homo

# UDA-ERP para PYMES

## Módulos de inventarios y manufactura

3 Tomo 3:
Pruebas de
seguridad



Esteban Crespo Martínez
Catalina Astudillo Rodríguez
Juan Maldonado Matute
Lenin Erazo Garzón



# UDA-ERP para PYMES
## Módulos de inventarios y manufactura

3 Tomo 3:
Pruebas de
seguridad

UDA ERP PARA PYMES
Módulos de inventarios y manufactura

Tomo 3: Pruebas de seguridad

© del texto: Esteban Crespo Martínez, Catalina Astudillo Rodríguez, Juan Maldonado Matute, Lenin Erazo Garzón, 2022
© de esta edición: Universidad del Azuay. Casa Editora, 2022

Primera edición: Casa Editora anterior, Ciudad, 2022

ISBN OBRA COMPLETA: 978-9942-618-57-3
ISBN: 978-9942-618-62-7
E-ISBN: 978-9942-618-63-4

Diseño y diagramación: Daniela Durán
Corrección de estilo: Oswaldo Encalada V.

Libro arbitrado por pares: Andrea Mory, Diego Pinto Auz, Rayner Durango

Impresión: PrintLab / Universidad del Azuay
en Cuenca del Ecuador

*Se prohíbe la reproducción total o parcial de esta obra, por cualquier medio, sin la autorización expresa del titular de los derechos*

CONSEJO EDITORIAL / UNIVERSIDAD DEL AZUAY

Francisco Salgado Arteaga
Rector

Genoveva Malo Toral
Vicerrectora Académica

Raffaella Ansaloni
Vicerrectora de Investigaciones

Toa Tripaldi
Directora de la Casa Editora

# UDA-ERP para PYMES
## Módulos de inventarios y manufactura

3 Tomo 3:
Pruebas de seguridad

Autores:
Esteban Crespo Martínez
Catalina Astudillo Rodríguez
Juan Maldonado Matute
Lenin Erazo Garzón

UNIVERSIDAD DEL AZUAY | Casa Editora

# Prólogo

**Este libro forma parte de los tres tomos que resumen el desarrollo de la primera etapa del proyecto de desarrollo de un Software ERP realizado en la Universidad del Azuay para las MIPYMES del Ecuador, orientado a los módulos de Manufactura e Inventarios.**

El proyecto mencionado surge como una solución para las MIPYMES, englobando además a pequeños emprendimientos de empresarios emergentes y artesanos de la región, cuyos recursos económicos por la naturaleza y tiempo de incubación de la empresa, no abastecen para la inversión de un producto para la gestión comercial. Con esta solución tecnológica, la Universidad se proyecta hacia la vinculación y servicio de los grupos mencionados.

Contar con la información justa en el momento oportuno ha significado 'poder', y solamente las personas o grupos de personas que se podían permitir financiar ese intercambio de información disfrutaron de ese privilegio. Esto genera la indagación de nuevas técnicas y habilidades para gestionar la información, muchas veces para aprovecharse de ella, con diversas motivaciones, en gran parte con fines de tendencia negativa, lo que genera riesgo, con diferentes niveles de impacto, por lo que es crucial protegerla, ya que constituye uno de los activos más importantes de una organización (Crespo, Ecu@Risk, Una metodología para la gestión de Riesgos, 2017).

Sin embargo, pese a la falta de conocimiento sobre cómo protegerla adecuadamente, o a la complejidad exigida por muchas normas internacionales y mejores prácticas de desarrollo, muchas organizaciones descuidan asegurarla.

Las pruebas de seguridad llevadas desde el punto de vista ético, sin duda, aportan considerablemente a reducir el riesgo que puede producirse en un sistema lleno de vulnerabilidades, garantizando así la confidencialidad, integridad y disponibilidad de la información y de los componentes informáticos involucrados en su captura, procesamiento, transporte y almacenamiento.

Este libro está dirigido a desarrolladores de aplicaciones empresariales del sector MIPYME, en los cuales se despliega el software de Manufactura e Inventarios como un recurso de consulta de conceptos y fundamentos teóricos para pruebas de seguridad. También está orientado a docentes y estudiantes de las asignaturas relacionadas a las áreas de seguridad informática.

# *Agradecimientos*

Iniciamos por expresar nuestro agradecimiento al Doctor Francisco Salgado Arteaga, Rector; a la Doctora Raffaella Ansaloni, Vicerrectora de Investigaciones; al Ingeniero Jacinto Guillén García, ex Vicerrector de Investigaciones; y al Ingeniero Oswaldo Merchán Manzano Decano de la Facultad de Ciencias de la Administración, autoridades de la Universidad del Azuay.

A la valiosa contribución otorgada por los revisores: Magister María José González, Doctora Patricia Ortega, Magister Andrés Patiño, Doctor Juan Gabriel Barros, docentes de la Universidad del Azuay, Magister Andrea Mory, docente de la Universidad Católica de Cuenca, Doctor Diego Pinto Auz, docente de la Universidad de las Fuerzas Armadas -ESPE y Magister Rayner Durango, docente de la ESPOL, quienes gracias a sus sugerencias enriquecieron el resultado final de este texto.

Finalmente, expresamos nuestro agradecimiento al Doctor Oswaldo Encalada de la Universidad del Azuay, por su apoyo en la revisión de estilo.

# *Dedicatoria*

Consideramos dedicar este esfuerzo a cada uno de los miembros de nuestras respectivas familias.

Queremos que sepan que son la brújula que guía cada paso de nuestras vidas, que son la inspiración para escalar aún más alto, que son el mayor escudo en momentos de guerra y nuestro consuelo cuando las adversidades nos hacen tropezar.

Queremos que sepan que el tiempo que no podemos estar con ustedes no es mal gastado, simplemente formamos las bases sólidas para construir un futuro mejor, inclusive para las generaciones futuras.

Queremos que sepan perdonar el estrés que muchas veces llevamos sobre nosotros por la intensidad del trabajo intelectual que implica generar nuevo conocimiento para un mundo mejor.

## Contenido

Este trabajo se enfoca a i) establecer una guía de pruebas de penetración basada en herramientas para este efecto, y ii) documentar las pruebas de penetración realizadas al software UDA-ERP para PYMES, que se viene desarrollando en la Universidad del Azuay.

# Índice

**1. Las pruebas de seguridad .....15**
1.1. Introducción .....15
1.2. Pentesting .....15
1.2.1. El hacker ético ..... 16
1.2.2. Riesgo informático ..... 16
1.2.3. El principio de defensa en profundidad.....17
1.3. Metodología ..... 19
1.4. Etapas del pentesting..... 20
1.4.1. Fase 1: Conceptualización..... 20
1.4.2. Fase 2: Preparar el laboratorio ..... 21
1.4.3. Fase 3: Obtención de información..... 21
1.4.4. Fase 4: Modelamiento de amenazas ..... 22
1.4.5. Fase 5: Análisis de vulnerabilidades ..... 22
1.4.6. Fase 6: Explotación ..... 22
1.4.7. Fase 7: Post explotación..... 22
1.4.8. Fase 8: Reporte ..... 23

**2. Aplicación de las pruebas de seguridad ..... 27**
2.1. Fase 1 - Conceptualización..... 27
2.1.1. Ámbito de evaluación ..... 27
2.2. Primero es lo primero..... 29
2.3. Fase 2 - Preparación del laboratorio ..... 30
2.3.1. Kali Linux ..... 30
2.3.2. Apex-Sert ..... 33
2.4. Fase 3: Obtención de información ..... 36

2.4.1. Identificación del contexto ..... 36
2.4.1.1. Resultados del reconocimiento ..... 42
2.4.1.2. Resultados del escaneo ..... 43
2.4.2. Identificación de vulnerabilidades en el código fuente..... 47
2.4.2.1. Recopilación de información ..... 48
2.4.2.2. Pruebas de autenticación..... 50
2.4.2.3. Ataque de fuerza bruta..... 53
2.4.3. Cross Site Scripting – XSS ..... 54
2.4.4. SQL Injection ..... 59
2.4.5. DDoS – Denegación de Servicios ..... 61
2.4.5.1. IP SPOOFING..... 64
2.4.5.2.SYN Flood..... 66
2.5. Fase 4 y 5: Modelamiento de amenazas y análisis de vulnerabilidades..... 69
2.5.1. Paso 1. Recopilar información básica ..... 70
2.5.2. Paso 2. Crear y analizar el modelo de amenazas .....71
2.5.3. Paso 3. Analizar las amenazas..... 73
2.5.4. Paso 4. Identificar las tecnologías y técnicas de mitigación ..... 79
2.5.5. Paso 5. Modelo de seguridad de documento y las consideraciones de implementación..... 82
2.5.6. Paso 6. Implementar y probar las mitigaciones ..... 82
2.5.7. Paso 7. Mantener el modelo de amenazas sincronizado con el diseño ..... 82
2.6. Fase 6: Explotación de vulnerabilidades..... 83
2.6.1. Autenticación ..... 83
2.6.1.1. Fortaleza de contraseñas ..... 84
2.6.1.2. Uso de segundo factor de autenticación ..... 84
2.6.1.3. Falta de uso de mecanismos de cifrado ..... 86
2.6.1.4. Fuerza bruta a SSH..... 87
2.6.2. Cross Site Scripting – XSS ..... 88
2.6.3. Identificación de vulnerabilidades en el código fuente..... 91
2.6.3.1. Resultados obtenidos con OWASP ZAP ..... 94
2.6.3.2. Resultados obtenidos con APEX – SERT .....110
2.6.4. SQL Injection ..... 111

2.6.5. DDoS – Denegación de servicios ..... 111
2.7. Fase 7: Post explotación ..... 116
2.8. Fase 8: Reporte ..... 116
2.8.1. Resumen ejecutivo ..... 116
2.8.2. Resultados obtenidos ..... 117
2.8.2.1. Puertos de escucha ..... 118
2.8.2.2. Contexto computacional ..... 118
2.8.2.3. Pruebas de vulnerabilidades de código ..... 119
2.8.2.4. Vulnerabilidad ante ataques de tipo XSS – Cross Site Scripting ..... 120
2.8.2.5. Vulnerabilidad ante ataques de inyección SQL ..... 120
**3. Glosario ..... 122**
**4. Bibliografía ..... 125**

# LAS PRUEBAS DE SEGURIDAD



# 1 Las pruebas de seguridad

## 1.1 Introducción

Para comprender de mejor manera los procedimientos técnicos que involucra el presente trabajo, es importante conocer algunos conceptos que forman parte del trabajo de pruebas de seguridad informática. Estos conceptos son tratados a continuación.

## 1.2 Pentesting

En la actualidad, existen muchas técnicas de pruebas de seguridad informática, entre ellas, la técnica de pruebas de penetración conocida también como pentesting.

El pentesting se resume en un conjunto de metodologías, procesos y procedimientos que son utilizados por quienes hacen este trabajo, mediante guías específicas y aprobadas para sobrepasar las protecciones de los sistemas de información, en las que se incluye el romper las características de seguridad integrada de un sistema. Este tipo de pruebas está asociado con el asesoramiento técnico, administrativo y las configuraciones operacionales y controles de un sistema. Normalmente las pruebas de penetración evalúan únicamente la seguridad de los sistemas de información tal cual fueron construidos (Broad & Bindner, 2013).

Es importante mencionar que, para este tipo de pruebas, los administradores de la plataforma y su equipo de trabajo pueden o no saber que las pruebas se están llevando a cabo.

En resumen, el pentesting es utilizado para la defensa proactiva y para la protección de los sistemas de información, donde se sugiere utilizar sistemas operativos especiales sobre un núcleo UNIX, scripts desarrollados, utilidades y aplicaciones (Stefinko, Piskozub, & Banakh, 2016), considerando que las pruebas sobre ambientes en producción pueden provocar daños colaterales; consideración que debe hacérsela sobre todo en sistemas com-

partidos en la nube, debido a la presencia de otros inquilinos (Li, y otros, 2015).

### 1.2.1. El hacker ético

El hacker ético es el profesional que estará encargado de las pruebas. Debe tener un alto nivel de conocimiento técnico para utilizar las herramientas y técnicas de penetración. Antes de proceder debe contar con el consentimiento, por escrito, de la alta dirección.

Un hacker ético es conocido también como un hacker de sombrero blanco, debido a que, como ya se lo había mencionado, cuenta con un permiso de la alta dirección para realizar las pruebas de vulnerabilidad de los sistemas. Una diferencia opuesta la tiene el hacker de sombrero negro, que es el que tiene la capacidad de romper un sistema sin autorización, y sin retroalimentar sobre las vulnerabilidades del mismo a un administrador (Broad & Bindner, 2013).

### 1.2.2. Riesgo informático

Muchas definiciones aparecen para la seguridad de la información. Una de ellas es: "la protección contra todos los daños sufridos o causados por la herramienta informática y originados por el acto voluntario y de mala fe de un individuo" (Royer, 2004).

Crespo, citando a Alcides, menciona que "el riesgo informático es un conjunto de normas y procedimientos que son aplicados para salvaguardar un sistema informático, cuya finalidad es garantizar que todos los recursos que conforman el sistema informático sean utilizados para el fin que fueron creados sin ninguna intromisión (Alcides, 2009)".

Así, considerando las definiciones anteriores, se puede argumentar que la seguridad de la información se fundamenta en tres principios básicos: confidencialidad, disponibilidad e integridad; entendiéndose por confidencialidad a los mecanismos que garantizan el acceso a la información a personas y organismos autorizados; por integridad a la consistencia de la información almacenada; y por disponibilidad a la característica de que la información debe estar disponible en el momento en que sea requerida (Crespo, 2016).

La vulnerabilidad hace referencia a las debilidades que existen en un sistema de información, lo que permite que pueda ser fácilmente atacado, evadiendo el control de acceso y la confidencialidad de los datos y las aplicaciones existentes (Cordero Torres, 2015). Las vulnerabilidades deben ser expresadas en una escala numérica para poder posteriormente cuantificar su impacto, y, citando a Burgos y Campos, se sugiere que éstas sean identificadas y valoradas individualmente (Burgos Salazar & Campos, 2008).

Las amenazas son los elementos que pueden dañar o alterar la información de una u otra forma. Estas generalmente pueden ser encontradas a partir de una vulnerabilidad existente. El riesgo, a su vez, es la probabilidad que tiene una amenaza para originarse y que puede generar un cierto impacto en la organización (Crespo, 2017).

### 1.2.3. El principio de defensa en profundidad

El principio de defensa en profundidad es un modelo en capas que sugiere establecer varios niveles de seguridad dentro del sistema de seguridad informático que mantiene una organización, los mismos que permitan mitigar o retrasar un ataque (Vásquez & López, 2016). Así, se puede fortalecer un sistema, haciendo que los atacantes desistan de vulnerarlo y se dediquen solamente a violar sistemas débiles.

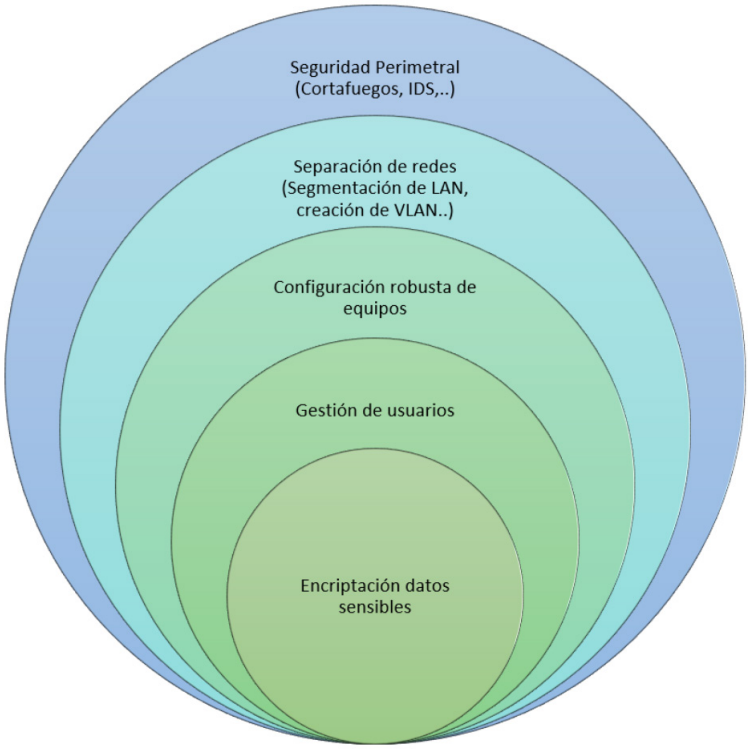



*Figura 1. Principio de defensa en profundidad. Fuente: (Gómez Vieites, 2011)*

El dejar abandonado un sistema luego de su desarrollo e implementación no garantiza que sea seguro y esté libre de vulnerabilidades. Por lo tanto, es imprescindible realizar pruebas de penetración en los sistemas informáticos post desarrollo y post producción, lo que permitirá analizar y documentar cualquier tipo de vulnerabilidades.

Esto hará que los desarrolladores escriban código seguro, y que los administradores de infraestructura se preocupen de desactivar o inhabilitar los servicios que son innecesarios, considerando además con un permanente cambio de contraseñas, aplicando el concepto de rotación y el uso de claves seguras, y aplicando las actualizaciones a los sistemas operativos y demás aplicaciones relacionadas.

## 1.3 Metodología

Muchas son las pruebas de penetración o pentesting que se pueden hacer a un objetivo informático. Sin embargo, es importante considerar que las más comunes y aceptadas son las pruebas de penetración de caja negra (black box), de caja blanca (white box) y de caja gris (gray box).

Según Caballero (2015), la prueba de caja negra consiste en realizar los ataques sin conocer la estructura interna de la organización, y es realizada solo con el conocimiento de una dirección IP o la URL provista por el personal del proyecto o el personal de TI.

La prueba de caja blanca consiste en realizar las pruebas de ataque considerando toda la información proporcionada, esto es, diagramas, detalles sobre el hardware, software, sistemas operativos, firewalls, etc. Esto ayuda a que los ataques sean objetivos, sin embargo, pueden quedar algunos detalles sueltos debido a que no se consideran otros elementos que podrían ser descubiertos con la técnica de ataque de caja negra (Caballero, 2015).

Para Caballero (2015), la prueba de caja gris simula un ataque realizado por un empleado descontento, el cual tiene un nivel de privilegios adecuado, además de contar con permisos de acceso a la red interna.

En este trabajo se realizaron pruebas tanto de caja negra como de caja blanca y caja gris. La primera etapa del ataque consistió en hacer un escaneo a ciegas (caja negra), solamente contando con la dirección IP perteneciente al software ERP, provisto por la directora del proyecto. Con la información obtenida en el ataque se ha podido identificar servicios y puertos de escucha que posiblemente no serían revelados en una prueba de caja blanca. Por otro lado, la información provista en la técnica de caja blanca ha servido para confirmar los resultados obtenidos en la técnica de caja negra.

En relación con la prueba de caja gris, se puede considerar a las pruebas internas con un usuario que tenga acceso a la aplicación, el mismo que utilizará los privilegios con los que cuenta; pero con sentido malicioso.

Se incluye también un estudio descriptivo-cuantitativo con la cual se realizará un estudio de interrelaciones realizando un análisis comparativo de los resultados de las diferentes herramientas utilizadas en cada una de las etapas del ataque, y por último también se aplicará la metodología explicativa y concluyente donde se evidenciarán los resultados de la causa y el efecto que tiene la prueba de penetración.

## 1.4 Etapas del pentesting

Las pruebas de penetración deben seguir un orden estructurado y lógico. Esto es, desde la conceptualización de las pruebas hasta la emisión del informe que incluye los resultados. En la siguiente sección se tratarán las etapas que se deberán considerar en las pruebas de penetración o pentesting.

### 1.4.1. Fase 1: Conceptualización

En esta etapa se define el alcance que tendrán las pruebas de penetración. Para este caso de análisis del software ERP, se han considerado los siguientes escenarios:

a. Partir de una prueba a ciegas (caja negra), donde lo único que se conoce es la dirección IP donde está alojado el servicio informático. Esto permitirá hacer un diagnóstico del entorno en el que se ejecuta el ERP, situación similar a la que se enfrentaría un hacker de sombrero negro. Previo a un ataque, es importante confirmar que se trata del entorno al que se realizarán las pruebas, ya que la falta de comunicación entre el hacker ético y el cliente, que muchas veces espera un análisis simple pueda llevar a una situación complicada por no haber contemplado una prueba intrusiva intensa.

b. Comprobar los resultados obtenidos mediante la aplicación de la prueba de caja blanca y caja gris, en los que, apoyándose en la información técnica proporcionada por el equipo de desarrollo y el líder del proyecto, se podrá comprobar y afirmar o rechazar los supuestos adquiridos en la etapa anterior.

La etapa de conceptualización es sumamente importante porque aquí se definen los objetivos de la prueba de penetración (Weidman, 2014). Es importante preguntar al cliente: ¿Qué le preocupa? ¿Qué parte de la infraestructura o del sistema es de la que sospecha vulnerabilidades? ¿Existen dispositivos delicados que deban ser considerados mientras se realiza el pentesting? (esto debido a que, en la práctica, se pueden encontrar dispositivos de cifrado de información crítica, equipos médicos y hospitalarios conectados a la red, termostatos, etc., así como también equipos virtualizados en los que existen otros clientes alojados).

Además de lo señalado anteriormente, es importante conocer el ámbito de direcciones IP que se van a analizar y vulnerar, el horario de pruebas, la información del contacto si ocurre algo serio, y, sobre todo, contar con la autorización escrita. Si el objetivo no forma parte de la organización contratante, es importante que este tercero conozca formalmente las pruebas a las que será sometido (Weidman, 2014).

