Pentesting sobre aplicaciones web basado en la metodología OWASP utilizando SBC de bajo costo

Andrés Felipe Muñoz Mayorga

Santiago Alejandro Pérez Solarte

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Programa de Ingeniería de Sistemas
Popayán, septiembre de 2017

Pentesting sobre aplicaciones web basado en la metodología OWASP utilizando SBC de bajo costo

Trabajo de grado presentado como requisito para obtener el título de Ingeniero de Sistemas

Andrés Felipe Muñoz Mayorga
Santiago Alejandro Pérez Solarte

Director: MSc. Siler Amador Donado

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Programa de Ingeniería de Sistemas
Popayán, septiembre de 2017

AGRADECIMIENTOS

En primer lugar, quiero agradecer a Dios por permitirme haber llegado a esta etapa tan importante de mi vida. A mi padre y a mi madre por su apoyo, afecto y cariño incondicional. Por ser un ejemplo de vida y darme fuerzas cuándo más las necesitaba.

A mi director Siler Amador Donado por orientarme y motivarme en este trabajo de investigación. Fue muy gratificante contar con su guía y apoyo para la culminación de este proyecto.

Gracias al grupo de investigación ALTENUA-MATDIS y al semillero de investigación CIBERSEGURIDAD por brindarnos los recursos materiales y económicos para la construcción del prototipo. Por aportar desde su conocimiento la comprensión matemática en este trabajo de investigación.

Gracias al ingeniero Carlos Alberto Ardila, al doctor Carlos Alberto Trujillo y el estudiante Juan Manuel Campo que desinteresadamente ayudaron en la propuesta y permitieron mejorar el trabajo presentado.

Andrés Felipe Muñoz Mayorga
Popayán, Septiembre de 2018

A mis padres por ser un ejemplo de vida y alentarme en la búsqueda del deseo de superación, resaltando el apoyo en los momentos de duda, desesperación, felicidad y por los que creí que no podía lograr.

Al director de tesis Siler Amador, por el grato recibimiento entre sus tesistas y por brindar esta oportunidad de desarrollar una nueva investigación dentro de la Universidad, por aportar su conocimiento en búsqueda de nuevas soluciones ante los problemas que se presentaron a lo largo de este proyecto de investigación.

A los ingenieros Carlos Ardila, Néstor Díaz, por brindar espacios de trabajo para debatir sobre conceptos claves que influyeron de manera positiva en este proyecto de investigación.

Santiago Alejandro Pérez Solarte
Popayán, septiembre de 2018

RESUMEN

Actualmente los servicios ofrecidos en internet están en aumento y evolución constante, tal aumento afecta la cantidad de datos a manejar, por lo cual se requiere que las empresas involucradas en este ambiente logren probar la seguridad en sus aplicaciones, servidores y activos de información. La manera de probar esto es mediante un pentesting, simulando el comportamiento de un atacante en un ambiente controlado, con el fin de descubrir vulnerabilidades en la organización. El objetivo de este trabajo de investigación es identificar de manera eficiente y eficaz las vulnerabilidades más comunes sobre las aplicaciones web, siguiendo la metodología OWASP. Se implementó un prototipo hardware y software sobre un cluster conformado por 6 dispositivos SBC de bajo costo, ejecutando tareas de manera paralela y distribuida para automatizar tanto el proceso del pentesting como la generación de reportes y determinar la severidad de los riesgos. Se realizó un experimento controlado sobre 5 sitios web, se compararon los resultados del prototipo contra cuatro herramientas especializadas en pruebas de penetración, con el fin de precisar la eficacia de las vulnerabilidades detectadas mediante un análisis de curvas ROC y medir la eficiencia del prototipo analizando los tiempos de ejecución. Como resultado se obtuvo un promedio de 12 vulnerabilidades en común que logró identificar tanto el prototipo como las herramientas seleccionadas en toda la prueba experimental. Para medir la eficiencia, el prototipo hardware y software en tiempo de ejecución se mantuvo en un rango entre los 740 a 2050 segundos aproximadamente para los 5 sitios web, siendo mejor que dos de las cuatro herramientas con las que se comparó. Finalmente, se puede concluir que las pruebas de concepto implementadas, identificaron las vulnerabilidades más comunes sobre todos los sitios web y se puede mejorar los tiempos de ejecución agregando más dispositivos SBC de bajo costo al implementar un cluster con las mismas características y configuración.

ABSTRACT

Nowadays the services offered on internet are increasing and in constant evolution, such increase affects the amount of data to manage, whereby is required that the companies in this environment achieve test security in your applications, servers and information assets. The way of test this is through a pentesting, simulating behavior of an attacker in a controlled environment, due to discover vulnerabilities in the organization. The objective of this researching project is to identify efficiently and effectively the most common vulnerabilities on the web applications, following the OWASP methodology. It implemented a prototype hardware and software on cluster conformed by 6 low cost SBC device, executing task in parallel and distributed way to automate as the process of pentesting as the generation of reports and determine the severity of the risks. It made a controlled experiment on 5 websites the results of the prototype were compared against four specialized tools in test of penetration, in order accuracy the efficient and effective of the vulnerabilities detected, through an analysis ROC curves y measure the efficient of the prototype analyzing the runtimes. As a result, an average of 12 common vulnerabilities was obtained which was identified as the prototype as the tools selected throughout the experimental test. To measure efficiency, the prototype hardware and software at runtime remained in a range between 740 to 2050 seconds approximately for the 5 websites, being better than two of the four tools with which were compared. Finally, it can be concluded that the proof of concept implemented, identified the most common vulnerabilities on all websites and you can improve the runtime by adding more low cost SBC devices when implementing a cluster with the same characteristics and configuration.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

1 CAPÍTULO I. INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Definición

El pentesting (prueba de penetración traducido al español) sobre aplicaciones web es un método utilizado para evaluar la integridad y seguridad de las aplicaciones mediante métodos de validación y verificación de la eficacia en los controles de seguridad que implementan las aplicaciones web [1]. En un alto nivel, esto significa, demostrar confidencialidad, integridad y disponibilidad de los datos, así como del servicio. El pentesting es el "arte" de probar el funcionamiento de las aplicaciones web de forma remota para encontrar vulnerabilidades de seguridad, sin conocer el funcionamiento interno de la aplicación en sí (prueba de caja negra). Normalmente, el equipo de pentesting tendría acceso a una aplicación como si fueran usuarios. El probador actúa como un atacante e intenta encontrar y explotar vulnerabilidades. En muchos casos se le dará una cuenta válida en el sistema [2].

Actualmente, muchas personas usan el pentesting como su principal técnica de prueba de seguridad, no se debe considerar como la única técnica de prueba. Gary McGraw en [3] resumió bien el pentesting cuando dijo: *"Si fracasas en una prueba, sabes que realmente tienes un problema muy grave. Si pasas una prueba, no sabes que problemas graves puedes tener en tu organización"*. Sin embargo, las pruebas de penetración enfocadas (es decir, las pruebas que intentan explotar vulnerabilidades conocidas detectadas en revisiones anteriores) pueden ser útiles para detectar si algunas vulnerabilidades específicas en realidad están en el código fuente de la aplicación web.

Un reporte a nivel global realizado por la firma DataSec analiza y presenta el estado de la seguridad sobre aplicaciones web analizadas desde el 1 de Abril del 2015 hasta el 16 de Marzo del 2016 [4]. Según los datos obtenidos reveló que las vulnerabilidades[1] con mayor ocurrencia y de alta preocupación fueron: Cross-Site Scripting (XSS[2]) y SQL Injection (SQLI[3]) con porcentajes del 25% y 35% respectivamente en el 2016. Según el documento OWASP Top 10 del año 2017 [5] las vulnerabilidades de SQLI y XSS proporcionan una serie de vectores de ataque que si son utilizados por un agente malicioso pueden causar un impacto negativo tanto a nivel técnico como para el negocio. Otra vulnerabilidad destacada es Vulnerable JavaScript Libraries, la cual duplica el porcentaje que tuvo en el año 2015, el cual fue del 15%.

[1] Agujero o debilidad de una aplicación, que puede ser por un mal diseño o configuración del entorno.
[2] Inserción de scripts maliciosos que son ejecutados por el navegador del usuario.
[3] Inserción o “inyección” de una consulta SQL que pretende generar un error en la aplicación.

A nivel continental se encuentran estadísticas durante la séptima Cumbre Latinoamericana de analistas de seguridad [6], donde el equipo de Kaspersky Lab dan a conocer que Brasil, México y Colombia son los países latinoamericanos con mayor número de ataques de malware en lo que va el 2017, con cifras de 53%, 17% y 9% respectivamente. La mayoría de los ataques fueron realizados vía web con una cifra alarmante del 85%, en otras palabras, ataques hacía servidores web compuestos por inserción de código malicioso, cuyo objetivo principal es el robo de datos e información sensible de las organizaciones. Mientras que el 15 % restante se realizó por medio de correo electrónico, donde se destaca el envío masivo de software malicioso como troyanos o ransomware que aprovechan las vulnerabilidades de los sistemas operativos para poderse propagar.

En el contexto colombiano, el 9 de enero del año 2018 fue lanzado el *ecenso* (censo online) por parte de la entidad estadística del país llamada Departamento Administrativo Nacional de Estadística, o por sus siglas DANE. Esto con el fin de saber cuántas personas viven en cada rincón del país, la condición en la que viven, la conformación del hogar, el nivel de educación, etc. Al paso de unos días, se dio a conocer una serie de vulnerabilidades sobre dicha aplicación web por medio de un weblog [7], donde hace hincapié al modo de cómo la aplicación almacena las contraseñas de los usuarios registrados, mencionando que la misma le muestra al usuario la contraseña en texto legible al momento de finalizar el registro y a la hora de recuperar la cuenta, siendo una falla grave, ya que las credenciales sensibles como lo es la contraseña de un usuario, deben ser previamente cifradas al menos con una función hash irreversible a la hora de guardar la información, para así evitar que agentes maliciosos puedan tener conocimiento de los datos de los usuarios.

Lo anterior indica que muchas organizaciones no realizan de manera adecuada el proceso de pruebas de seguridad dentro del ciclo de desarrollo del software sobre sus productos, dejando vulnerable a la organización ante posibles ataques, comprometiendo la integridad de los datos de los clientes y a las personas de la organización, incluso puede ocasionar pérdida de dinero y una mala reputación ante la sociedad.

1.1.2 Pregunta de investigación e hipótesis

¿Es posible identificar vulnerabilidades de manera eficiente y eficaz con la construcción de un prototipo hardware y software que automatice el pentesting sobre aplicaciones web utilizando dispositivos SBC[4] de bajo costo y que permita generar reportes de mediciones de riesgos de seguridad automáticos?

La hipótesis planteada en este trabajo de grado es la siguiente: E*s factible diseñar e implementar un prototipo hardware y software que permita identificar las vulnerabilidades más comunes sobre aplicaciones web, usando dispositivos SBC de bajo costo y siguiendo una metodología para determinar la severidad de los riesgos que posibilite generar reportes automáticos.*

[4] Single Board Computer (computador de placa simple de bajo costo).

1.1.3 Justificación

Con el rápido desarrollo de la tecnología en los últimos años, las aplicaciones web se han convertido en la principal aplicación utilizada en internet. Sin embargo, varios tipos de ataques contra las aplicaciones web han crecido rápidamente. Actualmente la mayoría de los sistemas tienen instalados firewalls, sistemas de detección y prevención de intrusos (IDS e IPS) y solo tiene abierto el puerto 80 del servidor web, pero no existe una solución fundamental para los problemas de seguridad en las aplicaciones web. Las amenazas sobre las aplicaciones web en internet incluyen ataques internos y externos, manuales y automatizados, maliciosos y no maliciosos, y así sucesivamente. Hay muchas vulnerabilidades sin solucionar hoy en día, que es la razón por la cual existen tantos ataques hacia las aplicaciones web [8].

La ausencia de implantación de controles de seguridad, hacen vulnerables a las aplicaciones web ante posibles amenazas[5], las cuales si son aprovechadas maliciosamente pueden ocasionar daños significativos a la integridad de los datos y activos de información, provocando un impacto negativo en la organización. Actualmente los servicios ofrecidos en el sector financiero (bancos, compañías de seguros, sociedades fiduciarias, etc.) están en aumento y evolución constante, sus transacciones se adaptan al contexto social, económico y tecnológico. Pero tal aumento afecta directamente el número de datos a manejar, la información confidencial, de los clientes y de la organización, por lo cual requiere que las organizaciones en este entorno puedan proteger la seguridad informática[6] y la seguridad de la información[7] [9].

Hoy en día existe software de pentesting automatizado que se puede encontrar de forma gratuita y de pago como: Acunetix, Nessus, Nexpose, Nikto, Burp Suite, OWASP ZAP entre muchas otras, las cuales permiten realizar pruebas a cualquier aplicación web y generar reportes automáticos, pero la mayoría de estas herramientas no identifican de manera eficiente y eficaz las vulnerabilidades descubiertas [10], lo que conlleva a realizar mayor esfuerzo para llevar a cabo esta tarea. Dichos problemas se presentan frecuentemente en las herramientas gratuitas por no tener un soporte oficial o las que lo tienen, presentan restricciones, como por ejemplo la cantidad de vulnerabilidades a encontrar o una licencia de prueba por un corto tiempo determinado, mientras que las herramientas de pago poseen un valor bastante elevado haciéndolas accesibles solo a empresas de tamaño grande. Las organizaciones de pequeño y mediano tamaño optan por el uso de herramientas gratuitas debido a que no cuentan con un presupuesto razonable para la adquisición de las herramientas que son de pago y para las capacitaciones.

Teniendo en cuenta lo anterior, es evidente que el uso de herramientas que automatizan el pentesting sobre aplicaciones web permiten ahorrar significativamente tiempo y costos en una organización, por tal motivo se ve la necesidad de diseñar un prototipo hardware y software con base en herramientas de código abierto utilizando dispositivos SBC de bajo costo como factor innovador gracias a su portabilidad y economía (ver tabla 1), que permitan identificar los tipos de ataques más comunes

[5] Evento cuya ocurrencia podría impactar en forma negativa en la organización.
[6] Infraestructuras de la tecnología de información y comunicación.
[7] Activos de información.

según el documento OWASP TOP 10, siguiendo una metodología de pentesting y otra metodología para el cálculo del riesgo que propone OWASP, con el fin de generar automáticamente reportes de pentesting sobre aplicaciones web.

Comparación Hardware		
	Raspberry PI 3 model B	**Lenovo ideapad 300**
Procesador	Quad Core 1.2GHz Broadcom BCM2837	6th Gen Intel Core i7-6500U (4M Cache, 2.5GHz)
GPU	VideoCore IV @ 300/400 MHZ	Radeon R5 M330 320 @ 1030 MHz
RAM	1GB	6 GB
Ethernet	10/100 Mbit/s	1000 Mbit/s
USB	4 puertos USB 2.0 + 1 micro USB para energía	2 puertos USB 2.0 + 1 puerto USB 3.0
Voltaje de carga	5V vía USB	100-240V vía AC Adapter
Dimensiones	85 x 56 mm	384 x 265 mm
Precio (COP)	$ 106.382,979	$ 1’914.863,22

Tabla 1 Comparación de dispositivos. Fuente: propia.

Es evidente que el uso de dispositivos SBC de bajo costo tiene una gran diferencia en tamaño y precio comparado ante un computador portátil. Además, sería un gran aporte en el área de pentesting, haciendo de estos dispositivos una gran alternativa para realizar dichas tareas. Aunque, su rendimiento es inferior con respecto a las demás características como la velocidad de procesamiento y capacidad de memoria RAM (por nombrar algunas), por tal motivo, el prototipo a implementar estará compuesto por dos o más dispositivos SBC (cluster[8]) de bajo costo que permita aumentar su rendimiento permitiendo realizar con normalidad las tareas de pentesting y al mismo tiempo, mantener su portabilidad y economía.

1.2 OBJETIVOS

1.2.1 Objetivo principal

Construir un prototipo hardware y software basado en dispositivos SBC de bajo costo, que automatice el pentesting sobre aplicaciones web, aplicando la metodología OWASP para identificar vulnerabilidades y generar reportes de mediciones de riesgos de seguridad automáticos.

1.2.2 Objetivos específicos

- Caracterizar el rendimiento de los dispositivos SBC de bajo costo para seleccionar los más apropiados en el contexto de pentesting sobre aplicaciones web.

[8] Conjunto de máquinas que trabajan juntas y que ejecutan la misma tarea bajo el control de un software.

- Diseñar e implementar un prototipo hardware y software, basado en la metodología OWASP que identifique los tipos de ataque más comunes siguiendo una metodología de pentesting.
- Aplicar la metodología de cálculo de riesgo según OWASP, con el fin de determinar la severidad de los riesgos identificados.
- Automatizar los reportes de mediciones de riesgos de seguridad en el pentesting sobre aplicaciones web, para generar el reporte técnico y ejecutivo.
- Evaluar el prototipo construido, para medir la eficiencia y eficacia en la identificación de vulnerabilidades mediante un experimento controlado.

1.3 METODOLOGÍA

El proyecto se desarrollará siguiendo la metodología de investigación-acción para ingeniería de software que propone [11]. Esta metodología propone dos ciclos, uno es el ciclo de la investigación y el otro es el ciclo de la solución al problema. Dichos ciclos tienen un comportamiento iterativo e incremental, además, contienen procesos que ayudan a realizar el proyecto de investigación. A continuación, en la Figura 1 es presentado de manera general la metodología de investigación a usar por medio de un diagrama de actividades.

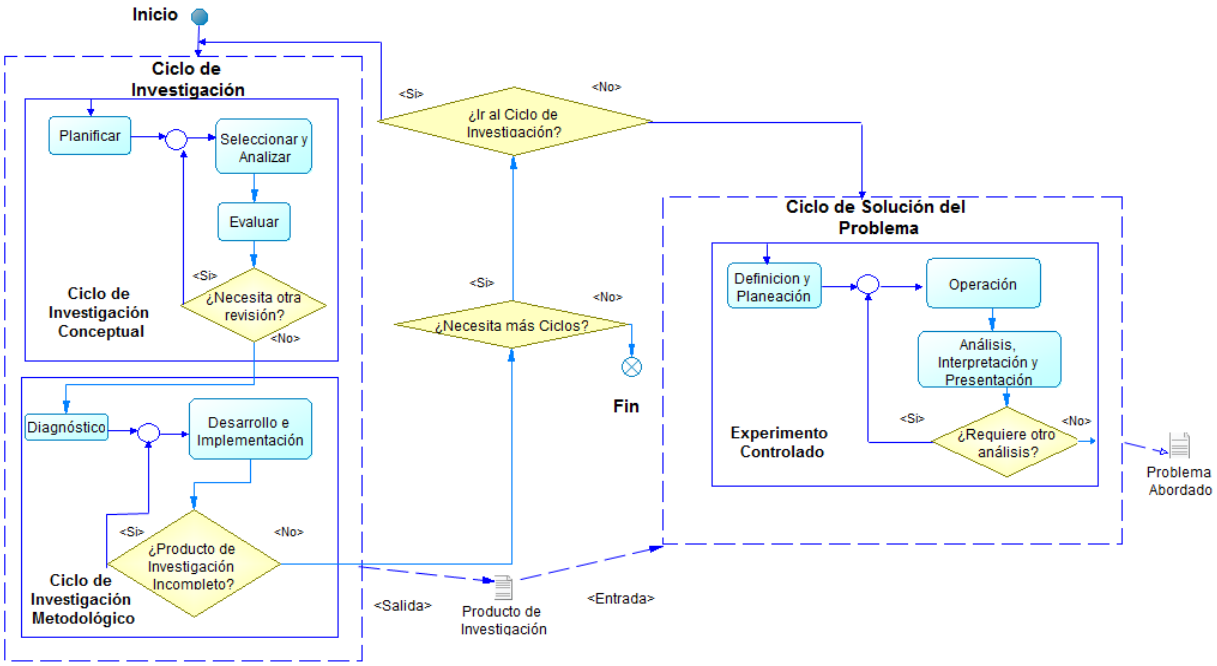


Figura 1. Diagrama de actividades metodología Investigación-Acción. Fuente: Adaptado de [11]

El ciclo de investigación está comprendido por dos ciclos adicionales que son: ciclo de investigación conceptual y ciclo de investigación metodológica. Además, dado que la metodología plantea tres procesos genéricos en cada ciclo para llevar a cabo una investigación (diagnóstico, acción y reflexión), se explicará en cada ciclo los procesos con nombres específicos y sus respectivas actividades.

Además, en este trabajo se incluye una etapa adicional relacionada con la Documentación, que se realizará en forma transversal a los ciclos mencionados y que

tendrá por objetivo la generación de la monografía y la elaboración del artículo científico.

1.4 Ciclo de investigación

1.4.1 Ciclo de investigación conceptual

Planificar: Este proceso se enfoca en la planeación del alcance y los objetivos que van a ser abordados en la revisión de la literatura. Definir el alcance de la investigación por medio de una pregunta de investigación. Estructurar las cadenas de búsqueda con sinónimos y operadores lógicos para que puedan ser procesadas sobre un motor de base de datos bibliográficas basada en la web, como por ejemplo Scopus. Por otra parte, se definen los criterios de inclusión y exclusión, para poder seleccionar los estudios obtenidos en: estudios primarios y estudios secundarios.

Seleccionar y analizar: Este proceso se enfoca en la recolección de estudios por medio de las cadenas de búsqueda planteadas anteriormente y en seleccionar los estudios primarios para la investigación. Para poder agrupar los resultados obtenidos, se aplican una serie de filtros para determinar qué estudios se consideran relevantes, los filtros son: eliminar estudios repetidos, aplicar los criterios tanto de inclusión como de exclusión. Analizar cada estudio considerado como primario para tener un punto de partida a la hora de realizar la propuesta que busca dar solución a la pregunta de investigación.

Evaluar: Este último proceso del ciclo de investigación conceptual está enfocado en garantizar que los estudios seleccionados hasta el momento sean acordes a la literatura. Debido a que la investigación será realizada por dos investigadores, se pretende realizar una reunión para resolver la discrepancia (si se presenta) en el análisis de los estudios y crear una lista final de los estudios que van a ser tomados. Evaluar los estudios mediante criterios de evaluación de la calidad, donde se aplican unos valores cuantitativos que indican si el estudio cumple con todos los criterios propuestos, por último, tabular los resultados respectivos de cada estudio.

1.4.2 Ciclo de investigación metodológico

Diagnóstico: Este proceso se enfoca en la recolección de información necesaria para poder llevar a cabo el desarrollo de la propuesta. Debido a que se utilizarán dispositivos SBC de bajo costo, se procede a realizar una caracterización de los dispositivos más usados en el ámbito del pentesting, con el fin de obtener las características más importantes de cada uno y si es factible su uso. Llevar a cabo una capacitación para identificar las características de los ataques según el documento OWASP Top 10. Analizar las diferentes herramientas que automaticen el pentesting para determinar qué tan eficientes y eficaces son a la hora de encontrar vulnerabilidades sobre aplicaciones web, además, determinar si siguen una metodología específica tanto para el pentesting como para el cálculo de riesgos.

Diseñar los respectivos diagramas para la construcción del prototipo hardware y software.

Desarrollo e implementación: En este proceso se lleva a cabo la implementación de la solución que busca resolver la pregunta de investigación. Construcción del prototipo hardware y software usando el dispositivo SBC de bajo costo seleccionado en el proceso anterior. Implementar la arquitectura de software y hardware. Seguir la metodología de riesgos que propone OWASP. Implementar la metodología de pentesting de OWASP. Desarrollar scripts que permitan automatizar el pentesting sobre aplicaciones web. Generar automáticamente los reportes técnico y ejecutivo.

1.5 Ciclo de solución del problema

Se requiere un método que guíe la solución del problema mediante el uso del producto de investigación en una situación real. Este método permite el análisis del enfoque de investigación desarrollado a fin de determinar su validez. Para ello se basó en un experimento controlado, el cual consiste en evaluar las diferentes opciones disponibles para lograr un objetivo, que muestre cómo las actividades genéricas del ciclo de solución del problema (diagnóstico, acción y reflexión) se pueden cumplir a partir de las actividades descritas para estos métodos de investigación [11]. Las actividades se realizarán así:

Definición y planeación: se define y planea el experimento. Esto incluye el personal y el ambiente, determinar las variables independientes (entradas) y las variables dependientes (salidas), selección de las PYMES, escoger el tipo de diseño, preparar la instrumentación (herramientas) y evaluación de validez.

Operación: consiste en tres principales pasos: preparación, ejecución y validación de datos. Se realizan pruebas funcionales y no funcionales al prototipo mediante una lista de chequeo que logren medir su rendimiento y funcionalidad.

Análisis, interpretación y presentación de datos recolectados: analizar los resultados con respecto al cálculo de riesgo empleando estadística descriptiva como medidas de tendencia central, medidas de dispersión, medidas de dependencia, determinar si fue posible verificar la propuesta planteada y una presentación de los hallazgos encontrados [12].

1.6 ESTRUCTURA DE LA TESIS

Capítulo I: Introducción. En este capítulo se presenta la problemática y la justificación que motivan la realización de este proyecto de investigación. Adicionalmente, se presentan los objetivos planteados y la estrategia utilizada para llevar a cabo el proyecto.

Capítulo II: Se describe el proceso de investigación que se llevó a cabo para la revisión de la literatura relevante acerca del pentesting sobre aplicaciones web,

presentando las actividades relacionadas de la revisión de la literatura, los criterios de selección y un análisis de los estudios con sus respectivos resultados.

Capítulo III: Se define, diseña y se modela una arquitectura para la implementación del cluster con los dispositivos SBC, se desarrollan scripts en el lenguaje de programación interpretado Python, se aplica la metodología según la guía de OWASP para realizar las pruebas de concepto, se calcula la severidad del riesgo a cada prueba realizada según la metodología de evaluación de riesgo de OWASP y se generan de manera automatizada el reporte tanto técnico como ejecutivo.

Capítulo IV: Se realiza un experimento controlado con las variables independientes y dependientes, se definen el grupo experimental y grupo de control, se establece el ambiente de pruebas en el cual se va a ejecutar la prueba experimental, se realiza un análisis mediante la curva ROC de los datos recolectados para determinar la eficacia en la detección de vulnerabilidades y se determina la eficiencia del prototipo hardware y software construido.

Capítulo V: Conclusiones y trabajos futuros. Se presentan las conclusiones obtenidas a partir de la realización del trabajo de investigación y posibles trabajos futuros. Como complemento al trabajo presentado, se presentarán los siguientes artefactos: (i) monografía del trabajo de grado, (ii) anexos, (ii) artículo técnico y (iv) un disco compacto con todos los artefactos mencionados anteriormente en formato digital.

2 CAPÍTULO II. REVISIÓN BIBLIOGRÁFICA

Se realizó una revisión bibliográfica de la literatura sobre un conjunto de bases de datos bibliográficas, con el fin de obtener artículos en inglés y español dentro de un intervalo de tiempo entre el año 2005 y 2017 que estén relacionados con el pentesting sobre aplicaciones web, automatización del pentesting, metodologías para el pentesting y estudios recientes acerca de la construcción de clusters usando dispositivos SBC de bajo costo. La estructura de este capítulo fue tomada de [13] y adaptada al contexto de esta investigación.

2.1 REVISIÓN DE LA LITERATURA DE PENTESTING SOBRE APLICACIONES WEB

2.1.1 Bases de datos y términos de búsqueda

Para realizar ésta investigación, se seleccionaron bases de datos que tuvieran las siguientes características: un motor de bases de datos bibliográficas basado en la

web, un motor de bases de datos bibliográficas capaz de realizar búsquedas filtrando por palabras claves y que contenga artículos relacionados con el área de seguridad de la información. Se realizó una revisión bibliográfica sobre los siguientes motores de bases de datos bibliográficas:

- Springer, sobre el tema de ciencias de la computación (Computer Science).
- Science@Direct, sobre el tema de ciencias de la computación (Computer Science).
- IEEE Xplore Digital Library.
- ACM Digital Library.
- SCOPUS.

Cabe resaltar que el motor SCOPUS recopila estudios que pueden ser encontrados en los demás motores de bases de datos bibliográficas (de ahora en adelante *fuentes bibliográficas*), la diferencia es la accesibilidad, ya que solo se puede acceder por medio de una entidad afiliada, en este se obtuvo acceso por medio de la Universidad del Cauca.

La lista de palabras clave usadas para encontrar una respuesta a la pregunta de investigación propuesta en el capítulo 1 son: *"penetration testing", "pentesting web", "penetration web application", "methodology", "methodologies", "testing tools", "suit", "web scanner", "vulnerability", "attack", "risk", "severity", "threat", "low cost computer", "single board computer", "sbc", "parallel computing", "distributed computing", "cost effectiveness," cluster computing".*

Tomando la lista de palabras claves vistas anteriormente como punto de partida y haciendo combinaciones de conectores lógicos como *"AND"* y *"OR"*, se obtuvieron 3 cadenas de búsqueda (ver tabla 1).

Con la **primera cadena** es posible encontrar estudios relacionados con las metodologías que pueden ser usadas para realizar un pentesting sobre aplicaciones web. Con la **segunda cadena** es posible encontrar estudios que hablen sobre el uso de herramientas para identificar vulnerabilidades y evaluar el riesgo asociado. Con la **tercera cadena** se obtienen los resultados con respecto a la construcción de clusters usando dispositivos SBC de bajo costo.

Cadenas de búsqueda	
1	(“penetration testing” OR “pentesting web” OR “penetration web application”) AND ((methodology OR methodologies OR “testing methodology”) OR framework OR standard OR process OR model) AND (nist OR osstmm OR owasp OR ptes OR issaf)
2	(“penetration testing” OR “pentesting web” OR “penetration web application”) AND ("testing tools" OR suit OR "web scanner") AND (vulnerability OR attack OR (risk OR severity) OR threat)
3	(“low cost computer” OR “single board computer” OR “sbc”) AND (“parallel computing” OR “distributed computing” OR “cost effectiveness”) AND (cluster computing)

Tabla 1. Cadenas de búsqueda utilizadas. Fuente: Propia.

2.1.2 Proceso de selección

El proceso de selección está dividido en 6 fases como se muestra en la figura 2:

1. **Búsqueda en las bases de datos**. Se listan las fuentes bibliográficas que se consideren necesarias, se construyen las cadenas de búsqueda con palabras clave y sus respectivos sinónimos, posteriormente se adaptan las cadenas a cada motor de búsqueda de las fuentes bibliográficas.
2. **Eliminar estudios repetidos**. Al realizar la búsqueda de estudios los resultados pueden ser redundantes, por tal motivo se eliminan dichos estudios y son almacenados en un gestor de referencias bibliográficas.
3. **Selección de estudios primarios**. A partir de los artículos encontrados se aplicaron los criterios de inclusión y exclusión (se lee el título, el resumen/abstract, introducción y conclusión si es necesario) los cuales son mencionados en la sección *“Criterios de selección”* para poder extraer los artículos primarios. Posteriormente cada estudio primario es almacenado.
4. **Selección final**. Una vez almacenado cada estudio primario, se procede a realizar una lectura completa de cada estudio.
5. **Eliminar inquietudes**. Si se presentan dudas e inquietudes una vez leídos los artículos, se acude a un experto en el tema para intentar resolver las diferencias.
6. **Evaluación de calidad**. Con base en los criterios de calidad que serán nombrados en la sección *Criterios de evaluación de calidad*, los estudios primarios son evaluados y clasificados con el fin de garantizar la evaluación apropiada de los estudios primarios.

