controller.controller import start
```

```
#!/usr/bin/env python

"""
Copyright (c) 2006-2019 sqlmap developers (http://sqlmap.org/)
See the file 'LICENSE' for copying permission
"""

import sys

sys.dont_write_bytecode = True

__import__("lib.utils.versioncheck")  # this has to be the first non-standard import

import logging
import optparse
import os
import warnings

warnings.filterwarnings(action="ignore", message=".*was already imported",
category=UserWarning)
warnings.filterwarnings(action="ignore", category=DeprecationWarning)

from lib.core.common import getUnicode
from lib.core.common import setPaths
from lib.core.data import logger
from lib.core.patch import dirtyPatches
from lib.core.patch import resolveCrossReferences
from lib.core.settings import RESTAPI_DEFAULT_ADAPTER
from lib.core.settings import RESTAPI_DEFAULT_ADDRESS
from lib.core.settings import RESTAPI_DEFAULT_PORT
from lib.core.settings import UNICODE_ENCODING
from lib.utils.api import client
from lib.utils.api import server

try:
    from sqlmap import modulePath
except ImportError:
    def modulePath():
        return getUnicode(os.path.dirname(os.path.realpath(__file__)),
encoding=sys.getfilesystemencoding() or UNICODE_ENCODING)
```

```
def main():
    """
    REST-JSON API main function
    """

    dirtyPatches()
    resolveCrossReferences()

    # Set default logging level to debug
    logger.setLevel(logging.DEBUG)

    # Initialize paths
    setPaths(modulePath())

    # Parse command line options
    apiparser = optparse.OptionParser()
    apiparser.add_option("-s", "--server", help="Run as a REST-JSON API server",
default=RESTAPI_DEFAULT_PORT, action="store_true")
    apiparser.add_option("-c", "--client", help="Run as a REST-JSON API client",
default=RESTAPI_DEFAULT_PORT, action="store_true")
    apiparser.add_option("-H", "--host", help="Host of the REST-JSON API server
(default \"%s\")" % RESTAPI_DEFAULT_ADDRESS,
default=RESTAPI_DEFAULT_ADDRESS, action="store")
    apiparser.add_option("-p", "--port", help="Port of the the REST-JSON API server
(default %d)" % RESTAPI_DEFAULT_PORT,
default=RESTAPI_DEFAULT_PORT, type="int", action="store")
    apiparser.add_option("--adapter", help="Server (bottle) adapter to use (default
\"%s\")" % RESTAPI_DEFAULT_ADAPTER,
default=RESTAPI_DEFAULT_ADAPTER, action="store")
    apiparser.add_option("--username", help="Basic authentication username
(optional)", action="store")
    apiparser.add_option("--password", help="Basic authentication password
(optional)", action="store")
    (args, _) = apiparser.parse_args()

    # Start the client or the server
    if args.server is True:
        server(args.host, args.port, adapter=args.adapter, username=args.username,
password=args.password)
    elif args.client is True:
        client(args.host, args.port, username=args.username, password=args.password)
    else:
        apiparser.print_help()

if __name__ == "__main__":
    main()
```