### 1.4.2. Fase 2: Preparar el laboratorio

La preparación del laboratorio exige la identificación de las herramientas que serán utilizadas para la ejecución del pentesting. Se sugiere utilizar el kit de pruebas Kali. Kali es una distribución basada en Linux que sirve para realizar pruebas de penetración. El sitio oficial de la herramienta es www.kali.org.

Por otro lado, se encuentra Vega, que es una herramienta gratuita para el escaneo de vulnerabilidades y una plataforma para pruebas de seguridad de entornos web, lo que permite la ejecución automatizada de análisis de inyección SQL (SQL Injection) o scripts cruzados (Cross-Site Scripting). El sitio oficial de la herramienta es https://subgraph.com/vega/.

Otra herramienta muy importante por la personalización de análisis guiados es NMAP. Esta herramienta permite el análisis de puertos, servicios y servidores en una red, con un nivel de detalle bastante interesante. El sitio oficial de la herramienta es https://nmap.org/.

Para este proyecto se plantea el uso del Metasploit Framework, que es un conjunto de scripts diseñados con la finalidad de comprometer un sistema informático, además de la personalización de los componentes que utiliza. Este framework de análisis puede ser encontrado en la distribución de Linux Kali.

### 1.4.3. Fase 3: Obtención de información

El siguiente paso consiste en obtener la información. Para este caso se han formulado las dos técnicas mencionadas en la fase anterior: caja negra y caja blanca.

Este análisis sugiere recolectar y analizar libremente cualquier origen de información (Weidman, 2014). En esta etapa se sugiere utilizar herramientas informáticas para análisis de puertos, identificación de sistemas operativos y usuarios conectados, para obtener una idea del entorno que será posteriormente comprometido.

### 1.4.4. Fase 4: Modelamiento de amenazas

Con la información obtenida en la sección anterior se pueden establecer planes de ataque. Si el software es adquirido a terceros, se puede investigar más sobre el mismo para encontrar vulnerabilidades y posibles actualizaciones provistas por el fabricante. A su vez, si el software fue desarrollado dentro de la organización, entonces el atacante puede tener acceso a la información interna mediante la vulneración de plataformas de desarrollo y código fuente (Weidman, 2014).

### 1.4.5. Fase 5: Análisis de vulnerabilidades

En este punto el atacante comienza a buscar e identificar amenazas, de forma activa, con la finalidad de establecer estrategias que permitan lograr su objetivo. Las pruebas con "exploits" no exitosos, generalmente confluyen en negación de servicios, desconfiguración de los sistemas de alerta, y otros elementos que pueden ser el fracaso de una vulneración satisfactoria.

Muchas herramientas automatizadas probablemente no son suficientes para los múltiples escenarios que pueden presentarse; es por ello que es importante considerar el uso de herramientas que puedan ser parametrizadas y ejecutadas de forma manual, tales como Kali, Nessus o NMAP.

### 1.4.6. Fase 6: Explotación

La explotación consiste en ejecutar varias tretas que aprovechen las vulnerabilidades descubiertas previamente, utilizando para ello las herramientas Metasploit y Vega, con el objetivo de encontrar formas de acceder al sistema evaluado. En esta sección se describen.

### 1.4.7. Fase 7: Post explotación

Una vez que se han explotado la mayoría de vulnerabilidades, se debe hacer un recuento de lo que se ha obtenido. Para ello se deben revisar los sistemas atacados, buscando archivos que han sido interesantes, de manera que puedan servir como mecanismos necesarios que habiliten la elevación de privilegios (Weidman, 2014).

También se deben evaluar los objetivos que no estuvieron disponibles, por alguna razón, en el momento en el que se realizó la etapa anterior.

### 1.4.8. Fase 8: Reporte

El reporte es el último paso que se debe realizar en una prueba de penetración. La escritura del mismo hace referencia a las pruebas realizadas, aspectos encontrados, y las sugerencias para corregir los problemas detectados. Es importante, además, considerar el redactar dos tipos de informes: uno netamente técnico y otro asimilable para quienes no son técnicos. El lenguaje utilizado en seguridad de la información suele ser muy denso, y posiblemente inentendible para el común de los mortales. Así, coincidiendo con (Weidman, 2014), se sugiere que existan dos tipos de reportes, uno técnico y otro que sea un resumen ejecutivo.

El resumen ejecutivo describe los objetivos de la prueba y los altos niveles de evaluación que se han realizado. Esto se lo hace con la intención de que la alta dirección incluya un programa de aseguramiento y seguridad de la información en su planificación anual.

(Weidman, 2014) sugiere abordar los siguientes bloques:

a. Antecedentes: en el que se indique el propósito de la prueba y la definición de los términos que pueden resultar no familiares para ejecutivos no técnicos.
b. Postura general: una revisión a la efectividad de las pruebas realizadas, los aspectos encontrados y las situaciones generales que causan vulnerabilidades, como es la ausencia de gestión de parches de seguridad.
c. El perfil del riesgo: Una revisión a la situación general de la seguridad de la organización en relación con otras organizaciones cuyas medidas son altas, moderadas y bajas. Además, se debería incluir una explicación de este ranking.
d. Hallazgos generales: un resumen general de los aspectos identificados, sustentados con estadísticas y mediciones de la efectividad de cualquier contramedida implementada.
e. Sumario de recomendaciones: una revisión de alto nivel de las tareas que son requeridas para corregir las debilidades encontradas en la prueba de penetración.
f. Mapa estratégico: Proporcionar al cliente metas a corto, mediano y largo plazo que deberá considerar en las mejoras.

Por otro lado, concordando con (Weidman, 2014), el reporte técnico es el documento que resume los detalles técnicos de las pruebas realizadas. Este debería incluir:

a. Introducción: Un inventario de los detalles, como es el ámbito, los contactos, etc.
b. Obtención de información: Detalles de las evidencias encontradas en la fase de obtención de información.

c. Evaluación de vulnerabilidades: detalles de las evidencias recolectadas en la etapa de análisis de vulnerabilidades.

d. Verificación de explotación de vulnerabilidades: detalles de los elementos encontrados en la fase de explotación

e. Post explotación: Detalle de las evidencias encontradas en la fase de post explotación.

f. Riesgo/Exposición: Una descripción cuantitativa de los riesgos descubiertos. Esta parte del informe debe incluir una estimación de la pérdida si las vulnerabilidades encontradas por el atacante fueran explotadas.

g. Conclusión: una reseña general final de las pruebas realizadas.

# APLICACIÓN DE LAS PRUEBAS DE SEGURIDAD



# 2 Aplicación de las pruebas de seguridad

## 2.1. Fase 1 - Conceptualización

El inicio de las pruebas de seguridad parte una vez que se haya identificado el objetivo del ataque. Para este caso, el objetivo del ataque, en esta primera etapa, es el servidor de pruebas del sistema ERP que se desarrolla en las instalaciones de la Universidad del Azuay.

La dirección URL del servidor donde se encuentran alojados los servicios y archivos del ERP es: http://172.16.1.87:8080/ords/f?p=552:LOGIN_DESKTOP::::::

### 2.1.1. Ámbito de evaluación

Lo que se pretende comprobar, de acuerdo con los requerimientos del líder del proyecto, es la seguridad del código, así como el sistema de archivos y vulnerabilidades propias del software. Por ello se delimitan las pruebas a los siguientes escenarios:

a. Los puertos de escucha del servidor deben estar controlados. Asimismo, no deben ser puertos considerados como conflictivos.

b. Identificación del contexto: sistema operativo, recursos utilizados, versión de la base de datos, usuarios iniciados, entre otras características.

c. Las pruebas deben identificar vulnerabilidades en el código. Para ello será importante considerar el uso de herramientas para el análisis de código fuente, conocidas también como SAST - Static Application Software Testing (herramientas para la prueba de software de aplicación estática). El proyecto OWASP Benchmark es una suite de pruebas libre y abierta, diseñada para evaluar la velocidad, la cobertura y la precisión de herramientas y servicios automatizados de detección de vulnerabilidades de software. Sin la capacidad de medir estas herramientas, es difícil entender sus fortalezas y debilidades, y compararlas entre sí. Cada versión del Benchmark OWASP contiene miles de casos de prueba que son completamente ejecut-

ables y explotables, cada uno de los cuales se correlaciona con el número CWE (Common Weakness Enumeration) apropiado para esa vulnerabilidad.

d. Pruebas de scripts (cross-site scripting - XSS) que permitirán identificar el cruce de variables de sesión entre una y otra página. Este tipo de vulnerabilidades se presentan siempre en aplicaciones que se ejecutan en navegadores o contenedores de sitios web. Un XSS es un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, poniendo en riesgo la integridad del sistema.

e. Pruebas de inyección SQL, que consisten en una técnica o método de infiltración de código intrusivo, que se aprovecha de una debilidad informática presente en una aplicación, en el nivel de validación de las entradas, para realizar operaciones sobre una base de datos. Con esta técnica se pretende extraer datos ocultos como usuarios y contraseñas de la aplicación, sobrescribir información valiosa o simplemente ejecutar comandos en el equipo donde se hospeda la base de datos.

## 2.2. Primero es lo primero

Antes de continuar, hay que descargar las responsabilidades y contar con las autorizaciones respectivas. Esto se resume en contar con la aprobación de la alta dirección para hacer la evaluación de las pruebas de seguridad. Obviar este importantísimo paso puede poner al evaluador de seguridad en el lado de los hackers de sombrero negro: sujetos perseguidos por la ley. Se sugiere, por lo tanto, redactar un documento en el que se contemple básicamente:

a. Las pruebas que se realizarán. Estas pruebas deben estar correctamente delimitadas. Para este caso, las pruebas se resumen en cuatro técnicas: DDoS o Negación de Servicios, pruebas de análisis de código fuente, Inyección SQL y Cross-Site Scripting. Se suma una quinta técnica que es la de identificación del contexto, que consiste en escaneo y recolección de datos relevantes, que posteriormente serán utilizados en el ataque.

b. Los sistemas y aplicaciones que se vulnerarán. El sistema que se intentará vulnerar será únicamente el servicio que se encuentra en el servidor de alojamiento con dirección IP 172.16.1.87

c. El personal a cargo de las pruebas de seguridad

d. El acuerdo de conocimiento del personal de la institución sobre las pruebas que van a ser realizadas, y que, por lo tanto, consideran que el evaluador de seguridad no está incumpliendo con ninguna norma o ley de delitos informáticos vigentes en la institución y en el país.

e. Un acuerdo de confidencialidad.

Para estas pruebas se ha planteado el documento de descargo que se encuentra en el anexo 1: Oficio de autorización y descargo.

## 2.3. Fase 2 - Preparación del laboratorio

El primer aspecto importante que se debe considerar es la preparación del laboratorio de análisis. Este proceso es clave, pues contando con las herramientas adecuadas se pueden conseguir los resultados esperados.

### 2.3.1. Kali Linux

Kali, como ya se había mencionado anteriormente, es una distribución Linux que contiene múltiples herramientas que sirven específicamente para pruebas de seguridad. Debido a que requiere ejecutarse en un entorno exclusivo, se sugiere que la imagen de este sistema operativo se ejecute en una máquina virtual, como por ejemplo Virtual Box.




*Figura 2. Entorno de pruebas Kali*

Es importante verificar que se encuentren habilitadas las interfaces de red. Para ello se debe ir a las opciones de la máquina virtual, a dispositivos, preferencias de red...

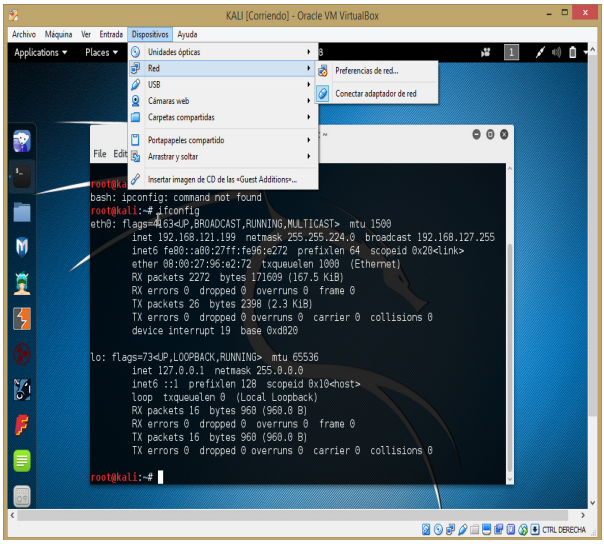



*Figura 3. Ajustes de interfaz de red en máquina virtual 1/3*

... y luego realizar los ajustes

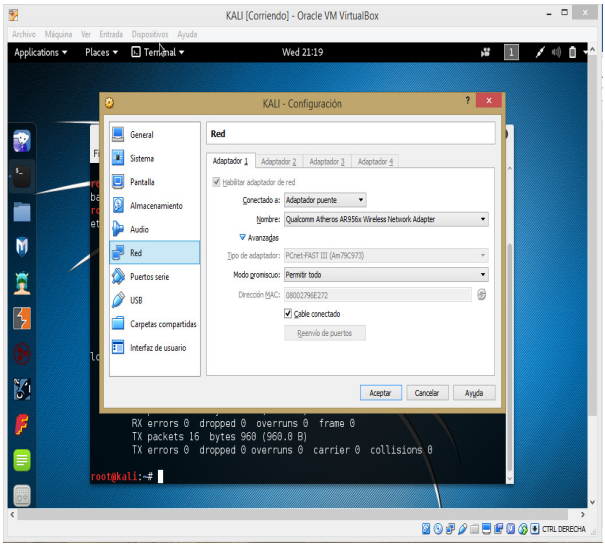



*Figura 4. Ajustes de interfaz de red en máquina virtual 2/3*

Debido a que se realizará un análisis a un host externo, la configuración debe establecerse a una conexión NAT. Esto es, cambiar el tipo de conexión de “adaptador puente” a “NAT”.

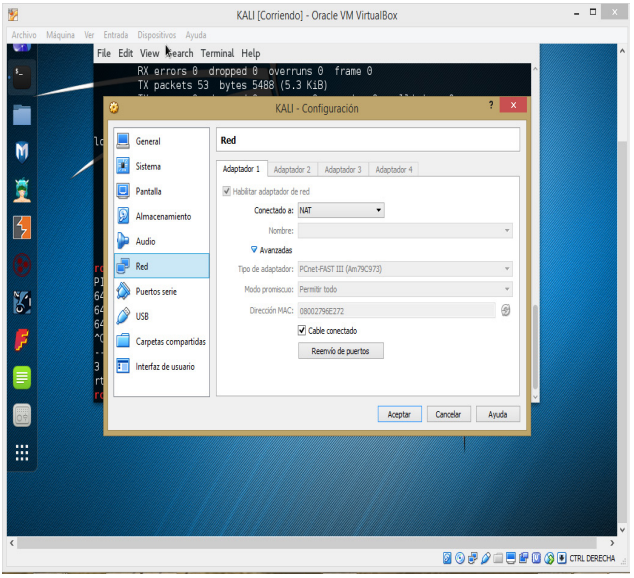



*Figura 5. Ajustes de interfaz de red en máquina virtual 3/3*

Se termina el ajuste comprobando, desde una ventana de terminal de Kali, la comunicación con el servidor que será analizado. Para cumplir este objetivo se ejecutó el comando: ping 172.16.1.87.

La respuesta a la petición ping realizada (pong) ha sido satisfactoria, tal como se puede comprobar en la figura 6.

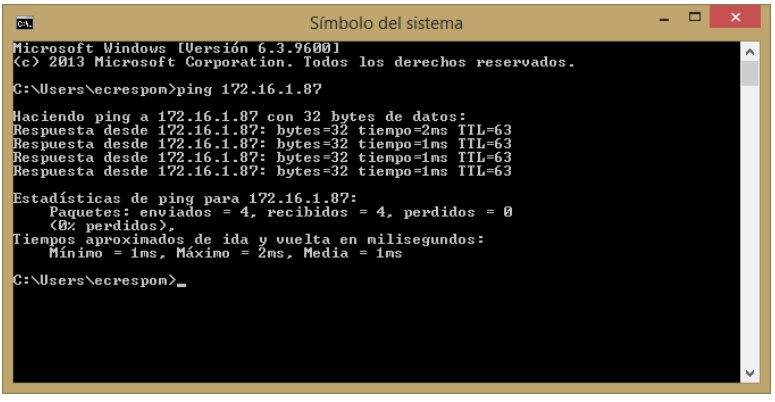



*Figura 6. Comprobación de comunicación con el servidor*

### 2.3.2. Apex-Sert

Finalmente, al tratarse de una aplicación elaborada con Oracle Apex, será importante incluir al análisis la herramienta Apex-Sert.

APEX-SERT es una aplicación APEX de código abierto que está diseñada para evaluar aplicaciones APEX en busca de vulnerabilidades de seguridad. Se instala en una instancia de APEX una vez y está disponible de forma inmediata para cualquier desarrollador que use las credenciales existentes. Una vez instalado, cualquier desarrollador puede ejecutar APEX-SERT en aplicaciones que residen en sus respectivos espacios de trabajo. El administrador APEX-SERT puede otorgar, incluso a un usuario en un espacio de trabajo, la capacidad de evaluar aplicaciones en espacios de trabajo ajenos. Las evaluaciones también se pueden programar para ejecutarse en un intervalo específico con los resultados que al finalizar, son enviados por correo electrónico.

Para preparar el laboratorio será necesario descargar la herramienta desde https://github.com/OraOpenSource/apex-sert/releases, y elegir la herramienta en formato zip (si se está trabajando en Windows), o tar (si se trabaja en Linux). Una vez descargada, descomprimirla. Ir luego al directorio "releases", ubicar el archivo sert_050000b2.zip y descomprimirlo. El resultado es un directorio "sert" con la siguiente estructura de contenidos:

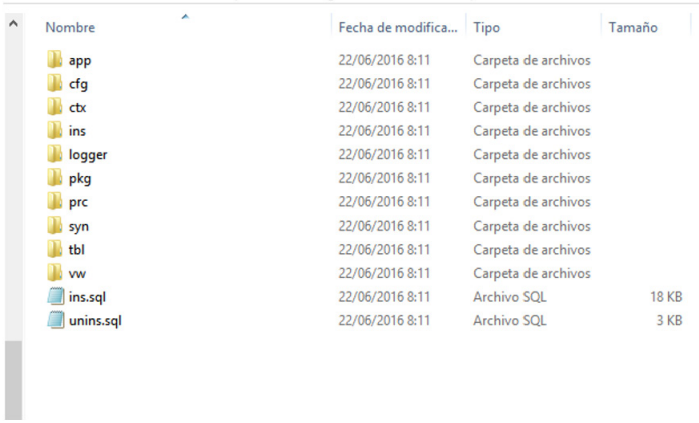



*Figura 7. Instalación del APEX-SERT. Estructura del archivo comprimido SERT*

Luego se procede a crear el tablespace ejecutando el comando siguiente:

```
sqlplus / as sysdba
```

Verificar que el comando se ejecute desde el directorio “sert” anteriormente identificado. Luego identificar los dos esquemas creados:

SV_SERT_050000: es la base que contiene la base de objetos APEX-SERT y los metadatos.

SV_SERT_APEX: es el esquema que puede ser utilizado como esquema de análisis gramatical para las aplicaciones APEX.

Una vez identificados los esquemas creados, se procede con la instalación:




*Figura 8. Instalación del APEX-SERT*

Luego preguntará por las contraseñas de los esquemas, del usuario administrador, y el correo y dirección del administrador SERT. También preguntará por otros elementos, como el esquema para análisis gramatical.

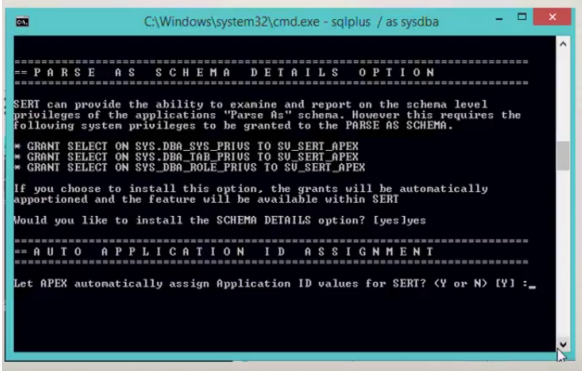



*Figura 9. Instalación del APEX-SERT*

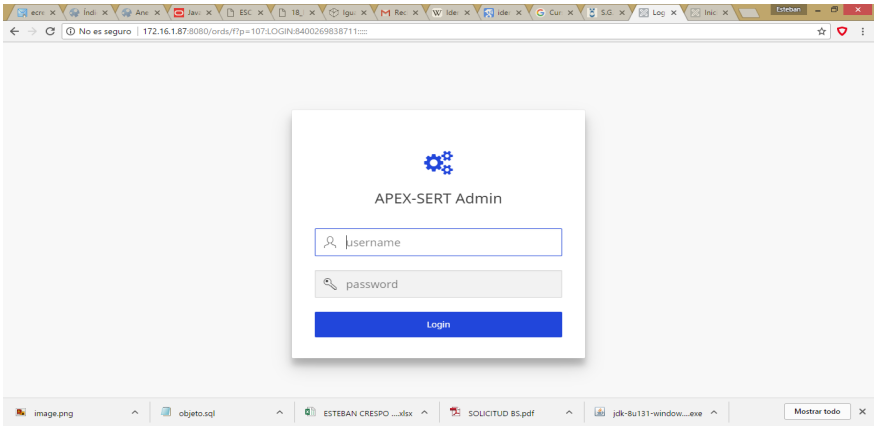



*Figura 10. Pantalla de inicio del administrador de APEX-SERT*

## 2.4. Fase 3: Obtención de información

Para las pruebas se han contemplado entonces cinco escenarios:

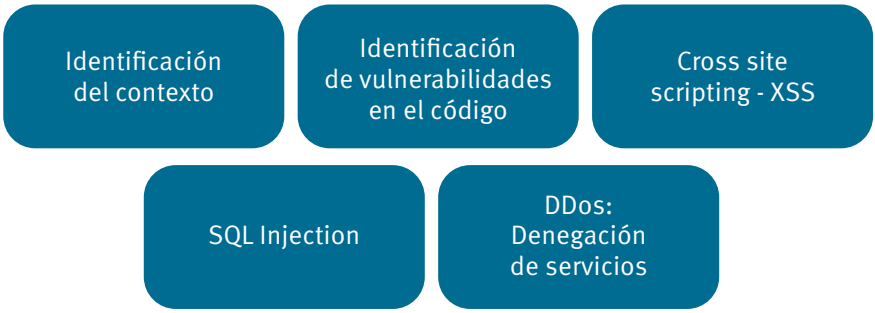



*Figura 11. Escenarios de pruebas de seguridad*

### 2.4.1. Identificación del contexto

La identificación del contexto permitirá identificar aspectos de vulnerabilidad que luego pueden ser aprovechados por otras técnicas. En este ámbito, se utilizarán dos técnicas: caja negra y caja blanca. Para ambos escenarios, las herramientas que se utilizarán son:

- NMAP
- Sparta

**NMAP** es una herramienta Open Source (de código abierto) utilizada en pruebas de penetración y auditoría de seguridad informática. El sitio web oficial del proyecto NMAP es http://www.nmap.org. La herramienta está disponible en el distro Kali.