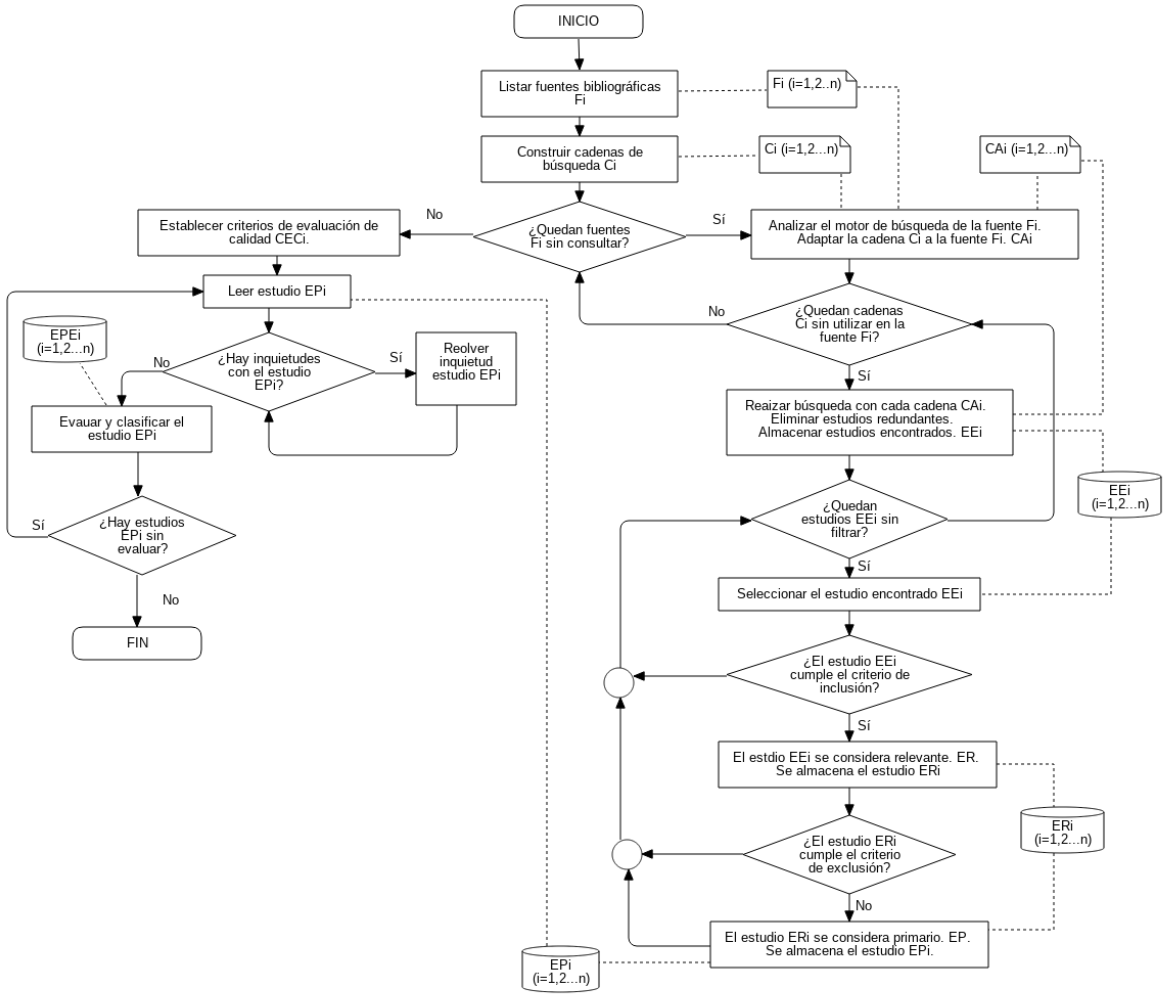


Figura 2. Proceso de selección para obtener estudios primarios. Fuente: Propia.

2.2 Criterios de selección

Una de las actividades esenciales durante la fase de planeación de esta investigación, es la definición de los criterios de inclusión y los criterios de exclusión. El objetivo de dichos criterios es ayudar a los investigadores, en el momento de seleccionar los artículos apropiados, y se emplea para reducir la cantidad de trabajos que serán analizados.

Para la selección de estudios relevantes, el *criterio de inclusión* se basó en un análisis del título, resumen y palabras claves de los artículos obtenidos en la búsqueda, ésto con el fin de determinar si están relacionados con el pentesting y si se centra en el ámbito de las aplicaciones web, si el artículo menciona el uso de algún modelo, proceso, framework o metodología para realizar el pentesting sobre aplicaciones web, y artículos relacionados con la construcción de clusters usando SBC de bajo costo.

Para determinar qué estudios eran lo suficientemente importantes para considerarlos como estudios primarios, el *criterio de exclusión* fue el encargado de excluir artículos que tuvieran los siguientes aspectos: el estudio presenta un proceso o metodología del pentesting sobre aplicaciones web pero no provee suficiente información acerca

de su uso o de su aplicación, si el estudio no presenta un tipo de evaluación para demostrar los resultado con respecto a las herramientas utilizadas para realizar el pentesting, si el artículo no contempla los ataques más comunes según el documento de OWASP TOP 10, si el estudio identifica vulnerabilidades en aplicaciones web pero solamente lo realizan de manera manual y si el estudio menciona el uso de SBC de bajo costo, pero se centra más en lo teórico que en lo práctico.

2.3 Criterios de evaluación de calidad

El objetivo de los criterios de evaluación de calidad es asegurar una apropiada evaluación de cada estudio que fue considerado primario. Los criterios de evaluación establecidos son:

- **CEC 1**. ¿El estudio presenta algún tipo de contribución: metodología, técnica, herramienta, enfoque, modelo, método, estrategia o framework?
- **CEC 2**. ¿El estudio presenta algún método de investigación basado en el análisis y descripción como: estudio empírico, estudio experimental, pruebas de concepto, teórica o caso de estudio?
- **CEC 3**. ¿El estudio menciona y aplica el tipo de contribución planteado?
- **CEC 4**. ¿El estudio realiza un análisis de los resultados obtenidos?

Para cada uno de los criterios de evaluación de calidad, aplicamos la siguiente valoración: S (sí) = 1, P (parcialmente) = 0.5, N (no) = 0. De esta manera, el resultado total para la evaluación de cada estudio (CEC1 + CEC2 + CEC3 + CEC4) puede resultar de la siguiente manera: 0, 0.5 y 1 (incompleto) 1,5 y 2 (regular), 2,5 (bueno), 3 (muy bueno) y 3,5 y 4 (excelente).

Para evaluar cada estudio primario, se establecieron reglas a cada criterio de evaluación, esto con el fin de complementar la parte cualitativa a la valoración:

- CEC 1. S, el estudio propone el uso o una nueva metodología, framework, modelo, técnica o herramienta; P, la contribución está presente pero no se describe claramente; N, la contribución no puede ser identificada o no está establecida.
- CEC 2. S, el estudio menciona que ha aplicado explícitamente algún método de investigación; P, el estudio presenta información relevante pero no especifica el método de investigación; N, el método de investigación no puede ser identificado o no está descrito.
- CEC 3. S, el estudio presenta de forma detallada el tipo de contribución que ha sido llevado a cabo; P, el tipo de contribución llevado a cabo es representado de forma breve; N, el estudio no describe claramente el tipo de contribución llevado a cabo.
- CEC 4. S, el estudio presenta un análisis detallado que explique los resultados obtenidos; P, los resultados son explicados brevemente; N, los resultados no pueden ser identificados o no están descritos.

2.4 Estudio de evaluación de calidad

Inicialmente se identificaron 97 estudios en los motores de bases de datos bibliográficas, estos de ahora en adelante se consideran como **encontrados**. Al eliminar los estudios redundantes, 84 estudios fueron considerados como **no repetidos**. Con el criterio de inclusión, 45 estudios fueron considerados como **relevantes**, mediante la lectura del título, resumen y palabras claves. Los 45 estudios fueron leídos completamente y con el criterio de exclusión, se obtuvieron 26 **primarios** (mirar tabla 2).

Bases de datos	Estudios			
	Encontrados	No repetidos	Relevantes	Primarios
Springer	10	6	4	2
Science@Direct	8	8	6	5
IEEE Xplore Digital Library	25	22	10	6
SCOPUS	46	43	20	10
ACM Digital Library	8	5	5	3
Total	97	84	45	26

Tabla 2. Resultados arrojados por las fuentes bibliográficas. Fuente: Propia.

A continuación, la tabla 3 muestra el resultado total de los estudios primarios luego de aplicar la evaluación de calidad. Cada estudio es enumerado por la columna *ID*, los nombres de los estudios primarios son presentados en la columna *Nombre del estudio* con su respectivo año de publicación en la columna *Año de la publicación*. Las columnas compuestas por *CEC (Criterio de Evaluación de Calidad)* corresponden a la valoración obtenida de los criterios de evaluación. Las columnas *Cuantitativo* y *Cualitativo*, muestra el resultado final de cada criterio.

ID	**Nombre del estudio**	**Año de la publicación**	**CEC**				**Calidad**	
			1	**2**	**3**	**4**	**Cuantitativo**	**Cualitativo**
1	Web security in the finance sector [9]	2017	S	S	P	S	3,5	E
2	Overview and open issues on penetration test [13]	2017	S	P	S	S	3,5	E
3	Large-Scale Analysis & Detection of Authentication Cross-Site Request Forgeries [14]	2017	S	S	S	S	4	E
4	Knowledge-based security testing of web applications by logic programming [15]	2017	S	S	P	S	3,5	E
5	Improving penetration testing through static and dynamic analysis [16]	2017	S	P	P	S	3	MB

6	Improving Penetration Testing Methodologies for Security-Based Risk Assessment [17]	2017	S	S	S	P	3,5	E
7	WebGuardia - An integrated penetration testing system to detect web application vulnerabilities [18]	2016	P	P	S	P	2,5	B
8	Analyzing the traffic of penetration testing tools with an IDS [19]	2016	P	S	P	p	2,5	B
9	An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0 [20]	2015	S	N	P	P	2	R
10	WAPTT - Web Application Penetration Testing Tool [21]	2014	S	P	P	P	2,5	B
11	A survey on web penetration test [22]	2014	S	P	S	S	3,5	E
12	Improving the Efficiency and Effectiveness of Penetration Test Automation [23]	2013	S	P	P	p	2,5	B
13	A comparison of the efficiency and effectiveness of vulnerability discovery techniques [24]	2013	S	S	S	S	4	E
14	Automated security test generation with formal threat models [25]	2012	S	S	P	S	3,5	E
15	An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection [26]	2011	S	S	P	P	3	MB
16	Towards a practical and effective security testing methodology [27]	2010	P	P	P	P	2	R
17	Evaluating single board computer clusters for cyber operations [28]	2017	S	S	S	S	4	E
18	Employing miniaturized computers for distributed vulnerability assessment [29]	2017	S	S	P	P	3	MB
19	Cybergrenade: Automated Exploitation of Local Network Machines via Single Board Computers [30]	2017	S	S	P	P	3	MB
20	Mobile clusters of single board computers: an option for providing resources to student projects and researchers [31]	2016	S	P	S	S	3,5	E
21	Building Supercomputer With Raspberry Pi [32]	2015	S	N	S	P	2,5	B
22	Iridis-pi: A low-cost, compact demonstration cluster [33]	2014	S	P	S	S	3,5	E
23	Affordable and Energy-Efficient Cloud Computing	2013	S	N	P	N	1,5	R

	Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment [34]							
24	A low-cost computer cluster for high-performance computing education [35]	2014	S	P	N	P	2	R
25	Security testing methodology for vulnerabilities detection of XSS in web services and WS-security [36]	2014	S	S	S	P	3,5	E
26	Security Testing Methodology for Evaluation of Web Services Robustness - Case: XML Injection [37]	2014	S	S	P	P	3	MB

Leyenda: S: sí; N: no; P: parcialmente; CEC: Criterio de evaluación de calidad; E: excelente; MB: muy bueno; B: bueno; R: regular; I: incompleto.

Tabla 3. Resultados de cada estudio primario con respecto a los criterios de evaluación de calidad. Fuente: Propia.

2.5 Análisis de Resultados

Para realizar la definición de algunos aspectos en el proceso de la revisión de la literatura se tomó en cuenta las palabras claves, los conceptos y el contexto de la investigación para una comprensión más detallada de cada estudio seleccionado, donde cada actividad proporciona la identificación de los siguientes aspectos: (i) escenarios de destino: aplicaciones web, dispositivos SBC de bajo costo, software y aplicaciones y aplicaciones web vulnerables; (ii) tipo de investigación: estudio empírico, estudio experimental, pruebas de concepto, experiencia industrial, teórica y caso de estudio;(iii) tipo de contribución: metodología, técnica, herramienta, enfoque, modelo, método, estrategia, framework y revisión de la literatura. Esta clasificación se puede ver en las figuras 3, 4 y 5.

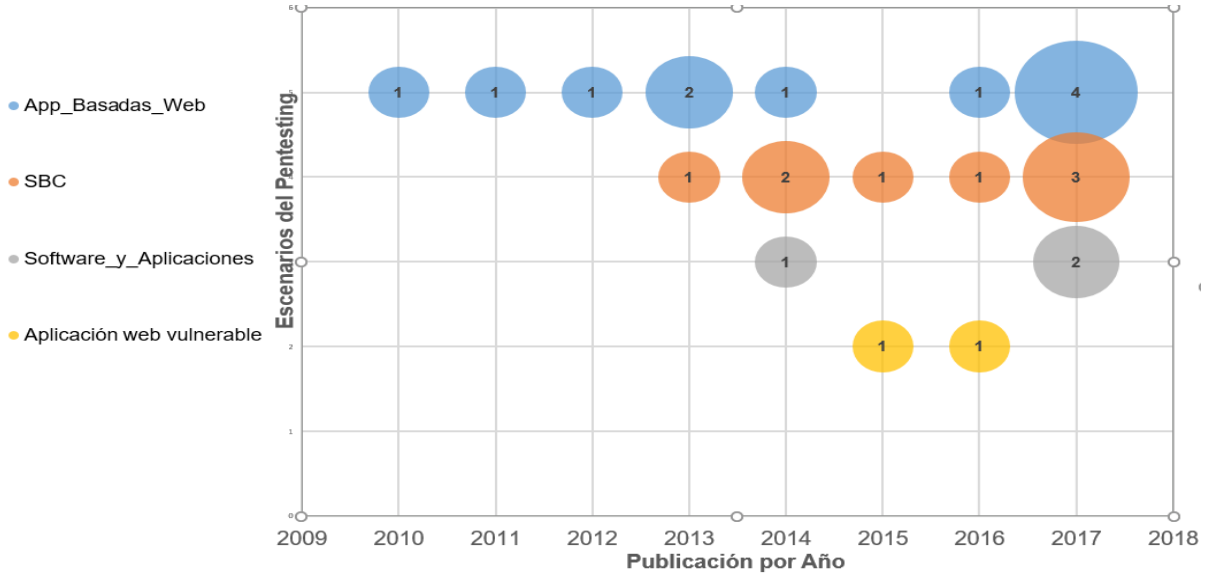


Figura 3. Diagrama de Burbujas de los Estudios y Escenario por Año. Fuente: Adaptado de [13]

Esta sección presenta una evaluación cuantitativa con respecto a los contextos donde se podrá llevar a cabo el proyecto de investigación. La figura 3 muestra un gráfico de burbujas con la distribución de los escenarios en relación con el año de publicación, donde el tamaño de la burbuja indica la cantidad de estudios relacionados con los anteriores aspectos mencionados en cada intersección de los ejes.

Los estudios se agrupan por años, por lo que es posible visualizar cómo se desarrolló el pentesting en los últimos años. Se puede observar en la figura 3 que el pentesting se ha aplicado con mayor frecuencia en el contexto de aplicaciones web en los últimos años. Por lo tanto, muestra que los escenarios de aplicaciones web es uno de los principales temas de investigación actualmente. Sin embargo, se puede ver como otros escenarios siguen siendo relevantes, por ejemplo, los dispositivos SBC de bajo costo

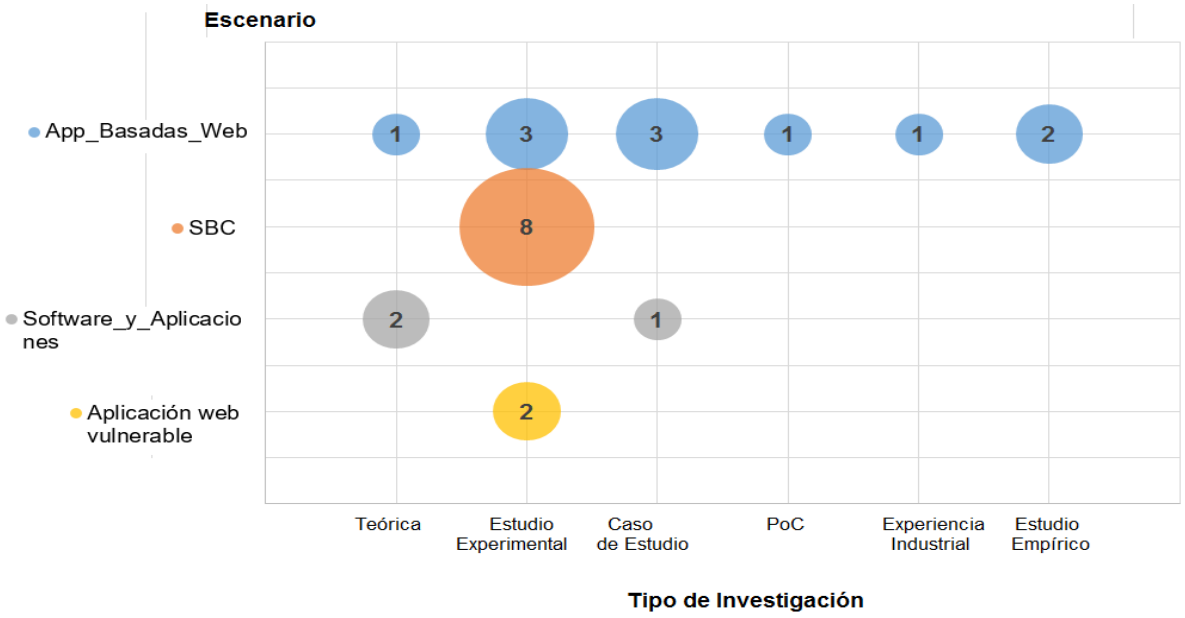


Figura 4. Diagrama de Burbuja de los Escenarios vs Tipo de Investigación. Fuente: Adaptado de [13]

La figura 4 muestra la relación entre el escenario de destino y el tipo de investigación. La mayoría de estudios primarios seleccionados se analizaron y caracterizaron como estudios experimentales. Este resultado apunta a que normalmente los trabajos de investigación se aplican a un área específica, buscando obtener el mejor resultado a través de una causa-efecto y no son estrategias generales que se pueden aplicar a cualquier contexto.

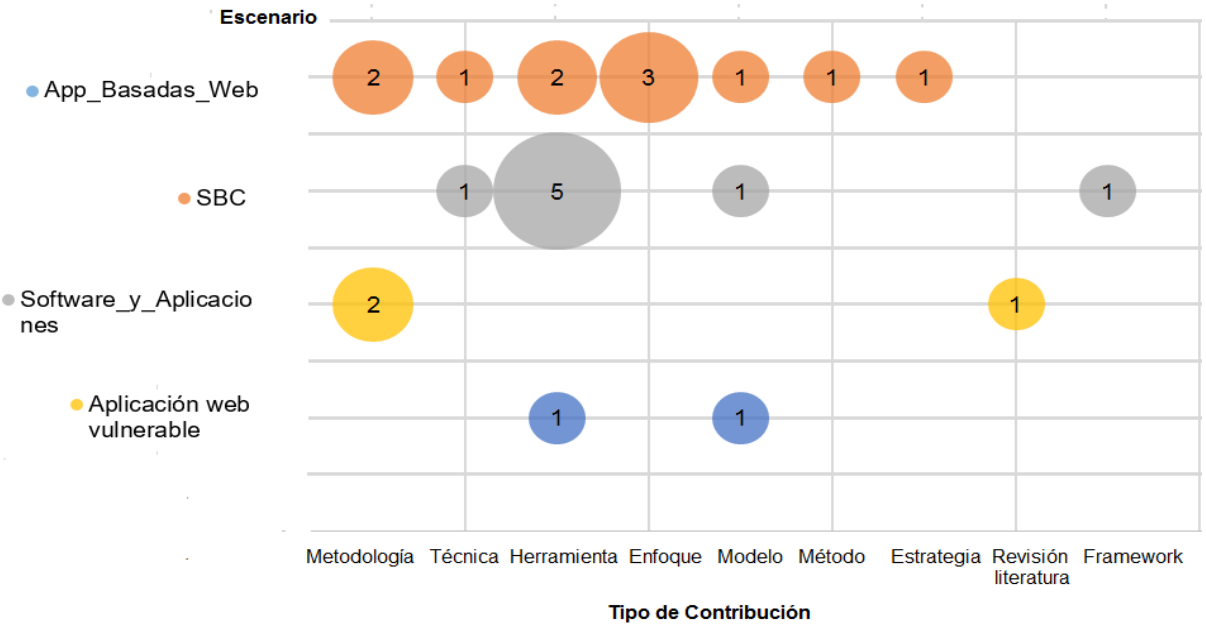


Figura 5. Diagrama de Burbuja de los Escenarios vs Tipo de Contribución. Fuente: Adaptado de [13]

Mientras tanto en la figura 5 muestra la relación entre el escenario y el tipo de contribución. Por lo tanto, se puede apreciar que la mayoría de estudios utilizan o desarrollan algún tipo de herramienta para llevar a cabo su trabajo de investigación en un determinado contexto. Este resultado parece coherente con la forma en que normalmente se desarrollan los trabajos en el área del pentesting, es decir, los trabajos tienden a desarrollar o aplicar alguna herramienta para poder evaluar o realizar el pentesting sobre alguna aplicación web.

2.6 Amenazas a la validez

Las principales amenazas identificadas que pueden comprometer la validez de los estudios primarios durante el proceso de selección son:

1. **Sesgo de publicación:** se refiere a la posibilidad de que algunos artículos no se seleccionen porque el proceso de búsqueda no arrojó los resultados deseados o porque la investigación se realizó sobre temas que no encajan en las conferencias, ponencias, revistas y artículos.

2. **Sesgo de selección de estudios primarios:** no se puede garantizar que todos los estudios primarios seleccionados fueron los ideales durante el proceso de búsqueda y evaluación. En este sentido, los criterios de calidad establecidos, así como la asignación de puntajes apuntan a mitigar esta amenaza.

3. **Falta de familiaridad con otros campos:** se definieron tres cadenas de búsqueda basadas en el conocimiento y experiencia de los autores, pero no se puede evitar por completo que algunos términos definidos en las cadenas de búsqueda tengan sinónimos que no se hayan identificado. Para minimizar esta

amenaza se adaptó y se refino cada cadena a las las bases de datos bibliográficas hasta que se encontraran los mejores resultados.

4. **Sesgo de selección de los estudios relevantes:** para la selección de artículos relevantes se pudo haber descartado algunos estudios, debido a que el análisis se basó en el título, resumen y palabras claves de los artículos obtenidos en la búsqueda, obviando el material o método del artículo.

2.7 CONCEPTOS PRELIMINARES

2.7.1 Pentesting

El pentesting es un proceso de evaluación para redes o sistemas informáticos que prueba los mecanismos de seguridad de las organizaciones o empresas al simular múltiples ataques en situaciones en las que se agrega nueva infraestructura, se instala software, se aplican actualizaciones al sistema, se instalan los parches de seguridad y se modifican las políticas de usuario. Algunas de las principales razones para adoptar las pruebas son: problemas de seguridad, protección de información, priorizar riesgos de seguridad y pérdida financiera. El objetivo del pentesting es incrementar la seguridad en los activos, datos y servicios de información. La seguridad de la información y las debilidades que se identifican en la prueba se consideran confidenciales y no se divulgarán hasta la resolución completa de las fallas encontradas. El pentesting es uno de los métodos más antiguos para evaluar la seguridad de un sistema informático. A principios de la década de los 70's, el Departamento de Defensa utilizó este método para demostrar las debilidades de seguridad en los sistemas informáticos e iniciar el desarrollo de programas para crear sistemas más seguros. El pentesting se puede hacer de forma manual o automática. La prueba manual requiere un equipo experto y experimentado para controlar todo y deben estar físicamente presentes en la duración de la prueba. El pentesting automático es una forma simple y segura de realizar todas las tareas relacionadas con la prueba. Además, dado que la mayoría del trabajo se realiza automáticamente, es más económico en términos de tiempo [22].

2.7.2 Prueba de concepto (Proof of concept - PoC)

Etimológicamente según la RAE (real academia española), *prueba* es: la razón, argumento, instrumento u otro medio con que se pretende mostrar y hacer patente la verdad o falsedad de algo, mientras *concepto* tiene como significado: idea que concibe o forma el entendimiento, por lo que se puede entender literalmente como una implementación de un método o idea (relativamente corto) realizada con el propósito de verificar que un concepto es susceptible de ser llevada a la práctica. En el área de la seguridad de la información, el concepto son las vulnerabilidades que se quiere encontrar y la prueba es una porción de código (de cualquier lenguaje de programación), que al ser ejecutado sobre un target se puede explicar la forma de cómo se puede explotar dicha vulnerabilidad.

Para mostrar un ejemplo sencillo de una prueba de concepto, se tomó como referencia la vulnerabilidad identificada como CVE-2017-7472 que permite a un usuario local realizar una denegación de servicio (DoS) a sistemas linux con versiones del kernel igual o inferiores a 4.10.13.

```
/*
Source: https://bugzilla.novell.com/show_bug.cgi?id=1034862
QA REPRODUCER:
gcc -O2 -o CVE-2017-7472 CVE-2017-7472.c -lkeyutils
. /CVE-2017-7472
(Will run the kernel out of memory)
*/
#include <sys/types.h>
#include <keyutils.h>
int main()
{
   for (;;)
      keyctl_set_reqkey_keyring(KEY_REQKEY_DEFL_THREAD_KEYRING);
}
```

El anterior código fue tomado de [38].

2.7.3 Black box test

La exploración de vulnerabilidades web de caja negra es una técnica que se ha adoptado ampliamente debido a la facilidad de uso, la automatización y la independencia de la tecnología utilizada en las aplicaciones web, no requiere obtener el código fuente y puede usarse repetidamente con un bajo costo. Se realiza cuando no se tiene un conocimiento específico de la red, simplemente se simula un atacante que no sabe absolutamente nada, más allá de una dirección IP. Tiende a probar las vulnerabilidades de seguridad de una aplicación web y generar informes y estadísticas sobre la hallazgos encontrados [26].

2.7.4 White box test

Esta prueba define claramente los objetivos para analizar, así como las tecnologías implementadas, usuarios y contraseñas, direccionamiento IP y políticas de seguridad, de los cuales se tiene un pleno conocimiento para elaborar vectores de ataque mucho más específicos [39]. Las pruebas de caja blanca se centran principalmente en la lógica interna y la estructura del código, resulta compleja y su costo es bastante alto [40].

2.7.5 Gray box test

La prueba de caja gris representa el término medio entre la caja negra y la caja blanca, en la que la cantidad de información sobre el objetivo no está completa pero tampoco es inexistente. Esta prueba define como información previa algunos objetivos, usuarios, o algunas tecnologías implementadas dentro de la red, se dice que el pentester solo tiene información parcial sobre la aplicación que va a probar [39].

2.7.6 OWASP (proyecto)

El Proyecto abierto de seguridad en aplicaciones web (OWASP) es una comunidad libre y abierta en todo el mundo centrada en mejorar la seguridad del software de aplicaciones. Todas la herramientas, documentos, foros y capítulos de OWASP son gratuitos y abierto a cualquiera interesado en mejorar la seguridad de aplicaciones web. El objetivo de este proyecto es recopilar todas las técnicas de prueba posibles, explicar estas técnicas y mantener actualizada la guía. El método de prueba de seguridad de aplicaciones web de OWASP se basa en el enfoque de caja negra. El tester no sabe nada o tiene poca información de la aplicación que será probada.

La misión es hacer que la seguridad de las aplicaciones sea "visible", para que las personas y las organizaciones puedan tomar decisiones sobre los riesgos de seguridad de las aplicaciones. Hay muchas formas de probar los defectos de seguridad y el proyecto OWASP capta la idea de los principales expertos sobre cómo realizar esta prueba de forma rápida, precisa y eficiente [2].

2.7.7 SBC

Un SBC (Single Board Computer) puede ser definido como un ordenador completo integrado en un único circuito. Incluye el microprocesador, la memoria, la unidad de procesamiento de gráficos y el resto de componentes de un SoC (Sistema en Chip), a los que se añaden otros elementos propios de una computadora, como unidades de almacenamiento y de entrada/salida (puertos USB, RJ-45 y otros) [41]. Los SBC se pueden ocultar en muchas áreas pequeñas para evitar la detección visual, lo que los hace perfectos para usar como máquinas de explotación [42].

2.8 METODOLOGÍAS Y ESTÁNDARES DEL PENTESTING

Hoy en día se puede afirmar que existen varias metodologías y estándares desarrolladas por empresas que permiten realizar un pentest, sin embargo, cada una de ellas fue elaborada para abarcar un objetivo en específico, lo que quiere decir que cada una tiene características diferentes. El autor en [23] afirma que, cualquier metodología o estándar que esté relacionado con el pentesting debe dar por lo menos la estructura general con respecto a los pasos que se deben seguir para realizar un pentest, por tal motivo, varios estudios se han enfocado en la caracterización de las

diferentes metodologías existentes con el fin de poder determinar cuáles son las mejores dentro de una área en específico (por ejemplo: Pentesting sobre aplicaciones web).

2.8.1 Metodologías existentes

A partir de los estudios primarios en la sección 2.1, se lograron identificar las siguientes metodologías, frameworks y estándares de seguridad:

- ISSAF (The Information System Security Assessment Framework).
- NIST SP 800-115 (National Institute of Standards and Technology).
- OSSTMM (Open Source Security Testing Methodology Manual).
- OWASP (Open Web Application Security Project).
- PTES (The Penetration Testing Execution Standard).