La primera prueba será utilizando NMAP con los parámetros básicos: -A, que habilita la detección del sistema operativo y la versión, y la opción -T4, que permite acelerar el proceso para visualizar el nombre del objetivo. Esta prueba también permite enlistar los puertos abiertos y el servicio asociado a cada puerto.

```
nmap -T4 -A 172.16.1.87
```

Si se agrega la opción -v, se tendrá un mayor nivel de detalle en los resultados visualizados. Agregando -vv, el detalle será aún mayor, sin embargo, el tiempo de análisis también se incrementa.

```
nmap -T4 -A -vv 172.16.1.87
```

El siguiente análisis debe ser realizado en especificación de puertos, asignando un orden de análisis:

-p ‹rango de puertos›: sólo sondear los puertos indicados, y, -r: analizar los puertos secuencialmente, no al azar.

Ejm: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080

Realizar, en primera instancia, un análisis a profundidad del puerto 8080.

```
nmap -T4 -A 172.16.1.87 -p U:8080 -r
```

La siguiente fase del análisis consiste en detectar el servicio y la versión del mismo. Para ello ha sido importante considerar los siguientes parámetros:

-sV: sondear puertos abiertos, para obtener información de servicio/versión

--version-intensity ‹nivel›: fijar de 0 (ligero) a 9 (probar todas las sondas)

--version-light: limitar a las sondas más probables (intensidad 2)

--version-all: utilizar todas las sondas (intensidad 9)

--version-trace: presentar actividad detallada del análisis (para depurar) (NMAP, 2017)

```
nmap -sV -T4 -A 172.16.1.87 -p U:8080 -r
```

Luego se utiliza la opción --version-all para hacer un análisis a profundidad de los servicios y versiones expuestas en el servidor de alojamiento.

```
nmap -T4 -A 172.16.1.87 -p U:8080 -r -sV
--version-all
```

Además, será importante identificar el sistema operativo que ejecuta el servidor. Para ello se ejecutará el análisis considerando los siguientes parámetros:

-O: activar la detección de sistema operativo (SO)
--osscan-limit: limitar la detección de SO a objetivos prometedores
--osscan-guess: adivinar el SO de la forma más agresiva (NMAP, 2017)

```
nmap -T4 -A 172.16.1.87 -O --osscan-guess
```

Procedimiento para el caso de que el firewall o el IDS no permita ejecutar el análisis.

```
EVASIÓN Y FALSIFICACIÓN PARA CORTAFUEGOS/IDS:

  -f; --mtu <valor>: fragmentar paquetes (opc.
con el MTU indicado)

  -D <señuelo1,señuelo2[,ME],...>: disimular
el análisis con señuelos

     N. del T.: «ME» es «YO» mismo.

  -S <Dirección_IP>: falsificar la dirección
IP origen

  -e <interfaz>: utilizar la interfaz indicada

  -g/--source-port <numpuerto>: utilizar el
número de puerto dado

  --data-length <num>: agregar datos al azar
a los paquetes enviados

  --ttl <val>: fijar el valor del campo
time-to-live (TTL) de IP

  --spoof-mac <dirección mac/prefijo/nombre
de fabricante>: falsificar la dirección MAC

  --badsum: enviar paquetes con una suma de
comprobación TCP/UDP falsa
```

(NMAP, 2017)

Legion es una aplicación GUI elaborada en Pyton, que permite realizar pruebas de penetración a una infraestructura de red, tanto en la etapa de escaneo como en la de enumeración. Las instrucciones oficiales de instalación están en http://kali.org/tools/legion

Legion está disponible en la distribución Kali. En la pestaña Hosts, se define el equipo al cual queremos hacer el diagnóstico de seguridad. La herramienta permite especificar, incluso, un rango de equipos que serán analizados.

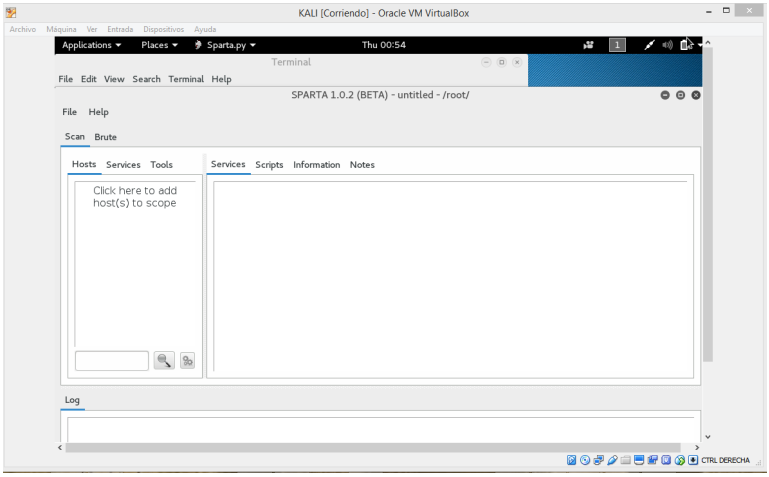



*Figura 12. Interfaz de Legion*

Para especificar un rango se debe utilizar una máscara; por ejemplo, 192.168.1.0/24. También se puede establecer un rango de análisis más corto utilizando: 192.168.1.1-10. Así analizará solo las 10 direcciones IP, a diferencia de la primera opción que analiza 254 objetivos.

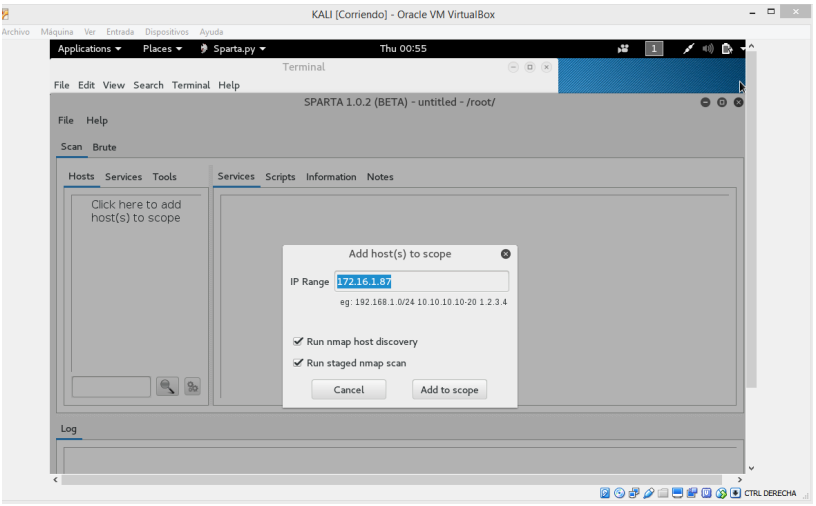



*Figura 13. Definición de un ámbito de análisis en Legion*

Como se indica en la figura anterior, Legion utiliza los servicios NMAP para descubrir los equipos de la red, y, por otro lado, hacer uso de los escenarios encontrados. Luego, agregamos el host que será analizado, haciendo clic en "Add to scope".

Una vez definido el objetivo o los objetivos que serán analizados, se deben especificar los filtros que se aplicarán en el análisis. En la opción de configuración (botón con dos ruedas dentadas), aparecerá el siguiente cuadro de diálogo:

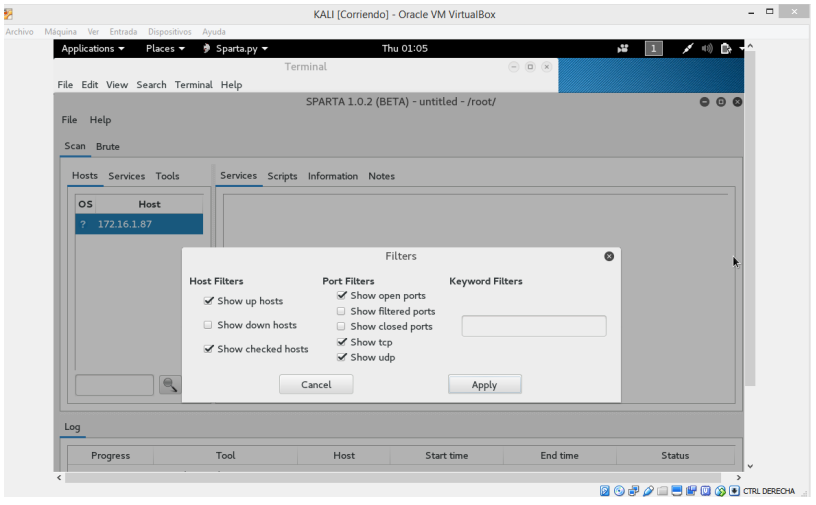



*Figura 14. Aplicación de filtros para identificación y análisis de puertos*

Los objetivos apagados o desconectados temporalmente, para este propósito, no serán considerados. Será importante hacer un análisis inicial de los puertos abiertos, tanto TCP como UDP, ya que muchos de ellos pueden presentar configuraciones de riesgo y además diversas vulnerabilidades. Se aplican los ajustes y se procede con el análisis.

Al dar un doble clic sobre el objetivo inicia el análisis. Ahora solo queda esperar hasta obtener los resultados.

Al dar un doble clic sobre el objetivo inicia el análisis. Ahora solo queda esperar hasta obtener los resultados.

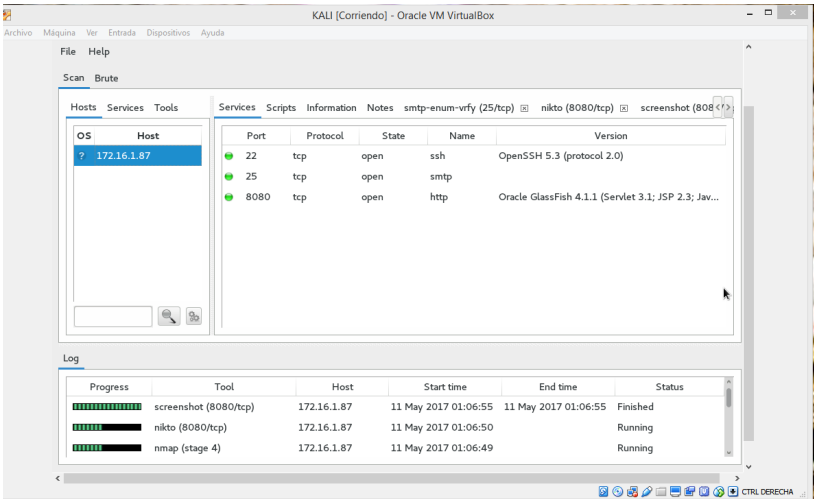


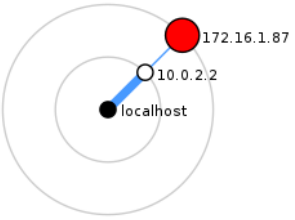
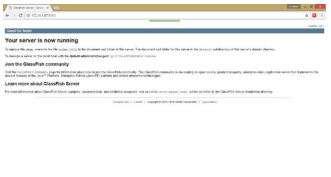

*Figura 15. Ejecución de la herramienta SPARTA*

A continuación, se presenta una tabla detallada con los resultados obtenidos en la etapa de reconocimiento y escaneo.

### 2.4.1.1. Resultados del reconocimiento

En la siguiente tabla se exponen los resultados adquiridos luego de la etapa de reconocimiento. Básicamente es un servidor virtual, que mantiene 7 puertos en escucha de los 15 encontrados como abiertos, dos puertos menos que los encontrados en el análisis de pentesting en la etapa de desarrollo del ERP.

*Tabla 1. Resultados del reconocimiento*

| Resultados | | |
|---|---|---|
| Herramienta Utilizada | NMAP | Sparta |
| Puertos encontrados | 7 puertos abiertos | 15 puertos abiertos, 7 filtrados. |
| Puertos abiertos | 22, 25, 1521, 4848, 7676, 8080, 8181 | 22, 25, 1521, 3700, 3820, 3920, 4848, 7676, 7776, 8080, 8181, 8686, 13335, 17210, 29573 |
| Sistema operativo | Virtualizado en Oracle Virtual Box | Oracle virtualbox |
| Topología |  | La herramienta no arroja resultados visuales para establecer la topología. |
| Captura de pantalla |  |  |

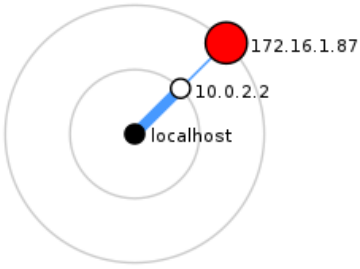
### 2.4.1.2.Resultados del escaneo

En los resultados del escaneo se pueden comprobar los resultados arrojados por ambas herramientas, en las que se puede ver una ventaja superior de Sparta frente a NMAP, debido obviamente a que Sparta utiliza diferentes motores de análisis, inclusive NMAP.

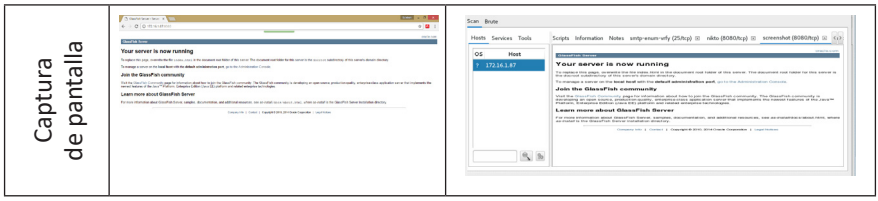
*Tabla 2. Resultados del escaneo*

| Resultados | | |
|---|---|---|
| Herramienta analizada | NMAP | Sparta |
| Puertos encontrados | 12 puertos abiertos | 23 puertos abiertos. |
| Puertos abiertos | 22, 25, 80, 1521, 4848, 5500, 5801, 5901, 6001, 7676, 8080, 8181 | 22, 25, 80, 1521, 1830, 3700, 4848, 5500, 5521, 5801, 5901, 6001, 7676, 7776, 8080, 8181, 8686, 9413, 24241, 26486, 31793, 33446, 36575 |

| | | |
|---|---|---|
| Servicios identificados | • 22/tcp, 5500/tcp open ssh versión: OpenSSH 5.3 (Protocol 2.0)<br>• 25/tcp open SMTP<br>• 1521/tcp open oracle-tns Oracle TNS Listener<br>• 4848/tcp open SSL/http Oracle GlassFish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7)<br>• 5801/5901 /tcp: puertos de escucha VNC<br>• 7676/tcp open java-message-service JAVA MESSAGE SERVICE 301<br>• 8080/tcp open http Oracle Glassfish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7)<br>• 8181/tcp open http Oracle Glassfish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7) | • 22/tcp open: Open SSH 5.3<br>• 25/tcp open: SMTP<br>• 80/tcp open: Apache 2<br>• 1521/tcp open Oracle-tns Oracle TNS Listener<br>• 1830/tcp open<br>• 3700/tcp open giop CORBA Naming service<br>• 4848/tcp open http Oracle GlassFish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7)<br>• 5500 /tcp open<br>• 5521 /tcp open<br>• 5801 /tcp open TigerVNC<br>• 5901 /tcp open VNC<br>• 6001 /tcp open<br>• 7676/tcp open Java Message Service 301<br>• 7776/tcp open<br>• 8080/tcp open http Oracle Glassfish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7)<br>• 8181/tcp open http Oracle Glassfish 4.1.1 (Servlet 3.1; JSP 2.3; Java 1.7)<br>• 8686/tcp open sun-as-jmx-rmi<br>• 9413/tcp open<br>• 24241 /tcp open JAVA RMI registry<br>• 26486/tcp open exoline TCP<br>• 31793/tcp open Oracle-tns Oracle TNS listener<br>• 33446/tcp open<br>• 36575/tcp open Oracle-tns Oracle TNS listener |
| Sistema Operativo | Virtualizado en Oracle Virtual Box | Oracle virtualbox |

| | | |
|---|---|---|
| Certificado | SSL-cert commonName=localhost/ organizationName=Oracle Corporation/ StateOrProvinceName = California /countryName = US<br>• Emisor: Local.<br>• Tipo de llave pública: RSA<br>• Algoritmo de firma: sha256 con cifrado RSA. | |
| Topología |  | |

| | | |
|---|---|---|
| Ataques SYN-ACK | Puerto 22: se obtiene el SSH-Hostkey de 1024 bits y llave DSA. SSH Hostkey es la llave criptográfica para la autenticación de equipos en el protocolo SSH.<br><br>Se obtiene además la llave de 2048 bits ssh-rsa<br><br>Puerto 4848: se obtiene el certificado público, cuyo algoritmo de firma es SHA256 con cifrado RSA. El certificado está vigente desde el 19 de septiembre de 2015 y expira el 19 de septiembre de 2025.<br><br>Puerto 8080<br><br>Métodos sustentados: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS<br><br>Métodos de riesgo potencial: PUT, DELETE<br><br>Puerto 8081<br><br>Métodos sustentados: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS<br><br>Métodos de riesgo potencial: PUT, DELETE | Puerto 8080: GlassFish Server Open Source 4.1.1<br><br>El servidor presenta posibilidades de fuga mediante ETags. La cabecera encontrada corresponde a 0xW/4626 0x1442682444000<br><br>La cabecera anti secuestro de clic (anti-clickjacking X-Frame-Options header) no está presente<br><br>Métodos sustentados: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS<br><br>HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE<br><br>Puerto 8181<br><br>La cabecera anti secuestro de clic (anti-clickjacking X-Frame-Options header) no está presente<br><br>Métodos sustentados: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS<br><br>HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE<br><br>El servidor presenta posibilidades de fuga mediante ETags. La cabecera encontrada corresponde al archivo README en la ubicación 172.16.1.87/icons<br><br>La cabecera de protección contra ataques Cross Site Scripting XSS no está definida.<br><br>El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida.<br><br>El servidor Apache es vulnerable a ataques XST (Cross Site Tracing<br><br>La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida. Esto podría permitir procesar el contenido del sitio de una manera diferente al tipo MIME.<br><br>Existe un directorio potencial de indexación que no debería estar presente: 172.16.1.87/icons/<br><br>El nombre de host '172.16.1.87' no concuerda con el nombre del certificado: Localhost |




## 2.4.2. Identificación de vulnerabilidades en el código fuente

En el mundo empresarial resulta difícil pensar en hacer pruebas de software hasta que este haya sido creado y se encuentre en su etapa de despliegue; por lo que es una práctica ineficaz y restrictiva debido a su costo. Una de las mejores alternativas para evitar problemas con los “bugs” o fallos que aparecen en el software de producción, es el de mejorar su ciclo de vida de desarrollo de software (SDLC), mejorando los niveles de seguridad en cada una de sus etapas.

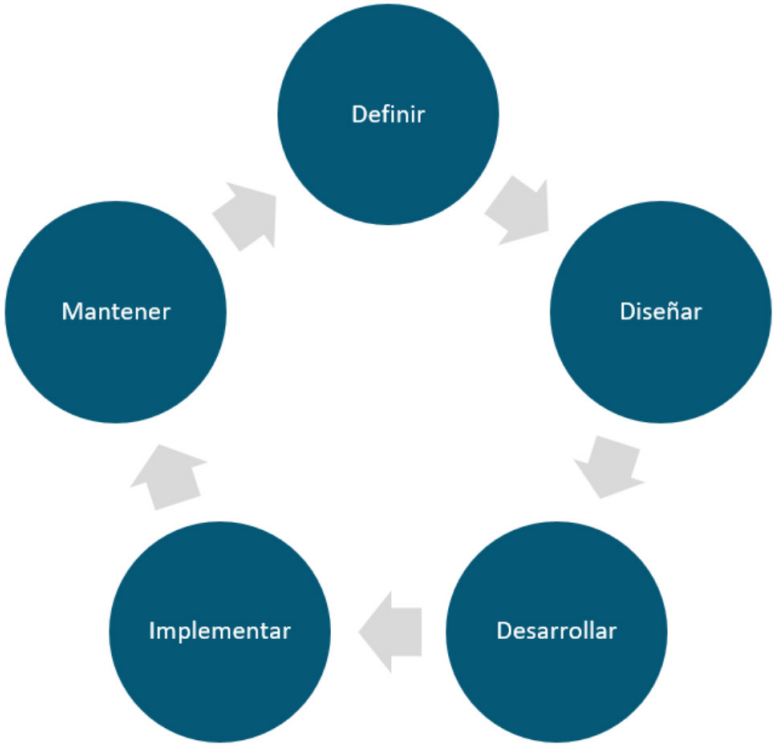



*Figura 16. Ciclo de vida de desarrollo de software. Fuente: (The OWASP Project, 2017)*

De acuerdo con el proyecto OWASP, las organizaciones, instituciones y empresas en general deberían inspeccionar a detalle cada una de las etapas del ciclo de vida de desarrollo de software, en las que se deben incluir pruebas para garantizar que la seguridad es adecuadamente cubierta por los controles utilizados a lo largo del proceso de desarrollo.

Un efectivo programa de pruebas debería incluir:

- **Gente:** garantizar una educación y sensibilización adecuadas.
- **Proceso:** garantizar la existencia de políticas y normas adecuadas.
- **Tecnología:** para asegurar que el proceso haya sido exitoso en su implementación.

La siguiente sección hace referencia a los pasos que deben ser contemplados en un proceso de validación de software, claro, desde el punto de vista de la seguridad.

a. Recopilación de información
b. Pruebas de gestión de la configuración
c. Pruebas de la lógica de negocio
d. Pruebas de autenticación
e. Pruebas de autorización
f. Pruebas de gestión de sesiones
g. Pruebas de validación de datos
h. Pruebas de denegación de servicio
i. Pruebas de servicios web
j. Pruebas de AJAX

Para la ejecución de estas pruebas se utilizará el sistema operativo Mantra Linux, producto del proyecto OWASP.

Mantra Linux es una distribución basada en XUBUNTU 14, que incluye herramientas de prueba específicamente para evaluar la seguridad del código fuente de aplicaciones y páginas web.

### 2.4.2.1.Recopilación de información

Para la recopilación de información, se hará primeramente un escaneo rápido utilizando Nikto, una herramienta que debe ser ejecutada desde el Shell de la distribución Mantra. Como ya se había evaluado previamente el objetivo del ataque, se sabía que el puerto 8080 era uno de los utilizados para escucha.

El comando que se va a ejecutar será:

```
nikto -host 172.16.1.87:8080 -C all
```

La herramienta arrojará las posibles vulnerabilidades de código fuente que tiene el objetivo. Con la opción -C all, se enlistarán todos los posibles directorios existentes, además del CGI-BIN.

La siguiente herramienta que se va a utilizar es OWASP Zed Attack Proxy, disponible en la distro Mantra.

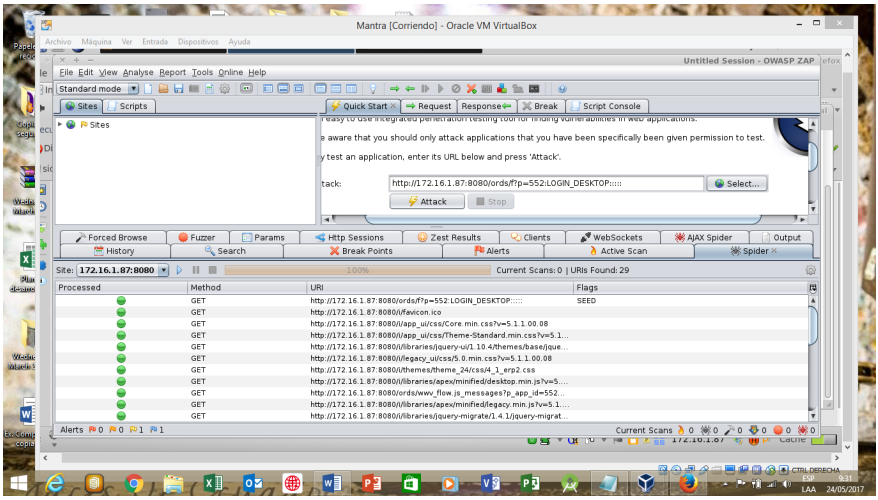



*Figura 17. Interfaz de la herramienta OWASP ZAP*

Para ello se ha indicado la URL que se va a analizar, la misma que ha sido ingresada en el campo URL Track de la pestaña Quick Start, y luego iniciar la exploración con Attack.

Las pestañas de resultados, en la parte inferior de la pantalla, permiten visualizar los resultados obtenidos en el análisis. Haciendo énfasis en los resultados, está la pestaña “Alerts”, que visualiza las alertas obtenidas en el ataque.

Active Scan visualiza los mensajes enviados al servidor, utilizando tanto métodos POST como GET, indica la URL y el camino analizado, el tiempo de ejecución y el tamaño de los paquetes enviados o recibidos. Los resultados obtenidos de este análisis serán tratados en la fase 4: Análisis de vulnerabilidades.

### 2.4.2.2. Pruebas de autenticación

No existe validación de usuario (tester) clave (tester). No se valida el número de cédula. La prueba 1212121212 fue aceptada. No maneja contraseñas fuertes.

Gibson Research Corporation (www.grc.com)

Para la validación de fortaleza de contraseña, según lo recomendado por (Spendolini, 2016), el uso de la herramienta https://www.grc.com/haystack.htm es importante, ya que visualiza el tiempo que le tomaría al atacante vulnerar la misma.

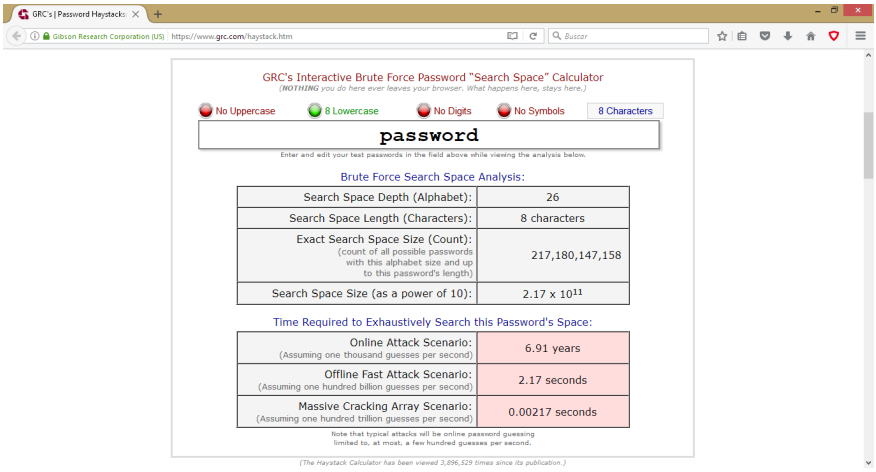



*Figura 18. Sitio WEB del proyecto GRC para cálculo de iteraciones en ataques de fuerza bruta*

Como se puede apreciar en la imagen, la herramienta visualiza el tiempo promedio que puede tomarle a una herramienta para vulnerar contraseñas, a través del mecanismo de ataque de fuerza bruta.

## W3AF

Para las pruebas de análisis de código ha sido importante el uso de la herramienta w3af. Para ello será importante ejecutar w3af, ya sea desde la consola de Kali o bien desde Mantra Linux, ejecutando el comando: root@kail:~# w3af

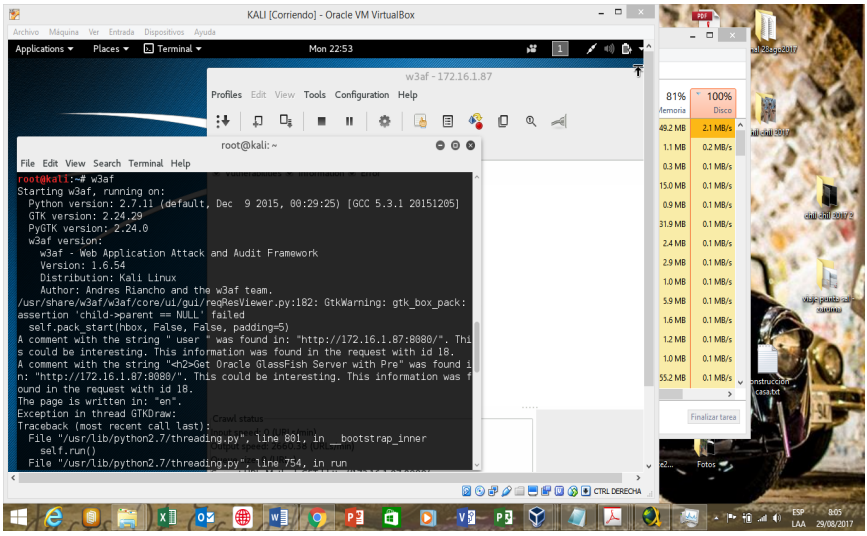



*Figura 19. Herramienta w3af para el análisis de vulnerabilidad en el código*

La herramienta w3af tiene una interfaz gráfica, sin embargo, los resultados de la exploración son visualizados en línea de comando. Una técnica interesante que adopta esta herramienta son las pruebas "fuzzing", que consisten en un proceso semiautomático que combina técnicas de prueba como ***son funcional, caja negra, exploratoria, seguridad, y resistencia*** para descubrir fallos de seguridad. Es utilizada por compañías de desarrollo de software y proyectos de código abierto para descubrir vulnerabilidades y mejorar la calidad de su software (Hernández, 2007).

Hernández (2007) menciona además que entre el gran número de entes que pueden ser analizados mediante esta técnica se encuentran:

- Formatos de archivo
- Entradas de datos estándar
- Protocolos de red (capa de aplicación)
- Pilas de protocolos
- Argumentos
- Variables de entorno
- Controles Active X

Entre los resultados que ha arrojado la herramienta se registran las siguientes vulnerabilidades:

- Comentario con la cadena “user” encontrada en **http://172.16.1.87:8080**
- Comentario con la cadena “Oracle Glass Fish” encontrada en **http://172.16.1.87:8080**
- Un punto de inyección encontrado e*n* **http://172.16.1.87:8080/copyright** referida desde **http://172.16.1.87:8080**
- La lista de las solicitudes fuzzables es:
    - Método GET: **http://172.16.1.87:8080**
- La siguiente lista de destinos no tiene protección contra ataques de secuestro de clic (Click Jacking) en la cabecera “X-Frame-Options Header”*:*
    - **http://172.16.1.87:8080/ords/f**
    - **http://172.16.1.87:8080/ords**
    - **http://172.16.1.87:8080/ords/wwv_flow.js_messages**

## WAPITI

Además, se han realizado pruebas con Wapiti, una herramienta para análisis de sitios web, que incluye la distribución Kali.

Wapiti ha sido ejecutado con los siguientes parámetros:

```
oot@kail:~# wapiti http://172.16.1.87:8080/ords -k -v 2 --color red -m
“-all,xss:get,exec:post”
```

Las opciones utilizadas son las siguientes:

-k: realizará el ataque desde el último punto analizado, en caso de que la herramienta ya haya sido ejecutada con anterioridad.

-v 2: visualiza en pantalla la ejecución de cada ataque realizado.

-- Color red: visualizará en color rojo cualquier vulnerabilidad encontrada.

-m “-all,xss:get,exec:post”: realiza el ataque utilizando cabeceras http para ataques Cross Site Scripting con el método GET, y ejecuta el ataque en el sitio aplicando el método post.

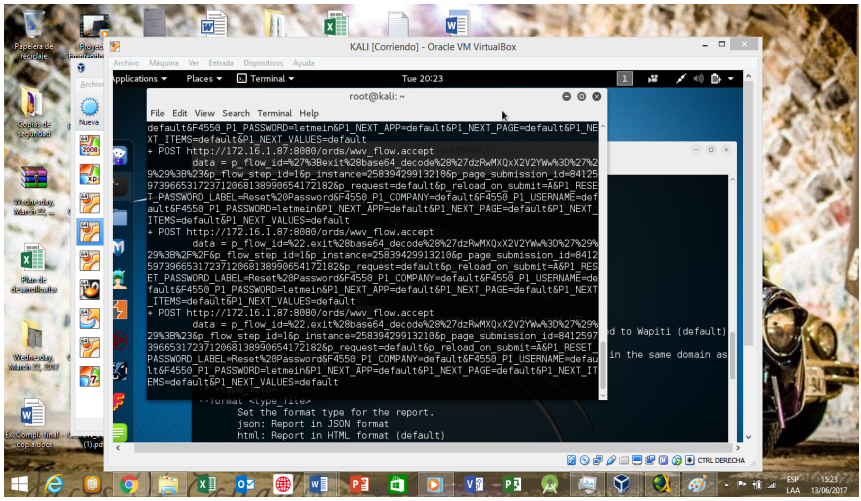



*Figura 20. Pruebas con Wapiti*

### 2.4.2.3. Ataque de fuerza bruta

Para esta prueba se utilizarán las herramientas Hydra y Crunch. Hydra que consiste en un software que realiza un sinnúmero de intentos de intrusión basada en un ataque de fuerza bruta, esto es, utilizando usuarios generalmente utilizados (admin, administrador, administrator) y una tabla de contraseñas generadas en base a un patrón. Para ello se han realizado los siguientes pasos, considerando obviamente que un ataque de este tipo puede tomar mucho tiempo, esto es, desde varias horas hasta varias semanas, o inclusive, varios años; dependiendo del tamaño de diccionario que se utilizará, y del grado de complejidad de la contraseña.

1. Crear un diccionario de ataque:
    a. nano diccionario_usuarios
    b. introducir algunos posibles usuarios
    c. grabar el archivo de diccionario.
2. Con la herramienta Crunch generar un diccionario de contraseñas basadas en aaaaaa11.
    a. Crunch 6 6 abcdefghijklmnopqrstuvwxyz1234567890 › diccionario_password
    b. El comando anterior ha generado 2.176’782.336 entradas de contraseñas aleatorias, basadas en todas las posibles combinaciones especificadas. Esto significa un archivo de 14GB en el que se encuentran las posibles contraseñas generadas de forma aleatoria en base al patrón señalado. Esto excluirá caracteres especiales puesto que no están contemplados en el patrón definido.

3. Se verifica que el puerto 22 se encuentre abierto en el equipo objetivo.
    a. Nmap 172.16.1.87 -n -v
    b. Hydra 172.16.1.87 ssh -s 22 -L diccionario_usuarios -P diccionario_password -f -vV
4. Hydra realizará 16600 intentos por cada tarea. Dependiendo de la longitud de la contraseña, la tarea puede tomar algunos minutos o inclusive semanas.
5. De lo evaluado, el firewall no bloquea intentos de conexión repetitivos, así como el equipo no bloquea los intentos de conexión fallidos.

### 2.4.3. Cross Site Scripting – XSS

Las pruebas Cross Site Scripting o XSS sirven para analizar aspectos que pueden ser utilizados por los atacantes para introducir un código malicioso en una aplicación web, generalmente en un formulario de un script del lado del navegador, para que llegue a un diferente usuario final. A pesar de que se trata de una de las vulnerabilidades más antiguas (tan antiguas como desde la creación de los navegadores), es también una de las más efectivas, además de las que han presentado una evolución interesante. Este tipo de ataques pueden presentarse de la siguiente manera:

- XSS Reflected: es un tipo de ataque que no se ejecuta conjuntamente con la aplicación web. Así, por ejemplo, cuando la página de un sitio se carga, en el contexto de este ataque únicamente se alcanza hasta la petición dada por un cliente. Normalmente estos ataques se asocian con el hurto de cookies, secuestro de sesiones, acceso al historial de navegación del objetivo e inclusive el ingreso al almacén de contraseñas que mantienen los navegadores.

    - Este tipo de ataque, al igual que muchas de las técnicas asociadas con XSS están directamente relacionadas con validaciones de los parámetros de entrada que son escuchados por una aplicación web. Por otro lado, los filtros y validaciones de los datos que pueden ser ingresados por un usuario en una aplicación son, por lo general, la razón principal para que se produzcan estas vulnerabilidades.

- XSS Stored: en este tipo de ataques, el objetivo no es únicamente el navegador del usuario, sino que puede afectar a todos los usuarios que ingresen a la aplicación. Es similar al anterior, pero mucho más peligroso, y, por lo tanto, más apetecido por el atacante. Ejemplos de este ataque se pueden ver en los foros o sitios donde se pueden redactar comentarios, es decir, campos de texto donde se pueden introducir etiquetas HTML o JavaScript.

- DOM XSS: el ataque DOM XSS se basa en los principios de los dos tipos de ataque anteriores, con la diferencia de que se aprovecha de la API DOM que tienen los navegadores para acceder a diferentes objetos que forman parte de él, como son las funciones en JavaScript. Así, se pueden manipular eventos, navegación y otras características que son ejecutadas en el lado del cliente.

- Flash XSS: consiste en explotar las etiquetas ‹object›, mediante el uso de ActionScript para acceder a las variables de videos o animaciones hechos en Flash. Además, se pueden utilizar funciones propias de flash como son getURL o loadMovie para conseguir tal efecto.

- CSRF: que es un ataque que funciona en tres pasos, donde i) se crea un script malicioso hospedado en un servidor web, controlado por el atacante; ii) el usuario víctima inicia sesión en una página web y mantiene su sesión activa; y iii) el atacante envía un script malicioso, utilizando generalmente la ingeniería social (mensajes, banners, anuncios son los más atractivos). Una vez que el usuario haya ejecutado el script malicioso, el atacante tendrá acceso a cookies y otros objetos de interés propios de los navegadores. Aquí, la actividad clave del ataque es la suplantación de identidad.

- XFS: consiste en la inyección de un código malicioso mediante la inclusión de frames para cargar código externo sin consentimiento de la víctima. El éxito está en la forma de manipular las variables que viajan a petición.

- XAS (Cross Agent Scripting): consiste en la técnica de inyección de un código en aplicaciones Web mediante la modificación de las cabeceras HTTP, en este caso, alterando el parámetro “User-Agent” del navegador web del atacante e instaurando cómo valor el código que se plantea inyectar.

- XRS (Cross Refer Script): funciona de manera muy similar al XAS, pero cambiando el valor de la cabecera Referer, con lo que se podrá posteriormente inyectar un código malicioso.

Para realizar estas pruebas de ataque XSS, se utilizará Vega. Vega es un analizador de vulnerabilidades gratuito, de código abierto, utilizado en pruebas de seguridad para plataformas y aplicaciones web.

Puede usarse no solo en pruebas XSS, sino también en análisis de vulnerabilidades que pueden ser aprovechadas por técnicas de inyección SQL, inyección Shell, entre otras.

La preparación del laboratorio para esta herramienta es simple. Se requiere instalar la herramienta y ejecutarla con privilegios administrativos. Se debe comprobar la comunicación con el objetivo e ingresar el destino de análisis, tal como lo enseña la figura a continuación:

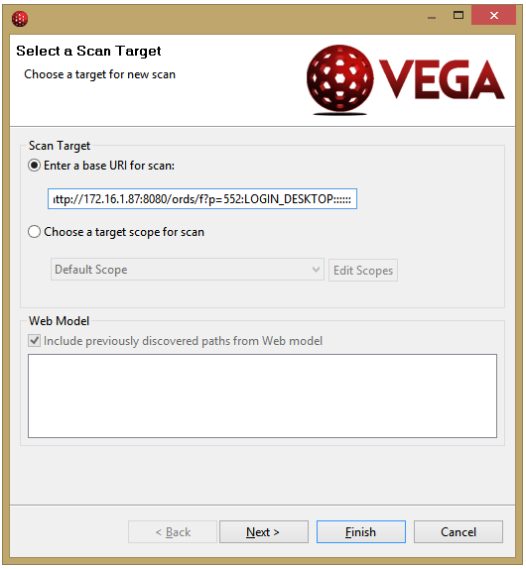



*Figura 21. Interfaz de la herramienta de análisis automatizada VEGA*

Una vez completada esta información, se deberá seleccionar los módulos de inyección que se utilizarán. Para este caso se utilizará “XSS Injection checks”. Sin embargo, se agregarán pruebas como “HTTP Trace Probes”, “HTTP Header injection checks” y “URL Injection checks”. Además, se hará una prueba XML para detectar servicios web disponibles.