2.8.1.1 ISSAF v0.2.1

La metodología ISSAF en su versión 0.2.1 provee un framework diseñado especialmente para evaluar la seguridad en redes, sistemas y controles de aplicaciones. Su estructura se divide en tres áreas o fases principales para llevar a cabo el pentesting, las cuales son [27]:

1. **Planeación y preparación**, consiste en establecer el entorno de prueba, las herramientas a usar, aspectos legales, requisitos y la estructura del informe final.
2. **Evaluación**, esta fase se considera la esencia de la metodología ISSAF, ya que se divide en nueve subfases y es la encargada de proveer el paso a paso para realizar el pentesting, lo que a grandes rasgos se conoce como reconocimiento, explotación y post-explotación.
3. **Reporte, limpieza y destrucción de artefactos**, es la última fase de la metodología ISSAF, con la que concluye el pentesting realizando un reporte completo y bien estructurado. Además, se procede a destruir los artefactos utilizados durante todo el pentesting, con el fin de evitar la divulgación de información sensible que pueda comprometer a la organización.

En la figura 6 se muestra de forma gráfica la metodología descrita [43].

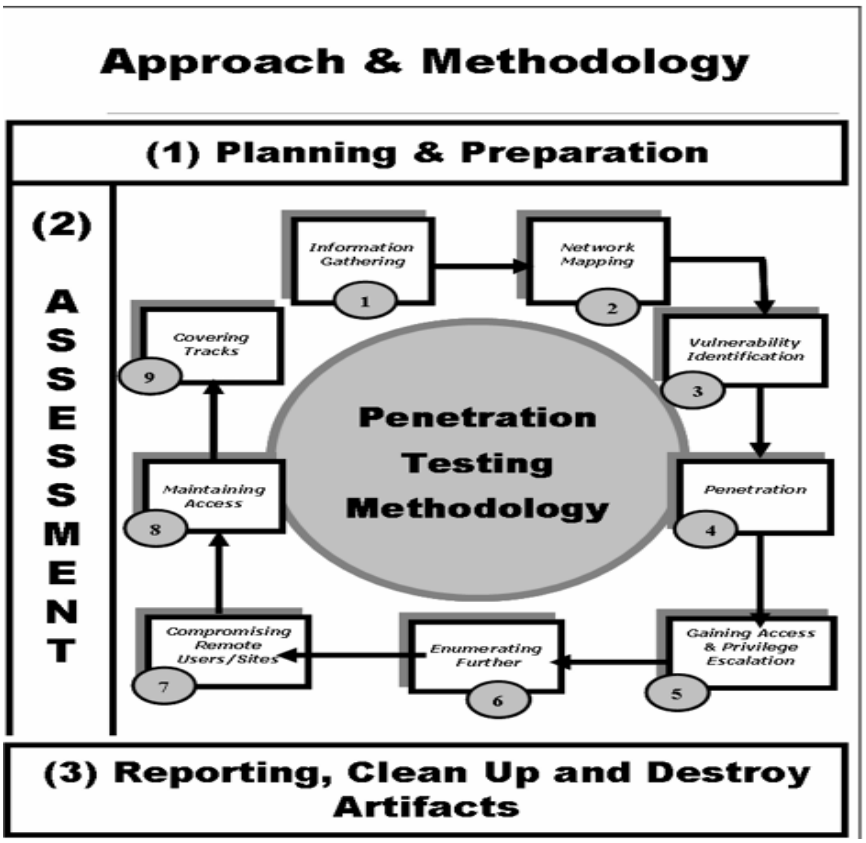


Figura 6. Actividades de la metodología ISSAF. Fuente: Tomado de: [23]

2.8.1.2 NIST SP 800-115

La metodología para pentesting propuesta por la *National Institute of Standars and Technology* (NIST) se encuentra en su versión SP 800-115 con el nombre de *"Technical Guide to Information Security Testing and Assessment"*. Para llevar a cabo el pentesting la metodología sigue cuatro fases como se puede observar en la figura 7 [44]. A continuación, se procede a explicar de forma general cada una de las fases.

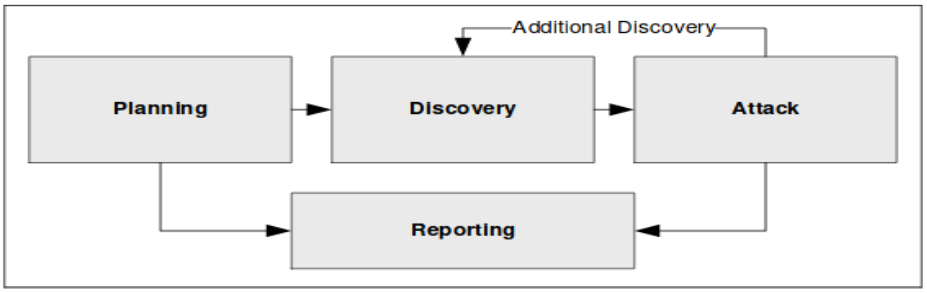


Figura 7. Estructura general de la metodología propuesta por la NIST. Fuente: Tomado de [24]

1. **Planeación**, es el primer paso de la metodología y es la encargada de definir los objetivos del pentesting, así como, la definición de reglas, alcance y aprobación.
2. **Descubrimiento**, esta fase hace referencia a todo lo relacionado con la recolección de información acerca del *target*. A su vez, esta fase se divide en

dos subfases que corresponden a la búsqueda de información relevante como lo es el nombre del host, información de la dirección IP, nombres y versiones de los servicios que el *target* está ejecutando. La segunda subfase hace referencia al análisis de vulnerabilidades, donde el pentester puede usar herramientas automáticas y bases de datos propias o públicas para la búsqueda de exploits.

3. **Ataque**, es la fase más importante, ya que se realizan las pruebas reales. El ataque como la fase **Descubrimiento** está dividida en más subfase, en este caso son cuatro, las cuales tienen como nombre: obtener acceso, escalar privilegios (usuarios), navegar por el sistema e instalar herramientas adicionales para mantener la prueba en el tiempo.
4. **Reporte**, última fase de la metodología si se mira con respecto a la enumeración, pero esta fase se realiza en paralelo con la **Planeación** y **Ataque**. Para realizar un reporte bien documentado, la metodología provee pasos para la mitigación de riesgos y recomendaciones tanto técnicas como no técnicas.

2.8.1.3 OSSTMM 3

OSSTMM en su versión 3 es una metodología diseñada y mantenida por la ISECOM (Instituto Superior de Empresa y Comunicación), a diferencia de las metodologías descritas, esta posee dos tipos de documentos: por un lado, está el manual completo dónde describe toda la documentación necesaria para seguir la metodología. Por otro lado está el manual en su versión resumida o *lite*, dónde hace énfasis en el flujo de trabajo de la metodología [45]. A continuación, se procede a explicar los pasos que provee esta metodología (mirar figura 8).

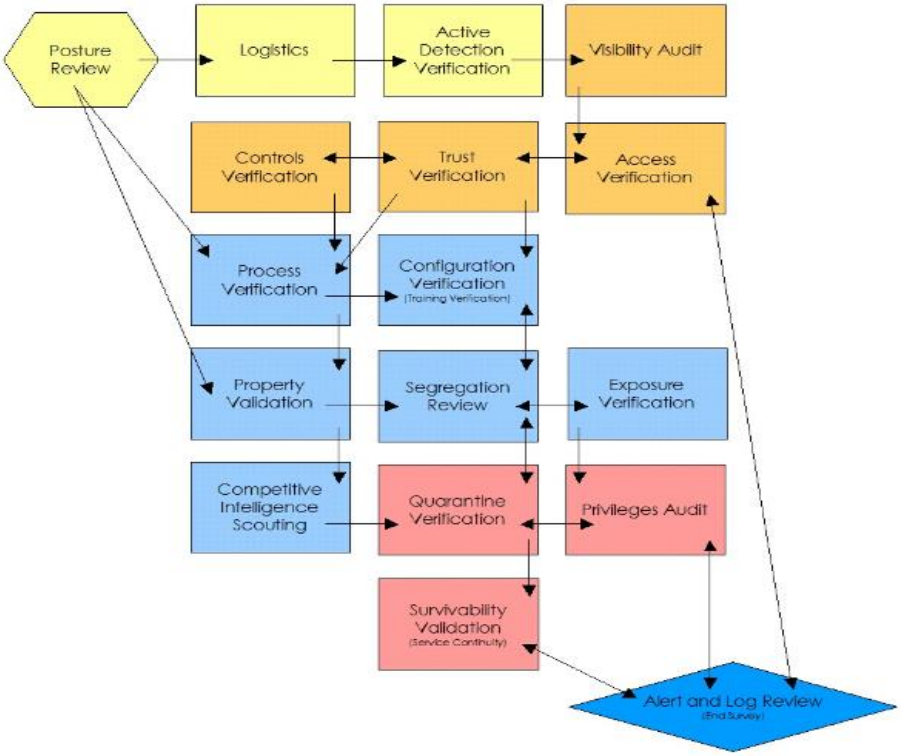


Figura 8. Flujo de la metodología OSSTMM. Fuente: Tomado de: [25]

El pentesting comienza con la definición del alcance. El alcance consta de tres tipos de clases: COMSEC (communications security), PHYSSEC (physical security) y SPECSEC (spectrum security), además, dichas clases se dividen en cinco canales antes de ser utilizadas por el pentester, las cuales son: humanos, físicos, inalámbricos, telecomunicaciones y redes de datos [45] (mirar tabla 4). Estos canales son usados para llevar a cabo el pentesting y cada uno tiene una especificación única para la evaluación de seguridad de acuerdo al escenario de prueba, siguiendo el flujo de la figura 8.

Área	Canal	Descripción
PHYSECC	Humano	Comprende el elemento humano de la comunicación donde la interacción es física o psicológica
	Físico	Pruebas de seguridad física donde el canal es de naturaleza física y no electrónica. Comprende el elemento tangible de seguridad donde la interacción requiere esfuerzo físico o un transmisor de energía para manipular.
SPECSEC	Comunicación inalámbrica	Comprende todas las comunicaciones electrónicas, señales y emanaciones que tienen lugar sobre el espectro electromagnético conocido. Esto incluye ELSEC como comunicaciones electrónicas, SIGSEC como señales y EMSEC, que son emanaciones sin ataduras por cables.
COMSEC	Red de datos	Comprende todos los sistemas electrónicos y redes de datos donde la interacción se lleva a cabo a través de líneas establecidas de cable y red cableada.
	Telecomunicaciones	Comprende todas las redes de telecomunicaciones, digitales o analógicas, donde la interacción se lleva a cabo a través de líneas telefónicas establecidas o líneas telefónicas similares.

Tabla 4. Descripción de los canales de la metodología OSSTMM. Fuente: Tomado de [25]

2.8.1.4 OWASP Testing Guide v4

OWASP provee un framework (ver figura 9) que puede ser usado en cualquier organización que se dedique al desarrollo de software, el cual puede ser visto como un punto de referencia que comprende actividades y tareas a realizar sobre las fases del ciclo de desarrollo de software [2]. El framework busca concientizar a los equipos de desarrollo a realizar pentesting sobre sus productos para mejorar la calidad de ellos, y se basa en otros proyectos como la guía de revisión de código (Code Review Guide) y la guía de desarrollo (Development Guide).

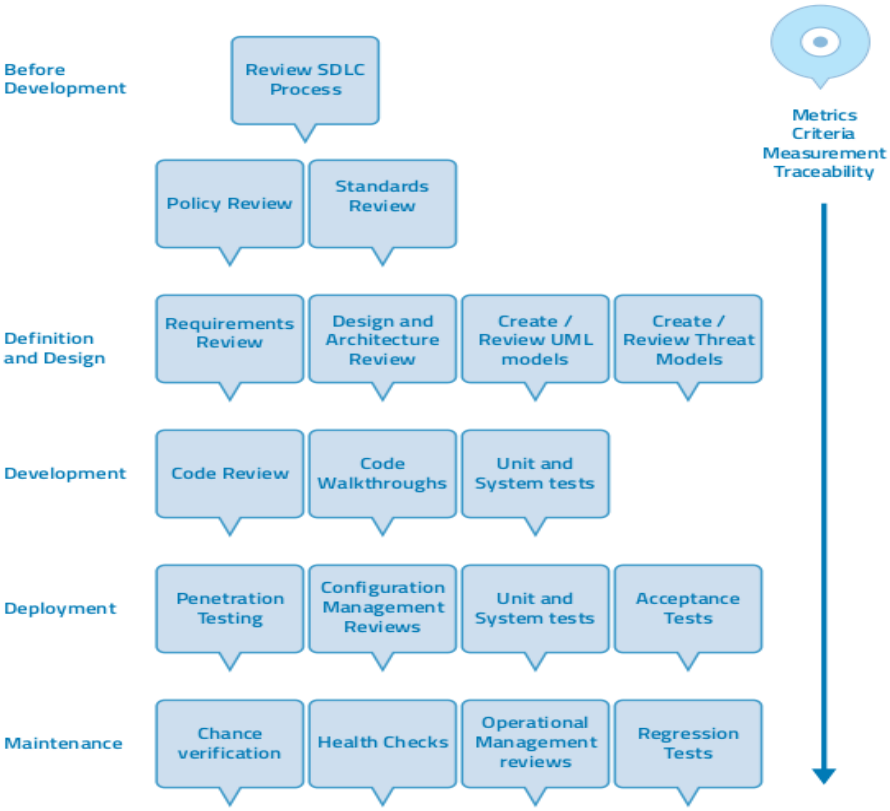


Figura 9. Framework propuesto por OWASP. Fuente: Tomado de [2]

Para este caso, el enfoque está dirigido hacia el pentesting que se encuentra en la actividad de **despliegue** (Deployment), por esa razón, OWASP desarrolló una metodología llamada OWASP Testing Guide que actualmente se encuentra en su versión 4, donde describe once subcategorías dando a conocer una descripción general de la misma, un resumen de cada prueba contenida en las subcategorías, el objetivo de la prueba, ejemplos de cómo realizar la prueba, algunas recomendaciones para evitar o mitigar el riesgo y por último lista las herramientas que pueden ser utilizadas para cada prueba.

Las subcategorías son las siguientes:

1. **Recolección de información:** esta categoría es de gran importancia, ya que brinda la fase inicial para poder comprender la configuración del servidor y la lógica de la aplicación. Se pueden utilizar herramientas como, un proxy HTTP, cookies, robots y huellas digitales para reunir información sobre las tecnologías, el mapa de rutas, sistema operativo, puertos abiertos y servicios asociados a cada puerto, sobre la aplicación y la configuración del servidor objetivo. Las plataformas de aplicaciones son amplias y variadas, pero algunos errores clave de configuración de la plataforma pueden comprometer la aplicación, de la misma manera que una aplicación no segura puede comprometer al servidor.

2. **Prueba para la gestión de configuración y despliegue:** la complejidad intrínseca de una infraestructura de servidor web interconectada y

heterogénea, que puede incluir cientos de aplicaciones web, hace de la gestión de la configuración y revisión un paso fundamental en la prueba e implementación de cada aplicación. Esta sección permite conocer la configuración de la aplicación, infraestructura de red, archivos backup en búsqueda de información sensible y métodos HTTP.

3. **Prueba para la gestión de identidad:** esta sección permite identificar los diferentes roles y permisos que poseen los usuarios en la aplicación web, así como realizar de manera correcta el proceso cuando se registra un usuario. Además, es posible enumerar el usuario y contraseña dependiendo de los mensajes de error que posea la aplicación en el momento de validar las credenciales.

4. **Prueba de autenticación:** la autenticación es el proceso de intentar verificar la identidad digital del remitente de una comunicación. Un ejemplo común de tal proceso es el proceso de inicio de sesión. Probar el esquema de autenticación significa entender cómo es el funcionamiento lógico de la aplicación web, para usar esa información y eludir este mecanismo como, por ejemplo: ataques de fuerza bruta, escalamiento de privilegios, aprovechar las credenciales por defecto y capturar la información que no es transmitida por un canal seguro.

5. **Prueba de autorización:** mediante esta subcategoría es posible acceder a recursos en línea que normalmente se pueden encontrar ocultos en el documento raíz que se encuentra alojados en el servidor. La autorización es el proceso, el cual se podrán validar las credenciales, analizar la configuración de roles y privilegios. Durante este tipo de evaluación, se podrá verificar si es posible evitar un esquema de autenticación, las cuales se hacen mediante las listas de control de acceso (ACL), también, se pueden encontrar vulnerabilidades en archivos transversales o encontrar la manera de escalar los privilegios.

6. **Prueba para la gestión de sesiones:** uno de los principales componentes de una aplicación web es el mecanismo por el cual controla y mantiene el estado para un usuario, está definido como el conjunto de controles que rigen cuando el usuario se encuentra interactuando. Esto abarca en general desde que el usuario está autenticado hasta que cierra sesión, como analizar si el sistema posee un control para cerrar sesión automáticamente cuándo el usuario se encuentre inactivo.

7. **Prueba de validaciones de entradas:** la debilidad más común de las aplicaciones web es no validar correctamente las entradas provenientes del usuario o del entorno antes de usarlas. Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones web, como ataques XSS, inyección de SQL, inyección de intérpretes, ataques al sistema de archivos y desbordamientos de búfer. Hay que desconfiar de todas las entradas en la aplicación web, tanto en formularios, campos de búsqueda como tener precauciones cuando el usuario ingresa una URL. Esa es la regla número uno, desafortunadamente las aplicaciones complejas a menudo tienen una gran

cantidad de puntos de entrada, lo que dificulta que un desarrollador aplique esta regla.

8. **Manejo de errores:** en esta subcategoría se logran encontrar los códigos de error generados por las aplicaciones o servidores web. Es posible hacer que estos errores se muestren mediante el uso de peticiones particulares, diseñadas manualmente o elaboradas con herramientas automáticas, con el fin de obtener información sobre bases de datos, componentes tecnológicos y códigos de estado como 1xx, 2xx, 3xx, 4xx y 5xx.

9. **Criptografía:** los datos sensibles deben estar protegidos cuando se transmiten a través de la red. Tales datos pueden incluir credenciales de usuario y tarjetas de crédito. Como regla general, si los datos deben protegerse cuando se almacenan, también deben estar protegidos durante la transmisión, para lograr esto es común utilizar el protocolo HTTPS para todo el tráfico que se genere en la aplicación web, así como utilizar los certificados digitales para permitir la autenticación entre el cliente y el servidor.

10. **Prueba de lógica de negocio:** este tipo de vulnerabilidad no puede ser detectado fácilmente por un escáner de vulnerabilidades, depende de las habilidades y la creatividad del pentester. Además, este tipo de vulnerabilidad suele ser específica de la aplicación, pero, al mismo tiempo, suele ser una de las más perjudiciales, si se explota. Este tipo de pruebas requieren desarrollar casos de abuso y utilizar muchas de las técnicas para realizar pruebas funcionales.

11. **Prueba de lado del cliente:** las pruebas del lado del cliente se refieren a la ejecución del código en el cliente, normalmente de forma nativa dentro de un navegador web o complemento del navegador. La ejecución del código en el lado del cliente es distinta de la ejecución en el servidor y las respuestas que genera este. Estas pruebas pueden explotarse cuando un usuario es capaz de tomar el control de un punto de entrada e inyectar código malicioso logrando modificar el contenido de la aplicación web.

2.8.1.5 PTES v1.0

El propósito de esta metodología no es establecer patrones complejos y rígidos para realizar un pentest, lo que propone son una serie de instrucciones que permiten realizar el proceso de evaluación de seguridad en un entorno específico. Por lo tanto, las normas técnicas ayudan a definir los procedimientos a seguir durante el pentest, permitiendo que la metodología provee una estructura básica para su inicio y ejecución. Las fases que presenta la metodología PTES son las siguientes [46]:

1. **Interacciones previas al compromiso,** esta fase propone definir el alcance, los objetivos específicos que se van a lograr en el pentest, los contratos y es aquí donde el pentester determina qué tipo de prueba es más conveniente (black-box, white-box o gray-box).

2. **Recolección de inteligencia**, el objetivo de esta fase es obtener y enumerar toda la información posible acerca del *target*.
3. **Modelo de amenazas**, esta fase se encarga de analizar los vectores de ataque que se pueden realizar según la información obtenida en la fase anterior, además de entender las contramedidas que permitan mitigar el riesgo si los ataques son materializados.
4. **Análisis de vulnerabilidades**, una vez que se hayan verificado los vectores de ataque, se recomienda realizar tanto pruebas automatizadas como manuales, con el fin de tener una mejor precisión en los resultados. Según [23] propone el uso de la herramienta automática Nessus para realizar esta fase.
5. **Explotación**, cuando se haya recolectado toda la información necesaria de las anteriores fases, se procede con la búsqueda y/o modificación de exploits en bases de datos propias o públicas para lograr obtener acceso en el *target*.
6. **Post-explotación**, esta fase tiene como objetivo mantener el acceso de la fase anterior y encontrar más fallas en el *target*, hasta que el tiempo fué estipulado en la fase 1.
7. **Reporte**, una vez se haya finalizado con toda la prueba, se procede a realizar un reporte técnico y ejecutivo. El reporte técnico contempla todos los hallazgos, herramientas y artefactos que fueron utilizados a lo largo de la prueba. EL reporte ejecutivo contiene los resultados de manera gráfica y sin tecnicismos, debido a que va dirigido al gerente o a la persona del área administrativa.

En la figura 10 se muestra de forma gráfica la metodología descrita.

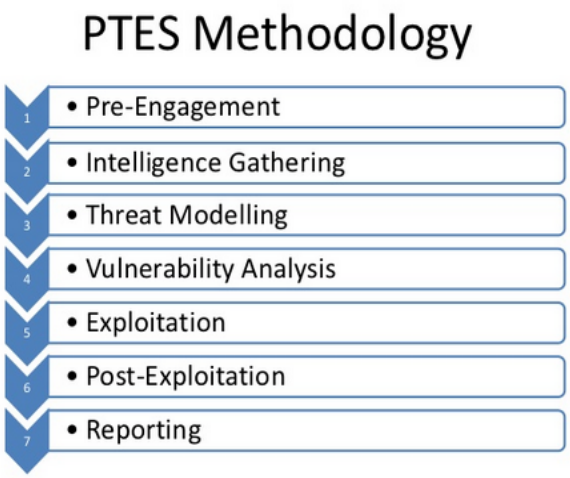


Figura 10. Actividades de la metodología PTES. Fuente: Tomado de [26]

2.8.2Comparación de las metodologías

Siguiendo las características fundamentales para la selección de una "*buena*" metodología propuestas por [27], se pretende identificar mediante las definiciones y conceptos proporcionados, la metodología de pentesting adecuada para llevar a cabo la propuesta de este trabajo de investigación. La tabla 5 muestra el resultado final de

la comparación. A continuación, se muestra la clasificación cualitativa utilizada para categorizar las metodologías:

- Aplica (A): La metodología proporciona definiciones y conceptos adecuados para dicha característica.
- Aplica parcialmente (AP): La metodología menciona definiciones y conceptos con respecto a la característica, pero sin una solidez necesaria.
- No aplica (NA): La metodología no menciona nada con respecto a la característica.

Soporte. Cuando es usada una tecnología para el desarrollo de soluciones informáticas, se debe tener en cuenta la versión y el soporte que brinda el framework o el lenguaje, así mismo es el caso de las metodologías. Una metodología que no es actualizada constantemente puede generar inseguridad al momento de su selección, ya que, algunos componentes o conceptos hayan cambiado y/o sean inutilizables con el tiempo. Para esta característica se determinó un valor numérico de diferencia en términos de años de la fecha actual y la fecha de lanzamiento de la versión estable de la metodología, el valor establecido fue de 5 años. OWASP Testing Guide v4 fue la única metodología que cumplió esta característica, debido a que cada 3 o 4 años es lanzada una versión estable de la metodología.

Planeación. La metodología debe proporcionar la ayuda necesaria al pentester para establecer la planificación adecuada de cada prueba a realizar. Por lo tanto, la planeación debe incluir la definición de las fases propias de la metodología, las herramientas que deben ser usadas en cada fase y el rendimiento esperado para cada actividad dentro de la prueba. Tanto como PTES, ISSAF y OWASP Testing Guide v4, describen de manera puntual las actividades que se deben realizar con respecto a la planeación del pentest como lo objetivos y el alcance que debe tener la prueba, así mismo, las herramientas que se deben usar en las actividades posteriores. En cambio, NIST SP 800-115 incluye tal característica, pero no tiene el suficiente detalle como las anteriores mencionadas.

Cobertura. Los conceptos y modelos que propone una metodología no deben ser ambiguos, pero no es excusa para limitar el alcance de este. Se refiere como alcance a la posibilidad de adaptar la metodología a diferentes escenarios como, por ejemplo: pentesting sobre redes, pentesting sobre aplicaciones web, etc. La única metodología que se ve limitada a esta característica es OWASP Testing Guide v4, ya que está enfocada a las aplicaciones web y servicios web, en cambio, OSSTMM, ISSAF, PTES y NIST SP 800-115 pueden ser adaptados a otro tipo de alcance.

Modelado. Como fue mencionado en la *cobertura*, la metodología debe evitar la ambigüedad en los conceptos y modelos que deba ser utilizadas en actividades posteriores. Los conceptos ayudan al pentester a modelar el flujo de acciones que se deben tomar a lo largo de las pruebas, además de modelar el sistema y el entorno del objetivo (target). OSSTMM, OWASP Testing Guide v4, ISSAF y PTES cumplen parcialmente con esta característica cuando se encuentran en la etapa de planificación de cada proceso de prueba.

Documentación. Finalmente, se puede considerar la documentación como una característica importante en un pentest, ya que no debe limitarse a las fases activas

como la explotación o post-explotación, sino que debe extenderse al registro de la configuración y entorno de la prueba, el progreso y la elaboración de los reportes finales con los hallazgos. Una metodología que ayude al pentester a realizar reportes no solo deja que evite la omisión de detalles, sino también a la redacción adecuada de la información para diferentes lectores (técnicos, ejecutivos). PTES, NIST SP 800-115 y OWASP Testing Guide v4 son las metodologías que no proveen la suficiente información como para elaborar una documentación que contenga y explique en detalle el proceso de cada actividad, así como la generación del reporte técnico y ejecutivo.

Metodología		**OWASP Testing Guide v4**	**PTES**	**ISSAF**	**NIST SP 800-115**	**OSSTMM**
Organización		OWASP	--	OISSG	NIST	ISECOM
Características	Soporte	A	NA	NA	NA	NA
	Planeación	A	A	A	AP	AP
	Cobertura	NA	AP	AP	AP	A
	Modelado	AP	AP	AP	NA	AP
	Documentación	AP	AP	A	AP	A

Tabla 5. Comparación de metodologías para el pentesting. Fuente: Adaptado de [27]

Con respecto a lo anterior, no existe una metodología que satisfaga todas las necesidades, unas tienen mejores características que otras, pero no significa que sea mejor ya que su enfoque es distinto. El autor en [47] afirma que si un pentester no utiliza una metodología que se adapte a las necesidades de la organización, los resultados acabarían en: un pentest incompleto donde no se logren abarcar todos los requerimientos que demanda el pentest, un desgaste tanto de tiempo, dinero y esfuerzo.

2.9 TIPOS DE ATAQUES MÁS COMUNES SEGÚN OWASP TOP 10

El software inseguro está debilitando las finanzas, salud, defensa, energía, y otras infraestructuras críticas. A medida que el software se convierte en algo crítico, complejo e interconectado, la dificultad de lograr la seguridad en las aplicaciones aumenta exponencialmente [48]. El ritmo vertiginoso de los procesos de desarrollo de software actuales, incrementa aún más el riesgo de no descubrir vulnerabilidades de forma rápida y precisa. Para identificar los tipos de ataques más comunes existe el proyecto OWASP TOP 10, el cual aborda los riesgos de seguridad de aplicaciones más importantes que enfrentan las organizaciones en la actualidad.

El OWASP Top 10 – 2017 [5] se basa principalmente en el envío de datos de más de 40 empresas que se especializan en seguridad de aplicaciones y una encuesta de la industria que fue completada por más de 500 personas. Esta información abarca vulnerabilidades recopiladas de cientos de organizaciones y más de 100.000

aplicaciones y APIs del mundo real. Las 10 principales categorías fueron seleccionadas y priorizadas de acuerdo con estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto. Uno de los principales objetivos del OWASP Top 10 es educar a los desarrolladores, diseñadores, arquitectos, gerentes y organizaciones sobre las consecuencias de las debilidades más comunes y más importantes de la seguridad de las aplicaciones web

Los nombres de los riesgos en el Top 10 derivan del tipo de ataque, el tipo de debilidad y el tipo de impacto que causan. A continuación, se presentan los nombres que reflejan con precisión los riesgos y están alineados con el marco de las debilidades del CWE para representar la terminología común y aumentar el nivel de conciencia sobre ellos:

- **A1: Inyección:** las fallas de inyección como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. El atacante puede modificar o crear una sentencia maliciosa para engañar al intérprete y ejecutar código arbitrario en la máquina objetivo o acceder a los datos sin la debida autorización [48].

- **A2: Pérdida de Autenticación:** Las funciones relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente en las aplicaciones, permitiendo a los atacantes comprometer usuarios y contraseñas, token de sesiones, o explotar fallas de implementación para asumir la identidad de otros usuarios temporal o permanentemente.

- **A3: Exposición de datos sensibles:** Muchas aplicaciones y APIs no protegen adecuadamente datos sensibles, tales como información financiera, de salud o Información Personalmente Identificable (PII). Los atacantes pueden robar o modificar estos datos para llevar a cabo fraudes con tarjetas de crédito, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales, como el cifrado, así como tener precauciones cuando se navega a través de los diferentes sitios web.

- **A4: Entidades Externas XML (XXE):** Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la red de área local (LAN), ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).

- **A5: Pérdida de Control de Acceso:** Las restricciones sobre el conjunto de privilegios de los usuarios autenticados no se aplican de manera adecuada. Los atacantes pueden explotar estos defectos para acceder de forma no autorizada, a funcionalidades y/o datos, cuentas de otros usuarios, ver archivos sensibles, modificar datos, cambiar derechos de acceso y permisos.

- **A6: Configuración de Seguridad Incorrecta:** La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, ad hoc o por omisión (o directamente por la falta de configuración). Son ejemplos: S3 buckets abiertos,

cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, frameworks, dependencias y componentes desactualizados, entre otros.

- **A7: Secuencia de Comandos en Sitios Cruzados (XSS):** Los XSS ocurren cuando una aplicación obtiene datos no confiables y los envía al navegador web sin una validación y codificación apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta JavaScript en el navegador. Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (defacement) los sitios web, o redireccionar al usuario hacia un sitio malicioso.

- **A8: Deserialización Insegura:** Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.

- **A9: Componentes con vulnerabilidades conocidas:** Los componentes como bibliotecas, frameworks y otros módulos se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, el ataque puede provocar una pérdida de datos o tomar el control del servidor. Las aplicaciones y API que utilizan componentes con vulnerabilidades conocidas pueden debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.

- **A10: Registro y Monitoreo Insuficientes:** El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotar a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos.

El OWASP Top 10 cubre una gran cantidad de riesgos, pero existen otros como Cross-Site Request Forgery (CSRF), Unvalidated Forward and Redirects, que debería considerar y evaluar en cada organización. Algunos de éstos se han publicado en versiones previas del Top 10, y otros no, incluyendo nuevas técnicas de ataque que son identificadas constantemente.