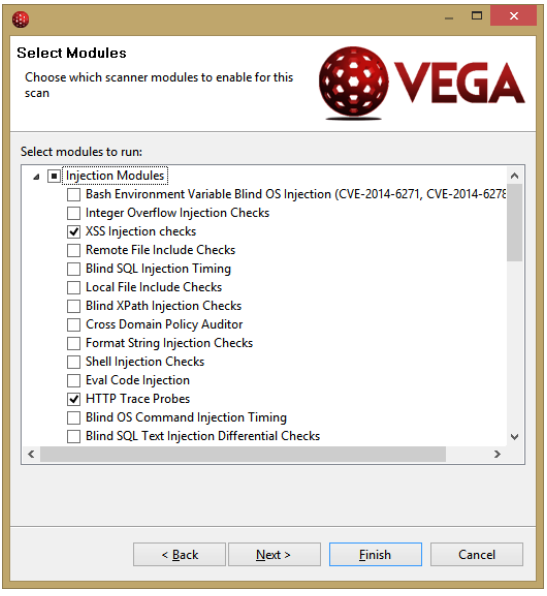



*Figura 22. Selección de los módulos de inyección en VEGA*

Por otro lado, será importante configurar los módulos de procesamiento de respuestas. Para ello se seleccionará:

- "HTTP Authentication Over Unencrypted HTTP"; esto debido a que el objetivo, tal como se revisó en la etapa de escaneo, no utiliza protocolos de cifrado.
- HTTP Header Checks, ya que será importante analizar las cabeceras HTTP a fin de determinar posibles vulnerabilidades.
- AJAX Detector; ya que existe una posibilidad de que una aplicación web utilice este tipo de contenido.
- E-Mail Finder; debido a que en el análisis se obtuvo como uno de los resultados, el uso del puerto 25, asociado al servicio de envío de correo electrónico.
- Internal IP; ya que será interesante conocer direcciones IP asociadas a la dirección IP expuesta.
- Credit Card Identification; ya que es necesario conocer si el software cuenta con algoritmos asociados a la identificación de una tarjeta de crédito.
- Insecure Script Include
- Interesting Metatag detection
- Source code disclosure module
- Character set not specified
- Cookie Security module
- Oracle Application Server Fingerprint Module
- Error Page Detection
- Unsafe or Unrecognized Character Set
- Form autocomplete
- Cookie Scope Detection
- Cleartext password over HTTP; esta prueba se la realiza cuando se tiene un objetivo que no utiliza SSL en el protocolo HTTP, y que generalmente lleva al transporte de usuarios y contraseñas sin cifrado.
- Empty Response Body Module
- WSDL, debido a que se probará la presencia de un web service.
- X-Frame Options Header Not Set
- Version control String Detection
- Insecure Cross-Domain Policy
- File Upload Detection, debido a que se encontraron vulnerabilidades en el servidor, el cual aceptaba métodos POST.
- Directory Listing Detection; pues será importante comprobar si el servidor arroja resultados de un listado de directorios.

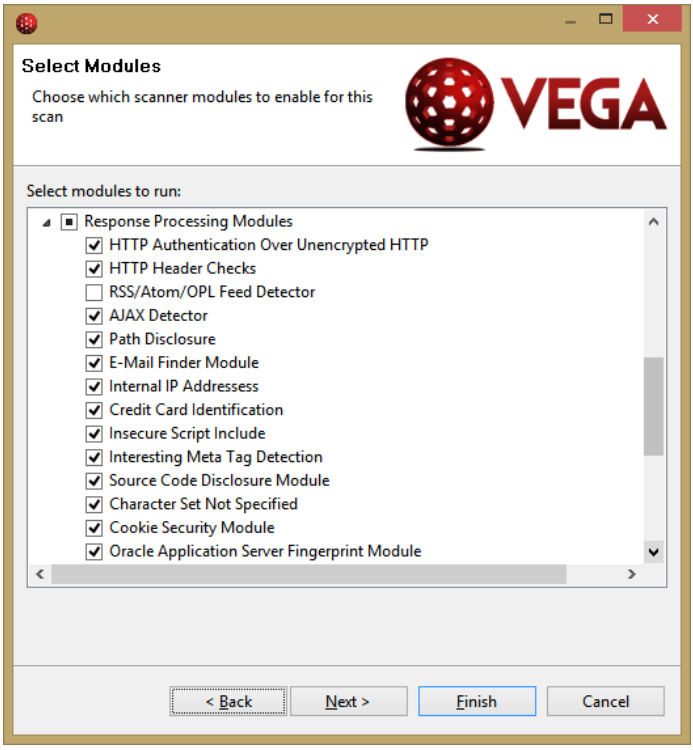



*Figura 23. Selección de los módulos de seguridad que serán analizados en VEGA*

En la primera prueba se utilizarán los siguientes parámetros por omisión, es decir, no se considerarán cookies que puedan ser descargadas desde el sitio que será analizado.

### 2.4.4. SQL Injection

SQL Injection es una técnica de inyección de código, utilizada para extraer información o dañar una base de datos. Además, es una de las técnicas de evasión de seguridad más utilizadas.

Existen múltiples formas de hacer este tipo de inyecciones de código. Entre ellas se pueden nombrar las siguientes:

- SQL en páginas web: utilizada cuando existen campos que requieren del ingreso de información del usuario, tales como el nombre de usuario (username) o el id de usuario (userid); pero en lugar de que ingrese esos datos, el atacante introducirá una sentencia SQL, como, por ejemplo
    - `txtUserId = getRequestString("UserId");`
    - `txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;`
- SQL basado en 1=1 es verdadero: si no hay nada que impida al usuario ingresar algo incorrecto, el atacante podrá introducir algo parecido a:
    - pepito OR 1=1. Así, en la aplicación, la sentencia SQL que se forma es: `SELECT * FROM Users WHERE UserId = pepito OR 1=1;`

  Así, retornarán todos los registros de la tabla "Users", debido a que 1=1 siempre será verdadero. Y es más peligroso aún si la tabla contiene usuarios y contraseñas.

- SQL basado en sentencias SQL agrupadas: debido a que muchas bases de datos admiten sentencias agrupadas, se puede enviar una sentencia SQL que liste todos los usuarios de una tabla, y que al mismo tiempo elimine otra. Por ejemplo:
    - User ID: *pepito; DROP TABLE Suppliers*
    - El resultado será: `SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;`

Utilizando Vega se realizarán los pasos similares a los utilizados en la prueba de inyección XSS, salvo las siguientes consideraciones en la configuración:

- Los módulos de inyección que se utilizarán son:
    - Blind SQL Injection Timing
    - Blind SQL Text Injection Differential Checks
    - Blind SQL Injection Arithmetic Evaluation Differential Checks

La segunda prueba que se va a realizar será con SQLMAP. Para ello será importante abrir la consola SQLMAP en la suite Kali. Luego se enlistarán los

gestores de bases de datos utilizando una inyección SQL. El siguiente comando habilita este análisis:

- *sqlmap – u http://172.16.1.87:8080/ords/f?p=552:LOGIN_DESKTOP:::::: --dbs*

La opción dbs enumera los gestores de bases de datos; -u hace referencia al objetivo URL que se pretende analizar.

En la ejecución de este comando se pudo obtener como resultado que, todos los parámetros probados aparentemente no son inyectables. En base a esto, se incrementa el nivel de agresividad a la prueba de intrusión. Se propone por lo tanto utilizar un nivel más avanzado, para lo cual se incluye el parámetro --level con el valor 3. Esta prueba, considerando los parámetros establecidos anteriormente, puede tomar hasta 20 minutos en ejecutarse.

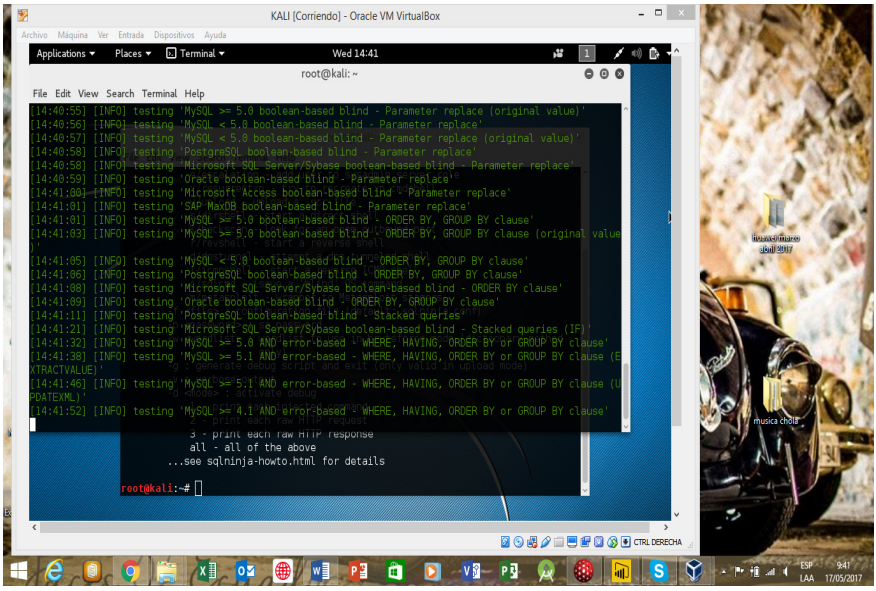



*Figura 24. Ejecución de la herramienta SQLMAP*

Luego de un ataque es necesario que, para una segunda prueba, se establezca el parámetro --technique=BEUS. Esto dará un tiempo al servidor para que se recupere del último ataque.

Ahora, con el mismo comando, pero con los parámetros establecidos anteriormente, se hará el análisis a cada uno de los campos. Esto es, al momento de proceder, en lugar de usar (C)ontinue, se lo hará utilizando (s)tring.

El comando que se va a ejecutar se escribirá de la siguiente manera:

```
sqlmap-uhttp://172.16.1.87:8080/ords/f?p=552:LOG-
IN_DESKTOP:::::: --dbs --level 3 --technique=BEUS
```

Los casos aplicados con el método *string han sido aplicados a los campos: usuario, contraseña, empresa. Los otros campos son omitidos debido a que son listas desplegables. Aceptar y cancelar hacen referencia a botones, por lo tanto, no serán comprobados en esta sección.*

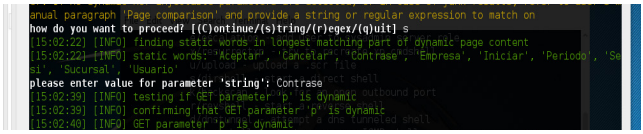



*Figura 25. Introducción de parámetros en la herramienta SQLMAP*

En la prueba anterior la herramienta notificó que el parámetro User-Agent no es inyectable. Por lo tanto, se sube a un nivel adicional que consiste en hacer la prueba con agentes aleatorios. De esta manera, se agrega la opción --random-agent al análisis.

### 2.4.5. DDoS – Denegación de Servicios

El ataque de denegación de servicios, conocido también como DoS (Denial of Service), es un tipo de ataque a una red de computadoras o a un sistema informático con el fin de afectar su disponibilidad a los usuarios. Generalmente se produce cuando una gran cantidad de conexiones o peticiones sobrecargan la capacidad de respuesta del objetivo, produciendo así su cierre inesperado.

Existen varios tipos de ataque del tipo DoS, entre ellos, se pueden citar los siguientes:

- UDP Flood (Saturación UDP): ataque enfocado al protocolo UDP, debido a que este protocolo no requiere mantener una sesión iniciada en el equipo remoto. El ataque entonces inunda los puertos de forma aleatoria al objetivo remoto con múltiples paquetes UDP. El equipo atacado tratará de resolver las peticiones de cada puerto, verificando las aplicaciones o servicios que lo han abierto, y como encuentra que no existe tal, tratará de enviar peticiones ICMP de respuesta, agotando de esta manera los recursos del servidor y dejándolo inaccesible.

- ICMP Flood (Saturación por Ping): es similar al ataque UDP Flood, pero saturando al objetivo mediante peticiones ICMP. Este tipo de ataques puede ser bloqueado a nivel de routers y firewalls cuando se incorporan listas de control de acceso (ACL).

- Service Port Flood (Ataque sobre Puertos de Servicio): en este tipo de ataques la inundación se la hace a los puertos más comunes, como es el 25 (SMTP) o el 80 (HTTP). Al ser puertos comunes, el

ataque es más difícil de afrontarlo. Sin embargo, los analizadores de tráfico (IDS) son muy útiles como contramedidas.

- HTTP Flood (Saturación HTTP): la saturación o inundación HTTP se consigue mediante la simulación de peticiones GET o POST válidas, aplicadas al servidor o equipo objetivo. El propósito es introducir un código maligno como Botnets, para convertir el equipo objetivo en un zombie, desde el cual se efectuarían ataques dirigidos a otros equipos.

- SYN Flood: por la forma en la que está concebido TCP/IP, el ataque SYN FLOOD es uno de los más comunes. La forma cómo funciona una conexión TCP es la siguiente: cuando un extremo desea iniciar una conexión hacia otro equipo, inicia la conversación con un mensaje 'SYN', el otro extremo ve este mensaje enviado y responde con un SYN+ACK. Finalmente, el extremo que empezó la conexión contesta con un ACK y de esta manera inicia la transmisión de los datos. El ataque de tipo Syn Flood acciona un número elevado de inicios de conexión que nunca son finalizados, dejando al servidor a la espera del ACK final, de esta manera, se consumen recursos de forma inesperada y el usuario ya no puede acceder a la información.

- Slowloris: este ataque envía únicamente las cabeceras de las peticiones al equipo objetivo. Al no ser completo, el equipo de destino estará siempre esperando completar el mensaje, lo que producirá un consumo de los recursos de la víctima y denegando el acceso a las peticiones legítimas.

- Ping of Death (Ping 'de la Muerte'): un requerimiento ICMP normalmente es de 32 bytes incluida la cabecera, por cuanto el equipo objetivo no tiene problema en responder la solicitud. Sin embargo, si se aumenta el tamaño del mensaje considerablemente (a 65535 bytes), y a una mayor frecuencia de lo normal (el ping, por defecto hace 4 peticiones), incluyendo varios equipos que ataquen de forma simultánea (por ejemplo, con equipos infectados con botnets) el equipo destino no podrá resolver todas las peticiones haciendo que la capacidad de respuesta se sature.

- NTP Amplification (Amplificación NTP): un ataque de amplificación NTP es una forma de ataques distribuidos de denegación de servicio (DDoS) que se basa en la utilización de servidores NTP públicamente accesibles para abrumar a un sistema de la víctima con el tráfico UDP.

- Blended Flood (Ataque combinado): este tipo de ataques combina varias de las técnicas mencionadas anteriormente. Debido a su alto grado de complejidad, muchas tecnologías de equipos cortafuegos o identificadores de intrusos no son capaces de detenerlos.

- Zero-day DDoS Attack (Ataques 'Día Cero'): los ataques de día cero se aprovechan de las vulnerabilidades de los equipos para las cuales aún no existen actualizaciones de seguridad. Como ejemplo de tipo de ataque puede relacionarse al producido al final de la primera quincena de mayo del 2017, con el ransomware WannaCry.

Para la realización de estas pruebas se plantea el uso de dos herramientas informáticas, disponibles en la distro Kali: HPing3 y Metasploitable.

HPing3, que es una herramienta de uso libre distribuida bajo un esquema de licenciamiento GNU, y sirve para generar paquetes a discreción, es decir, crear y analizar paquetes TCP/IP, esto es, ICMP, TCP, UDP y RAW-IP. Su uso, para este laboratorio, se enfoca a pruebas de inundación TCP/SYN (SYN FLOOD ATTACK).

El primer uso de hping3 será para comprobar la disponibilidad del equipo atacante. Como se puede ver en la figura a continuación, se envían paquetes de 46 bits de longitud que, sumados la cabecera y bits de verificación, suman 54 bits.

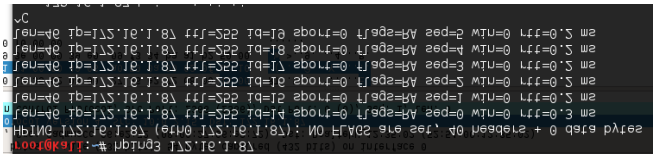


*Figura 26. Ejecución de una prueba ICMP utilizando hping3*

Para la validación de lo señalado anteriormente se puede utilizar un analizador de tráfico como Wireshark.

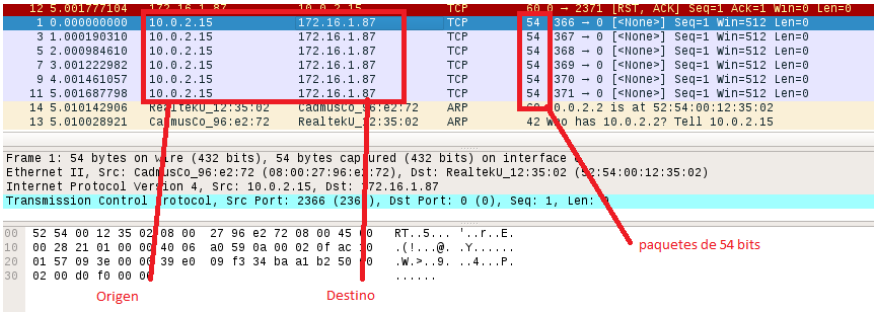



*Figura 27. Captura de paquetes ICMP enviados por hping3, con la herramienta Wireshark*

Como se pudo ver en la figura 27, se confirma el envío de paquetes de 54 bits.

### 2.4.5.1. IP SPOOFING

Para la prueba del IP Spoofing, primero se suplantará la identidad del atacante de manera que no sea identificado por el firewall o el identificador de intrusos. Para ello se utilizará el siguiente comando:

```
hping3 -a 2.3.3.1 172.16.1.87
```

donde 2.3.3.1 es la dirección de camuflaje que se ha asignado al equipo atacante.

Tal como se observa en la figura a continuación, el analizador de tráfico afirma el ataque con la identidad del atacante, variada. Sin embargo, ha sido bloqueada la conexión debido a que el ataque fue fácil de interceptar, ya que son muchas peticiones a un puerto específico. Lo siguiente será generar peticiones aleatorias para intentar sobrepasar el firewall que protege la aplicación.

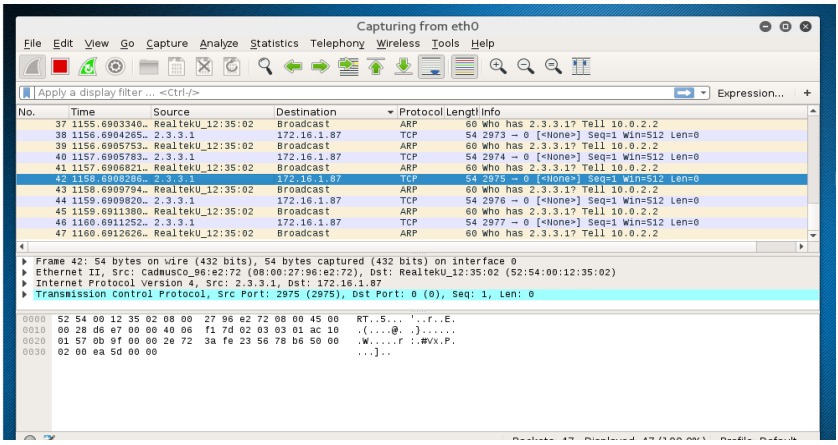



*Figura 28. Captura de tráfico al resultado del comando hping3 -a 2.3.3.1 172.16.1.87*

Para burlar de cierta forma al firewall, se ejecutará el ataque con el siguiente comando:

hping3 --rand-source 172.16.1.87 -p 8080

De esta forma enviará peticiones desde direcciones de origen, aleatorias, que terminarán en el puerto 8080 del equipo objetivo.

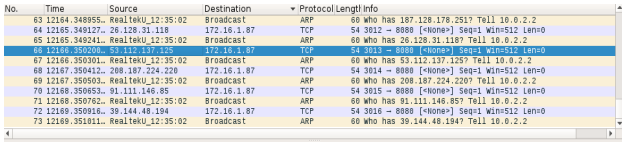



*Figura 29. Comprobación del tráfico resultante a la ejecución del comando hping3 --rand-source 172.16.1.87 -p 8080*

Como se pudo ver en la gráfica anterior, el origen del ataque ahora se ha convertido en direcciones aleatorias. De esta manera, el firewall podría asumir que se tratan de peticiones normales desde varias direcciones.

Finalmente, se procede con el comando para hacer la denegación de servicios por inundación IP (IP flooding):

```
hping3 --rand-source -d 600 172.16.1.87 -p
8080 --flood
```

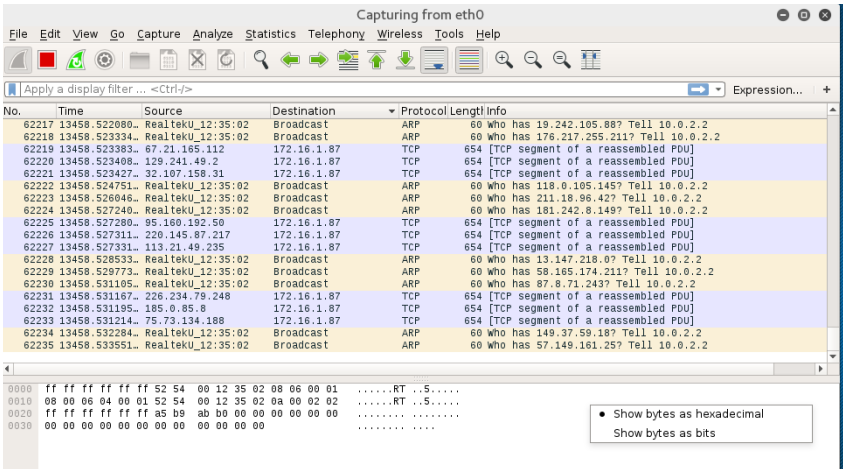



*Figura 30. Comprobación del tráfico resultante a la ejecución del comando hping3 --rand-source -d 600 172.16.1.87 -p 8080 --flood*

Luego se comprobó que el equipo quedó fuera de servicio, aspecto que puede ser verificado haciendo nuevamente un ping al servidor. La figura a continuación refleja lo mencionado anteriormente.

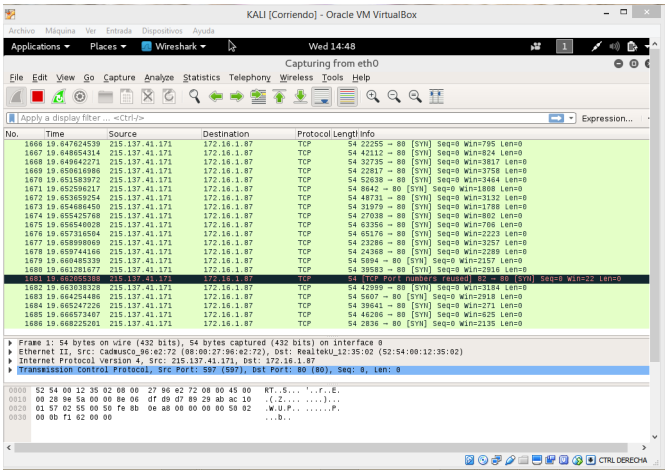



*Figura 31. Resultado de la denegación por servicio por inundación de paquetes*

### 2.4.5.2. SYN Flood

Para el ataque SYN Flood será necesario el uso del Metasploit Exploit Framework del toolkit Kali.

Los pasos realizados son los siguientes:

a. Escribir msfconsole en línea de comandos.

b. Una vez iniciado el Metasploit Framework, escribir:

c. use /auxiliary/dos/tcp/synflood

d. Luego se procede a configurar el objetivo de ataque. Para ello se pueden ver las opciones disponibles con el comando: show options.

e. El objetivo, en este caso es: set rhost 172.16.1.87

f. Configurar el puerto al que se atacará: set rport 8080

g. Se probará el ataque con una réplica de envío de datos cada 500 ms, que es el tiempo establecido por omisión por la herramienta.

h. Finalmente, se escribe “exploit” en la línea de comandos para iniciar el ataque.

Utilizando el sniffer se puede comprobar el ataque tal como lo muestra la figura a continuación:

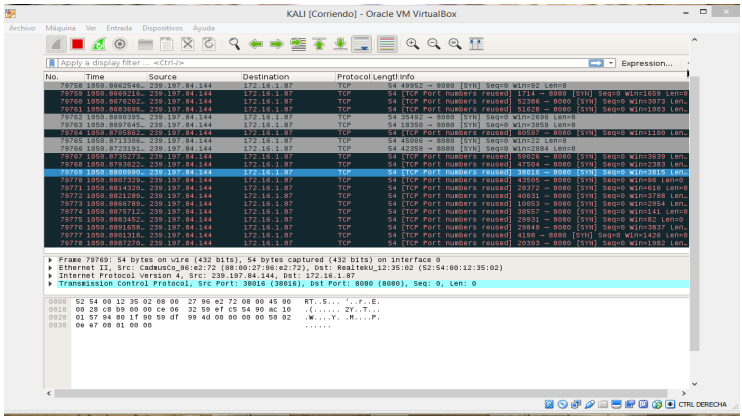



*Figura 32. Captura del tráfico generado por ataque SYNFlood*

El servidor aún sigue respondiendo. Se hará la prueba reduciendo el tiempo de envío entre cada paquete, y aumentando el tamaño del paquete a 65535 bytes.

Para ello:

1. Set timeout 100
2. Set snaplen 65535
3. Exploit

El ataque puede ser observado en el capturador de paquetes:

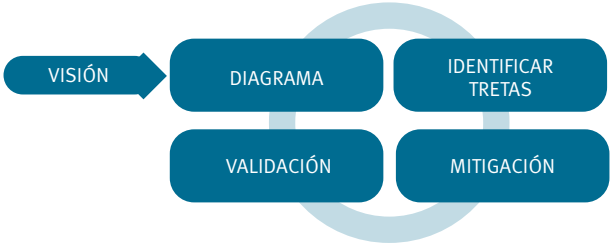



*Figura 33. Captura del tráfico generado por ataque SYNFlood con un tamaño de paquetes aumentado.*

La siguiente prueba es reducir aún más el tiempo de envío de paquetes:

Para ello:

1. Set timeout 30
2. exploit

El resultado:

## No se puede acceder a este sitio web

**172.16.1.87** ha tardado demasiado tiempo en responder.

- Busca 172 8080 ords en Google

ERR_CONNECTION_TIMED_OUT

Lo cual afirma que el servidor no está preparado para responder a ataques de tipo SYN Flood.

## 2.5. Fase 4 y 5: Modelamiento de amenazas y análisis de vulnerabilidades

Según The OWASP Project (2017), el modelamiento de amenazas es una representación estructurada de toda la información que afecta a la seguridad de una aplicación, y que se ha vuelto bastante popular en los últimos años. Menciona que Microsoft ha publicado un libro sobre su proceso y que incluye a este proceso como una actividad clave en su ciclo de vida de desarrollo seguro (S-SLDC).

El modelamiento de amenazas habilita a cualquier desarrollador o arquitecto de software a:

- Comunicar acerca del diseño de seguridad de sus sistemas.
- Analizar aquellos diseños para situaciones potenciales de seguridad usando la metodología provista.
- Sugerir y gestionar elementos de mitigación para situaciones de seguridad.

El modelo puede estar representado de la siguiente manera:

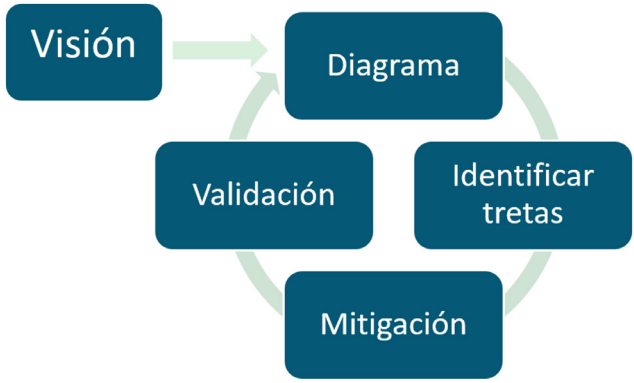



*Figura 34. El proceso de modelamiento de amenazas. Fuente: (Microsoft, 2017)*

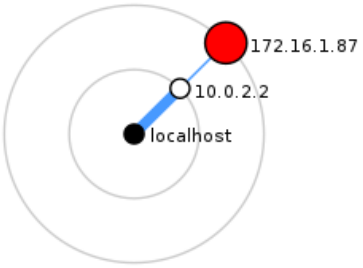
Microsoft (2017), en su análisis de modelo de amenazas sugiere realizar 7 pasos, los mismos que son descritos en la siguiente sección.

### 2.5.1. Paso 1. Recopilar información básica

En este caso se identificarán los escenarios de casos de uso. Será importante obtener un diagrama de flujo de datos para cada escenario. Para esta situación, el caso de uso será el acceso del usuario desde la web. También sugiere determinar los límites y el ámbito del entorno de destino, comprender los límites entre los componentes que son de confianza y los que no lo son, y además comprender el modelo de administración y configuración de cada componente. Deben considerarse además las dependencias externas.

Para las pruebas de seguridad realizadas los siguientes elementos encontrados pueden ser considerados como información básica:

*Tabla 3. Información básica*

| | |
|---|---|
| Puertos encontrados | 17 puertos abiertos, 7 filtrados. |
| Puertos abiertos | 22, 25, 1521, 3700, 3820, 3920, 4848, 7676, 7776, 8080, 8181, 8686, 13335, 17210, 29573 |
| Sistema operativo | Virtualizado en Oracle Virtual Box |
| Topología |  |
| Screenshot |  |
| Herramienta de desarrollo | Oracle APEX 5 |

*Además, se debe agregar que el objetivo al cual se está haciendo el ataque, actualmente, no puede ser alcanzado desde fuera.*

### 2.5.2. Paso 2. Crear y analizar el modelo de amenazas

Esta etapa hace referencia a la creación de una lista de amenazas, en la que se deberá incluir el título, una descripción breve, el activo (activos), los impactos, el riesgo, las técnicas de mitigación, el estado de mitigación y número del error.

Las posibles amenazas para este escenario de análisis pueden resumirse en la siguiente tabla:

*Tabla 4. Modelamiento de amenazas*

| Vulnerabilidad | Amenaza |
|---|---|
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta. |
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques MITM |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques MITM |
| Puerto TCP 8080 abierto | Reemplazo de un recurso específico mediante el método POST. |
| Puerto TCP 8080 abierto | Eliminación de un recurso específico mediante el método DELETE. |
| Puerto TCP 8080 abierto | HTTP Method (‘Allow’ Header) Métodos de riesgo potencial: PUT, DELETE |
| Puerto TCP 8080 abierto | Reemplazo de un recurso específico mediante el método POST. |
| Puerto TCP 8181 abierto | Eliminación de un recurso específico mediante el método DELETE. |
| Puerto TCP 8181 abierto | HTTP Method (‘Allow’ Header) Métodos de riesgo potencial: PUT, DELETE |

| | |
|---|---|
| La cabecera anti secuestro de clic (anti-clickjacking X-Frame-Options header) no está presente | Ataques clickjacking |
| El servidor presenta posibilidades de fuga mediante ETags | WebBrowsing Fingerprinting |
| La cabecera de protección contra ataques Cross Site Scripting XSS no está definida | Ataques XSS |
| Puerto TCP 1521 abierto | Envenenamiento TNS (Rocha, 2014) |
| Puerto TCP 3700 abierto | Portal of Doom. Usado comúnmente por troyanos (Gallo, 2011) |
| Puerto TCP 3820 abierto | Denegación de servicios |
| Puerto TCP 3920 abierto | Denegación de servicios |
| Puerto TCP 4848 abierto | Denegación de servicios |
| Puerto TCP 7676 abierto | Denegación de servicios |
| Puerto TCP 7776 abierto | Denegación de servicios |
| Puerto TCP 8080 abierto | Denegación de servicios |
| Puerto TCP 8181 abierto | Denegación de servicios |
| Puerto TCP 8686 abierto | Denegación de servicios |
| Puerto TCP 13335 abierto | Denegación de servicios |
| Puerto TCP 17210 abierto | Denegación de servicios |
| Puerto TCP 29573 abierto | Denegación de servicios |
| Puerto TCP 31780 abierto | Denegación de servicios |
| Puerto TCP 33562 abierto | Denegación de servicios |
| El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida. | Denegación de servicios |
| La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida | Denegación de servicios |
| El nombre de host '172.16.1.87' no concuerda con el nombre del certificado | Suplantación de identidad |

| | |
|---|---|
| Servidor acepta solicitudes ICMP | Saturación ICMP |
| Acepta métodos POST y GET | Inundación HTTP |
| TCP utiliza un mecanismo de negociación de tres vías. | Syn Flood |
| Servidor acepta solicitudes ICMP | Ping de la muerte |
| Servidor acepta solicitudes ICMP | Inundación IP |
| Contraseñas débiles | Fuerza bruta |
| Contraseñas débiles | Ataque de diccionario |
| Cookies | Predicción de credenciales |

### 2.5.3. Paso 3. Analizar las amenazas

De acuerdo con lo que sugiere (Microsoft, 2017), en este paso se debe valorar el riesgo que puede producirse por cada amenaza, y determinar cómo se actuará ante ellas, una vez identificadas las amenazas del entorno. En un pentesting se analizan las posibles amenazas que pueden ser utilizadas para vulnerar los elementos anteriormente hallados. Se pueden usar algunas categorías de efecto para calcular la exposición al riesgo; por ejemplo, daños potenciales, capacidad de reproducción, aprovechamiento, usuarios afectados y capacidad de descubrimiento.

Haciendo uso de las herramientas que sugiere ECU@Risk para la identificación de amenazas, se puede catalogar que las anteriormente mencionadas pueden caer dentro de los siguientes grupos:

| [NO_INTENCIONADO.2] Errores del administrador | |
|---|---|
| Tipos de activos: • [IE] información electrónica • (IP) información en papel • [SW] aplicaciones (software) • [HW] equipos informáticos (hardware) • [IC] infraestructura de comunicaciones | Dimensiones: 1. [D] disponibilidad 2. [I] integridad 3. [C] confidencialidad |
| Descripción: errores involuntarios de personas con responsabilidades de instalación y operación. | |

| [NO_INTENCIONADO.4] Errores de configuración | |
|---|---|
| Tipos de activos: • (IE) información electrónica (IP) información en papel | Dimensiones: 1. [I] integridad |
| Descripción: introducción de datos de configuración erróneos. Prácticamente todos los activos dependen de su configuración y ésta de la diligencia del administrador: privilegios de acceso, flujos de actividades, registro de actividad, encaminamiento, etc. Ver: EBIOS: no disponible | |

| [PROVOCADO.1] Alteración en la configuración | |
|---|---|
| Tipos de activos: • (IE) información electrónica | Dimensiones: 1. [D] disponibilidad 2. [I] integridad |
| Descripción: desastres debidos a la actividad humana: alteración deliberada al sistema informático. Origen: entorno (accidental) humano (accidental o deliberado). | |

| [EL.1] Difusión de software dañino | |
|---|---|
| Tipos de activos: • [SW] software • [IE] información electrónica | Dimensiones: 1. [D] disponibilidad 2. [I] integridad 3. [C] confidencialidad |
| Descripción: propagación inocente de virus, espías (spyware), gusanos, troyanos, bombas lógicas, malware en general. | |

*Por otro lado, es necesario considerar los criterios de clasificación de las amenazas de acuerdo con la matriz de impacto:*

*Tabla 5. Matriz de valoración de impacto. Fuente: (Crespo, Ecu@Risk, Una metodología para la gestión de Riesgos, 2017)*

| Matriz de Impacto | Valoración |
|---|---|
| E – Extremo | 5 |
| A – Alto | 4 |
| M – Moderado | 3 |
| B – Menor | 2 |
| L – Leve | 1 |

*De esta manera, y en relación con los grupos de clasificación anteriormente mencionados, se pueden calificar las amenazas según lo expuesto a continuación, donde C, D, I y V hacen referencia a i) confidencialidad, ii) disponibilidad, iii) integridad, y iv) valoración general (este último es el valor más alto obtenido de los tres anteriores).*

*Tabla 6. Valoración de amenazas*

| Vulnerabilidad | Amenaza | ID Amenaza | C | D | I | V |
|---|---|---|---|---|---|---|
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta. | [NO_INTENCIONADO.4] | | | 4 | 4 |
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques MITM | [NO_INTENCIONADO.4] | | | 4 | 4 |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta | [NO_INTENCIONADO.4] | | | 3 | 3 |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques MITM | [NO_INTENCIONADO.4] | | | 3 | 3 |
| Puerto TCP 8080 abierto | Reemplazo de un recurso específico mediante el método POST. | [PROVOCADO.1] | 4 | 3 | 4 | 4 |
| Puerto TCP 8080 abierto | Eliminación de un recurso específico mediante el método DELETE. | [PROVOCADO.1] | 3 | 3 | 4 | 4 |
| Puerto TCP 8080 abierto | HTTP Method (‘Allow’ Header) Métodos de riesgo potencial: PUT, DELETE | [NO_INTENCIONADO.4] | | | 3 | 3 |
| Puerto TCP 8181 abierto | Reemplazo de un recurso específico mediante el método POST. | [PROVOCADO.1] | 4 | 3 | 4 | 4 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Puerto TCP 8181 abierto | Eliminación de un recurso específico mediante el método DELETE. | [PROVOCADO.1] | 3 | 3 | 4 | 4 |
| Puerto TCP 8181 abierto | HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE | [NO_INTENCIONADO.4] | | | 3 | 3 |
| La cabecera anti secuestro de clic (anti-clickjacking X-Frame-Options header) no está presente | Ataques clickjacking | [PROVOCADO.1] | 2 | 2 | 2 | 2 |
| El servidor presenta posibilidades de fuga mediante ETags | WebBrowsing Fingerprinting | [PROVOCADO.1] | 2 | 2 | 2 | 2 |
| La cabecera de protección contra ataques Cross Site Scripting XSS no está definida | Ataques XSS | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 1521 abierto | Envenenamiento TNS (Rocha, 2014) | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 3700 abierto | Portal of Doom. Usado comúnmente por troyanos (Gallo, 2011) | [EL.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 4848 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 5500 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 5521 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 5801 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 5901 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 6001 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 7676 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 7776 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Puerto TCP 8080 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 8181 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 8686 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 9413 abierto | Denegación de servicios | [PROVOCADO.1] | | | | |
| Puerto TCP 24241 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 26486 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 31793 abierto | Oracle TNS Listener / Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 33446 abierto | Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Puerto TCP 36575 abierto | Oracle TNS Listener / Denegación de servicios | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida. | Denegación de servicios | [PROVOCADO.1] | 3 | 1 | 3 | 3 |
| La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida | Denegación de servicios | [PROVOCADO.1] | 3 | 1 | 3 | 3 |
| El nombre de host '172.16.1.87' no concuerda con el nombre del certificado | Suplantación de identidad | [PROVOCADO.1] | 4 | 1 | 3 | 4 |
| Servidor acepta solicitudes ICMP | Saturación ICMP | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Acepta métodos POST y GET | Inundación HTTP | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| TCP utiliza un mecanismo de negociación de tres vías. | Syn Flood | [PROVOCADO.1] | 3 | 3 | 3 | 3 |
| Servidor acepta solicitudes ICMP | Ping de la muerte | [PROVOCADO.1] | 1 | 3 | 1 | 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Servidor acepta solicitudes ICMP | Inundación IP | [PROVOCADO.1] | 1 | 3 | 1 | 3 |
| Contraseñas débiles | Fuerza bruta | [PROVOCADO.1] | 4 | 2 | 2 | 4 |
| Contraseñas débiles | Ataque de diccionario | [PROVOCADO.1] | 4 | 2 | 2 | 4 |
| Cookies | Predicción de credenciales | [PROVOCADO.1] | 3 | 2 | 2 | 3 |
| Interesting Meta Tags Detected | Revelación no intencionada de información | [NO_INTENCIONADO.2] | 2 | 1 | 1 | 1 |

### 2.5.4. Paso 4. Identificar las tecnologías y técnicas de mitigación

Después de identificar las amenazas que se van a solucionar, se debe determinar las técnicas de ataque disponibles para cada una. En este caso, será la herramienta que facilitará el análisis.

*Tabla 7. Herramientas utilizadas en el análisis de amenazas*

| Vulnerabilidad | Amenaza | Herramienta considerada en el análisis |
|---|---|---|
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta. | Crunch<br>Hydra |
| Puerto TCP 22 abierto | El puerto 22 no es cifrado y por lo tanto es propenso a ataques MITM | Wireshark |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques de fuerza bruta | Crunch<br>Hydra |
| Puerto TCP 25 abierto | El puerto 25 no es cifrado y por lo tanto es propenso a ataques MITM | Wireshark |
| Puerto TCP 8080 abierto | Reemplazo de un recurso específico mediante el método POST. | VEGA<br>Wapiti |
| Puerto TCP 8080 abierto | Eliminación de un recurso específico mediante el método DELETE. | VEGA<br>Wapiti |
| Puerto TCP 8080 abierto | HTTP Method (‘Allow’ Header) Métodos de riesgo potencial: PUT, DELETE | Kali Linux<br>APEX-Sert<br>W3AF<br>NIKTO<br>OWASP |
| Puerto TCP 8080 abierto | Reemplazo de un recurso específico mediante el método POST. | VEGA<br>Wapiti |
| Puerto TCP 8181 abierto | Eliminación de un recurso específico mediante el método DELETE. | VEGA<br>Wapiti |

| | | |
|---|---|---|
| Puerto TCP 8181 abierto | HTTP Method ('Allow' Header) Métodos de riesgo potencial: PUT, DELETE | Kali Linux<br>APEX-Sert<br>W3AF<br>NIKTO<br>OWASP |
| La cabecera anti secuestro de clic (anti-clickjacking X-Frame-Options header) no está presente | Ataques clickjacking | VEGA, OWASP |
| El servidor presenta posibilidades de fuga mediante ETags | WebBrowsing Fingerprinting | VEGA, OWASP |
| La cabecera de protección contra ataques Cross Site Scripting XSS no está definida | Ataques XSS | VEGA, OWASP |
| Puerto TCP 1521 abierto | Envenenamiento TNS (Rocha, 2014) | Metasploit exploit framework |
| Puerto TCP 1830 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 3700 abierto | Portal of Doom. Usado comúnmente por troyanos (Gallo, 2011) | Metasploit exploit framework |
| Puerto TCP 4848 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 5500 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 5521 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 5801 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 5901 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 6001 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 7676 abierto | Denegación de servicios | Metasploit exploit framework |

| | | |
|---|---|---|
| Puerto TCP 7776 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 8080 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 8181 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 8686 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 9413 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 24241 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 26486 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 31793 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 33446 abierto | Denegación de servicios | Metasploit exploit framework |
| Puerto TCP 36575 abierto | Denegación de servicios | Metasploit exploit framework |
| El sitio utiliza SSL y la cabecera de transporte seguro estricto (Strict-Transport-Security HTTP Header) no está definida. | Denegación de servicios | Nikto, VEGA, OWASP |
| La cabecera de opciones de tipo de contenido X-Content-Type-Options no está establecida | Denegación de servicios | Nikto, VEGA, OWASP |
| El nombre de host '172.16.1.87' no concuerda con el nombre del certificado | Suplantación de identidad | W3AF, WebEscarab |
| Servidor acepta solicitudes ICMP | Saturación ICMP | hping3, Metasploit exploit framework |
| Acepta métodos POST y GET | Inundación HTTP | Nikto, VEGA, OWASP |

| | | |
|---|---|---|
| TCP utiliza un mecanismo de negociación de tres vías. | Syn Flood | hping3, nping, Metasploit exploit framework |
| Servidor acepta solicitudes ICMP | Ping de la muerte | hping3, Metasploit exploit framework |
| Servidor acepta solicitudes ICMP | Inundación IP | hping3, Metasploit exploit framework |
| Contraseñas débiles | Fuerza bruta | Crunch<br>Hydra |
| Contraseñas débiles | Ataque de diccionario | Rainbow |
| Cookies | Predicción de credenciales | |

### 2.5.5. Paso 5. Modelo de seguridad de documento y las consideraciones de implementación

En el modelo de análisis de las amenazas propuesto por Microsoft se sugiere que guarde la documentación de todo lo que se descubra en el análisis de modelo de amenazas. Es importante, por lo tanto, para un atacante o para la parte opuesta, documentar toda amenaza, vulnerabilidad, frecuencia de repetición, técnica de ataque, pruebas y resultados obtenidos.