2.10 CÁLCULO DE MEDICIÓN DE RIESGOS

Los atacantes pueden potencialmente utilizar diferentes rutas a través de su aplicación para perjudicar una empresa u organización. Cada uno de estos caminos representa un riesgo que puede o no ser suficientemente grave como para merecer atención. En la figura 11 se puede observar el flujo de un riesgo de seguridad en las aplicaciones web, la cual surge a través de un ataque que ha aprovechado una vulnerabilidad y es capaz de evitar los controles de seguridad que se hayan

implementado, afectando así a un recurso o función del sistema, causando un impacto negativo en la organización, en el peor de los casos perdida de dinero.

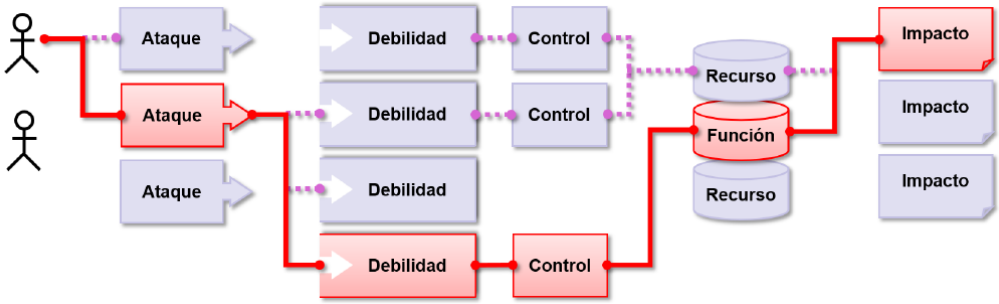


Figura 11. Flujo de un riesgo en la seguridad de las aplicaciones web. Fuente: Tomado de [5]

Algunas veces, estos caminos son fáciles de encontrar y explotar, mientras que otras son extremadamente difíciles. De la misma manera, el perjuicio ocasionado puede no tener consecuencias o puede dejarlo en la quiebra. A fin de determinar el riesgo para las organizaciones, se puede evaluar la probabilidad asociada a cada agente de amenaza, vector de ataque, debilidad de seguridad y combinarlo con una estimación del impacto técnico y de negocio para las empresas. Todos estos factores determinan el riesgo en general.

Idealmente habría un sistema de calificación de riesgo universal que calcularía con precisión todos los riesgos para todas las organizaciones. Pero una vulnerabilidad que es crítica para una organización puede no ser muy importante para otra. Si una organización de interés público utiliza un sistema de gestión de contenido (CMS) para manipular información pública y el sistema de salud utiliza el mismo CMS para tratar datos sensibles, los agentes de amenaza y los impactos en el negocio son muy distintos para la misma aplicación. Es fundamental comprender el riesgo para su organización en función de los agentes de amenaza aplicables a su negocio y los impactos comerciales [48].

La metodología de evaluación de riesgo de OWASP [5] [48] para el Top10 se basa en el modelo de riesgo estándar: *riesgo* = *probabilidad* * *impacto*, pero este incluye tres factores de probabilidad para cada vulnerabilidad (prevalencia, posibilidad de detección y facilidad de explotación) y un factor de impacto técnico. La escala de riesgos para cada factor utiliza el rango de 1 (bajo), 2 (medio) y 3 (alto). La prevalencia de una vulnerabilidad es un factor que normalmente no es necesario calcular. Para los datos de prevalencia, se han obtenido estadísticas de un conjunto de organizaciones distintas y se han calculado el promedio de los datos agregados para elaborar el Top 10 de probabilidad de existencia según la prevalencia. Esta información se combinó con los dos factores de probabilidad (posibilidad de detección y facilidad de explotación) para calcular la tasa de probabilidad de cada vulnerabilidad. Esta tasa fue multiplicada por el impacto técnico promedio estimado de cada elemento, para finalmente elaborar la clasificación de riesgo total (ver figura 12) cuanto mayor sea el resultado, mayor será el riesgo. La detectabilidad, la facilidad de explotación y el impacto se calcularon analizando los CVEs reportados asociados a las 10 categorías principales.

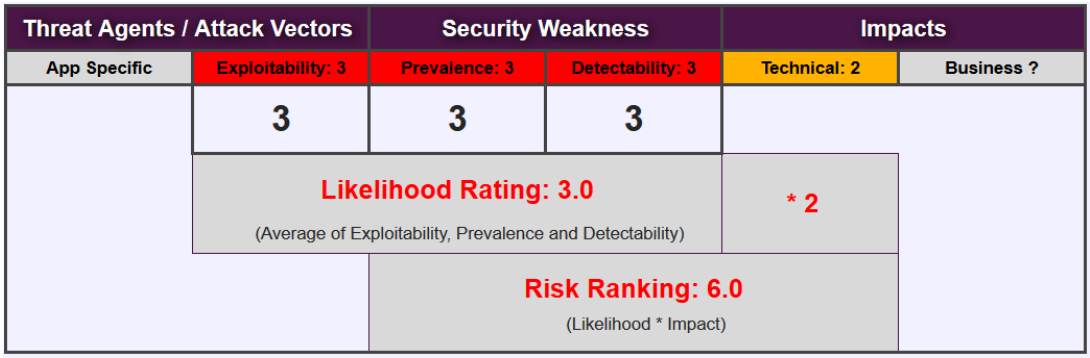


Figura 12. Cálculo de riesgo de un agente de amenaza. Fuente: Tomado de [5]

Se debe considerar que la metodología no tiene en cuenta la probabilidad del agente de amenaza y el impacto de negocio específico. Cada organización deberá decidir cuánto riesgo en las aplicaciones web y APIs está dispuesta a asumir dada su cultura, industria y el entorno regulatorio.

2.11 HERRAMIENTAS QUE AUTOMATIZAN EL PENTESTING

Debido a que las empresas pueden ahorrar tiempo, dinero y esfuerzo con herramientas que permitan automatizar el pentesting, hoy en día, están a disposición una gran cantidad de productos software tanto de pago como de código abierto que hacen posible esta tarea. Sin embargo, tales herramientas no detectan las vulnerabilidades de una manera eficiente, por tal motivo varios estudios se han enfocado a evaluar y analizar el comportamiento de las herramientas en entornos controlados para determinar cuáles son las más recomendables a la hora de realizar un pentesting, así mismo, otros autores realizan estudios similares pero con el objetivo de desarrollar una nueva herramienta que permita mejorar la eficiencia y eficacia a la hora de encontrar vulnerabilidades [21].

Cabe resaltar que la elección de una herramienta es una tarea importante que se debe realizar en las fases previas del pentest, ya que no todas permiten una configuración personalizada o un módulo de generación de reportes automáticos [22], que tal y como se mencionó en la sección de *Metodologías y estándares del pentesting*, los reportes es el elemento fundamental de las últimas fases y es el medio de presentar todos los hallazgos encontrados a lo largo del pentest, el cual va dirigido tanto al equipo de desarrollo (reporte técnico) como a la parte ejecutiva (reporte ejecutivo).

A continuación, en la tabla 6 se hace una comparación de 5 herramientas standalone enfocadas automatizar el pentesting sobre aplicaciones web. Acunetix WVS, Nessus, Netsparker y Nexpose son de licencia comercial, por otro lado, está OWASP ZAP que es open source y tiene soporte por la comunidad de usuarios. Los criterios para evaluar dichas herramientas fueron tomados del estudio [8] el cual está enfocado en evaluar el rendimiento y los artefactos de salida que genera cada herramienta. Además, se añadieron 6 criterios como el *precio* en pesos colombianos (COP) de la

licencia anual para la versión empresarial de cada una, el *sistema operativo* donde pueden ser instaladas y/o ejecutadas, *reporte basado en OWASP* para identificar los tipos de ataques más comunes según el top 10 2013 o 2017, la *calidad del reporte ejecutivo,* el cual es de vital importancia en la toma de decisiones, *pruebas personalizadas* para elegir las pruebas que le interesen al cliente, y por último las *recomendaciones* las cuales pueden ser generales (dando remediaciones para varias tecnologías) o específicas, brindando una sugerencia precisa para el tipo de tecnología que posee la aplicación. Cabe destacar que todas las herramientas fueron evaluadas con la misma aplicación web en un entorno controlado para analizar las características individuales.

Herramientas/ Características	Acunetix WVS	Nessus	Netsparker	OWASP ZAP	Nexpose
Versión	11	7.0.3	3.0	2.7.0	6.5
Sistema operativo	Windows	Windows	Windows	Windows Linux	Windows Linux
Licencia	Comercial	Comercial	Comercial	Open Source	Comercial
Fase en el Pentest	Pre-Prueba de Concepto Prueba de Concepto	Pre-Prueba de Concepto	Pre-Prueba de Concepto	Pre-Prueba de Concepto Ataque	Pre-Prueba de Concepto Prueba de Concepto
Eficacia de detección de vulnerabilidades	Alto	Medio	Alto	Medio	Medio
Calidad del Reporte Técnico	Alto	Alto	Alto	Medio	Medio
Reporte Basado en OWASP TOP 10 2013 o 2017	2013 2017	2013	2013 2017	2017*	No
Calidad del Reporte Ejecutivo	Medio	Bajo	Alto	Bajo	Alto
Pruebas personalizadas	Sí	No	Sí	Sí	No
Recomendaciones Específicas	Sí	No	No	Sí	Sí
Actualización del software	Continuo	Continuo	Continuo	Continuo	Continuo
Tiempo total de la prueba (aprox)	2274 seg	1260 seg	1884 seg	1800 seg	180 seg
Precio COP	$38’257.862	$6'569.651	$17’615.064	$0	$14’265.944

Leyenda: *: Algunas pruebas las realiza de forma manual y automática.

Tabla 6. Herramientas que permiten automatizar el pentesting sobre aplicaciones web. Fuente: Adaptado de [8]

De las herramientas evaluadas solamente Acunetix WVS y Netsparker proveen la funcionalidad de generar reportes basados en el documento OWASP TOP 10 2017 para las vulnerabilidades más comunes, en cambio, OWASP ZAP por defecto genera un reporte sencillo donde muestra las vulnerabilidades con su respectivo riesgo, no

obstante, la herramienta permite la adición de plugins escritos en el lenguaje de programación JAVA que permiten mejorar y adicionar diferentes tipos de reportes.

La principal desventaja de la mayoría de herramientas comerciales son sus elevados precios, siendo asequible por grandes empresas, además carecen de interoperabilidad ya que solamente pueden ser ejecutadas sobre el sistema operativo Windows. La desventaja que tienen las herramientas open source es su **nivel medio-bajo** de eficiencia y eficacia al momento de detectar vulnerabilidades, asimismo, se debe tener conocimientos avanzados de pentesting sobre aplicaciones web, ya que las pruebas que no son automatizadas, el usuario debe realizarlas manualmente.

A continuación, en la figura 13 se presenta de forma gráfica el tiempo requerido por cada herramienta para ejecutar el pentesting sobre una aplicación web. Por otro lado las figuras 14 y 15 muestran la cantidad de vulnerabilidades encontradas tanto por la severidad de riesgo de cada una, como el número de vulnerabilidades por A' que provee el documento OWASP TOP 10 2017 respectivamente.

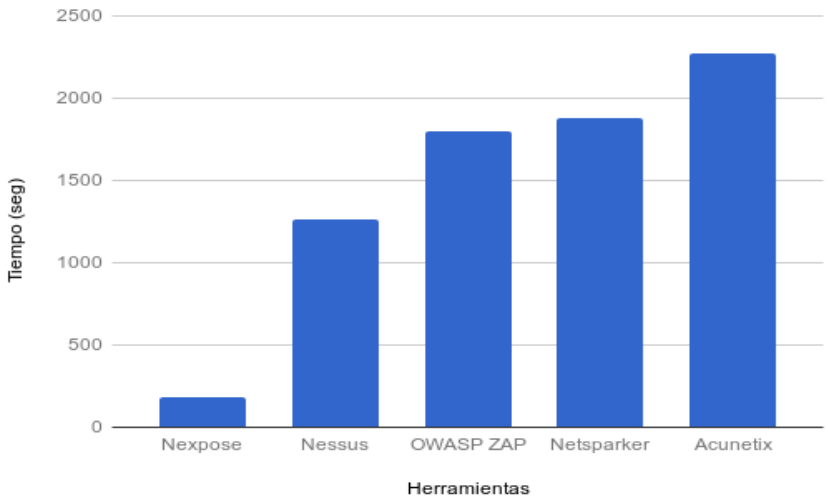


Figura 13. Tiempo total de la prueba. Fuente: Propia.

La herramienta Netsparker fue una de las herramientas que menos tiempo tomó en realizar el pentest con un total de 180 segundos y fue la única que detectó vulnerabilidades con severidad de riesgo altas en la aplicación web, a diferencia con OWASP ZAP, nexposey Acunetix que solamente identificaron vulnerabilidades de severidad media y baja con tiempos de 1800, 1884, 2274 segundos respectivamente. Nessus fue la segunda herramienta más rápida en acabar el pentest seguida de Nexpose con un total de 1260 segundos, pero la eficacia a la hora de detectar vulnerabilidades fue baja debido a que no encontró ninguna vulnerabilidad de severidad baja, media, alta, solamente arrojó datos de información.

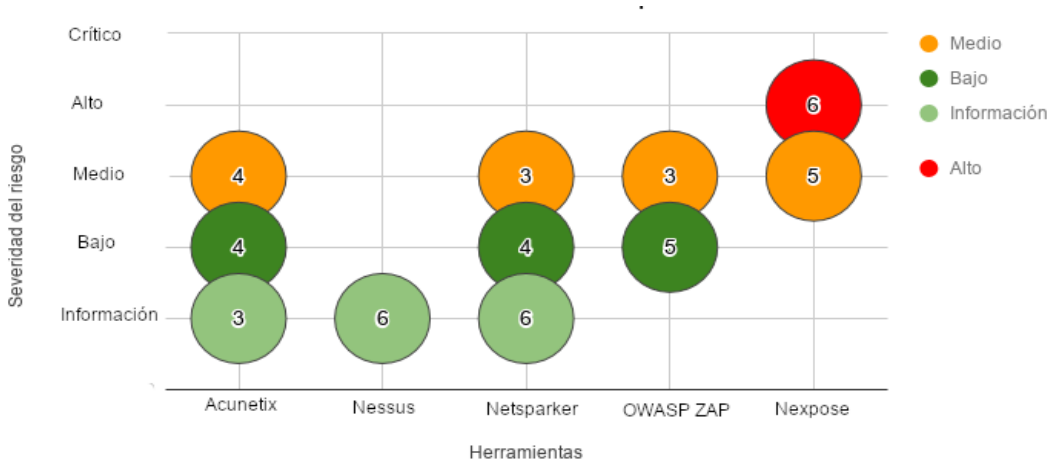


Figura 14. Vulnerabilidades encontradas por severidad. Fuente: Propia.

Al comparar los resultados de las figuras 14 y 15, se encontró una discrepancia al momento de comprobar la cantidad de vulnerabilidades encontradas en la herramienta Acunetix WVS. Se compararon los reportes generados de la herramienta. El reporte técnico menciona que se detectaron 11 vulnerabilidades sobre la aplicación web, mientras que en el reporte basado en el documento OWASP TOP 10 afirma que fueron 15 vulnerabilidades 8 para el A3, 3 para el A6 y 4 para el A9. Todo conlleva a que Acunetix WVS al momento de generar el reporte basado en OWASP TOP 10 menciona que algunas vulnerabilidades pueden ser falsos positivos y que requiere una verificación manual.

La mayoría de herramientas encontraron vulnerabilidades asociadas a las que presenta OWASP TOP 10, sin embargo, los resultados expuestos por Nessus no entran en ninguna categoría de los ataques más comunes según OWASP TOP 10. Por otro lado, las vulnerabilidades identificadas por Nexpose solo corresponden a la categoría de *Datos sensibles* correspondiente al A3.

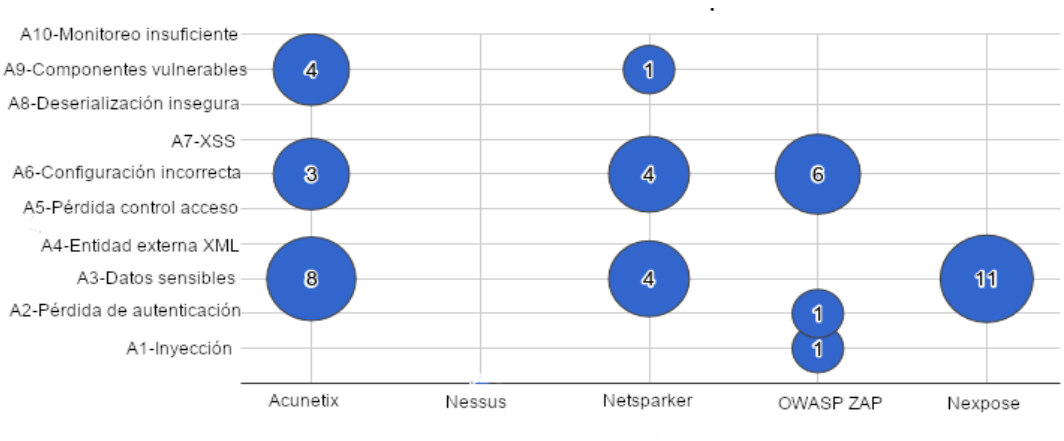


Figura 15. Vulnerabilidades encontradas por A. Fuente: Propia.

Finalmente, la herramienta que mejores resultados arrojó según los criterios mencionados fue Netsparker, debido a que realiza un análisis interno que permite comprobar si la vulnerabilidad encontrada existe en realidad o no, además, soporta la

generación de reportes con base al OWASP TOP 10 2013 y 2017. También permite personalizar las pruebas si se desea utilizar un enfoque de caja blanca o caja gris, en caso de tener algún conocimiento sobre la aplicación web y seleccionar las tecnologías que la aplicación tiene implementada para arrojar resultados más eficientes.

A continuación, en la tabla 7 se pueden observar una serie de herramientas web para automatizar el pentesting con los criterios más relevantes de la tabla 5, debido a que estas herramientas son más limitadas en el momento de ejecutar las diferentes pruebas sobre la aplicación web.

Características/ Herramientas	Detección de vulnerabilidades	Calidad Reporte	Basado en OWASP Top 10 2013 o 2017	Pruebas personalizadas	Membresía	Tiempo (seg)
Pentest-tool	Bajo	Bajo	No	Sí	Gratis Comercial	243
QualysGuard	Medio	Medio	2013	No	Gratis Comercial	425
WhiteHat Sentinel	Medio	Medio	2013	Sí	Comercial	332
SUCURI	Bajo	Bajo	No	No	Gratis	180
UpGuard	Bajo	Bajo	No	No	Gratis	310
Strictly	Bajo	Bajo	No	No	Gratis	70
Detectify	Bajo	Bajo	2017	No	Gratis Comercial	321
AsafaWeb	Bajo	Bajo	No	No	Gratis	124

Tabla 7. Herramientas online para escanear vulnerabilidades web. Fuente: Adaptado de [8]

La gran mayoría de las herramientas online no tardan demasiado en realizar el pentesting hacia las aplicaciones web, lo cual implica que la detección de vulnerabilidades se clasifique con una severidad de riesgo **baja**. Además, las calidades de los reportes no son muy detallados y solamente Qualys y Detectify permiten identificar las vulnerabilidades más comunes según el documento OWASP Top 10 2017. Adicionalmente existen otras herramientas que permiten ejecutar el pentesting, pero únicamente si verifica que es el propietario de la aplicación web. Por tal motivo no fue posible probar las herramientas de Acunetix y Netsparker para las versiones online.

Otra solución propuesta por OWASP es la herramienta Nettacker [49], que a diferencia de las ya mencionadas, esta se especializa en abarcar un proceso en específico, que es la recolección de información. No quiere decir que solo recopila la información y la presenta por medio de su interfaz, sino que incluye módulos que permiten realizar búsqueda de vulnerabilidades ya sea a targets en concreto o a redes internas, así mismo implementa diferentes protocolos para evitar el bloqueo o detección de dispositivos intermedios como los IDS o firewalls.

En el caso de este trabajo de tesis se propone explorar la mayoría de las herramientas anteriormente descritas, con el propósito de identificar las fortalezas y debilidades de sus módulos para poder realizar un pentest. Ya que en la tabla 5 se encuentran herramienta con licencias comerciales, se procede a estudiar sus versiones de prueba gratuita, teniendo en cuenta que no todas las características van a estar disponibles.

2.12 ELABORACIÓN DEL REPORTE

Cada metodología de pentesting tiene actividades de post-testing o post-execution, las cuales están ubicadas en las últimas fases y tienen como nombre genérico la *elaboración del reporte*. En este punto los hallazgos encontrados, las herramientas utilizadas y otros artefactos que fueron necesarios para realizar el pentest deben ser documentados en términos de vulnerabilidades siguiendo (i) una escala cuantitativa que representa el valor numérico del riesgo si la vulnerabilidad encontrada es explotada, y (ii) una escala cualitativa que es la traducción del valor numérico del riesgo a la definición de severidad y estado del sistema como: riesgo alto, medio y bajo.

Como fue mencionado, el reporte es un documento de salida de los procesos de cada metodología enfocada al pentesting y por tal motivo debe tener al menos una estructura básica que explique los componentes necesarios que debe tener un reporte. Según la metodología NIST SP 800-115, un reporte debe contener las siguientes características [44]: (i) actividades de mitigación que permitan abordar las vulnerabilidades encontradas, (ii) debe ser un punto de referencia para realizar el seguimiento y evaluación de los requisitos de seguridad establecidos en la organización, (iii) llevar a cabo un análisis costo/beneficio que ayude a mejorar la seguridad del sistema.

2.12.1 Reporte ejecutivo

Este reporte está dirigido al *personal ejecutivo o administrativo*, el cual es el encargado de crear estrategias y tomar decisiones que permitan mitigar los posibles ataques y minimizar el riesgo para su organización a partir del estado general del sistema. Como lo menciona un integrante del instituto SANS [50], el reporte ejecutivo es un pequeño párrafo que contiene de manera puntual las actividades que fueron llevadas a cabo, la metodología utilizada, los hallazgos descritos en alto nivel con sus respectivas recomendaciones, acompañado de un diagrama para su mejor comprensión.

2.12.2 Reporte técnico

A diferencia del reporte ejecutivo, el reporte técnico está dirigido al *personal técnico* de la organización que son los encargados de interpretar las estrategias y decisiones tomadas por el personal ejecutivo en soluciones que permitan remediar las vulnerabilidades presentes en el sistema. La presentación de cada hallazgo debe ser

lo más detallado posible. En este reporte entran artefactos como el diagrama de red que represente los dispositivos involucrados en el pentest, el cronograma de actividades, una lista de chequeo que compruebe las vulnerabilidades existentes en el sistema, y la clasificación de cada vulnerabilidad encontrada con su respectiva definición y evaluación de riesgo pertinente.

2.12.3 Metodología para la elaboración del reporte

Algunos autores han desarrollado metodologías propia para la redacción de los reportes, como en [50] que describe de manera detallada las actividades que se deben seguir para poder realizar un reporte tanto ejecutivo como técnico.

Planeación del reporte: esta actividad debe establecer los objetivos del reporte, ya que estos ayudan al lector a centrarse en los puntos principales de los hallazgos y a establecer el alcance de lo que se va a realizar. Debe contemplar el tiempo exacto que va a durar el pentesting, así como un cronograma que permita establecer las fechas de cada prueba para que puedan ser monitoreadas por el equipo técnico. Por último, se recomienda clasificar el contenido del reporte según las políticas de seguridad de la organización.

Recolectar información: considerada la actividad más importante ya que debido a la naturaleza del pentesting, se puede utilizar más de una herramienta o dispositivos por lo que el pentester necesita colocar evidencia de cada resultado obtenido. Para esto, lo mínimo que se debe documentar es: la descripción de la prueba, capturas de pantalla que ayuden a corroborar la prueba realizada y los diferentes estados del sistema a medida que se van completando las pruebas, ya que una prueba (sin intención) puede lograr deshabilitar el sistema por un tiempo como, por ejemplo: denegación de servicio (denial of service) o un desbordamiento de búfer (buffer overflow).

Escribir el primer borrador: esta actividad es la encargada de ir recopilando y redactando de manera organizada toda la información recolectada de la anterior actividad. Claro está que un reporte no se hace en un solo día, por eso se recomienda un manejo de versiones o borradores que ayuden a identificar y corregir elementos faltantes para la presentación del reporte final.

Revisión y finalización: como se mencionó, el borrador necesita ser revisado por otra persona, en el caso de que el pentesting haya sido realizado por un equipo, todos los integrantes del equipo deben revisar y editar el reporte, para así obtener mejores resultados como también mantener la calidad. Incluso, una vez actualizado el borrador, el equipo de calidad puede revisar el reporte para asegurar que se siguen los estándares de calidad establecidos por la organización.

Por otra parte los integrantes del proyecto OWASP de Turquía, publicaron en el año 2012 una lista de chequeo (checklist)[9] que ayuda a categorizar las pruebas con

[9] Checklist puede ser encontrada por medio de la siguiente dirección: https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/wasclist/web_app_sec_checklist_2012_en.xls

respecto a la metodología OWASP Testing GUide v4. La lista es un documento con extensión xls que permite llevar el control de las subcategorías de la metodología de pentesting, añadiendo otros parámetros fundamentales para la elaboración del reporte como lo son: el rol responsable encargado de remediar el fallo, la explicación del control de seguridad, el nivel de riesgo y el estado de la vulnerabilidad en el sistema, además permite generar gráficas tanto de cada subcategoría como el consolidado de todas, llamado *resultado total*.

2.13 ESTUDIO DE DISPOSITIVOS SBC DE BAJO COSTO

2.13.1 Contexto general

Hoy en día existen una gran cantidad de dispositivos SBC en el mercado, unos con mejores características y con precios más económicos. Para proyectos empresariales o de investigación que demande el uso de la computación en paralelo/servicios en la nube, los dispositivos SBC pueden ser una buena elección a la hora de implementar cluster que utilicen nodos de bajo costo. Para las organizaciones de tamaño pequeño y mediano permite reducir notablemente los costos relacionados a la adquisición de hardware nuevo, así como los costos operacionales [31]. Además, gracias a su reducido tamaño las infraestructuras pueden ser fácilmente transportables por sus usuarios.

Actualmente existen trabajos relacionados acerca del pentesting con dispositivos SBC. Un tipo de SBC es la Raspberry Pi, con el cual se han desarrollado plataformas o estaciones de trabajo portátiles para realizar pentesting sobre redes inalámbricas con el fin de lograr un consumo de recursos computacionales menor a los que utiliza un computador convencional de última generación [51]. Algunos autores expertos en el tema de seguridad, toman como referencia trabajos previos donde caracterizan diferentes dispositivos SBC de bajo costo (teniendo en cuenta, el procesamiento, costo, consumo de energía, entre otras) para realizar ejercicios o retos que publican organizaciones para la simulación de agentes externos que realicen accesos no autorizados a los sistemas corporativos [52]. Otros autores han realizado un caso de estudio para demostrar el poder de un cluster formado por 32 Raspberry Pi, que intentan identificar y romper el hash de contraseñas de usuarios que se almacenan en un servidor de autenticación, ante un algoritmo de cifrado que utiliza una función hash y la técnica salt para darle una protección extra al hash de una contraseña, reduciendo notoriamente la velocidad que toma este proceso [28].

A continuación, se enumeran una serie de SBC de bajo costo, que se puede adquirir en el mercado. Aunque no es un listado exhaustivo, los ejemplos mostrados dan una idea general sobre este tipo de hardware [53].

2.13.1.1 Raspberry Pi 3 Modelo B

La Raspberry Pi es considerada por su fabricante como una computadora del tamaño de una tarjeta de crédito, fue lanzado al mercado con muy buena acogida en febrero 2012. El proyecto raspberry cuenta con un sistema operativo basado en Linux llamado Raspbian, la Raspberry Pi es el dispositivo SBC más popular y económico (con respecto a sus características) en el mercado, muy avanzado y funcional. La principal ventaja de este dispositivo es su precio y la facilidad de adquisición, siendo asequible a través de varios sitios web [32].

2.13.1.2 PandaBoard

Este SBC se encuentra al nivel de los ordenadores portátiles de bajo consumo, presenta mejores prestaciones que el Raspberry Pi. Además, de un microprocesador de una gama superior. Entre las ventajas de este sistema, comparadas con el Raspberry Pi, se encuentran una potencia superior del microprocesador. La principal desventaja está en el precio, que es aproximadamente cinco veces más al precio del Raspberry Pi.

2.13.1.3 ODROID-U2

El ODROID-U2 es uno de los SBC más potentes actualmente disponibles en el mercado. Integra un microprocesador de 4 núcleos a 1.7 GHz con 2 GB de RAM, características que superan ampliamente las del Raspberry Pi y las del PandaBoard. La ventaja de este SBC es la capacidad de montar un procesador de cuatro núcleos, lo que permite una gran flexibilidad para la programación en entornos de memoria compartida, así como su velocidad de 1.7 GHz. Su principal desventaja es la asequibilidad, ya que está restringida su compra en algunos países.

2.13.1.4 Orange PI

Es una computadora de una sola placa de código abierto fabricada por Shenzhen Xunlong Software CO Limited. Se puede ejecutar Android, Ubuntu, Debian, así como las imágenes de Rasberry Pi y Banana Pi. Utilizan los microprocesadores AllWinner H2, H3 y H5 SoC, A64 Quad-core Cortex-A53 64bit o ARM Cortex-A5 32bit, entre otros, y tienen desde 256MB las placas más pequeñas hasta 2 GB DDR3 SDRAM de RAM pudiendo tener según el modelo Ethernet, Bluetooth, WiFi, e incluso 2G. Se pueden construir un ordenador, un centro multimedia (Smart TV), un servidor de archivos, instalar Android, instalar Linux y mucho más, porque Orange Pi 2 se basa en código abierto.

2.13.1.5 Banana Pi M2 Ultra

El Banana Pi Ultra proporciona un Allwinner R40 SoC más rápido, y tiene una GPU ARM Mali-400 MP2 en lugar de una PowerVR SGX544MP2. Los principales cambios de características en esta placa de 92 x 60 mm es la adición de un conector SATA habilitado por el nuevo R40, junto con la pérdida resultante de uno de los cuatro puertos host USB. También cuenta con 2 GB de memoria RAM, lo cual es inusual para un SOC ARMv7 de cuatro núcleos. El M2 Ultra está equipado con GbE, WiFi, Bluetooth, micro-USB OTG y un conector compatible con Raspberry Pi de 40 pines.

2.13.1.6 Parallella

La Parallella es otro SBC de tamaño compacto con un gran potencial para aplicaciones de alto rendimiento y seguridad. Fue lanzado al mercado en el año 2014. Parallella tiene una CPU ARM A9 de doble núcleo, un coprocesador Epiphany de 16 núcleos y 1 GB de RAM, además tiene un consumo de energía equivalente a 5 voltios de CC a 2,5 A y 5 vatios de potencia. La edición de microservidor de la Parallella tiene un costo de $ 99.00, mientras que la edición de escritorio tiene un costo de $ 149.00 USD. Parallella es pequeña, autónoma y extremadamente eficiente en el consumo de energía [28].