### 2.5.6. Paso 6. Implementar y probar las mitigaciones

Microsoft (2017) hace referencia a la implementación de contramedidas y prueba de las mismas. Para el caso del pentesting, se hará referencia a la etapa de explotación de vulnerabilidades.

### 2.5.7. Paso 7. Mantener el modelo de amenazas sincronizado con el diseño

El paso 7, para el caso de una prueba de penetración, hace referencia a la mantención del acceso, conocido también como fase Post-Explotación. En ella un atacante de sombrero negro elimina cualquier evidencia de ataque, algo que no lo hace el atacante ético.

## 2.6. Fase 6: Explotación de vulnerabilidades

El servidor con dirección 172.16.1.87, que aloja al software ERP que mantiene la Universidad del Azuay como proyecto, presenta las vulnerabilidades que serán detalladas y analizadas en esta sección.

### 2.6.1. Autenticación

Se ha podido evidenciar que el sitio no maneja un certificado digital, así como tampoco un protocolo de cifrado SSL.

*Tabla 8. Recomendación frente a amenazas de autenticación*

| *Puerto 8080* | | |
|---|---|---|
| *Vulnerabilidad* | *Análisis* | *Recomendación* |
| El nombre de host '172.16.1.87' no concuerda con el nombre del certificado: Localhost | No existe una relación entre el nombre del host y el nombre del certificado. Esto no garantiza que la conexión que se está estableciendo sea realizada con el servidor auténtico. | Cuando el servicio ingrese a producción es importante considerar la implementación de un certificado digital de tipo Organization SSL, que asegura no solamente que el sitio web es válido, sino que garantiza la autenticidad de la organización que la respalda. |

### 2.6.1.1. Fortaleza de contraseñas

Como resultado de la evaluación con la herramienta provista por GRC (Gibson Research Corporation), se puede mencionar lo siguiente:

*Tabla 9. Fortaleza de contraseñas frente a escenarios de ataque.*

| Escenario | Contraseña 1 | Contraseña 2 | Contraseña 4 | Contraseña 3 |
|---|---|---|---|---|
| | password | Password | P@5sw*rD | .Pa5sw0rD.! |
| Profundidad de análisis (alfabeto) | 26 | 52 | 95 | 95 |
| Longitud (caracteres) | 8 | 8 | 8 | 11 |
| Cuenta de las posibles contraseñas con el tamaño de alfabeto y longitud de clave | 217.180'147.158 | 54,507,958,502,660 | 6,704,780,954,517,120 | 5,748,511,570,879,116,626,495 |
| Escenario de ataque Online | 6.91 años | 17.33 siglos | 2.13 miles de siglos | 1.83 billón de siglos |
| Escenario de ataque Offline | 2.17 segundos | 9.08 minutos | 18.62 horas | 18.28 siglos |
| Escenario de arreglo de cracking masivo | 0.00217 seg. | 0.545 seg. | 1.12 minutos | 1.83 años |

Se puede ver en la tabla anterior una variación de la palabra password (comúnmente utilizada), así como el tiempo que le tomaría a un atacante vulnerar la contraseña. Eso sugiere entonces que la aplicación debería considerar el uso de contraseñas robustas, con una longitud superior a los 8 caracteres, y considerar una rotación de máximo 45 días.

### 2.6.1.2. Uso de segundo factor de autenticación

El proyecto ERP actualmente no ofrece la protección de un segundo factor de autenticación, por cuanto pueden realizarse ataques de repetición contra usuarios y contraseñas. Se sugiere, por lo tanto, incluir al menos uno de los siguientes mecanismos de validación:

*Tabla 10. Mecanismos de validación sugeridos*

| Mecanismo | Objetivo | Aspectos negativos |
|---|---|---|
| Captcha (texto) | Evitar los ataques de tipo repetición originados por técnicas automatizadas. | Puede ser incómodo para el usuario tratar de comprender el texto de un captcha. |
| Captcha (gráficos) | Evitar los ataques de tipo repetición originados por técnicas automatizadas. | Son vulnerables a sistemas de reconocimiento de imágenes. Otros son débiles si utilizan algoritmos de cifrado débiles como un MD5. |
| Código de verificación | Evitar los ataques de tipo repetición originados por técnicas automatizadas. | Requieren del envío de un toquen por SMS o correo electrónico, que puede generar costos por la operadora. Puede darse el caso de que el mensaje no llegue al destinatario y el usuario no pueda hacer inicio de sesión. |
| Verificación de correo electrónico | Evitar los ataques de tipo repetición originados por técnicas automatizadas. | El correo electrónico de verificación podría no llegar al destinatario. Se requiere un punto de contacto para que el cliente haga las consultas en caso de que las tenga. |
| Aprovisionamiento de correo electrónico. | Evitar los ataques de tipo repetición originados por técnicas automatizadas. Garantizar la identidad del usuario. Se envía un hipervínculo al correo electrónico del cliente para que valide el espacio de trabajo al cual se está conectando. | El correo electrónico de verificación podría no llegar al destinatario. |
| Encrypted Tablespaces | Al habilitar esta opción, APEX utilizará el TDE (Transparent Data Encryption) para cifrar los archivos de datos asociados. TDE podría cifrar todos los archivos de bases de datos que son escritos en el disco, haciéndolos ilegibles a cualquiera que trate de acceder a los mismos. | La opción de cifrado TDE (que forma parte del Oracle Advanced Security Option) está disponible únicamente en ediciones Enterprise de la base de datos Oracle. Por otro lado, si se ha utilizado una llave para cifrar tablespaces fuera del esquema, se deberá configurar y abrir Oracle Wallet. |

### 2.6.1.3. Falta de uso de mecanismos de cifrado

Utilizando el analizador de tráfico Wireshark se pudo comprobar que los datos viajan sin cifrar. Resultó fácil capturar al usuario y la contraseña de la aplicación, tal como se puede ver en la figura a continuación:

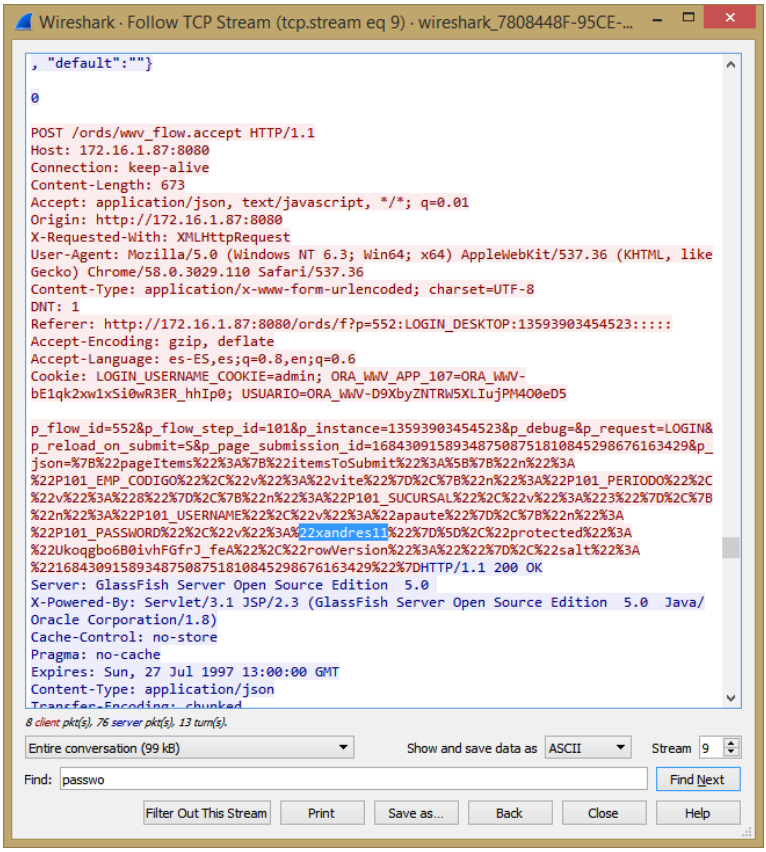



*Figura 35. Captura de la contraseña del usuario utilizando un analizador de tráfico*

### 2.6.1.4. Fuerza bruta a SSH

Se había determinado en la etapa de reconocimiento que existe un puerto SSH para la administración remota del equipo. Esto significó que el servidor es vulnerable a accesos SSH no autorizados.

*Tabla 11. Recomendación ante vulnerabilidades de puertos SSH en escucha*

| *Vulnerabilidad* | *Análisis* | *Recomendación* |
|---|---|---|
| Puerto SSH abierto | Se realiza un ataque de fuerza bruta a la dirección IP donde se encuentra alojado el proyecto ERP utilizando la herramienta HYDRA.<br><br>De lo evaluado, el firewall no bloquea intentos de conexión repetitivos, así como el equipo no bloquea los intentos de conexión fallidos | Se recomienda limitar las direcciones IP desde donde podrá alcanzarse el equipo objetivo.<br><br>Es recomendable contar con una infraestructura de identificación de intrusos para mitigar los ataques de acción repetitiva.<br><br>Se recomienda además el uso de contraseñas complejas, con una longitud mínima de 8 caracteres. |

### 2.6.2. Cross Site Scripting – XSS

Como resultado del análisis obtenido por la herramienta Vega, se mencionan las siguientes vulnerabilidades:

*Tabla 12. Recomendaciones ante vulnerabilidades XSS*

| *Vulnerabilidad* | *Análisis* | *Recomendación* |
|---|---|---|
| Cleartext password over HTTP | La prueba realizada fue la ejecución del comando GET / ords/f?p=552:LOGIN_DESKTOP::::::, lo que permitiría la divulgación de usuarios y contraseñas a intrusos en la red. | Se recomienda el uso del protocolo HTTPS y cifrado en la transmisión de información, además de la implementación y el uso de certificados digitales emitidos por entidades certificadoras de confianza. |
| Form Password field with autocomplete enabled | La prueba inicia con la aplicación del comando GET / ords/f?p=552:LOGIN_DESKTOP::::::<br><br>El error aparece debido a la existencia de un campo de contraseña, cuyo atributo no ha sido establecido a Off. Esto puede llevar a que algunos browsers almacenen valores ingresados por los usuarios de forma local, los mismos que pueden ser utilizados posteriormente. | Para deshabilitar el autocompletamiento, se debe utilizar el código: <INPUT TYPE=”password” AUTOCOMPLETE=”off”> |
| Interesting Meta Tags Detected | A través del comando GET / ords/f, se pudo obtener ‹meta name=”viewport” content=”width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no”/›, esto hace referencia a que los meta tags pueden revelar información acerca de la plataforma o la configuración. | Evaluar el uso de los meta tags y el tipo de información que se está entregando. La importancia de las meta etiquetas (meta tags) es que los motores de búsqueda los leen con el fin de comparar si estas palabras clave y la descripción están relacionados con el contenido visible. |

| | | |
|---|---|---|
| X-Frame Options Not Set | Con GET / se ha detectado que no se ha establecido el encabezado de respuesta HTTP X-Frame-Options. Este encabezado permite al recurso especificar su política con respecto a si se puede incluir en marcos en otros dominios, así como qué dominios están permitidos. Cuando se ha establecido el encabezado, esto puede ayudar a mitigar los ataques de clickjacking contra los navegadores compatibles con esta característica. Si no se ha establecido el encabezado, el recurso afectado puede utilizarse en ataques de tipo clickjacking. | La cabecera X-Frame-Options sirve para prevenir que la página pueda ser abierta en un frame, o iframe. De esta forma se pueden prevenir ataques de clickjacking sobre el sitio web.<br><br>Se recomienda establecer los parámetros de X-Frame de acuerdo a las necesidades del negocio. Los parámetros son:<br>DENY<br>SAMEORIGIN<br>ALLOW-FROM URI<br><br>Esto evita los ataques de tipo clickjacking.<br><br>Para evadir los ataques XSS, se sugiere además incluir la siguiente cabecera:<br><br>`header('X-XSS-Protection: 1;mode=block');` |

| | | |
|---|---|---|
| X-Frame Options Not Set | Luego de ejecutar el comando GET /, se ha comprobado que el recurso no ha especificado un conjunto de caracteres en la respuesta. Si no se especifica un juego de caracteres, el navegador puede hacer suposiciones sobre el conjunto de caracteres basado en el contenido del recurso. De esta manera se puede presentar un problema de seguridad si el recurso afectado contiene contenido generado dinámicamente, que se origina en el lado del usuario. En tal caso, los usuarios malintencionados pueden potencialmente aprovechar la forma en que los navegadores específicos interpretan los caracteres para que se produzca contenido malintencionado. Por ejemplo, un atacante puede evitar un filtro de secuencias de comandos de sitios cruzados al codificar su carga útil malintencionada en un conjunto de caracteres alternativo, que se puede ejecutar dependiendo de cómo el navegador interpreta el contenido codificado. | Se debe crear un conjunto de caracteres válidos para respuesta. Por ejemplo:<br><br># charset<br><br>AddCharset utf-8 .html<br><br>AddCharset utf-8 .js<br><br>AddCharset utf-8 .css |

### 2.6.3. Identificación de vulnerabilidades en el código fuente

*Tabla 13. Recomendaciones ante vulnerabilidades en el código fuente*

| Vulnerabilidad | Análisis | |
|---|---|---|
| Server leaks inodes via ETags, header found with file /<br>, fields: 0xW/4626 0x1442682444000 | Puerto 8080: GlassFish Server Open Source 4.1.1<br><br>El servidor presenta posibilidades de fuga mediante ETags. La cabecera encontrada corresponde a 0xW/4626 0x1442682444000<br><br>Un ETag es similar a una huella digital. Es un mecanismo que proporciona validación del caché web, lo que permite al cliente realizar peticiones condicionales, lo que hace que los caches sean más eficientes, y ahorran además ancho de banda, ya que el servidor analiza si el contenido ha cambiado antes de enviar una respuesta. | Establecer el parámetro<br><br>`FileETag None`<br><br>En el servidor glassfish. |
| The anti-clickjacking X-Frame-Options header is not present. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | |
|---|---|---|
| Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS | GET y POST son los métodos más comunes para brindar acceso a la información provista por un sistema web, sin embargo, existen algunos adicionales según la referencia RFC 2616 (que describe el estándar HTTP 1.1).<br><br>Sin embargo, los métodos PUT, DELETE, y TRACE deben ser manejados con mucho cuidado.<br><br>PUT: permite que el cliente cargue nuevos archivos al servidor web. Un atacante puede aprovecharse de esta vulnerabilidad subiendo archivos infectados, generalmente ejecutables; o simplemente utilizando el servidor de la víctima como un repositorio de archivos.<br><br>DELETE: método que permite al cliente eliminar archivos en el servidor web. Un atacante puede hacer uso de esta vulnerabilidad para eliminar el contenido del sitio web, o ejecutar un ataque de denegación de servicios DoS.<br><br>TRACE: este método simplemente responde a cualquier cadena que haya sido enviada al servidor, y que es utilizada generalmente para propósitos de depuración (debug). Este método puede ser utilizado para realizar ataques del tipo Cross Site Tracing. | Si estos métodos son estrictamente necesarios, se debe limitar a un número de usuarios con privilegios, en caso contrario, debería prescindirse de ellos. |

| | | |
|---|---|---|
| OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server. | PUT: permite que el cliente cargue nuevos archivos al servidor web. Un atacante puede aprovecharse de esta vulnerabilidad subiendo archivos infectados, generalmente ejecutables; o simplemente utilizando el servidor de la víctima como un repositorio de archivos. | Si estos métodos son estrictamente necesarios, se debe limitar a un número de usuarios con privilegios, en caso contrario, debería prescindirse de ellos. |
| OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server. | DELETE: método que permite al cliente eliminar archivos en el servidor web. Un atacante puede hacer uso de esta vulnerabilidad para eliminar el contenido del sitio web, o ejecutar un ataque de denegación de servicios DoS. | Si estos métodos son estrictamente necesarios, se debe limitar a un número de usuarios con privilegios, en caso contrario, debería prescindirse de ellos. |

### 2.6.3.1. Resultados obtenidos con OWASP ZAP

*Tabla 14. Recomendaciones ante vulnerabilidades en código fuente: cabeceras*

| Vulnerabilidad | Método | URL | Análisis | Solución |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/app_ui/css/Core.min.css?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/app_ui/css/Theme-Standard.min.css?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/favicon.ico | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/legacy_ui/css/5.0.min.css?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/libraries/apex/minified/desktop.min.js?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/libraries/apex/minified/legacy.min.js?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/libraries/jquery-migrate/1.4.1/jquery-migrate-1.4.1.min.js?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/libraries/jquery-ui/1.10.4/themes/base/jquery-ui.min.css?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/libraries/jquery/2.2.3/jquery-2.2.3.min.js?v=5.1.1.00.08 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/themes/theme_24/css/4_1.css | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/themes/theme_24/css/4_1_erp2.css | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/themes/theme_24/images/LOGOERP.PNG | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/i/themes/theme_24/js/4_1.js | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/ords/f?p=552:LOGIN_DESKTOP::::: | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/ords/f?p=552:LOGIN_DESKTOP::::: | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Content-Type-Options header missing | GET | http://172.16.1.87:8080/ords/wwv_flow.js_messages?p_app_id=552&p_lang=en&p_version=5.1.1.00.08-21244798 | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |

| | | | | |
|---|---|---|---|---|
| X-Content-Type-Options header missing | POST | http://172.16.1.87:8080/ords/wwv_flow.accept | La cabecera anti-MIME-Sniffing header X-Content-Type-Options no ha sido establecida a 'nosniff'.<br><br>Esto permite que antiguas versiones de Internet Explorer y Chrome ejecuten un MIME-sniffing en el cuerpo de respuesta (body response), causando potencialmente que la información provista por el cuerpo de respuesta sea visualizada como otro tipo de contenido diferente al declarado originalmente. | Asegurarse de que el servidor de aplicaciones / web establezca la cabecera Content-Type de forma apropiada y que establezca el encabezado X-Content-Type-Options en 'nosniff' para todas las páginas web. Si es posible, se debe asegurar que el usuario final utilice un navegador web moderno compatible con los estándares, que no realice MIME-sniffing o que pueda ser dirigido por la aplicación web / servidor web para que no realice MIME-sniffing. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/app_ui/css/Core.min.css?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/app_ui/css/Theme-Standard.min.css?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/legacy_ui/css/5.0.min.css?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/apex/minified/desktop.min.js?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/apex/minified/legacy.min.js?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/jquery-migrate/1.4.1/jquery-migrate-1.4.1.min.js?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/jquery-ui/1.10.4/themes/base/jquery-ui.min.css?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/jquery/1.12.3/jquery-1.12.3.min.js?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/libraries/jquery/2.2.3/jquery-2.2.3.min.js?v=5.1.1.00.08 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/themes/theme_24/css/4_1.css | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/themes/theme_24/css/4_1_erp2.css | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/i/themes/theme_24/js/4_1.js | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |
| X-Frame-Options header not set | GET | http://172.16.1.87:8080/ords/wwv_flow.js_messages?p_app_id=552&p_lang=en&p_version=5.1.1.00.08-21244798 | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

| | | | | |
|---|---|---|---|---|
| X-Frame-Options header not set | POST | http://172.16.1.87:8080/ords/wwv_flow.accept | La cabecera X-Frame-Options no ha sido incluida en la respuesta HTTP response para protegerse en contra de los ataques 'ClickJacking'. | La mayoría de los navegadores web modernos admiten el encabezado HTTP X-Frame-Options. Se debe asegurar de que está configurado en todas las páginas web devueltas por el sitio. Si se espera que la página se enmarque sólo por las páginas en el servidor (por ejemplo, es parte de un FRAMESET), entonces se deberá utilizar SAMEORIGIN, de lo contrario si nunca se espera que la página sea enmarcada, se debe utilizar DENY. ALLOW-FROM permite a sitios web específicos encuadrar la página web en navegadores sustentados. |

### 2.6.3.1. Resultados obtenidos con APEX – SERT

APEX es un entorno de desarrollo declarativo, esto quiere decir que cada objeto en una aplicación APEX es actualmente almacenado como unos metadatos en un conjunto de tablas de una base de datos. Cuando una página es traducida, APEX llama a un conjunto de sus propios procedimientos PL/SQL, que, a su vez, leen los metadatos correspondientes y la utilizan para generar cualquier componente que forma parte de la página. Así, cuando una nueva página, reporte, calendario, diagrama o un proceso dado es creado, no se crean nuevos objetos en la base de datos. Más bien, la información acerca del componente es almacenada como metadata por APEX y es vuelta a llamar cuando el usuario actualiza la página (Spendolini, 2016).

Todos los metadatos de APEX son expuestos a través de un conjunto de vistas simples llamada "APEX Views". Estas vistas proveen de un panorama de todo lo que hacen los metadatos en un entorno APEX, desde el propio espacio de trabajo hasta una columna de un informe. Debido a que una gran porción de APEX está basada en metadatos, es posible utilizar herramientas automatizadas como SERT, que inspeccionan los valores de muchos atributos, permitiendo determinar la mejor alternativa de aseguramiento (Spendolini, 2016).

Los APEX Views pueden ser accedidos desde cualquier esquema definido en la base de datos, pero retorna datos solo si las vistas son consultadas desde un esquema que está relacionado a un workspace o desde SYS y SYSTEM. Estos pueden ser accedidos desde cualquier herramienta que se conecte directamente a la base de datos y no solamente desde el APEX (Spendolini, 2016).