2.14 Caracterización de los dispositivos SBC

2.14.1 Criterios de selección de dispositivos SBC

Para esta investigación, se pretende analizar un conjunto de dispositivos SBC de bajo costo que son recomendados en [54] por medio de unos criterios de evaluación que ayuden a identificar y clasificar los más pertinentes según sus características para llevar a cabo el desarrollo de la propuesta. Con el fin de identificar cuáles de los dispositivos SBC eran los más adecuados para la ejecución de las diversas tareas que demanda un pentesting, así mismo manteniendo la portabilidad y economía del prototipo, se tomaron las siguientes características que fueron consideradas como las más relevantes a la hora de realizar el proceso de evaluación:

1. Precio (COP).
2. Procesador.
3. Memoria RAM.
4. Conexión Ethernet.
5. Conexión Wi-Fi.
6. Sistemas operativos que soporta.

Debido a que algunas de las características nombradas no tienen la misma relevancia con respecto a las demás, es posible que algunos dispositivos sean considerados aptos por tener características técnicas muy superiores sin tener en cuenta otras

como precio. Un ejemplo para el anterior caso puede ser: el dispositivo parallella tiene consigo un procesador ARM A7 con 2 núcleos, 1 GB de memoria RAM y un precio aproximado de $99 USD, todas esas características lo hacen uno de los dispositivos potencialmente elegibles para este trabajo, sin embargo, el contexto regional donde se desarrolla la investigación no maneja la misma moneda para la adquisición del dispositivo que en este caso es el dólar (USD), por lo tanto la conversión del precio al peso colombiano (COP) tendría un costo más elevado. Es por esa razón que se estableció un valor porcentual a cada característica de la siguiente manera:

1. **Precio (COP)**, representa el 40% debido a que se requiere un dispositivo de bajo costo, el cual es uno de los factores diferenciadores de la propuesta.

2. **Procesador**, esta característica se divide es dividida en cuenta la cantidad de núcleos para las tareas en paralelo con un 9% y en la frecuencia de reloj medida en GHZ para realizar un mayor número de operaciones por segundo, también con un 9%. Todo esto equivale al 18%.

3. **Memoria RAM**, representa el 15% ya que se requiere un dispositivo que tenga la suficiente capacidad de cargar un sistema operativo que funcionen sobre arquitecturas ARM, además de potenciar la funcionalidad multitask (multitarea) sin que haya una reducción de rendimiento.

4. **Conexión alámbrica (Ethernet)**, ya que la propuesta consta de la construcción de un cluster donde cada dispositivo va a estar conectado a un switch y la velocidad en que se envían los mensajes está ligada a la velocidad de transferencia (Fast Ethernet o Gigabit Ethernet), por tal motivo, se procede a establecer esta característica con un valor del 12%.

5. **Sistemas operativos que soporta**, se tomó esta característica porque se plantea utilizar un sistema operativo basado en Linux especializado en pentesting como lo es Kali Linux. Con respecto a lo anterior, algunos dispositivos no soportan tal distribución por compatibilidad de la arquitectura o por otros factores, por lo tanto, esta característica representa el 10%.

6. **Conexión inalámbrica (Wi-Fi)**, esta última característica está relacionada con la flexibilidad que proporciona el dispositivo con respecto a la conectividad y al envío de mensajes por red, reduciendo la dependencia a una conexión cableada, por tal motivo representa el 5%.

2.14.2 Criterios de evaluación de selección

El objetivo principal de los criterios de evaluación de selección es asegurar una apropiada evaluación para cada dispositivo SBC de bajo costo, con el fin de obtener los mejores y que son considerados aptos para el desarrollo de la propuesta. Los criterios de evaluación se establecieron de la siguiente manera:

Primero se estableció una valoración general que será aplicada a cada característica del dispositivo. La valoración se divide en un valor cuantitativo y cualitativo así: MA

(muy alto) = 1, A (alto) = 0.5, M (medio) = 0,25, B (bajo) = 0. Luego se establecieron rangos con respecto a las especificaciones que corresponden a cada característica del dispositivo a evaluar unido con la valoración general. A continuación, en la tabla 8 se muestra la distribución mencionada.

Característica		**Especificación**	**Puntaje**	
			Cuantitativo	**Cualitativo**
Precio (COP)		Menor o igual que $333.000	0	B
		Entre $222.000 y $333.000	0,25	M
		Entre $140.000 y $222.000	0,5	A
		Entre $0 y $140.000	1	MA
Procesador	Número de núcleos	1	0	B
		2	0,25	M
		4	0,5	A
		8	1	MA
	Frecuencia de reloj (GHZ)	Menor que 1	0	B
		1	0,25	M
		Entre 1.2 y 1.8	0,5	A
		Mayor que 1.8	1	MA
Capacidad de memoria RAM (Gb)		Entre 0 y 0.128	0	B
		Entre 0.128 y 1	0,25	M
		Entre 1 y 2	0,5	A
		Mayor que 2	1	MA
Conexión Ethernet		No	0	B
		Fast	0,5	A
		GbE	1	MA
Conexión WI-FI		No	0	B
		b, g, n solamente	0,25	M
		b/g/n	0,5	A
		AC	1	MA
Sistemas operativos		No soporta Linux	0	B
		Soporta Linux	0,5	A
		Soporta Kali Linux ARM	1	MA

Leyenda: B: bajo, M: medio, A: alto, MA: muy alto.

Tabla 8. Rangos de las especificaciones con respecto a las características del dispositivo SBC. Fuente: Propia.

2.14.3 Análisis de resultados

A continuación, la tabla 9 muestra de una forma resumida los resultados obtenidos después de aplicar los criterios de evaluación de selección para cada dispositivo SBC. Los resultados son presentados de la siguiente manera: cada dispositivo se encuentra

enumerado por la columna *ID*, seguido de su correspondiente nombre en la columna *Dispositivo*. La columna *Total* está dividida en otras dos columnas llamadas *Cuantitativo* y *Cualitativo*. El valor cuantitativo es la sumatoria de cada multiplicación entre el **porcentaje** de cada característica correspondientes a la sección criterios de selección de dispositivos SBC, por el **valor de la columna métrica**. El valor cualitativo representa el nivel de aptitud de cada dispositivo, para eso se definieron ciertos rangos: se considera MUY ALTO cuando el valor cuantitativo es mayor que 0,7, ALTO cuando el valor supera el 0,5 pero menor que 0,7, MEDIO cuando el valor se encuentra entre 0,3 y 0,5, finalmente BAJO representa un valor inferior a 0,3. Cabe resaltar que no se muestra la totalidad de dispositivos SBC que fueron sometidos a la evaluación de calidad por fines prácticos, sin embargo, en el **Anexo A** se muestra la tabla completa con los 40 dispositivos evaluados.

ID	Dispositivo	Total	
		Cuantitativo	Cualitativo
1	Banana Pi BPI-M2 Ultra	0,885	MUY ALTO
2	Orange Pi Prime	0,8125	MUY ALTO
3	Nano Pi M2A	0,81	MUY ALTO
4	RaspBerry PI 3 Model B	0,75	MUY ALTO
5	Firefly-ROC-RK3328-CC	0,7125	MUY ALTO

Tabla 9. Los 5 mejores dispositivos SBC para el desarrollo de la propuesta. Fuente: Propia.

El objetivo de esta caracterización era identificar cuáles son los dispositivos SBC que más se adaptan al contexto de la propuesta de investigación. Se tomó como referencia un puntaje mayor a 0,7 del total, para los dispositivos que tuvieran las mejores características a la hora de realizar las tareas que demanda la realización de un pentesting; los dispositivos que superaron tal valor y los que perfectamente pueden ser seleccionados para todo el proyecto son: Banana Pi BPI-M2 Ultra, Orange Pi Prime, NanoPi M2A, RaspBerry PI 3 Model B y Firefly-ROC-RK3328-CC, con un puntaje total de 0,885, 0,8125, 0,81, 0,75 y 0,7125 respectivamente. Así mismo, se establecieron los demás puntos de referencia, de 0,5 al 0,7 se encuentran los dispositivos que pueden ser seleccionados como una segunda opción, pero con un rendimiento menor. Desde el 0,3 al 0,5 se catalogan los dispositivos que posiblemente pueden ser utilizados, pero no se cumplen los requisitos mínimos para el desarrollo de la propuesta. Finalmente, se encuentran los dispositivos que van del rango de 0 a 0,3 los cuales se descarta totalmente su participación para el desarrollo de este proyecto.

2.14.4 Selección del dispositivo SBC

Una vez evaluados los distintos SBC disponibles en el mercado para llevar a cabo el proyecto de investigación se consideró seleccionar el dispositivo RaspBerry pi 3 Model B. La razón fundamental de la elección se basa en que este sistema presenta la mejor relación potencia/precio de los analizados y soporta el sistema operativo kali Linux el cuál se pretende utilizar para poder realizar las pruebas de pentesting. Cabe

destacar también el reducido tamaño del dispositivo, lo que redundará en un sistema final más compacto y fácil de manejar. Además, es más asequible en la región dónde se pretende realizar el trabajo de investigación, y gracias al trabajo colaborativo del semillero de investigación de criptografía[10] se disponen de seis dispositivos RaspBerry pi 3 Model B, los cuales ahorran dinero, tiempo y facilitan la construcción del cluster. También dispone de una comunidad activa de usuarios muy amplia que ayudan con el soporte en el momento que se presenta algún tipo de problema en la configuración de los dispositivos.

3 CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO HARDWARE Y SOFTWARE

En este capítulo se describen los pasos llevados a cabo para construir el cluster, así, como su respectiva configuración para su correcto funcionamiento, arquitectura software y hardware, diagramas de red, instalación del software necesario y la configuración de este a través de los diferentes archivos y comandos necesarios.

3.1 ¿Qué es un Beowulf cluster?

Un Beowulf cluster [55] es un conjunto de dispositivos hardware (normalmente idénticos) que están conectados entre sí por medio de una red LAN para un mejor rendimiento. La estructura básica de Beowulf cluster se compone de un dispositivo principal llamado *maestro*, el cual se encarga de distribuir mantener y organizar los procesos que serán enviados a los demás dispositivos denominados *esclavos*, los cuales ejecutan las tareas de manera distribuida y/o paralela. El envío y recepción de datos dentro del cluster se realiza mediante un software middleware que están descritos en la sección 3.2 para poder simular que el procesamiento se está haciendo en un solo dispositivo.

Algunas características para tener en cuenta de este tipo de cluster son:

- Los dispositivos deben estar 100% dedicados y no deben ser usados para otro propósito.
- Todos los dispositivos deben ejecutar componentes software (sistema operativo, aplicaciones, etc) con licencias open source.
- La red donde se comunican los diferentes nodos debe ser dedicada y no debe ser usada para otro propósito.

[10] Semillero de investigación conformado por la FIET y la FACNED de la Universidad del Cauca enfocado al área de la criptografía.

El objetivo de utilizar un Beowulf cluster para este proyecto fue distribuir los procesos para las diferentes pruebas de concepto a realizar sobre un pentesting de manera eficiente y eficaz entre los diferentes dispositivos SBC de bajo costo.

A la hora de definir la estructura lógica del cluster [56], existen alternativas para su montaje, debido a que un cluster puede ser diseñado para resolver problemas específicos. Por tal motivo, hay que saber a priori el problema que se desea resolver con el cluster para poder decidir qué tipo de cluster se adapta mejor a las necesidades. Principalmente existen cuatro tipos de clusters:

Almacenamiento de datos: Ofrecen un sistema de archivos consistente y transparente a través de todos los servidores que conforman el cluster. Permite que el servidor pueda leer y escribir simultáneamente de un sistema de ficheros.

Alta disponibilidad (failover): Ofrecen disponibilidad continua de servicios y por medio del proyecto Heartbeat transfieren servicios de un nodo a otro en caso de que un nodo se vuelva inoperativo.

Balanceo de carga: Envían solicitudes de servicios de red a múltiples nodos del cluster para balancear la carga entre los nodos. El balanceo de carga ofrece escalabilidad costo-eficiencia puesto que se puede igualar el número de nodos necesarios en función de los requerimientos de la carga. Si un nodo se vuelve inoperativo, el software de balanceo detecta el fallo y redirige las peticiones hacia otros nodos, lo que es conocido como la tolerancia a fallos. Estos problemas no son visibles para clientes externos, por lo que este tipo de cluster ofrece un servicio completamente transparente para el usuario.

Alto rendimiento: Conocido como computación de alto rendimiento (por sus siglas en inglés HPC), es un tipo de cluster donde los nodos están interconectados por medio de un dispositivo intermedio (switch, hub, etc) que permite la comunicación entre ellos para poder ejecutar tareas de forma paralela y/o distribuido. Está compuesto por varios procesadores, memoria RAM, rápida administración del sistema y amplios almacenes de información.

El tipo de cluster usado en este proyecto de investigación es el de ***alto rendimiento,*** debido a que están destinados a utilizar más potencia y brindar altas prestaciones en cuanto a capacidad de procesamiento para a la disposición de una operación, tarea, o problema mediante la computación paralela. Los cluster de este tipo suelen tener una gran cantidad de nodos, aunque desde afuera puede parecer como un sistema único, el funcionamiento interno debe ser capaz de resolver cualquier tarea que se envíe a un nodo específico del cluster, proporcionando acceso rápido, confiable y concurrente en un framework de almacenamiento.

3.2 Tecnologías para la comunicación

Debido a la naturaleza de los clusters HPC en los cuales los procesos son paralelizados entre varios nodos, la comunicación entre los diferentes nodos se hace

necesaria. Por ello hay que escoger un modelo de comunicación. Los modelos más usados son MPI (Message Passing Interface), PVM (Parallel Virtual Machine), PyRO (Python Remote Objects) y APACHE HADOOP.

3.2.1 MPI – Message Passing Interface

Es un sistema estandarizado y portable de paso de mensajes, diseñado para funcionar en una amplia variedad de computadoras paralelas. El estándar define la sintaxis y semántica de un conjunto de funciones de librerías que permiten a los usuarios escribir programas portables en los principales lenguajes utilizados por la comunidad científica (Fortran, C, C++). Desde su aparición, la especificación MPI se ha transformado en el estándar dominante en librerías de paso de mensajes para computadoras paralelas. En la actualidad se dispone de diversas implementaciones, algunas provistas por los fabricantes de computadoras de alto rendimiento hasta otras de reconocidos proyectos open source, tales como MPICH y LAM/MPI, muy utilizadas en los clusters Beowulf [57].

3.2.2 PyRO - Python Remote Objects

Es una librería que permite crear aplicaciones en las que los objetos pueden comunicarse entre sí a través de la red, con el mínimo esfuerzo de programación. Puede usar llamadas a métodos normales de Python, con casi todos los parámetros posibles y el tipo de valor de retorno, Pyro se encarga de ubicar el objeto correcto en la computadora correcta para ejecutar el método adecuado. Proporciona un conjunto de potentes funciones que permiten crear aplicaciones distribuidas de forma rápida y sin esfuerzo. Pyro es una biblioteca pura de Python y se ejecuta en diferentes plataformas y versiones de Python [58].

3.2.3 Apache HADOOP

Es una plataforma de software de código abierto para el almacenamiento distribuido y procesamiento distribuido de grandes conjuntos de datos en clústeres informáticos construidos del hardware de productos básicos. Los servicios de Hadoop prevén almacenamiento de datos, procesamiento de datos, acceso a los datos, gestión de datos, seguridad y operaciones. Algunas de las razones por las que las organizaciones usan Hadoop es su capacidad para almacenar, administrar y analizar grandes cantidades de datos estructurados y no estructurados de manera rápida, confiable, flexible y de bajo costo [59].

3.2.4 PVM

Es un paquete software que permite a un conjunto heterogéneo de computadores Unix o Windows compartir sus recursos (como procesador y memoria RAM) con el objetivo de mejorar el rendimiento o tiempo de ejecución de un proceso o tarea al distribuir la carga de trabajo entre varios nodos y pueda ser visto para un programa

que ejecute el usuario como una plataforma de computación en paralelo. Soporta un modelo sencillo pero muy completo para el paso de mensajes entre diferentes nodos. Cuando el PVM es instalado correctamente, este es capaz de aprovechar los recursos compartidos y suministrar un alto nivel de rendimiento y funcionalidad [60].

Para este trabajo de investigación el modelo de comunicación seleccionado fue MPI (Message Passing Interface), debido a su gran soporte que presenta actualmente para el lenguaje interpretado Python, cuya curva de aprendizaje es baja y se tiene pensado implementar los diferentes algoritmos o scripts con dicho lenguaje. Este modelo de comunicación está destinado para funcionar con clusters homogéneos, ya que todos los dispositivos a utilizar van a ser del mismo modelo y tipo. Además, se encontraron algunos trabajos relacionado con MPI y dispositivos SBC de bajo costo [61], [57], [62] los cuales permitirán brindar un soporte en el momento de la instalación y configuración, ya que para los otros modelos de comunicación no se encontró un gran número de proyectos relacionados.

3.3 Arquitectura de red del cluster

El cluster fue diseñado para realizar procesos de manera eficaz y eficiente, de tal manera está orientado a realizar tareas en paralelo permitiendo ejecutar diferentes procesos al mismo tiempo y mejorar el rendimiento general del sistema [62], evitando así el cuello de botella con los procesos secuenciales. Además, sigue una arquitectura maestra/esclavo donde un dispositivo maestro asigna trabajo a dispositivos esclavos para lograr optimizar los recursos del cluster.

En este trabajo se utilizaron cables de red y un switch Ethernet para conectar todos los dispositivos. La interconexión utilizada fue una topología de estrella para proporcionar comunicaciones entre los diferentes nodos implementando la interfaz de paso de mensajes (MPI) debido a la estandarización, rendimiento, portabilidad, funcionalidad y la disponibilidad con el cluster Beowulf.

En la figura 16 se puede observar el diagrama de red del cluster con los diferentes RaspBerry PI y los dispositivos de red utilizados.

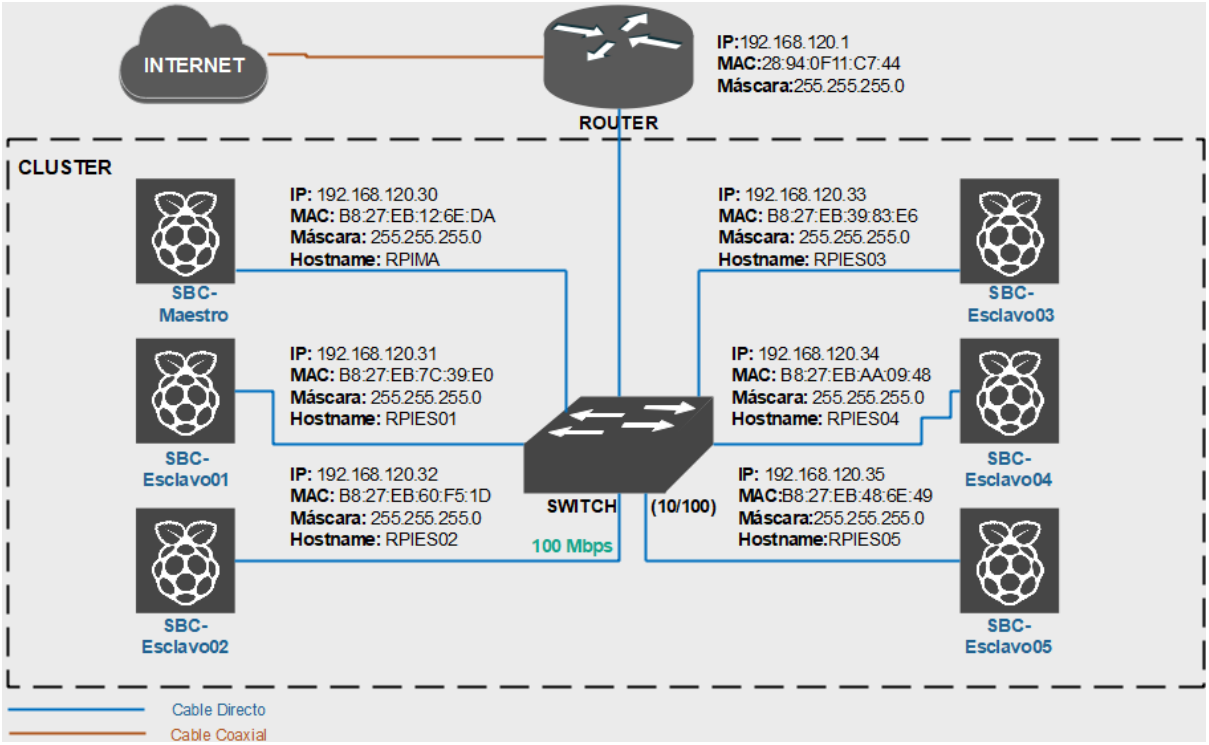


Figura 16. Diagrama de Red del Cluster con Raspberry PI. Fuente: Propia

3.4 Construcción y configuración del cluster con RaspBerry PI

Para el montaje del cluster se utilizaron cinco componentes principales: hardware, sistema operativo especializado en el pentesting como Kali Linux, librería MPI y multiprocesing, y un switch Ethernet. En nuestro caso hemos elegido la Raspberry Pi 3 modelo B como el tipo hardware, como se mencionó en la sección 2.14.4. A continuación, en la tabla 10 se muestra el listado de los materiales que fueron utilizados para el montaje del cluster:

Descripción	Cantidad	Precio Unitario (COP)	Precio Total
Raspberry pi 3 modelo B	6	$128.706	$772.236
Fuente de alimentación	1	$ 55.000	$55.000
Switch 10/100 8 puertos	1	$30.000	$30.000
Tarjeta micro SD 16 GB clase 10 (80 MB/s)	6	$30.000	$180.000
Cable de red UTP R-J45 Cat 5e	7	$3.000	$21.000
Pi rack por 6 capas	1	$184.000	$184.000
Total			$1.242.236

Tabla 10. Presupuesto de los materiales para construir el cluster. Fuente: Propia.

Una vez elaborado el diagrama de red y teniendo los materiales necesarios indicados en el presupuesto, se procedió a construir el cluster para su posterior configuración. En la figura 17 se puede observar la estructura física del cluster.

Figura 17. Cluster conformado por 6 Raspberry Pi. Donde el primer dispositivo de arriba hacia abajo es el maestro y los otros cinco son los esclavos Fuente: Propia.

3.5 Configuración e instalación de los dispositivos SBC

3.5.1Configuración del dispositivo maestro

Para la instalación y configuración de cada Raspberry Pi [63], [64] fue necesario el uso de los siguientes elementos software:

- Kali Linux ARM en su versión 2018.1a
- PuTTY (Windows) o el comando ssh (Distribuciones Linux) para la conexión mediante el protocolo SSH.
- dd (Dataset definition) para extraer una copia bit a bit de la información en cada tarjeta de almacenamiento micro SD.

El procedimiento para la instalación del sistema operativo fue la siguiente:

1. Descargar la imagen[11] de kali Linux para el hardware ARM, disponible para el dispositivo Raspberry PI en su versión 3.
2. Grabar la imagen del sistema operativo Kali Linux en la tarjeta micro SD.
3. Cambiar el nombre del hostname a RPMA
4. Deshabilitar la interfaz GUI del sistema operativo e iniciar la opción CLI (Command Line Interface) para el arranque del sistema.
5. Habilitar el protocolo SSH para realizar la configuración posteriormente
6. Realizar una configuración de red estática para el dispositivo.

La explicación detallada de la instalación del sistema operativo Kali Linux se puede ver en el **Anexo B**.

3.5.2Instalación de MPI

Debido que se requiere una norma estándar de paso de mensaje para aplicaciones de memoria distribuida que utilizan computación paralela MPICH es un software gratuito, de alto rendimiento, portable y disponible para la mayoría de distribuciones Linux. Tal como está, MPICH puede ejecutar los programas en lenguaje C y Fortran. Dado que Raspberry Pi tiene el entorno de codificación Python preinstalado, es más fácil instalar un intérprete de Python a MPI. Para observar la instalación de MPI ver el **Anexo C**.

3.5.3Configuración de los dispositivos esclavos

Una vez realizado los pasos anteriores, se debe extraer una copia bit a bit de la tarjeta SD perteneciente al nodo maestro y copiarla en los nodos esclavos. Luego se procede a conectar cada dispositivo para cambiar el nombre del host por RPIESC01, RPIESC02, RPIESC03, RPIESC04, RPIESC05 y configurar la dirección IP de manera estática para evitar problemas de conexión. En la figura 18 se puede observar las direcciones IP para cada nodo, donde la primera dirección está asignada para el nodo maestro y las otras dos direcciones para los esclavos.

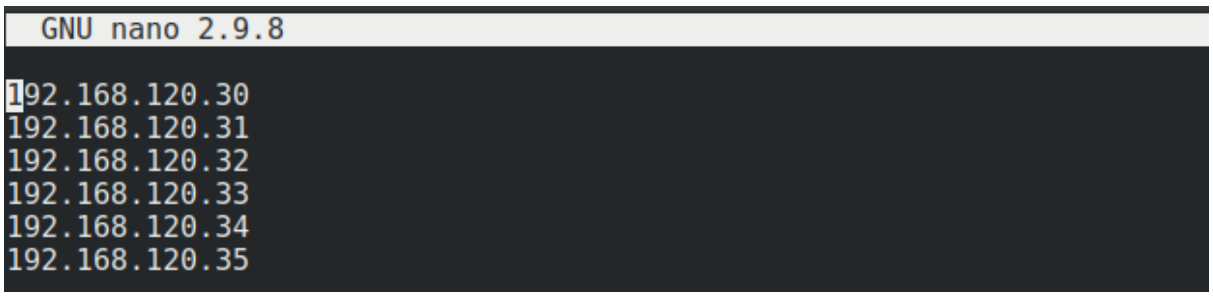


Figura 18. Configuración de red de los dispositivos SBC

[11] Enlace para descargar la imagen de Kali Linux https://www.offensive-security.com/kali-linux-arm-images/

3.5.4Configuración de las claves SSH para cada dispositivo

Para acceder a los dispositivos sin necesidad de un usuario y contraseña se generaron las llaves SSH respectivas y se compartío la llave pública en cada dispositivo que conforma el cluster. Al hacer esto, MPI podrá comunicarse con cada dispositivo sin preocuparse por las credenciales. Los detalles para esta configuración se pueden observar en el **Anexo D**.

3.5.5Configuración de la base de datos

Para almacenar los resultados de cada pentest y generar los reportes automáticos, fue necesario crear una base de datos para centralizar la información. El sistema gestor de base de datos (SGBD) utilizado fue PostgresSQL en su versión 10, el cual es de código abierto, tiene compatibilidad con sistemas basados en Linux/UNIX, además permite crear bases de datos relacionales y orientadas a objetos.

La instalación del SGBD se realizó sobre un servidor lógico dedicado ofrecido por los servicios de Microsoft Azure con una suscripción de estudiante para que la base de datos fuera accesible por todos los dispositivos que conforman el cluster, además para ahorrar tiempo en la configuración del servidor, ya que dicho servicio ofrece todas las alternativas necesarias para el despliegue y sus respectivas reglas de seguridad.

El SGBD sobre el servidor lógico de Microsoft Azure crea un usuario y una base de datos por defecto llamada postgres. Una vez efectuada la conexión remota al servidor de base de datos se debe cambiar al usuario *postgres* y establecer una sesión emitiendo los siguientes comandos:

$ sudo -u postgres -i
$ psql

Finalmente se obtiene un shell que permite la interacción con el servidor de base de datos de postgreSQL como se puede ver en la figura 19.

Figura 19. Acceso correcto a la base de datos PostgreSQL. Fuente: Propia.

Las sentencias para crear la base de datos y los permisos correctos (como evitar eliminar tablas, crear o borrar bases de datos y crear otros roles) del rol que está autorizado para manipular la misma son:

=# CREATE DATABASE scanlynx;

=# CREATE ROLE pentester NOSUPERUSER NOCREATEDB NOCREATEROLE LOGIN
=# SET ROLE pentester

El diagrama resultante de la base de datos se muestra en la figura 20, donde la tabla ***pentest*** almacenará los datos correspondientes de la aplicación a la que se realizará el pentest, la tabla ***prueba***, contendrá la información con respecto a las pruebas implementadas como son: código de la prueba que puede ser por ejemplo OTG-INFO-002, la descripción, el objetivo y los respectivos riesgos de probabilidad e impacto técnico (mirar sección 3.6.5), así como la referencia a la ***subcategoría*** y ***a_top10_2017*** a la que pertenece. La tabla ***detalle***, es la que almacenará los respectivos resultados y recomendaciones de las pruebas ejecutadas, así como también el riesgo total. Finalmente, ***riesgo_cualitativo*** contendrá la severidad del riesgo que puede tener el resultado de una prueba como CRÍTICO, ALTO, MEDIO y BAJO.

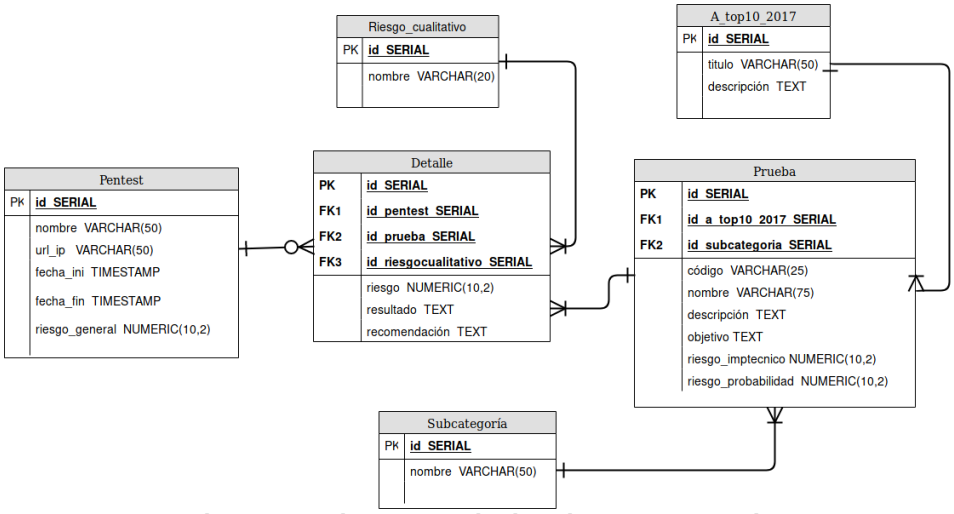


Figura 20. Diagrama relacional. Fuente: Propia.

3.6 Implementación de los tipos de ataques más comunes según OWASP

3.6.1Selección de las pruebas

Para el desarrollo de las pruebas que simulan los ataques más comunes, se tuvo en cuenta el documento OWASP TOP 10 2017, con el fin de abarcar por lo menos una prueba de cada categoría que plantea. Además, se tomó en cuenta las 11 subcategorías de la guía OWASP Testing Guide v4, definidas en la sección 2.9, para que el usuario o cliente pueda escoger las subcategorías que desea al momento de realizar el pentesting sobre su aplicación web y ejecutarlas por demanda, característica adicional a diferencia de otras herramientas que no permiten escoger las pruebas dinámicamente.

3.6.2 Pruebas implementadas

Las diferentes pruebas que establece la metodología OWASP requieren de una entrada como la dirección IP, URL o el dominio de la aplicación web para que se logre ejecutar de manera correcta y se pudiera almacenar sus respectivos resultados en la base de datos. Debido a que algunas pruebas necesitan las salidas de otras para su correcto funcionamiento, se realizó un diagrama de secuencia el cual muestra el flujo y la dependencia que hay entre las diferentes pruebas (si las hay), este diagrama se puede consultar en el **Anexo E.**

Una vez identificado y analizado el flujo de cada prueba se realizó un control de las pruebas que se desarrollaron, para ello se elaboró una tabla con la categoría según el **A Top 10**, el **código** y **descripción** de las pruebas por cada **subcategoría** según la metodología OWASP Testing Guide v4. En la tabla 11 se puede observar las pruebas implementadas que comprende este proyecto.

A Top 10	Test ID	Test Description
Information Gathering		
A9	OTG-INFO-002	Fingerprint Web Server
A6	OTG-INFO-003	Review Webserver Metafiles for Information Leakage
A9	OTG-INFO-004	Enumerate Applications on Webserver
A6	OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage
A6	OTG-INFO-007	Map execution paths through application
A9	OTG-INFO-008	Fingerprint Web Application Framework
A9	OTG-INFO-009	Fingerprint Web Application
A6	OTG-INFO-010	Map Application Architecture
Configuration and Deploy Management Testing		
A6	OTG-CONFIG-001	Test Network/Infrastructure Configuration
A6	OTG-CONFIG-002	Test Application Platform Configuration
A6	OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces
A6	OTG-CONFIG-006	Test HTTP Methods
A3	OTG-CONFIG-007	Test HTTP Strict Transport Security
A4	OTG-CONFIG-008	Test RIA cross domain policy
Identity Management Testing		
A2	OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account
Authentication Testing		
A3	OTG-AUTHN-001	Testing for Transported over an Encrypted Channel
A2	OTG-AUTHN-002	Testing for default credentials
A2	OTG-AUTHN-003	Testing for Weak lock out mechanism
A2	OTG-AUTHN-006	Testing for Browser cache weakness
Authorization Testing		
A5	OTG-AUTHZ-001	Testing Directory traversal/file include
A5	OTG-AUTHZ-003	Testing for Privilege Escalation

A8	OTG-AUTHZ-004	Testing for Insecure Direct Object References
Session Management Testing		
A2	OTG-SESS-002	Testing for Cookies attributes
A2	OTG-SESS-003	Testing for Session Fixation
A2	OTG-SESS-006	Testing for logout functionality
A2	OTG-SESS-007	Test Session Timeout
Input Validation Testing		
A7	OTG-INPVAL-001	Testing for Reflected Cross Site Scripting
A7	OTG-INPVAL-002	Testing for Stored Cross Site Scripting
A5	OTG-INPVAL-003	Testing for HTTP Verb Tampering
A1	OTG-INPVAL-005	Testing for SQL Injection
A1		Oracle Testing
A1		MySQL Testing
A1		SQL Server Testing
A1		Testing PostgreSQL
Error Handling		
A6	OTG-ERR-001	Analysis of Error Codes
Cryptography		
A3	OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection
A3	OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels
Business Logic Testing		
A10	OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use
Client Side Testing		
A7	OTG-CLIENT-001	Testing for DOM based Cross Site Scripting
A1	OTG-CLIENT-003	Testing for HTML Injection
A1	OTG-CLIENT-005	Testing for CSS Injection

Tabla 11. Pruebas implementadas siguiendo la metodología OWASP. Fuente: Propia.

En el **Anexo F** se detalla la descripción y objetivo de cada prueba implementada de la tabla 11.

Con respecto a lo anterior **Information gathering**, fue la subcategoría en la que se tuvo más énfasis al momento de implementar las pruebas, ya que es el punto de partida donde proporciona información relevante acerca de la configuración del servidor, la lógica de la aplicación, puertos y servicios abiertos, tecnologías del servidor web, lenguajes de programación, gestor de contenidos, extensiones criptográficas, bases de datos, sistema operativo y mapa de rutas de la aplicación. Todo esto comprende la fase inicial para realizar el pentesting sobre cualquier aplicación web, por un lado, se obtienen un amplio panorama del alcance y características que tiene el sistema a probar, por otro lado, su salida es la entrada para las pruebas contenidas en las subcategorías que le proceden según la metodología, haciendo que las demás fases del pentest tengan resultados satisfactorios.

En la figura 21 se puede observar las pruebas implementadas por cada subcategoría.

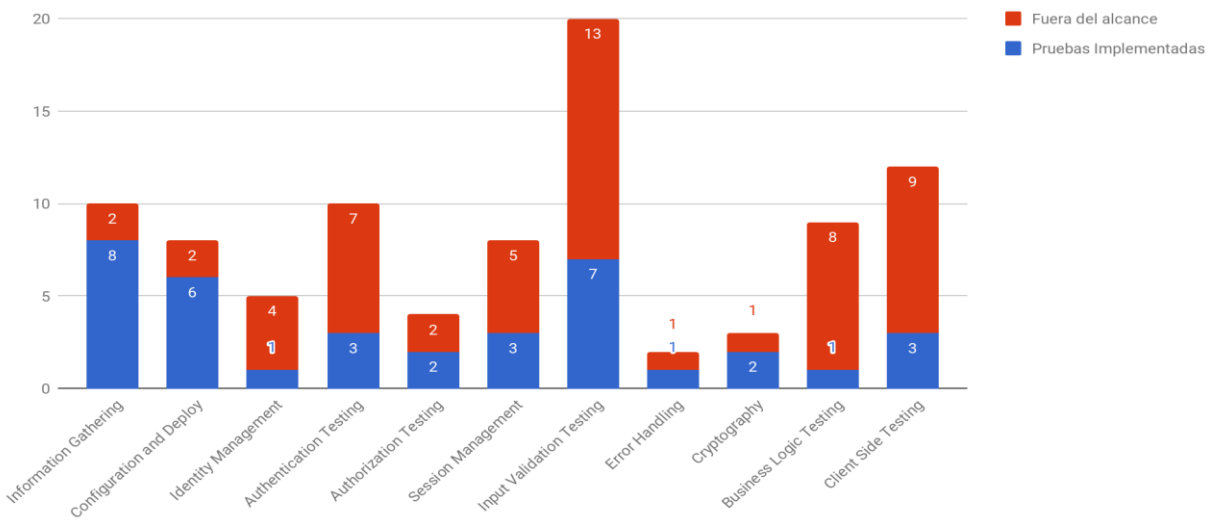


Figura 21. Pruebas implementadas por cada subcategoría. Fuente: Propia.

Las pruebas implementadas por A's son de vital de importancia, ya a que permite comprender e identificar los 10 tipos de riesgos más comunes que sufren las aplicaciones web en la actualidad. Para este trabajo de investigación se logró hacer un mayor énfasis en el A6, el cual consiste en la configuración de seguridad incorrecta, siendo este uno de los mayores problemas generados por los desarrolladores de software, debido a que no realizan el esfuerzo necesario en el ciclo de vida del desarrollo de software, especialmente en la fase de definición y diseño, permitiendo a los atacantes realizar acciones como: explotar vulnerabilidades conocidas, elevar privilegios dentro de la aplicación, acceder a páginas, archivos y/o directorios con información sensible que se encuentran sin algún tipo de protección. Además, se logró realizar varias pruebas para el A1 (inyección), el cual puede causar divulgación, pérdida o corrupción de información, pérdida de auditabilidad o denegación de acceso.

En la figura 22 se pueden observar las pruebas de concepto que se lograron implementar por el tipo de A.

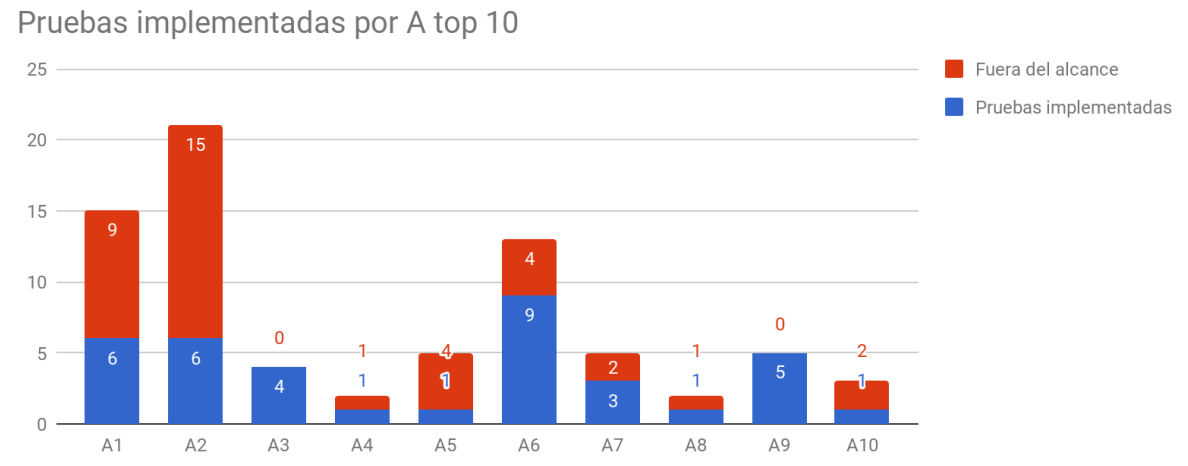


Figura 22. Pruebas implementadas por A'. Fuente: Propia

Cabe destacar que la decisión de implementar dichas pruebas se basó por una parte en el valor del riesgo que estas aportan, donde la mayoría encajan en el riesgo medio y alto (mirar sección 3.6.5 para ver el cálculo de riesgos). Por la otra parte estas pruebas son contempladas por la mayoría de herramientas tanto comerciales como open source.

3.6.3 Herramientas utilizadas

Para la implementación de las pruebas según la metodología OWASP Testing Guide v4, se utilizó el lenguaje de programación Python en su versión 3.x debido a que la mayoría de herramientas enfocadas hacia el pentesting sobre aplicaciones web están escritas sobre Python, dichas herramientas como algunas librerías (por ejemplo: requests y mechancalsoup) cuentan con licencias open source GNU GPL, lo que quiere decir que es posible analizar y modificar según las necesidades del proyecto.

Claro está que todas las pruebas implementadas no se desarrollaron desde cero, por eso, se vió la necesidad de utilizar herramientas de terceros que facilitaran el desarrollo del proyecto. A continuación, se listan las herramientas utilizadas:

- **Nmap:** Herramienta escrita en los lenguajes C, Lua y Python, la cual permite escanear y descubrir redes, dispositivos, servicios, sistema operativo y puertos sobre un servidor o aplicación web. Además, ofrece técnicas para evadir dispositivos de seguridad intermedios como los firewalls, IDS, IPS o algún mecanismo de seguridad que bloquee tráfico malicioso en la capa de red [65].

- **Whatweb:** Herramienta escrita en Ruby que permite reconocer las tecnologías web con sus respectivas versiones como gestor de contenidos, librerías JavaScript, servidores web, direcciones de correo, dispositivos embebidos, framework web y bases de datos [66].

- **WAFW00F:** Herramienta escrita en Python que detecta e identifica firewall o controles de seguridad de las aplicaciones web, enviando una petición HTTP y analiza la respuesta para determinar algún producto WAF (Web Application Firewall), si eso no tiene éxito, analiza las respuestas devueltas por el servidor y utiliza un algoritmo que permite estimar si es un WAF o una solución de seguridad que está respondiendo activamente a los ataques de la herramienta [67].

- **BruteSpray:** Herramienta escrita en Python que toma la salida de nmap de los puertos abiertos con su respectivo servicio y realiza un ataque de fuerza bruta con un diccionario de credenciales para un usuario y contraseña, el diccionario se puede personalizar dependiendo de la aplicación web [68].

- **theHarvester:** Herramienta escrita en Python que permite la recolección de subdominios, direcciones de correo electrónico, hosts virtuales, puertos abiertos, banners, nombres de usuario de diferentes motores de búsqueda que

puede ser de gran ayuda para la elaboración de los diccionarios para realizar ataques de fuerza bruta de credenciales [69].

- **Dirsearch:** Herramienta escrita en Python que por medio de fuerza bruta y diccionarios permite obtener tanto nombres de directorios como de archivos no indexados dentro de la aplicación web, los cuales, normalmente contienen información sensible [70].

- **TestSSL:** Herramienta escrita en BASH, la cual permite reconocer los tipos de protocolos (capa de transporte) configurados en un servidor web para el envío y recepción de datos cifrados, por ejemplo: SSLv3, TLS1.1, TLS1.2, etc. Además, permite identificar las vulnerabilidades correspondientes (si las hay) para los diferentes protocolos que encuentra, con su respectiva referencia.

También fueron muy útiles algunos comandos del sistema operativo Kali Linux como **wget** para descargar archivos o pàginas web, el comando **hosts** para realizar búsqueda sobre DNS y **grep** para realizar búsquedas sobre archivos y filtrar información de manera recursiva.

3.6.4 Proceso para repartir las pruebas de manera distribuida y paralela

Para distribuir las pruebas implementadas en cada dispositivo, fue necesario implementar un balanceador de carga con la capacidad de identificar cuáles pruebas requerían mayor tiempo de ejecución y las dependencias unas de otras, esto se hizo posible con ayuda de funciones nativas de MPI como **scattering** y **gathering**. Scattering (dispersión), es la función que permite distribuir o dispersar el conjunto de pruebas que debe realizar cada dispositivo, en este caso el dispositivo maestro es el encargado de ejecutar esta función. Una vez los dispositivos esclavos reciban el conjunto de pruebas, estos proceden a su ejecución de manera independiente, generando los resultados, recomendaciones y su respectivo riesgo. Finalmente, cuando todos los dispositivos hayan terminado sus procesos, ejecutan la función definida como Gathering (recoger/reunir) para notificarle al dispositivo maestro que el proceso ha finalizado para así completar el pentest.

La computación paralela permitió ejecutar algunas pruebas simultáneamente gracias a la librería **multiprocessing** de python [71]**,** mediante la cual fue posible aprovechar los múltiples recursos de los dispositivos como son los cuatro núcleos que poseen y así permitir que cada proceso funcione de manera independiente (dentro de cada dispositivo) para las pruebas que consumen demasiado tiempo de ejecución o que tengan aspectos en común. Algunas de las pruebas que se ejecutan de forma paralela son OTG-INFO-002 (huellas digitales del servidor web), OTG-INFO-008 (huellas digitales del framework) y OTG-INFO-009 (huellas digitales de la aplicación), las cuales se implementaron con ayuda de la herramienta *whatweb*. Asimismo, ocurrió un comportamiento similar en las pruebas OTG-INFO-002 (enumerar aplicaciones en el servidor web) y OTG-INFO-010 (mapa de arquitectura de la aplicación), pero estas pruebas se hicieron con ayuda de la herramienta *nmap*.

3.6.5 Uso de la metodología del cálculo de riesgo según OWASP

Para estimar la severidad del riesgo asociado a cada prueba se utilizó la metodología del cálculo de riesgo, definida en la sección 2.10. El documento OWASP Top 10 2017 calcula el riesgo para la explotabilidad, prevalencia, detectabilidad e impacto técnico para cada categoría, pero no tiene en cuenta tanto la probabilidad del agente de amenaza como el impacto del negocio. Para el agente de amenaza, cuyo objetivo es estimar la probabilidad de un ataque exitoso, se tuvo en cuenta cuatro factores, cada uno definido con valores asociados para lograr calcular el promedio de la probabilidad. Estos factores fueron modificados en un rango de 1 a 3 con el fin de obtener la misma escala de riesgos que utiliza la metodología OWASP Top 10. A continuación se describen los factores para el agente de amenaza:

- **Nivel de habilidad:** ¿Técnicamente cómo está capacitado el grupo de agentes de amenaza? Sin habilidades técnicas (1), Algunas habilidades técnicas (1), Usuario avanzado de informática (2), Habilidades de red y programación (2), Habilidades de penetración en seguridad (3).

- **Motivación:** ¿Qué tan motivado esta el grupo de agentes de amenaza para encontrar y explotar vulnerabilidades? Bajo o sin recompensa (1), posible recompensa (2), alta recompensa (3).

- **Oportunidad:** ¿Qué recursos y oportunidades se requieren para que el grupo de agentes de amenaza logre encontrar y explotar vulnerabilidades? Acceso total o requiere recursos costosos (1), Acceso especial o requiere recursos (2), Algunos accesos o requiere algunos recursos (3), Sin acceso o no requiere recursos (3).

- **Tamaño:** ¿Qué tan grande es el grupo de agentes de amenaza? Desarrolladores (1), administradores de sistemas (1), Usuario de la red interna (2), Socios (2), Usuarios autenticados (2), Usuarios anónimos de internet (3).

Cabe destacar que las preguntas para calcular el valor del agente de amenaza fueron contestadas por los estudiantes, tomando como referencia el peor de los casos.

La severidad del riesgo asociado a cada una de las pruebas se puede ver en la tabla 12. En el **Anexo G** se puede visualizar detalladamente el valor cuantitativo y cualitativo de cada uno de los factores para el agente de amenaza.

Prueba	Vectores de Ataque		Debilidades de Seguridad		Impacto	Puntaje
ID	Agente de Amenaza	Explotabilidad	Prevalencia	Detectabilidad	Técnico	
OTG-INFO-002	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	MODERADO:2	2,2
OTG-INFO-003	1,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,5
OTG-INFO-004	2,0	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,7
OTG-INFO-005	1,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,5
OTG-INFO-007	1,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,5

OTG-INFO-008	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	MODERADO:2	2,2
OTG-INFO-009	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	MODERADO:2	2,2
OTG-INFO-010	2,0	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,7
OTG-CONFIG-001	1,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,5
OTG-CONFIG-002	1,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,5
OTG-CONFIG-005	1,7	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,6
OTG-CONFIG-006	2,0	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,7
OTG-CONFIG-007	1,7	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	GRAVE:3	2,1
OTG-CONFIG-008	1,7	PROMEDIO:2	COMÚN:2	FÁCIL:3	GRAVE:3	2,1
OTG-IDENT-004	1,7	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,1
OTG-AUTHN-001	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	GRAVE:3	2,2
OTG-AUTHN-002	2,2	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,3
OTG-AUTHN-003	2,2	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,3
OTG-AUTHZ-003	2,2	PROMEDIO:2	COMÚN:2	PROMEDIO:2	GRAVE:3	2,0
OTG-AUTHZ-004	1,7	DIFÍCIL:1	COMÚN:2	PROMEDIO:2	MODERADO:2	1,6
OTG-SESS-002	1,7	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,1
OTG-SESS-003	1,5	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,1
OTG-SESS-007	1,7	FÁCIL:3	COMÚN:2	PROMEDIO:2	GRAVE:3	2,1
OTG-INPVAL-001	2,5	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,8
OTG-INPVAL-002	2,5	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,8
OTG-INPVAL-005	2,2	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,5
OTG-INPVAL-005	2,5	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,6
OTG-INPVAL-005	2,5	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,6
OTG-INPVAL-005	2,5	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,6
OTG-INPVAL-005	2,2	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,5
OTG-ERR-001	1,7	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,6
OTG-CRYPST-001	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	GRAVE:3	2,2
OTG-CRYPST-003	2,0	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	GRAVE:3	2,2
OTG-BUSLOGIC-007	1,7	PROMEDIO:2	DIFUNDIDO:3	DIFÍCIL:1	MODERADO:2	1,9
OTG-CLIENT-001	2,2	FÁCIL:3	DIFUNDIDO:3	FÁCIL:3	MODERADO:2	2,8
OTG-CLIENT-003	2,2	FÁCIL:3	COMÚN:2	FÁCIL:3	GRAVE:3	2,5
OTG-CLIENT-009	2,2	PROMEDIO:2	DIFUNDIDO:3	PROMEDIO:2	MODERADO:2	2,3

Tabla 12. Cálculo del riesgo para cada una de las pruebas implementadas. Fuente: Adapatado de [5]

El valor de la columna *agente de amenaza*, es el promedio de los factores anteriormente mencionados que logra contemplar la prueba. La *explotabilidad, prevalencia*, *detectabilidad* e *impacto técnico*, son valores que ya vienen definidos en el documento OWASP TOP 10 2017. Para calcular la *probabilidad* para cada prueba se lleva a cabo un promedio entre el agente de amenaza, explotabilidad, prevalencia y detectabilidad, teniendo así su respectivo valor en la columna *puntaje*.

Para calcular el impacto de negocio se debe solicitar al cliente o usuario responder cuatro preguntas, cada una definida con valores asociados que vienen siendo los factores necesarios para poder calcular impacto general. A continuación, se describen las preguntas a contestar:

- **Daño financiero:** ¿Cuál sería la pérdida financiera si se llegara a explotar alguna vulnerabilidad? Menor que la mitigación de la vulnerabilidad (1), menor al beneficio anual (1), mayor al beneficio anual (2), bancarrota (3)

- **Daño a la reputación:** Si alguna de las vulnerabilidades es explotada. ¿Cuál sería el daño en términos de reputación de negocio? Daño mínimo (1), pérdida de cuentas (2), pérdida de la buena imágen (2), pérdida del valor de marca (3)

- **Incumplimiento:** ¿Cuál sería el incumplimiento en términos de compromiso del negocio, si su aplicación es vulnerada? Incumplimiento menor (1), claro incumplimiento (2), incumplimiento de alto perfil (3)

- **Violación de la privacidad:** ¿Cuánta información personal sería revelada, si alguna vulnerabilidad es explotada? Información de una persona (1), información de centenares de personas (2), información de miles de personas (3), información de millones de personas (3)

Al igual que el agente de amenaza, el impacto de negocio se obtiene al promediar los valores de cada pregunta que fue contestada por el cliente o usuario. Luego se promedian los dos valores del impacto (técnico según la tabla 12 y del negocio) y se obtiene el impacto general.

Una vez calculado los valores cuantitativos para la probabilidad y el impacto de cada prueba, se deben multiplicar dichos valores para obtener la severidad general del riesgo (**Severidad general del riesgo = Probabilidad X Impacto**) y con este resultado conseguir el riesgo cualitativo. Esto se hace por medio de una escala (mirar tabla 13), la cual está dividida en tres rangos donde es posible obtener los valores **BAJO**, **MEDIO** y **ALTO** con intervalos entre 0 a 3, 3 a 6 y 6 a 9, respectivamente.

Niveles de Probabilidad e Impacto	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Tabla 13. Nivel de probabilidad e impacto para calcular la severidad del riesgo. Fuente.Tomado de [29]

Después de que se hayan clasificado los riesgos, el propietario o las personas encargadas de la aplicación web deberán realizar una lista priorizada de qué riesgos solucionar o mitigar. Como regla general, los riesgos críticos deben de ser corregidos al instante, no importa si su costo es alto o bajo o si para solucionarlos son demasiados complejos.

3.7 Despliegue del prototipo hardware y software

Lo anterior describe la configuración e implementación del cluster para su correcto funcionamiento, sin embargo, este solo funciona como una aplicación que es ejecutada por consola, en otras palabras, si se desea realizar un pentest sobre cualquier aplicación web se debe trasladar físicamente el cluster al sitio o tener un administrador encargado que ingrese el arduo comando para iniciar la tarea. Para dar solución a lo anterior y para que cualquier usuario final no tenga inconvenientes a la hora de realizar un pentest, se decidió implementar una aplicación web sencilla y amigable que proporciona las funcionalidades que se han venido tratado a lo largo de este capítulo.

Las tecnologías utilizadas para la implementación de la aplicación web fueron Flask, como el microframework para desarrollo web, la librería de Python llamada pdfkit la cual interactúa con la herramienta wkhtmltopdf (escrita en C++) que permite transformar contenido HTML a PDF para la generación de los reportes tanto ejecutivo como técnico. Cabe destacar que el tiempo tomado para la generación de los reportes oscila entre 5-10 segundos debido al renderizado de las gráficas.

A continuación, se muestra por medio de un diagrama de despliegue la comunicación entre los componentes de los distintos nodos que conforman el prototipo hardware y software, así como la disposición y relación física de los artefactos y dispositivos como servidores web, bases de datos, cluster (Back-end) y la aplicación web (Front-end)

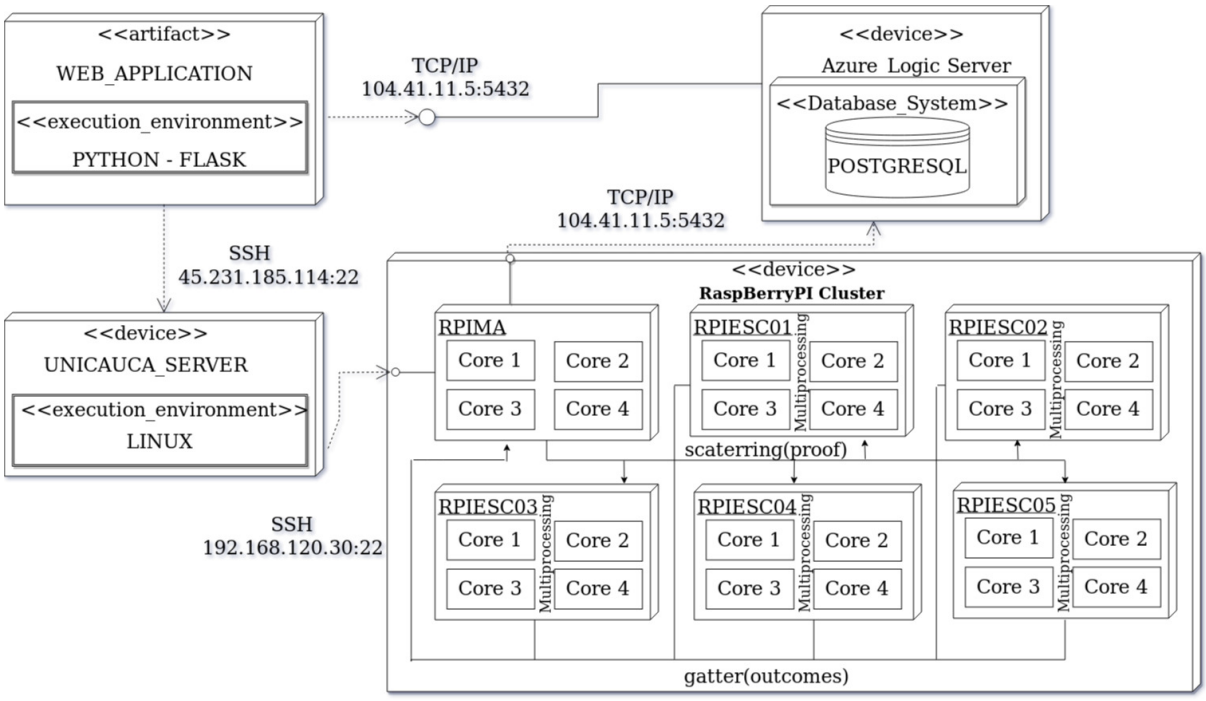


Figura 23. Diagrama de despliegue del prototipo hardware y software. Fuente: Propia.

Según la figura 23, la aplicación web establece una conexión por un lado con el servidor de base de datos alojado en los servicios de Microsoft Azure por medio del protocolo TCP/IP para crear y obtener la información correspondiente al pentest como es la dirección URL/IP de la aplicación web que proporciona el usuario para realizar

el pentest. Por otro lado, se establece una comunicación con un servidor alojado en la Universidad del Cauca el cual ejecuta una distribución de Debian utilizando el protocolo SSH (Secure SHell) para establecer una conexión directa con el cluster y poder enviar el comando necesario para realizar el pentest a la aplicación web deseada, esto se hizo de esta manera ya que físicamente el cluster se encuentra dentro de las instalaciones de la Universidad del Cauca. Finalmente, el servidor establece una comunicación nuevamente por SSH con el **dispositivo maestro** del cluster para así realizar el pentest y almacenar toda la información en la base de datos en Azure.

3.8 Interfaz de la aplicación web y generación de reportes

En esta sección se presenta brevemente la apariencia de la aplicación web y cómo se genera el reporte técnico y ejecutivo.

La figura 24, presenta la página principal de la aplicación web llamada **ScanLynx**, donde se le da la opción al usuario de ingresar la dirección IP/URL y un nombre para poder reconocerlo fácilmente de la aplicación web a la que desearía realizar un pentest.

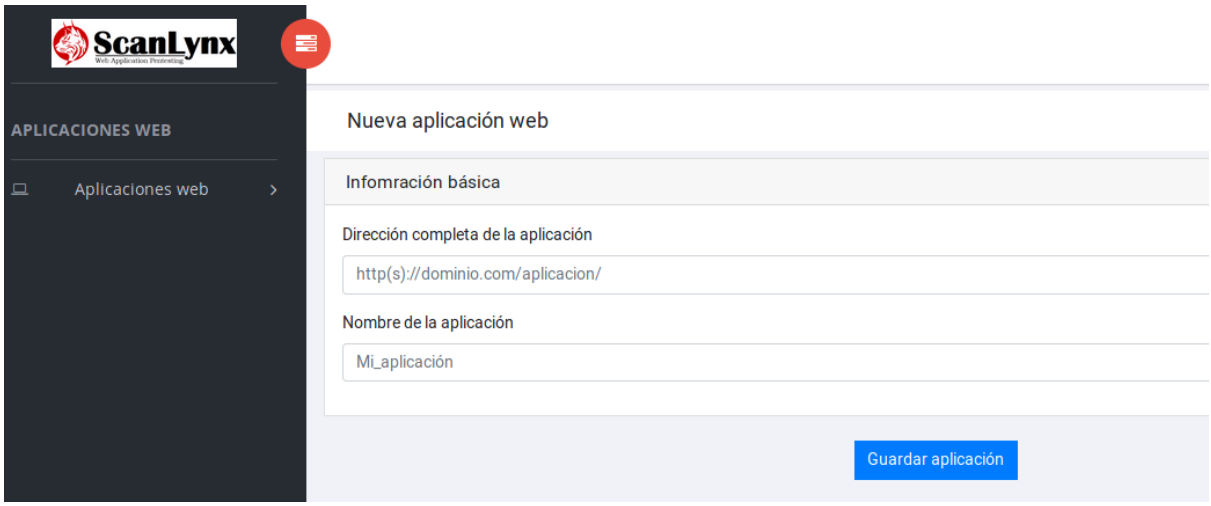


Figura 24. Página principal ScanLynx. Fuente: Propia.

Una vez almacenada la información la figura 25 representa la interfaz donde el usuario puede apreciar una lista de todas las aplicaciones web que ha ingresado, como también los estados respectivos estados de ejecución **no iniciado**, **en progreso**, y **finalizado** Por último se presentan 3 acciones para administrar los elementos de la lista, el primero es ejecutar el pentest donde se le despliega en una página diferente las diferentes subcategorías que el usuario desee escoger, la segunda es eliminar una aplicación web con toda su información, la tercera y última, es la generación de reportes tanto técnico como el ejecutivo.