Una sesión APEX, desde el navegador, puede visualizarse de esta manera:

*http://172.16.1.87:8080/ords/f?p=552:1:3027767055940:::::*

El valor "3027767055940" que es un número generado aleatoriamente, hace referencia a un identificador de sesión, que es única para diferenciar a otras sesiones iniciadas, la misma que es usada en casi la mayoría de URLs APEX. Una ventaja que se tiene con esto es que, para lograr un adecuado secuestro de sesión, no es suficiente solo con copiar la URL y pegarla en otra ventana. El ID de sesión no es el único elemento que garantiza una conexión segura, sino que a esto se suma el uso de cookies de sesión, que son almacenadas en el equipo del usuario (Spendolini, 2016).

Esta cookie contiene un valor que, cuando es combinada con el ID de sesión, valida al usuario como un usuario APEX. El estado de la sesión es almacenado en la base de datos y no en la cookie del usuario, lo que hace más segura la sesión. Por otro lado, la mayoría de los procesos son realizados en la base de datos en el lado del servidor, haciendo que el trabajo de búsqueda sea más eficiente. El tiempo de vigencia de una sesión APEX puede variar y terminar según uno de los siguientes eventos: primero, tan rápido como el usuario finalice su sesión, la misma automáticamente caduca y todos los

valores asociados en el estado de sesión también son purgados (Spendolini, 2016).

Si el usuario cierra todas las ventanas de los navegadores, la cookie de sesión expira hasta terminar la sesión. Pero cerrar solamente la pestaña de navegación no es suficiente, ya que puede mantener la sesión del usuario activa.

Por omisión, los trabajos calendarizados que efectúa APEX, hace que las sesiones sean purgadas de inmediato, o en el peor de los casos cada 24 horas. Esta actividad programada se ejecuta cada 8 horas, si es que no ha sido modificada.

### 2.6.4. SQL Injection

No se encuentran vulnerabilidades en cuanto a inyección SQL.

### 2.6.5. DDoS – Denegación de servicios

Como resultado del análisis obtenido por la herramienta metasploit y el comando hping3, se mencionan las siguientes vulnerabilidades:

*Tabla 15.Resultados del análisis de vulnerabilidades*

| *Vulnerabilidad* | *Análisis* | *Resultados* |
|---|---|---|
| IP Flooding 8080 | Se realiza una prueba ejecutando el comando hping3 --rand-source -d 600 172.16.1.87 -p 8080 –flood, cuyo resultado fue la negación de servicio del servidor. | Vulnerable |
| SYN Flood 8080 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 8080<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El resultado fue la negación de servicio. | Vulnerable |

| | | |
|---|---|---|
| Envenenamiento 1521 TNP | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 1521<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 3700 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 3700<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 3820 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 3820<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 3920 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 3920<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |

| | | |
|---|---|---|
| SYN Flood 4848 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 4848<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 6001 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 6001<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 7676 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 7676<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 7776 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 7776<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |

| | | |
|---|---|---|
| SYN Flood 8181 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 8181<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 8686 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 8686<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 9413 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 9413<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 24241 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 24241<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |

| | | |
|---|---|---|
| SYN Flood 26486 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 26486<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 31793 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 31793<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 33446 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 33446<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |
| SYN Flood 36575 | Se realiza la prueba utilizando el Metasploit Framework con la opción: auxiliary/dos/tcp/synflood, con los parámetros:<br>RHOST: 172.16.1.87<br>RPORT: 36575<br>SNAPLEN: 65535<br>TIMEOUT: 30<br>El servicio no ha sido afectado. | Ok |

## 2.7. Fase 7: Post explotación

En la etapa de post explotación el atacante busca mantener establecida la conexión con el servidor y extraer o alterar toda la información que le sea posible, considerando además la eliminación de las huellas o rastros que haya dejado durante el ataque. Este paso no es considerado en un hackeo ético, pues la parte interesada debe conocer los elementos que fueron vulnerados para validarlo con el informe de pentesting y así tomar los correctivos necesarios.

## 2.8. Fase 8: Reporte

### 2.8.1. Resumen ejecutivo

La seguridad de la información y la seguridad informática están basadas en tres principios fundamentales: i) la disponibilidad, que hace referencia a que la información debe estar disponible en el momento en que se la requiera, ii) la confidencialidad, que se refiere al acceso autorizado a la misma, y iii) la integridad, que hace relación a que la información debe ser alterada bajo conocimiento. Además, en el siglo XXI se dice que la información es el recurso más valioso para una organización, y es más valioso para quien la tenga en el momento oportuno, además de que sepa cómo utilizarla; por lo tanto, según la frase propuesta por Francis Bacon quien sugiere que la "información es poder", para una empresa u organización es sinónimo de ventaja competitiva.

La evolución de la informática, las computadoras y el rápido crecimiento de las múltiples transacciones de negocio, obligan a las empresas a anticiparse a los diversos escenarios de riesgo en los que la información y los dispositivos computacionales que facilitan su captura, procesamiento, transmisión y almacenamiento están inmersos; escenarios como ataques de hackers, usuarios internos, amenazas lógicas, entre una gran variedad adicional.

Una de las técnicas de evaluación que apoya al análisis de brechas de seguridad es el pentesting, análisis que debe ser realizado por una persona con conocimientos técnicos, pero con principios éticos. Esta última característica diferencia a un atacante común, conocido en esta época como un "hacker de sombrero negro", que es una persona que vulnera la seguridad de la organización con el fin de beneficiarse de la información en ella almacenada.

El presente documento contempla las diversas técnicas utilizadas en el pentesting realizado tanto al servidor que aloja al producto informático, como al software ERP desarrollado por la Universidad del Azuay, utilizando la plataforma APEX 5 se contemplan las seis etapas que comprende una prueba de

penetración que sugiere (Caballero, 2015): i) Conceptualización, etapa que permite definir el alcance de las pruebas que se realizarán, ii) la preparación del laboratorio, en la que se definen algunas de las herramientas que servirán para el inicio de las pruebas de seguridad, iii) la obtención de información, que hace referencia a las etapas de reconocimiento y escaneo, en las que se identifican posibles objetivos para luego explorar con mayor profundidad algunas características intrínsecas que puedan ser aprovechadas; iv) el análisis de las vulnerabilidades encontradas en la etapa anterior, v) la explotación de esas vulnerabilidades mediante la selección de adecuadas herramientas que permitan lograr ese propósito; y vi) la post explotación, etapa en la que se contempla la destrucción de evidencias del ataque y la conservación de la conexión y los accesos logrados para extraer la información.

Las pruebas efectuadas han sido realizadas dentro de las instalaciones de la Universidad del Azuay, considerando el ambiente de desarrollo en el que actualmente se encuentra el proyecto ERP. En un futuro, se sugiere realizar este conjunto de pruebas en un ambiente de pre producción y además en el ambiente de producción, ya que el escenario podría variar de acuerdo con las características propias de cada uno de estos. Con relación a las pruebas en la etapa de desarrollo, se descubrieron nuevos puertos abiertos y otros que han sido cerrados, producto de la dinámica de los sistemas operativos.

### 2.8.2. Resultados obtenidos

En el alcance de la investigación se ha definido la exploración y análisis de las vulnerabilidades y posibles amenazas que comprometerían al proyecto ERP en una situación de riesgo informático. Entre ellas se han identificado las siguientes:

1. Puertos de escucha
2. Identificación del contexto computacional, que contempla al sistema operativo, base de datos, lenguaje de programación, usuarios iniciados, entre otros.
3. Vulnerabilidades en el código fuente
4. Vulnerabilidad ante ataques de tipo XSS – Cross Site Scripting
5. Vulnerabilidad ante ataques de inyección SQL.

#### 2.8.2.1. Puertos de escucha

Los puertos de escucha del servidor deben estar controlados. Asimismo, no deben ser puertos considerados como conflictivos.

En este segundo análisis se detectaron los puertos: 22, 25, 80, 1521, 1830, 3700, 4848, 5500, 5521, 5801, 5901, 6001, 7676, 7776, 8080, 8181, 8686,

9413, 24241, 26486, 31793, 33446, 36575; considerándose como peligroso el puerto 22, 5500 y 5901 y como conflictivos los puertos 25, 80, 8080 y 8181.

El puerto 22, referente al servicio SSH debe ser utilizado con cuidado, debido a que el mismo ofrece la posibilidad de gestionar un servidor de manera remota. Debe, por lo tanto, considerarse la restricción de acceso limitando a solo ciertos equipos que puedan establecer la comunicación, además del uso de contraseñas robustas (8 caracteres mínimo, combinados con caracteres alfabéticos entre mayúsculas y minúsculas, números y caracteres especiales). Se sugiere además el considerar el uso de un servicio VPN con canal cifrado para establecer una conexión segura. El mismo tratamiento debe ser considerado con el uso de los puertos 5801 y 5901, relacionados con el servicio de soporte remoto gráfico VNC.

Entre los intentos de vulneración al servicio, se pudo determinar que la infraestructura no controla los intentos de acceso con contraseñas fallidas.

El puerto 25 es utilizado por el servicio SMTP. Si el servicio no es utilizado, se sugiere cerrarlo, en caso contrario, se recomienda cambiar el puerto de escucha (por ejemplo, al puerto 26) y realizar conexiones seguras, utilizando certificados digitales y cifrado. El mal uso del puerto SMTP puede comprometer la dirección IP pública de la institución, catalogándola como generadora de SPAM.

Se visualiza la página por omisión del Apache, versión 2.2.15, al acceder por el puerto 80. Se recomienda no publicar esta información, o en su defecto, cambiar la página por omisión.

El puerto 5801 está abierto debido al servicio de conexión remota TigerVNC. Además, mantiene el puerto 5901 como servicio de soporte remoto VNC, con nivel de seguridad de autenticación medio.

Los puertos 8080 y 8181 están expuestos sin cifrado. Están siendo utilizados por Oracle Glassfish 5.0. Se recomienda por lo tanto el uso de un certificado digital y el cifrado de información durante la transmisión y recepción. Es importante considerar el uso de un IDS/IPS para detectar intentos de ataque de inundación HTTP, o hacer frente a herramientas como WebEscarab, que permite la descarga completa de un sitio web, lo que permitirá a un atacante efectuar una suplantación de identidad mediante la técnica de phishing. Mantiene potenciales riesgos en los servicios PUT, DELETE, y TRACE que deberían ser evaluados en su utilización.

#### 2.8.2.1. Contexto computacional

Según lo sugerido (Weidman, 2014), este análisis sugiere recolectar y analizar libremente cualquier origen de información mediante el uso de herramientas informáticas para análisis de puertos, identificación de sistemas operativos y usuarios conectados, lo que ha dado una idea del entorno que posteriormente será comprometido.

El sistema operativo encontrado es Linux Centos, que corre bajo un escenario virtualizado. La divulgación de uso de los puertos 1521 y 5521, revelaron que la base de datos utilizada es Oracle. Además, se pudo determinar que la herramienta de desarrollo es APEX 5. Mediante la técnica de sniffing, se pudo encontrar al usuario "apaute" como un usuario que realizó un inicio de sesión en la página. El motor web utilizado es Oracle GlassFish 5.0, con Servlet 3.1, JSP 2.3 y Java 1.8.

Entre los ataques realizados, se verificó que el equipo es vulnerable a la denegación de servicios. Se realizaron pruebas de inundación IP, inundación SYN, ping de la muerte y saturación HTTP. De todas ellas, el equipo entró en denegación de servicios cuando se aplicó la inundación por IP y por SYN.

En cuanto a autenticación, se ha podido observar que el sitio utiliza el algoritmo propio de APEX para bloquear los ataques de fuerza bruta. Sin embargo, no cuenta con un segundo factor de autenticación. Se sugiere implementar un segundo factor para garantizar la autenticación de un usuario a la plataforma.

### 2.8.2.2. Pruebas de vulnerabilidades de código

Para las pruebas de vulnerabilidad de código se han utilizado herramientas de intrusión y herramientas para el análisis de código fuente, conocidas también como SAST - Static Application Software Testing (herramientas para la prueba de software de aplicación estática).

Existen múltiples métodos para la detección de fallos de seguridad en software (Hernández, 2007), entre los que se pueden citar: i) análisis del código fuente, ii) análisis estático del archivo binario, iii) análisis dinámico del archivo binario (en tiempo de ejecución), iv) fuzzing, v) métodos híbridos (combinando las técnicas anteriormente mencionadas). En este caso, se ha aplicado la técnica de fuzzing, que consiste en un conjunto de pruebas aleatorias uniformes, es decir, selecciona los datos que serán probados de manera aleatoria de acuerdo con la distribución de probabilidad uniforme y envía los mismos al conjunto de entradas del programa que será probado (Hernández, 2007).

Entre las vulnerabilidades encontradas se ha podido encontrar el no establecer parámetros de configuración, como son la cabecera anti-MIME-Sniffing header X-Content-Type-Options o la cabecera contra protección de ataques XSS; así como también el uso de métodos de riesgo potencial como son PUT y DELETE.

Por otro lado, el nombre de host '172.16.1.87' no concuerda con el nombre del certificado: Localhost.

El sitio web, al no tener establecidas la cabecera X-Frame-Options que no ha sido incluida en la respuesta HTTP es susceptible a ataques 'ClickJacking'.

Por naturaleza, APEX, mantiene sesiones con cookies. Si se cierran todas las pestañas, la sesión se termina. Pero cerrar solamente la pestaña de navegación no es suficiente, ya que puede mantener la sesión del usuario activa. Se sugiere que esta situación sea comunicada al usuario final.

### 2.8.2.3. Vulnerabilidad ante ataques de tipo XSS – Cross Site Scripting

Las pruebas de scripts (Cross-site Scripting - XSS) permiten identificar el cruce de variables de sesión entre una y otra página.

Debe considerarse establecer el parámetro de configuración de la cabecera a: header('X-XSS-Protection: 1;mode=block'), con el fin de evitar los ataques de tipo cross site scripting.

Como se había mencionado anteriormente, este tipo de vulnerabilidades se presentan siempre en aplicaciones que se ejecutan en navegadores o contenedores de sitios web, y es el XSS un vector de ataque que puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, poniendo en riesgo la integridad del sistema.

### 2.8.2.4. Vulnerabilidad ante ataques de inyección SQL

Las pruebas de inyección SQL, es un conjunto de técnicas o métodos de infiltración de código intrusivo que se aprovechan de una debilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.

La aplicación ERP desarrollada por la UDA, en su ambiente de desarrollo, no presenta vulnerabilidades relacionadas con esta práctica.

# GLOSARIO



# 3 Glosario

**APEX:** Oracle Application Express es una herramienta RAD (Rapid Application Development) que se ejecuta con una base de datos Oracle. Permite desarrollar prototipos de aplicaciones WEB de forma segura y rápida.

**Cabecera HTTP:** hace referencia a los parámetros que se envían en una petición o respuesta HTTP al cliente o al servidor, para proporcionar información esencial sobre la transacción en curso, y además es enviada automáticamente por el navegador.

**Exploit:** consiste en un mecanismo para explotar vulnerabilidades del objetivo de ataque; ya sea mediante el uso de fragmentos de software, fragmentos de datos o secuencias de comandos.

**Inyección SQL:** es un método de infiltración de código intruso que se aprovecha de una vulnerabilidad en un sistema informático presente en una aplicación, en el nivel de validación de las entradas para realizar operaciones sobre una base de datos.

**Meta tag:** se trata de una etiqueta que se encuentra en el código fuente de una página web y describe el contenido de la misma. Ejemplo: ‹meta http-equiv=”Content-Type” content=”text/html; charset=utf-8” /›.

**OWASP:** (Open Web Application Security Project) es un proyecto de código abierto cuyo objetivo es determinar y combatir las causas que hacen que un software sea inseguro. Su comunidad está conformada por empresas, organizaciones educativas y particulares de todo el mundo.

**Renderizado web:** es un software que toma el contenido HTML, XML, JPG, entre otros, e información de formateo (XSL, CSS), los

combina y visualiza el contenido resultante de esta acción en el navegador.

**Sandbox:** entorno de pruebas que se encuentra separado del entorno de producción.

**SAST - Static Application Software Testing:** es una herramienta para la prueba de software de aplicación estática, es decir, sin necesidad de ejecutar el código de la aplicación.

**Sniffer:** conocido también como analizador de tráfico, es un software que captura las tramas de una red de datos generada por sus múltiples protocolos.

**SQL:** es un lenguaje de consulta estructurada, específico del dominio que da acceso a la gestión de bases de datos relacionales.

**TAR:** formato de compresión de archivos usado en sistemas operativos UNIX y LINUX.

**Virtualización:** hace referencia al proceso de crear una representación basada en software (o virtual), en lugar de una física.

**X-Frame Options:** es un encabezado HTTP que puede ser usado para prevenir ataques de secuestro de clic, o establecer parámetros de renderización de una página.

**XSS:** el Cross Site Scripting es una vulnerabilidad que presentan las aplicaciones web que son aprovechadas por el atacante para introducir código Java Script o similares, evitando medidas de control.

**ZIP:** Formato de compresión de archivos comúnmente usado en sistemas operativos Windows.

# BIBLIOGRAFÍA



# 4 Bibliografía

Alcides, G. (2009). *Seguridad informática*. Antioquía, Colombia: Universidad de Antioquía.

Broad, J., & Bindner, A. (2013). *Hacking with Kali : Practical Penetration Testing Techniques*. USA: Syngress.

Burgos Salazar, J., & Campos, P. G. (2008). Modelo Para Seguridad de la Información en TIC. Concepción, Chile: Universidad del Bío-Bío.

Caballero, A. (2015). *Hacking con Kali Linux*. Lima.

Cordero Torres, G. (01 de 07 de 2015). Estudio comparativo entre las metodologías MAGERIT y CRAMM, utilizadas para análisis y gestión de riesgos de seguridad de la información. Cuenca, Ecuador.

Crespo, E. (2016). *Metodología de Seguridad de la Información para la gestión del Riesgo Informático aplicable a MPYMES*. Universidad de Cuenca, Cuenca.

Crespo, E. (2017). Ecu@Risk, Una metodología para la gestión de Riesgos. *Enfoque UTE*, 107-121.

Gallo, F. (2011). *Inseguridad Informática*. España.

Goldrath, E., & Abraham, (. (s.f.). *Insights en Operaciones V 5.1*.

Gómez Vieites, Á. (2011). *Enciclopedia de la Seguridad Infomática. México: AlfaOmega*.

Heizer, H., & Render, B. (2008). *Dirección de la Producción: Decisiones Tácticas*. Pearson Educación S.A.

Hernández, A. (2007). Fuzzing para pruebas de seguridad en software. *braioverflow.org*.

IEEE. (2008). *IEEE Standard for Software and System Test Documentation. IEEE Std 829™-2008 (Revision of IEEE Std 829-1998)*. IEEE Computer Society.

International Software Testing Qualifications Board (ISTQB). (2015). *Standard Glossary of Terms Used in Software Testing. Version 3.01. All Terms*. Obtenido de http://www.istqb.org/downloads/finish/20/193.html

Keck, J. (2005). A conceptual framework for next-generation enterprise systems? *Journal of Enterprise Information Management*.

Li, R., Abendroth, D., Lin, X., Guo, Y., Baek, H., Eide, E., . . . Merwe, J. (2015). POTASSIUM: penetration testing as a service. *ACM SoCC 2015 - Proceedings of the 6th ACM Symposium on Cloud Computing* (págs. 30-42). Salt Lake City, Utah: ACM.

Microsoft. (2017). *SDL Threat Modeling Tool*. Obtenido de https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

Myers, G. (2004). *The Art of Software Testing*. John Wiley & Sons.

NMAP. (19 de 04 de 2017). *Resumen de Opciones*. Obtenido de https://nmap.org/man/es/man-briefoptions.html

Ptak, C. A., & Schragenheim, E. (2000). *ERP Tools, Techniques and Applications for Integrating the Supply Chain*. APICS Serie.

Ptak, C. A., & Smith, C. (2011). Orlicky's Material Requirements Planning. McGraw-Hill Companies. Inc.

Rocha, L. (29 de Mayo de 2014). *Count Upon Security*. Obtenido de Incident Handling and Hacker Techniques: https://countuponsecurity.com/2014/05/29/simple-and-practical-attack-part-2/

Royer, J.-M. (2004). *Seguridad en la Informática de Empresa: riesgos, amenazas, prevención y soluciones*. Barcelona: Eni ediciones.

Sommerville, I. (2005). *Ingeniería de Software*. Madrid: Pearson Addison Wesley.

Spendolini, S. (2016). *Expert Oracle Application Express Security*. Apress.

Stefinko, Y., Piskozub, A., & Banakh, R. (2016). Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. *Modern Problems of Radio Engineering, Telecommunications and Computer Science, Proceedings of the 13th International Conference on TCSET 2016* (págs. 488-491). Ukraine: TCSET 2016.

Süer, G., Saiz, M., & Rosado-Varela, O. (1998). *KNOWLEDGE-BASED SYSTEM FOR MASTER PRODUCTION SCHEDULING*.

The OWASP Project. (2017). *OWASP Testing Guide 4.0*. N/E: The OWASP Project.

Vásquez, S., & López, D. (14 de 03 de 2016). Estudio comparativo entre las metodologías Microsoft Secure Risk Management y Octave. Cuenca, Azuay, Ecuador.

Weidman, G. (2014). *Penetration Testing, A Hands-On Introduction to Hacking (1)*. EEUU: No Starch Press.



ISBN: 978-9942-618-63-4