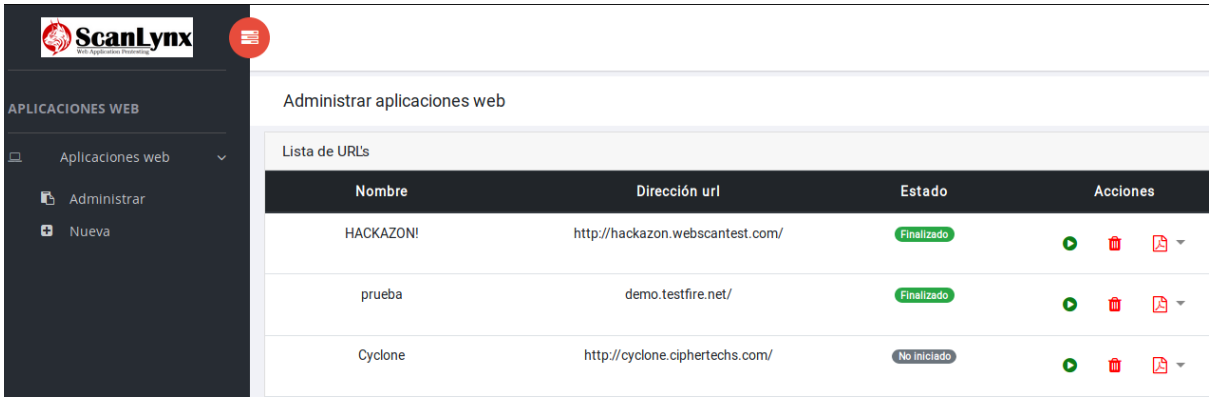


Figura 25. Administrar aplicaciones web. Fuente: Propia.

Al querer ejecutar el pentest de cualquier aplicación web, el usuario podrá observar y seleccionar (según sus necesidades) una o todas las 11 subcategorías que plantea la guía OWASP Testing Guide v4 para poder realizar el pentest (mirar figura 25). En cada subcategoría se encuentran las preguntas definidas en la sección 3.6.5 para el cálculo de riesgo de negocio, las cuales deben ser contestadas por el usuario debido a que él es el único que puede estimar tal riesgo.

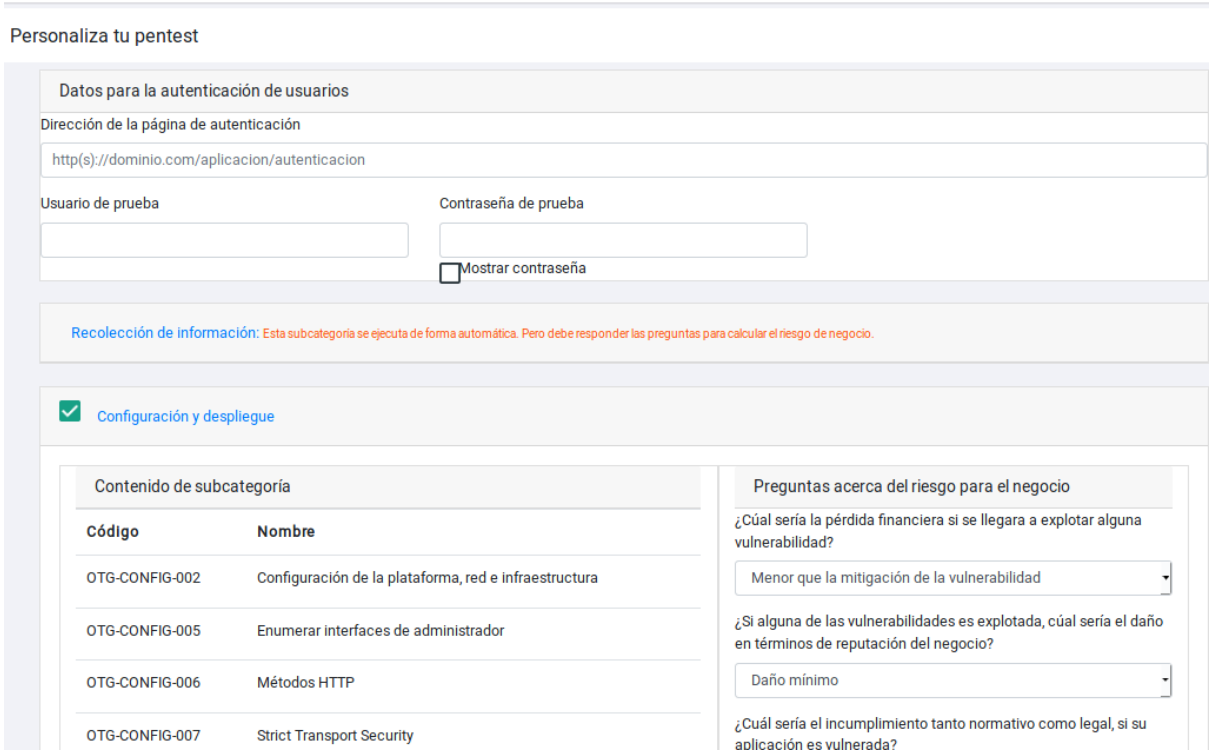


Figura 26. Elección de las pruebas de concepto a realizar. Fuente: Propia.

Por último, los reportes se generan en formato PDF como se puede apreciar en la figura 26 donde se muestra como portada el nombre del prototipo implementado y la información básica como la dirección IP/URL de la aplicación probada, la fecha de inicio y fin junto su respectiva duración y la cantidad de vulnerabilidades encontradas en términos de severidad.

Figura 27. Presentación del reporte ejecutivo y técnico. Fuente: Propia.

En el **Anexo H** se puede observar los reportes completos.

4 CAPÍTULO IV. EXPERIMENTO CONTROLADO

En esta última etapa se llevó a cabo la prueba experimental del prototipo hardware y software construido, con el fin de seguir un método que guíe la solución del problema [12], aplicado a diferentes grupos de aplicaciones web, en el cual se verificó la hipótesis planteada y se analizaron e interpretaron los datos recolectados mediante las curvas ROC y análisis de rendimiento. Para lograr esto se realizaron las siguientes etapas:

4.1 Definición y Planeación

Esta etapa va enfocada a la definición de las actividades para definir el programa de medición y análisis teniendo en cuenta los objetivos tanto el general como los específicos y la hipótesis. Para lograr esto se realizó un experimento controlado donde se crearon dos grupos:

- Grupo de control: son los que proporcionan una línea base para observar si el procedimiento aplicado tiene un efecto distinto al realizado en el grupo experimental.
- Grupo experimental: en este grupo se manipulan las variables independientes para luego observar su efecto en las variables dependientes (mirar sección 4.1.2).

FInalmente se comparan ambos resultados y si la proporción de los resultados deseados es mayor en el grupo experimental que en el grupo de controlado, se puede afirmar que los resultados son satisfactorios, en caso contrario serían contraproducentes.

Con respecto a lo anterior se establecieron una serie de actividades para alcanzar dicho objetivo:

- Identificar las variables tanto dependientes como independientes.
- Validar los datos recolectados de la medición mediante estadística descriptiva y analítica.
- Analizar los resultados obtenidos en la medición y generar las respectivas conclusiones.

4.1.1Hipótesis

La hipótesis planteada a verificar es la siguiente: *es posible identificar las vulnerabilidades más comunes sobre aplicaciones web de manera eficiente y eficaz, utilizando un cluster conformado por SBC de bajo costo para determinar la severidad de los riesgos y que permita generar reportes automáticos.*

4.1.2Definición de variables

Variable independiente: las variables seleccionadas para el experimento fueron la dirección URL o dirección IP de la aplicación web, el riesgo de negocio definido para cada aplicación y el tipo de prueba que se llevó a cabo (full pentesting, customizable pentesting) y así lograr observar el efecto en la variable dependiente.

Variable dependiente: para observar la conducta que tiene el experimento y evaluar los efectos de la variable independiente se analizó los resultados de los reportes tanto técnico como ejecutivo de cada aplicación probada.

4.1.3Grupos experimentales y grupos de control

Los grupos para realizar el experimento controlado se escogieron de acuerdo a las vulnerabilidades más comunes (basadas en el documento OWASP Top 10 2017) que

presentan las aplicaciones web y que permitieron determinar la severidad de los riesgos por cada vulnerabilidad encontrada.

A continuación, en la tabla 14 se listan diez aplicaciones web que cumplen con los requisitos anteriormente mencionados, las cuales están divididas en el grupo experimental (tratamiento que se desea investigar) y grupo de control (tratamiento normal).

Grupo	Desarrollado por	URL
Experimental	Acunetix	http://testphp.vulnweb.com/
	IBM	http://demo.testfire.net/
	Netsparker	http://php.testsparker.com
	NTOSpider	http://www.webscantest.com/
	Rapid7	http://hackazon.webscantest.com/
Controlado	Grupo Enigma	https://www.enigmagroup.org/
	HackThis	https://www.hackthis.co.uk/
	OWASP	http://cyclone.ciphertechs.com/
	HP	http://zero.webappsecurity.com/
	OWASP	https://juice-shop.herokuapp.com/

Tabla 14. Aplicaciones web vulnerables. Fuente: Propia.

4.1.4Ambiente de pruebas

En este paso fue necesario especificar el ambiente de pruebas o de ejecución para poder recolectar los diferentes datos. Para esto se estableció una fecha fija donde se contempla el tiempo requerido del experimento controlado, se intentó que todas las actividades descritas en esta sección fueran en un mismo lugar geográfico y con la misma infraestructura de red, con el fin de no tener alteraciones de datos en cuanto a la detección de vulnerabilidades y los tiempos de ejecución. Asimismo, se estableció un entorno de trabajo el cual comprende el uso de dos equipos computacionales que fueron los encargados de ejecutar las diferentes herramientas para el pentesting sobre aplicaciones web. Por un lado, está el **equipo 1** con arquitectura de 64 bits, el cual fue el encargado de ejecutar las siguientes herramientas: Acunetix, Netsparker, Nessus y OWASP ZAP, todas con sus respectivas aplicaciones de escritorio. Por otro lado, está el prototipo al cual hacemos referencia como **equipo 2** con arquitectura ARMv8, este fue el encargado de ejecutar la herramienta **ScanLynx**. A continuación, en la tabla 15, se describe de manera detallada el ambiente de pruebas que se realizó para este experimento.

Encargado de la medición	Andrés Muñoz, Santiago Pérez
Espacio temporal	Un mes (1/08/2018 - 3/09/2018)
Espacio geográfico	Universidad del Cauca, salón 101
Medios de recolección	Computacional
Entorno de trabajo	**Equipo 1**: Computador portátil - Lenovo ideapad 300 **Software utilizado**: Acunetix, Netsparker, Nessus, OWASP ZAP **Procesador**: Intel Core i7-6500U 6ta gen **Velocidad del procesador**: 2.5 GHz **RAM**: 6 GB **Sistema operativo**: Debian 9 x64 **Equipo 2**: Cluster Raspberry Pi 3 model B **Software utilizado**: ScanLynx **Procesador**: Broadcom Quad Core x6 **Velocidad del procesador**: 1.2 GHz x6 **RAM**: 1GB x6 **Sistema operativo**: Kali Linux 2018.1 ARM

Tabla 15. Detalles del ambiente de prueba. Fuente: Propia.

Cabe resaltar que la razón por la cual el cluster solo ejecutó la herramienta **ScanLynx**, es por la arquitectura con la que fueron diseñados las demás herramientas por lo que fue necesario ejecutarlas sobre un computador con una arquitectura de 64 bits.

Finalmente se debe aclarar que el pentesting ejecutado sobre las diferentes aplicaciones web fue a través de la red cableada en horas de la noche, debido a que la velocidad del ancho de banda en la Universidad del Cauca es de 90 Mbits/sg comparado con la velocidad en horas de la mañana o tarde que es en promedio de 35 Mbits/sg según el test de velocidad de virgin mobile[12].

4.2 Procedimiento del experimento controlado

El primer paso que se hizo fue seleccionar las herramientas de pentesting para comparar las vulnerabilidades identificadas y tiempos de ejecución por el prototipo hardware y software **ScanLynx** contra las herramientas de pentesting seleccionadas **Netsparker**, **Acunetix**, **Nessus y OWASP ZAP** definidas en la sección 2.11.

El procedimiento que se aplicó al grupo controlado de acuerdo a las variables independientes fue el siguiente:

- Tipo de prueba: se escogieron algunas pruebas de concepto que tienen implementadas las herramientas seleccionadas y el prototipo implementado (**customizable pentesting**)

[12] https://www.virginmobile.co/web/virgin/sac/servicios/medidor-de-velocidad

- Riesgo de negocio: el riesgo cuantitativo seleccionado para el impacto de negocio fue **2.0**, el cual se promedió con el impacto técnico para obtener el riesgo general del impacto.

Las pruebas seleccionadas en este grupo de control se enfocaron en identificar vulnerabilidades definidas en las subcategorías de la metodología OWASP como: **configuración y gestión de despliegue, pérdida de control de acceso**, **recolección de información**, **manejo de errores** y **gestión de la sesión**, por tal motivo se escogió un pentesting personalizado y se seleccionó un riesgo de cuantitativo de 2.0 lo que cualitativamente equivale a *bajo*, ya que se estimaba que los riesgo identificados en estas aplicaciones web era de una severidad *media* o *baja***.**

Los resultados arrojados en la detección de vulnerabilidades no fue el más óptimo debido a que las herramientas como **Acunetix y Nessus** no lograron finalizar el pentesting para la aplicación *https://juice-shop.herokuapp.com* y el prototipo implementado **ScanLynx** no logró terminar las pruebas de penetración para la aplicación web *http://cyclone.ciphertechs.com*/cyclone, lo cual no permite realizar un análisis minucioso de las vulnerabilidades identificadas y tiempos de ejecución.

El procedimiento aplicado al grupo experimental fue el siguiente:

- Tipo de prueba: se escogieron todas las pruebas de concepto que tienen implementadas las herramientas seleccionadas y el prototipo implementado (**full pentesting**)
- Riesgo de negocio: el riesgo cuantitativo seleccionado para el impacto de negocio fue **3.0**, el cual se promedió con el impacto técnico para obtener el riesgo general del impacto.

Para este grupo experimental se esperaba obtener las vulnerabilidades más comunes con una severidad de riesgo *bajo*, *medio* y *alto*, por tal motivo se escogió un riesgo cuantitativo de 3.0 lo que equivale a un riesgo cualitativo *medio*, de esta manera se ejecutaron todas las pruebas de concepto que tienen implementadas las herramientas seleccionadas y el prototipo hardware y software.
Los resultados de las vulnerabilidades identificadas por todas las herramientas fueron satisfactorios, debido a que se logró ejecutar el pentesting en cada aplicación sin problemas al momento de realizar todas las pruebas de concepto.

4.2.1Análisis mediante ROC

Para comparar los resultados de *ScanLynx* contra las herramientas seleccionadas y conocer si la detección de vulnerabilidades en el grupo experimental se realizó de manera eficaz fue necesario un análisis mediante las curvas ROC [72].

Lo primero que se hizo fue identificar las vulnerabilidades que había detectado el prototipo sobre cada aplicación y así comparar con los resultados de cada una de las herramientas, de las cuales solo se seleccionaron las pruebas que tenían en común tanto el prototipo como el resto de herramientas, debido a que estás tiene implementadas más pruebas de concepto que el prototipo hardware y software.

Una vez realizado este procedimiento sobre cada aplicación web se procede a identificar los siguientes criterios entre el prototipo implementado contra cada una de las herramientas (*ScanLynx vs Netsparker, ScanLynx vs Nessus, ScanLynx vs Acunetix, ScanLynx vs OWASP ZAP*), se tomó como **verdad absoluta** las vulnerabilidades que detectaron cada una de las herramientas seleccionadas:

- **VP (Verdaderos positivos):** vulnerabilidades en común que logró detectar tanto *ScanLynx* como cada una de las herramientas.

- **(VN) Verdaderos Negativos:** fueron las vulnerabilidades que la aplicación web tenía, pero no logró detectar el prototipo ni tampoco la herramienta en cuestión.

- **(FP) Falsos positivos:** vulnerabilidades que detectó *ScanLynx,* pero no detectó la herramienta comparada.

- **(FN) Falsos negativos:** se tuvieron en cuenta las vulnerabilidades que detectó la herramienta, pero no logró detectar el prototipo implementado.

Luego se procedió asignar un valor binario: **1**: si detectó una vulnerabilidad; **0**: no detectó vulnerabilidad. En la tabla 16 se puede observar el procedimiento realizado sobre la aplicación web *http://php.testparker.com* entre ScanLynx contra Netsparker, con la vulnerabilidad clasificada según él **A'**, su valor binario y su respectivo resultado (medida de evaluación)

Herramientas / Vulnerabilidades	**ScanLynx**	**Netsparker**	**Medida de evaluación**
A1 (Injection)			
SQL Injection	0	1	FN
Testing for HTML Injection	0	0	VN
A2 (Broken Authentication and Session Management)			
Weak lock mechanism	1	1	VP
Session Fixation	1	0	FP
Cierre de sesión automático	1	0	FP
Test Session Timeout	0	0	VN
A3 (Sensitive Data Exposure)			
HTP HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	0	FP
A4 (XML EXternal Entity)			
Test RIA cross domain policy	0	1	FN
A5 (Broquen Access Control)			

Testing Directory traversal/file include	0	0	VN
A6(Security Misconfiguration)			
Map execution paths through application	1	1	VP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	0	FP
Revisar comentarios y metadatos	0	0	VN
Infrastructure y platform Configuration	0	1	FN
Test File Extensions Handling for Sensitive Information	0	1	FN
A7 (XSS)			
Cross Site Scripting	0	1	FN
A8 (Insecure Deserialization)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Using Components with Known Vulnerabilities)			
Apache server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	0	FP
Open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Insufficient Login and Monitoring)			
Test Defenses Against Application Mis-use	1	0	FP

Tabla 16. Vulnerabilidades detectadas entre ScanLynx vs Netsparker. Fuente: Propia.

Para observar los resultados entre el prototipo implementado contra cada una de las herramientas que se ejecutó sobre cada aplicación web puede ver el **Anexo I**

Una vez se encontró la medida de evaluación para cada vulnerabilidad se procedió a obtener los resultados de las razones de los verdaderos positivos, falsos positivos y verdaderos negativos:

- **VPR (Razón de Verdaderos Positivos o sensibilidad):** mide hasta qué punto se clasificaron las vulnerabilidades correctamente, entre todos los casos positivos disponibles durante la prueba

$$VPR = VP / (VP + FN)$$

- **FPR (Razón de Falsos Positivos):** define cuántas vulnerabilidades son incorrectas entre todos los casos negativos disponibles durante la prueba

$$FPR = FP / (FP + VN)$$

- **SPC (Razón de Verdaderos Negativos o especificidad):** probabilidad de clasificar correctamente una vulnerabilidad que sea definida como negativa

$$SPC = VN / (FP + VN) = 1 - FPR$$

4.3 Análisis, interpretación y presentación de los datos recolectados para medir la eficacia del prototipo construido

A continuación en la tabla 17 se presentan los resultados con la medida de evaluación de todas las herramientas comparadas ejecutadas en la aplicación *http://php.testparker.com*

Aplicación: php.testsparker.com	**ScanLynx VS**			
Medidas de evaluación	**Netsparker**	**OWASP ZAP**	**Acunetix**	**Nessus**
VP (Verdaderos Positivos)	7	2	13	8
FP (Falsos Positivos)	9	14	3	8
FN (Falsos Negativos)	6	0	6	1
VN (Verdaderos Negativos)	4	10	4	9
VPR (Razón de Verdaderos Positivos)	0,54	1	0,68	0,89
FPR (Razón de Falsos Positivos)	0,7	0,58	0,43	0,47
SPC (Razón de Verdaderos Negativos)	0,31	0,42	0,57	0,53

Tabla 17. Medidas de evaluación entre ScanLynx contra cada una de las herramientas para la aplicación web testparker. Fuente: Propia.

El mejor método posible de predicción se situaría en un punto en la esquina superior izquierda, o coordenada (0,1) del espacio ROC, representando un 100% de sensibilidad (ningún falso negativo) y un 100% también de especificidad (ningún falso positivo). A este punto (0,1) también se le llama una clasificación perfecta.

La diagonal divide el espacio ROC. Los puntos por encima de la diagonal representan los buenos resultados de clasificación (mejor que el azar), puntos por debajo de la línea de los resultados pobres (peor que al azar).

En la figura 28 se puede observar el comportamiento de la curva ROC para la aplicación *http://php.testparker.com*

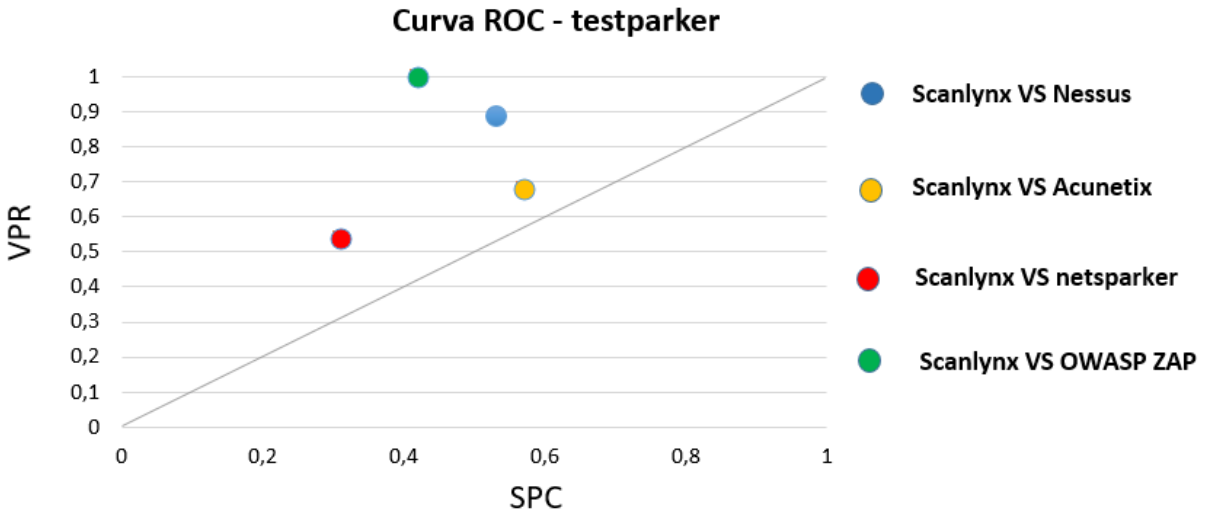


Figura 28. Curva ROC de ScanLynx Vs cada una de las herramientas para la aplicación web testparker. Fuente: Propia.

Para la aplicación web *php.testparker,* la predicción de ScanLynx contra Netsparker (punto rojo) representa una buena clasificación de las vulnerabilidades, debido a que está por encima de la **línea de no-discriminación** (diagonal), aunque se detectaron **7** verdaderos positivos (ver la tabla 15 para cada uno de los valores mencionados), estos fueron las vulnerabilidades que detectaron en común, también obtuvo el mejor valor en especificidad **0,31**, ya que Netsparker detectó **6** vulnerabilidades que no logró identificar ScanLynx. Esto se debe a la gran cantidad de pruebas que tiene implementada la herramienta Netsparker. La comparación con OWASP ZAP arrojó la clasificación perfecta para la razón de los falsos positivos **1**, debido a que esta herramienta no logró detectar vulnerabilidades distintas a las que identificó ScanLynx, pero la clasificación de la razón de los falsos positivos fue regular **0,58,** ya que se obtuvieron demasiados falsos positivos **14** (vulnerabilidades que detectó ScanLynx pero no identificó OWASP ZAP), una razón de este resultado es la poca detección de vulnerabilidades que realizó esta herramienta sobre esta aplicación web. Con respecto a Acunetix obtuvo el valor deficiente para la especifidad en toda la prueba **0,57**, esto se debe a la cantidad de falsos negativos **6**, debido a que es una herramienta comercial que permite generar reportes basados en OWASP. El último análisis con Nessus permitió obtener un buen resultado en la sensibilidad **0,89**, ya que identificó solamente una vulnerabilidad que no detectó ScanLynx, esto caracteriza la probabilidad de que se clasificó correctamente las vulnerabilidades detectadas por el prototipo.

A continuación en la tabla 18 se muestran los resultados con las medidas de evaluación de todas las herramientas comparadas ejecutadas en la aplicación *http://demo.testfire.com*

http://demo.testfire.com	**Scanlynx VS**			
Medidas de evaluación	**Netsparker**	**OWASP ZAP**	**Acunetix**	**Nessus**
VP (Verdaderos Positivos)	10	5	11	8
FP (Falsos Positivos)	6	10	4	7

FN (Falsos Negativos)	3	1	5	2
VN (Verdaderos Negativos)	2	4	1	4
VPR (Razón de Verdaderos Positivos)	0,77	0,83	0,69	0,80
FPR (Razón de Falsos Positivos)	0,75	0,71	0,80	0,64
SPC (Razón de Verdaderos Negativos)	0,25	0,29	0,20	0,36

Tabla 18. Medidas de evaluación entre ScanLynx contra cada una de las herramientas para la aplicación web testfire. Fuente: Propia.

En la figura 29 se puede observar el comportamiento de la curva ROC para la aplicación *http://php.testparker.com*

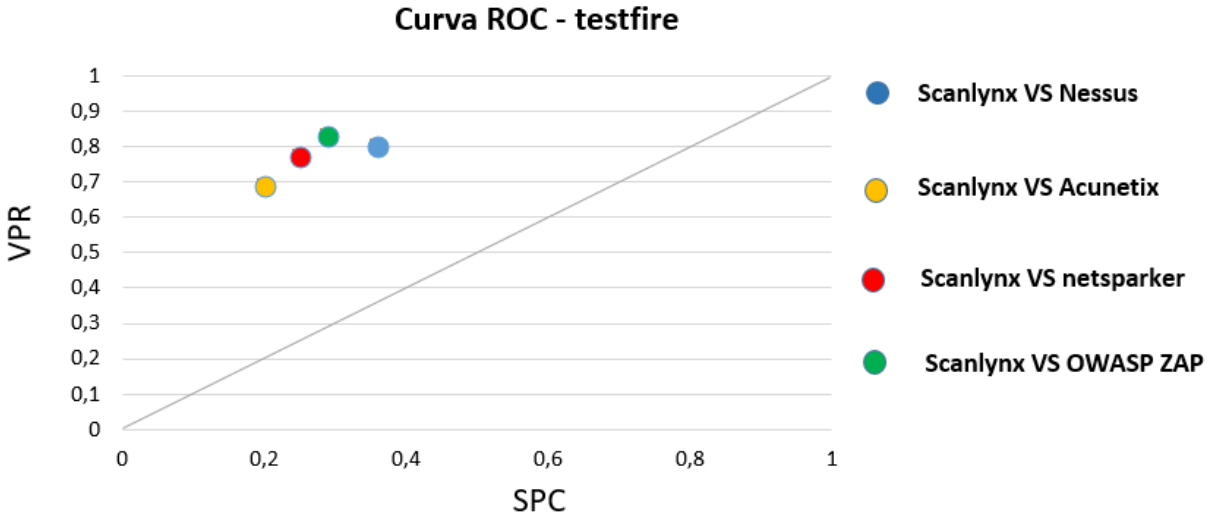


Figura 29. Curva ROC de ScanLynx Vs cada una de las herramientas para la aplicación web testfire. Fuente: Propia.

La mejor predicción en especificidad la obtuvo Scanlynx contra Acunetix (punto naranja), debido a que solamente se detectó **1** verdadero negativo (ver tabla 16 para cada uno de los valores mencionados), esto quiere decir que ninguna de las dos herramientas en cuestión lograron identificar una vulnerabilidad que tenía la aplicación web. La comparación contra OWASP ZAP arrojó el mejor resultado para la sensibilidad (razón de verdaderos positivos) **0,83**, ya que fue el menor valor en falsos negativos (**1**) en toda la prueba, esto significa que OWASP ZAP detectó una vulnerabilidad que no identificó ScanLynx). Con respecto a Nessus se obtuvo el valor más deficiente en especificidad **0,36**, debido a que se detectaron **4** verdaderos negativos, el peor valor en toda la prueba, esto se debe a que esta herramienta detecta vulnerabilidades en la fase inicial del pentest (**recolección de información**) pero olvida las demás pruebas de concepto. La última comparación contra Netsparker arrojó una predicción buena tanto en sensibilidad como en especificidad, debido a que se detectaron **10** verdaderos positivos (vulnerabilidades que detectaron en común) y **3** falsos negativos (vulnerabilidades identificó Netsparker, pero no logró identificar ScanLynx), puesto que esta herramienta tiene implementadas más pruebas

con respecto al prototipo y gracias a esto se presentaron más vulnerabilidades en común.

En general la predicción de las herramientas en cuestión para la aplicación web fue buena, dado que está por encima de la **línea de no-discriminación**.

A continuación en la tabla 19 se muestran los resultados con las medidas de evaluación de todas las herramientas comparadas ejecutadas en la aplicación web *http://testphp.vulweb.com*

http://testphp.vulweb.com	**Scanlynx VS**			
Medidas de evaluación	**Netsparker**	**OWASP ZAP**	**Acunetix**	**Nessus**
VP (Verdaderos Positivos)	13	5	15	11
FP (Falsos Positivos)	5	12	3	6
FN (Falsos Negativos)	2	0	3	0
VN (Verdaderos Negativos)	2	4	2	4
VPR (Razón de Verdaderos Positivos)	0,87	1	0,83	1,00
FPR (Razón de Falsos Positivos)	0,71	0,75	0,60	0,60
SPC (Razón de Verdaderos Negativos)	0,29	0,25	0,40	0,40

Tabla 19. Medidas de evaluación entre ScanLynx contra cada una de las herramientas para la aplicación web testphp. Fuente: Propia.

En la figura 30 se puede observar el comportamiento de la curva ROC para la aplicación web *http://testphp.vulweb.com*

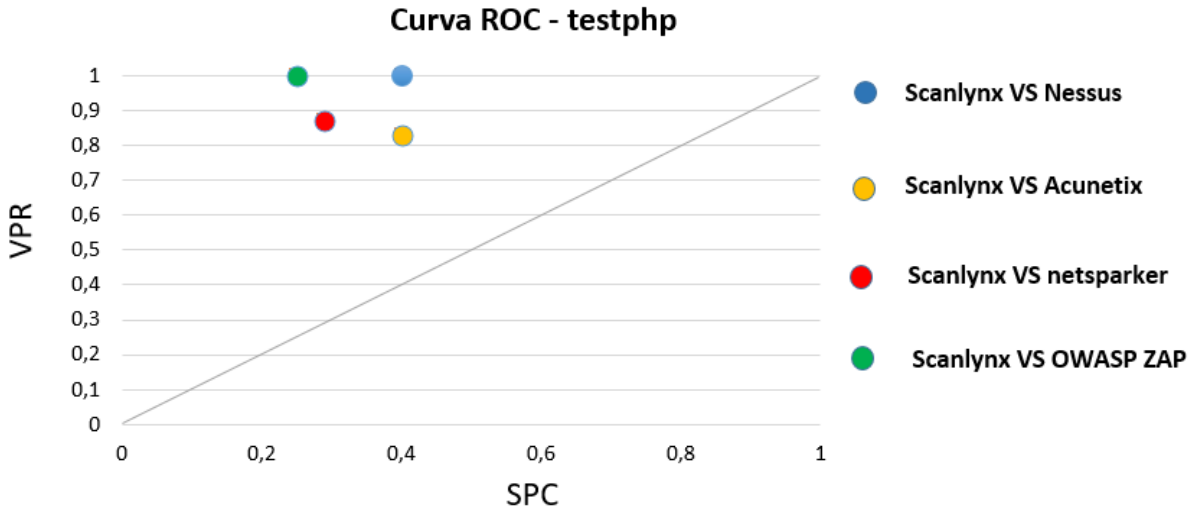


Figura 30. Curva ROC de ScanLynx Vs cada una de las herramientas para la aplicación web testphp. Fuente: Propia.

La predicción de ScanLynx contra OWASP ZAP y Nessus en sensibilidad (probabilidad de que se clasificaron correctamente las vulnerabilidades detectadas por ScanLynx) fue una calificación perfecta **1**, puesto que ninguna de las dos herramientas detectó falsos negativos. La predicción de ScanLynx contra Netsparker (punto rojo) en razón de los verdaderos positivos fue buena, ya que tuvieron en común **13** vulnerabilidades detectadas (verdaderos positivos) y solamente el prototipo no logró identificar **2** vulnerabilidades que detectó Netsparker. Con respecto Acunetix se obtuvo una buena sensibilidad **0,83**, dado que se presentó la mayor ocurrencia de verdaderos positivos **15,** esto se debe a la gran cantidad de pruebas de concepto que tiene implementada esta herramienta enfocadas en la metodología de OWASP, pero en razón de los verdaderos negativos fue el resultado más deficiente **0,40** en toda la prueba porque Scanlynx no logro detectar **3** vulnerabilidades (falsos negativos) que identificó Acunetix.

En general el comportamiento de la prueba en todo el grupo experimental fue bueno, ya que se logró tener una calificación perfecta en razón de los verdaderos positivos. Esto quiere decir que la probabilidad de que se detectaron de manera eficaz las vulnerabilidades que lograron detectar OWASP ZAP y Nessus fue del 100%.

A continuación en la tabla 20 se muestran los resultados con las medidas de evaluación de todas las herramientas comparadas ejecutadas en la aplicación web *http://www.webscantest.com*

http://www.webscantest.com	**Scanlynx VS**			
Medidas de evaluación	**Netsparker**	**OWASP ZAP**	**Acunetix**	**Nessus**
VP (Verdaderos Positivos)	11	6	14	10
FP (Falsos Positivos)	9	14	6	10
FN (Falsos Negativos)	3	2	2	1
VN (Verdaderos Negativos)	3	3	3	4
VPR (Razón de Verdaderos Positivos)	0,79	0,75	0,88	0,91
FPR (Razón de Falsos Positivos)	0,75	0,82	0,67	0,71
SPC (Razón de Verdaderos Negativos)	0,25	0,18	0,33	0,29

Tabla 20. Medidas de evaluación entre ScanLynx contra cada una de las herramientas para la aplicación web webscantest. Fuente: Propia.

En la figura 31 se puede observar el comportamiento de la curva ROC para la aplicación web *http://www.webscantest.com*

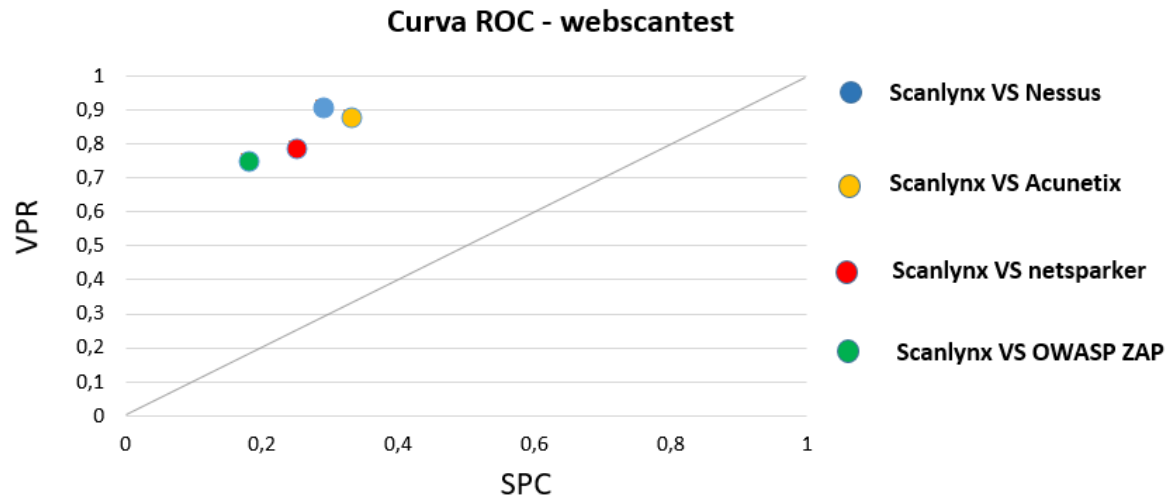


Figura 31. Curva ROC de ScanLynx Vs cada una de las herramientas para la aplicación web webscantest. Fuente: Propia.

Para la aplicación web *webscantest* la mejor predicción en cuanto a especificidad fue contra OWASP ZAP **0,18**, debido a que ScanLynx con esta herramienta no lograron detectar 3 vulnerabilidades que tenía la aplicación web y se obtuvieron **14** falsos positivos (vulnerabilidades que detectó el prototipo, pero no OWASP ZAP), esto se debe a las pocas vulnerabilidades que detecta la herramienta en cuestión.
El mejor valor para la razón de los verdaderos positivos **0,91** fue contra Nessus, puesto que dicha herramienta solamente identificó una vulnerabilidad en la fase de recolección de información que no logro detectar ScanLynx. La comparación contra Netsparker fue satisfactoria, ya que se obtuvo una buena predicción (punto rojo) con respecto a las medidas de evaluación (sensibilidad y especificidad), debido que se tuvieron en común **11** vulnerabilidades que tenía la aplicación web y solamente el prototipo no lo logró detectar **3** vulnerabilidades que identificó la herramienta Netsparker.

Con respecto a Acunetix se logró detectar la mayor cantidad de vulnerabilidades en común **14** verdaderos positivos, esto da indica que el prototipo implementado está realizando las pruebas de concepto de manera satisfactoria y está produciendo el efecto esperado en la identificación de las vulnerabilidades más comunes.

El resultado de la prueba en general fue satisfactorio debido a que la predicción para la sensibilidad y especificidad no varió mucho, ya que los valores para la razón de verdaderos positivos estuvo en un rango entre **0,75 y 0,91** y la razón de los verdaderos negativos estuvo en un rango entre **0,18 y 0,33** resultados cercanos a la prueba perfecta para la sensibilidad y especificidad.

A continuación en la tabla 21 se muestran los resultados con las medidas de evaluación de todas las herramientas comparadas ejecutadas en la aplicación web *http://www.hackazon.webscantest.com*

http://www.hackazon.webscantest.com	**Scanlynx VS**			
Medidas de evaluación	**Netsparker**	**OWASP ZAP**	**Acunetix**	**Nessus**

VP (Verdaderos Positivos)	13	5	14	13
FP (Falsos Positivos)	7	14	6	6
FN (Falsos Negativos)	2	3	3	1
VN (Verdaderos Negativos)	2	2	2	4
VPR (Razón de Verdaderos Positivos)	0,87	0,63	0,82	0,93
FPR (Razón de Falsos Positivos)	0,78	0,88	0,75	0,60
SPC (Razón de Verdaderos Negativos)	0,22	0,13	0,25	0,40

Tabla 21. Medidas de evaluación entre ScanLynx contra cada una de las herramientas para la aplicación web hackazon. Fuente: Propia.

En la figura 32 se puede observar el comportamiento de la curva ROC para la aplicación web *http://www.hackazon.webscantest.com*

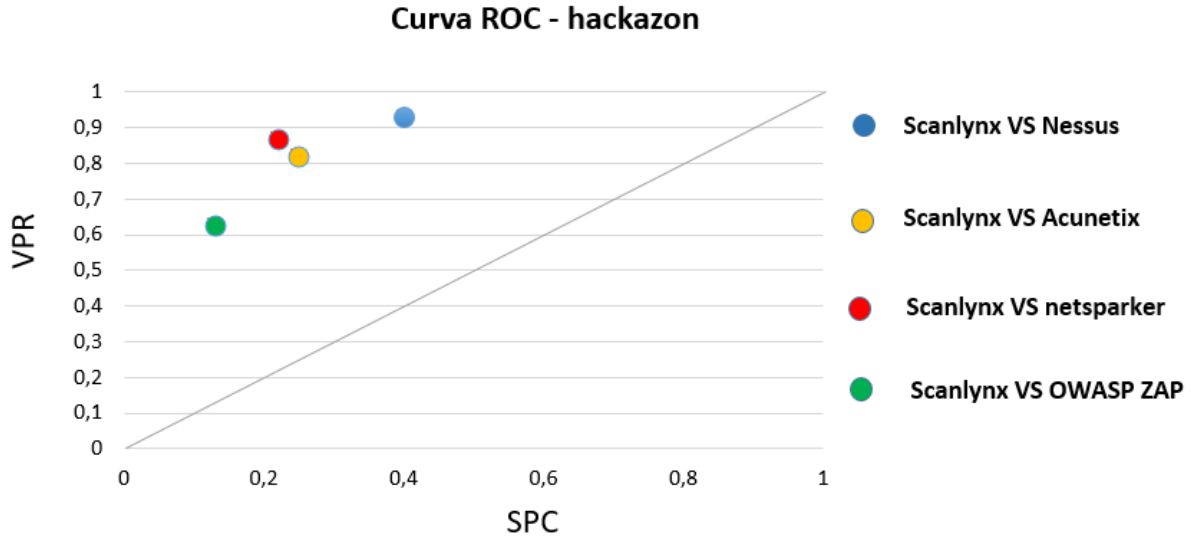


Figura 32. Curva ROC de ScanLynx Vs cada una de las herramientas para la aplicación web hackazon. Fuente: Propia.

Los valores de especificidad y sensibilidad para las herramientas Acunetix y Netsparker fueron muy similares, puesto que detectaron **14** y **13** vulnerabilidades en común (verdaderos positivos)respectivamente, tuvieron **2** verdaderos negativos (vulnerabilidades que no detectó ni el prototipo hardware y software y las herramientas seleccionadas), se obtuvieron **3** y **2** vulnerabilidades que detectaron las herramientas pero no logró identificar ScanLynx y el número de vulnerabilidades que encontró el prototipo, pero no descubrieron las herramientas fue de **6** y **7** (falsos positivos) respectivamente. Esto se debe al gran número de pruebas de concepto que poseen ambas herramientas que están estipuladas en la metodología OWASP. Nessus fue la herramienta que obtuvo el mínimo valor en toda la prueba para los falsos negativos **1**, generando el mejor resultado para la sensibilidad **0,93**, pero el valor más deficiente en razón de los verdaderos negativos **0,40,** debido a los 4 verdaderos negativos, ya que esta herramienta detecta la mayor parte de vulnerabilidades cuya severidad es **baja** y descuida hacer énfasis en otras

vulnerabilidades que pueden ocasionar un riesgo **alto** o **medio** para la aplicación web. El último análisis para OWASP ZAP arrojó el mejor resultado para la razón de verdaderos negativos **0,13** (especificidad), ya que solamente hubo **2** verdaderos negativos (vulnerabilidades que no detectó ScanLynx ni OWASP ZAP), pero se obtuvo el valor más deficiente para la sensibilidad **0,63** en toda la prueba por las 3 vulnerabilidades que identificó la herramienta, pero no logró detectar el prototipo.

La predicción en toda la prueba para la aplicación web fue buena, puesto que se encontraron muchas vulnerabilidades en común (verdaderos positivos) y pocos falsos negativos, esto permitió que los resultados para la sensibilidad y especificidad estuvieran por encima de la **línea de no-discriminación.**

4.4 Análisis para determinar la eficiencia al encontrar vulnerabilidades conocidas con otras herramientas

Para comparar que tan eficiente fue el prototipo al momento de encontrar/detectar las vulnerabilidades más conocidas según OWASP se analizaron los tiempos de ejecución de 4 herramientas dedicadas al pentesting sobre aplicaciones web siendo 3 comerciales (Nessus, Netsparker, Acunetix) y 1 open source (OWASP ZAP), el grupo de objetivos seleccionados para este análisis fueron 5 aplicaciones web desarrolladas por empresas dedicadas a la seguridad de la información con el fin de poder ser vulneradas. La configuración de las diferentes herramientas incluido el prototipo (con sus 6 dispositivos SBC) se hicieron de manera **Full Pentest**, en otras palabras, se cargan todas las pruebas implementadas de cada herramienta y se ejecutan sobre los diferentes objetivos. Es necesario aclarar, que los tiempos de ejecución también pueden verse afectados por el lenguaje de programación utilizado para desarrollar las herramientas, así como el ejecutable que generan (binario o script), por esa razón, dicha característica fue descartada.

En tabla 22, se muestra una perspectiva general en términos de segundos de cada herramienta con su respectiva aplicación web.

	TestSparker	**Hackazon**	**WebscanTest**	**DemoTestfire**	**Vulnweb**
Nessus	660	600	540	900	2520
Netsparker	480	1260	1666	960	1980
Acunetix	2819	470	4740	4560	2157
ZAP	1440	6260	4880	2469	2940
ScanLynx	1380	2580	2410	2760	1800

Tabla 22. Tiempo de ejecución requerido por cada herramienta en segundos. Fuente: Propia.

A simple vista es difícil distinguir cuál fue la herramienta más eficiente a la hora ejecutar el pentest a cada aplicación web, por esa razón, se utilizó una gráfica de curvas (ver figura 33) para identificar de manera directa los tiempos, siendo el eje X

las diferentes aplicaciones web y el eje Y el tiempo en una escala de 0-7000 segundos.

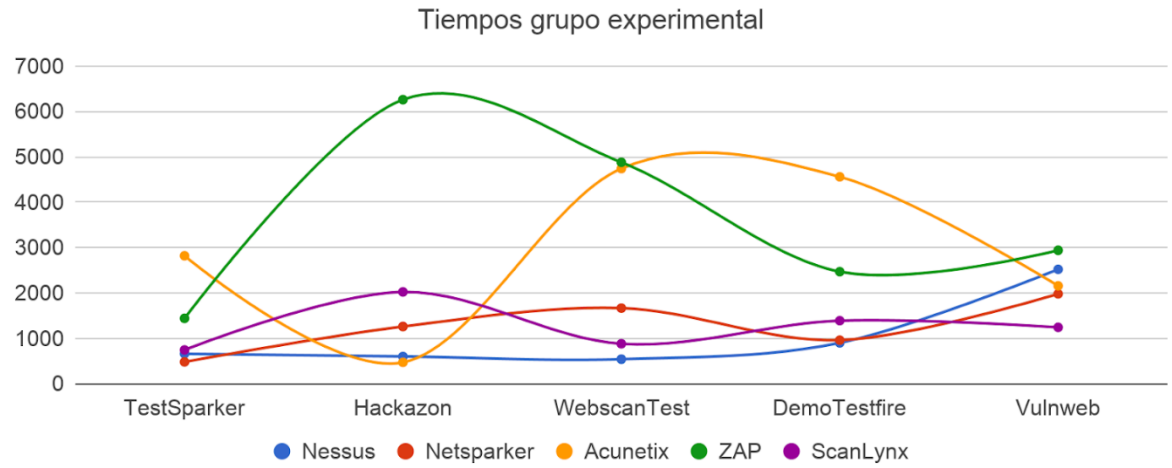


Figura 33. Gráfica de tiempo de ejecución requerido por cada herramienta. Fuente: Propia.

De esta manera rápidamente se pudo identificar que el prototipo (curva morada) se mantiene en un rango de tiempo entre los 740 a los 2050 segundos aproximadamente, tiempo relativamente bueno con respecto a Acunetix y OWASP ZAP, donde sus intervalos fueron demasiado variables de una aplicación web a otra. Sin embargo, no ocurre lo mismo con Nessus y Netsparker, ya que estos forman una curva con un intervalo de tiempo inferior a la del prototipo, esto es debido a las pruebas ejecutadas, ya que de entrada el tiempo mínimo requerido por el prototipo al momento de realizar un pentest es de 10 minutos (600 segundos) por la ejecución de la prueba con código OTG-SESS-007 que describe "*cierre de sesión por inactividad*", prueba que las demás herramientas no tienen implementada.

Comparativamente, la eficacia del prototipo con respecto a las otras herramientas está dentro del rango ideal demostrando que es superior a una herramienta open source que fue desarrollada por el proyecto OWASP y que está actualmente en constante mantenimiento, así como la herramienta comercial Acunetix.

4.5 Análisis para determinar la eficiencia del cluster al encontrar vulnerabilidades conocidas

Además de comparar la eficiencia en cuanto al tiempo del cluster usando los 6 dispositivos SBC contra las otras herramientas, se realizó otra medición de eficiencia individual del cluster construido usando 1, 2 4 y 6 dispositivos SBC para ejecutar el pentest a las aplicaciones web vulnerables del grupo experimental. Cabe resaltar que todos los análisis se realizaron con la misma cantidad de pruebas de concepto.

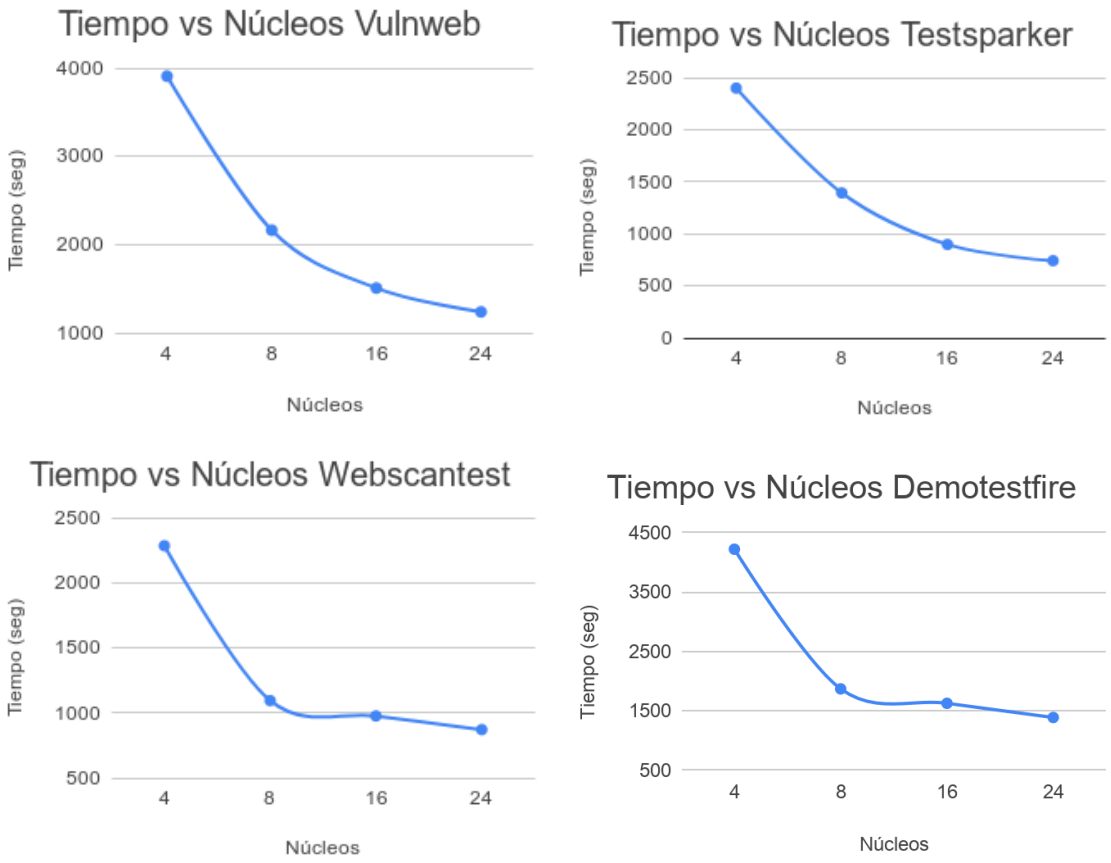


Figura 34. Gráficas de tiempos del cluster usando 1, 2, 4 y 6 dispositivos. Fuente: Propia.

Con respecto al conjunto de gráficas (figura 34) que se pueden apreciar arriba de este párrafo, se puede notar un cambio bastante significativo en el uso de 1 dispositivo SBC para la ejecución del pentest a 2 dispositivos, esto es debido al cambio de arquitectura, mientras con 1 se hace de manera secuencial (una prueba de concepto tras otra), con 2 se realiza de forma distribuida y paralela, optimizando más de 1000 segundos lo cual equivale a 16 minutos aproximadamente.

Al aumentar la cantidad de dispositivos SBC en el cluster se pudo observar que cada vez se optimiza el tiempo de ejecución, sin embargo, no hay una reducción tan marcada como el caso anterior. Esto se debe a la cantidad de pruebas de concepto implementadas, ya que el procesamiento se realiza de forma distribuida y como la prueba con código OTG-SESS-007 tiene una duración fija de 600 segundos (10 minutos) las demás pruebas que están alojadas en los otros dispositivos tienen ese intervalo de tiempo para completar sus respectivas tareas.

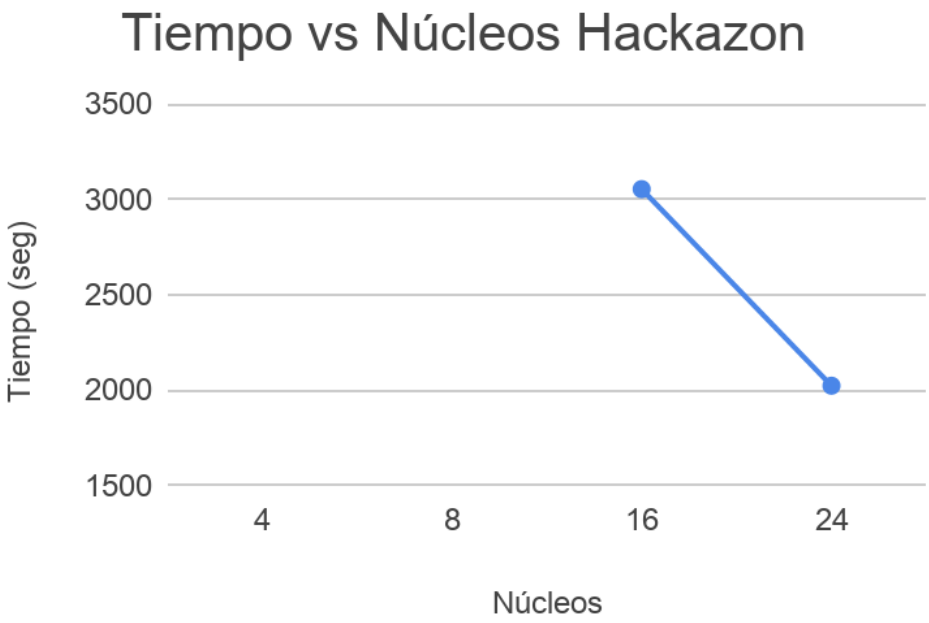


Figura 35. Tiempos del cluster sobre la aplicación web vulnerable Hackazon. Fuente: Propia.

Según la figura 35, se puede decir que es un caso especial donde se muestra las limitaciones por parte del hardware. La aplicación web *Hackazon* fue la única que reportó un número elevado de url's a probar, las cuales fueron exactamente 200, además se sabe que el dispositivo SBC Raspberry Pi 3 modelo B que es usado en este proyecto posee una memoria RAM de 1 GB, por lo tanto al realizar el pentest sobre dicha aplicación web con 1 y 2 dispositivos no se logró capturar un tiempo final, debido a un reinicio del sistema por la falta de espacio de almacenamiento en RAM del dispositivo maestro, este inconveniente ocurrió 3 veces de igual manera. Una vez adicionados 2 equipos más al cluster solucionó el problema anterior, los siguientes pentests con 4 y 6 dispositivos fueron satisfactorios, obteniéndose una diferencia de 1000 segundos aproximadamente.

Para finalizar se puede afirmar que al aumentar más dispositivos SBC junto a la implementación de más pruebas de concepto puede mejorarse el rendimiento (pero no de forma lineal) en cuanto al tiempo de ejecución con respecto a las otras herramientas que se encuentran en el mercado y que permiten la automatización del pentesting sobre aplicaciones web.

4.6 Conclusión del experimento controlado

Para medir la eficacia en la identificación de vulnerabilidades del prototipo hardware y software, fue posible gracias al análisis mediante la curva ROC, en el cual se observó que las diferentes herramientas utilizadas son bastante robustas en determinadas pruebas de concepto. OWASP ZAP se centra en área de **Inyección**, ya que siempre detectó Inyección SQL y XSS (Cross Site Scripting) en cuatro de los sitios web vulnerables, siendo estas vulnerabilidades catalogadas como críticas para cualquier aplicación web. Nessus se enfoca en la fase inicial del pentesting

(**Recolección de Información**), detectando tecnologías del servidor y la aplicación, sistema operativo, servicios de la aplicación, lenguajes de programación y puertos abiertos. Acunetix es muy fuerte en el área denominada **Incorrecta Configuración de Seguridad**, puesto que siempre identificó el mapa de rutas de la aplicación, métodos HTTP utilizados por el sitio web, meta archivos del servidor web, configuración de la plataforma e infraestructura, análisis de errores de código, vulnerabilidades de seguridad conocidas y pudo enumerar las interfaces de administrador, así como directorios ocultos o no indexados. Además, detectó en todo el grupo experimental la presencia de XSS e inyección SQL. Netsparker examina de manera minuciosa el área de **Exposición de Datos Sensibles**, debido a que inspecciona la presencia del encabezado HSTS por medio de las cabeceras de respuesta por parte del servidor web, analiza los puertos que utilizan el protocolo SSL y TLS y verifica si la información se transmite a través del protocolo HTTP en lugar de HTTPS o si se utilizan protocolos de cifrado débil.

Estas características mencionadas de cada herramienta permitirán al prototipo construido acercarse más a la coordenada (0,1) del espacio ROC, donde no se logren presentar en gran medida los falsos negativos ni falsos positivos, teniendo así una predicción ideal perfecta.

En general el comportamiento de la prueba de ScanLynx contra las herramientas seleccionadas fue buena, puesto que la predicción para la razón de verdaderos positivos y la razón para los verdaderos negativos estuvo por encima de la **línea de no-discriminación** y en dos sitios web vulnerables se alcanzó la predicción perfecta para la sensibilidad con respecto a OWASP ZAP y Nessus, esto significa que estas herramientas no lograron detectar vulnerabilidades adicionales a las que identificó el prototipo hardware y software construido. Con respecto a las verdaderos positivos se obtuvo un promedio de **12** vulnerabilidades en común para toda la prueba experimental, un resultado satisfactorio, si se considera que algunas de las herramientas no tienen implementado pruebas de concepto en la subcategoría de **Pruebas de Gestión de la Sesión** y herramientas como OWASP ZAP y Nessus no poseen demasiadas pruebas para toda la fase del pentesting, lo cual conlleva a clasificar algunas vulnerabilidades como falsos positivos (vulnerabilidades que detectó ScanLynx pero no identificaron las herramientas). Por su parte Netsparker y Acunetix tienen demasiadas pruebas enfocadas en la metodología OWASP, lo cual permitió clasificar la mayoría de pruebas implementadas por el prototipo como verdaderos positivos y lograr obtener resultados satisfactorios en toda la prueba experimental.

5 CAPÍTULO V. CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Análisis de los objetivos de investigación

Para la realización de este trabajo de investigación se definieron un conjunto de objetivos que fueron cumplidos de manera sistemática siguiendo las actividades anteriormente descritas. A continuación, se presenta un resumen en donde se muestra cómo se cumplió con cada objetivo propuesto.

- **OE1.** Caracterizar el rendimiento de los dispositivos SBC de bajo costo para seleccionar los más apropiados en el contexto de pentesting sobre aplicaciones web.

En la sección Estudio de Dispositivos SBC de Bajo Costo se realizó una caracterización de los dispositivos actuales, se estableció unos criterios de selección y evaluación, se analizaron los resultados para cada SBC y por último se escogió el dispositivo que fuera más acorde con las necesidades del proyecto de investigación.

- **OE2.** Diseñar e implementar un prototipo hardware y software, basado en la metodología OWASP que identifique los tipos de ataque más comunes siguiendo una metodología de pentesting.

En el capítulo Diseño e Implementación del Prototipo Hardware y Software se estableció la arquitectura de red del cluster, la tecnología para la comunicación entre los diferentes dispositivos SBC, la construcción y configuración del cluster con Raspberry Pi y las pruebas de concepto implementadas para identificar las vulnerabilidades más comunes.

- **OE3.** Aplicar la metodología de cálculo de riesgo según OWASP, con el fin de determinar la severidad de los riesgos identificados.

En la sección Uso de la Metodología del Cálculo de Riesgo Según OWASP se tuvo en cuenta la explotabilidad, prevalencia, detectabilidad e impacto técnico para cada categoría según el documento OWASP Top 10-2017, se describieron los factores para el agente de amenaza y se definieron cuatro preguntas para calcular el impacto de negocio y lograr obtener la severidad general del riesgo.

- **OE4.** Automatizar los reportes de mediciones de riesgos de seguridad en el pentesting sobre aplicaciones web, para generar el reporte técnico y ejecutivo.

En la sección Despliegue del Prototipo Hardware y Software e Interfaz de la aplicación web y generación de Reportes se especificaron las tecnologías utilizadas para transformar el contenido del reporte en formato PDF y lograr visualizar el reporte técnico y ejecutivo a través de la interfaz de la aplicación web ScanLynx.

- **OE5.** Evaluar el prototipo construido, para medir la eficiencia y eficacia en la identificación de vulnerabilidades mediante un experimento controlado.

En el capítulo Experimento Controlado se definieron una serie de actividades para medir la eficacia de las vulnerabilidades identificadas mediante un análisis de curvas ROC y se realizó un análisis para determinar la eficiencia del cluster comparando el tiempo de ejecución que tardó sobre cada aplicación web con respecto a otras herramientas.

- **OG.** Construir un prototipo hardware y software basado en dispositivos SBC de bajo costo, que automatice el pentesting sobre aplicaciones web, aplicando la metodología OWASP para identificar vulnerabilidades y generar reportes de mediciones de riesgos de seguridad automáticos.

Como resultado de cumplir cada uno de los cinco objetivos propuestos, se cumplió con el objetivo principal de manera satisfactoria. Se logró construir un cluster (prototipo hardware y software) junto con una aplicación web llamada ScanLynx que permitiera identificar vulnerabilidades conocidas en cinco sitios web vulnerables y lograr generar los reportes técnicos y ejecutivos con su respectiva severidad del riesgo.

5.2 Conclusiones

El prototipo hardware y software llamado ScanLynx, tuvo un buen comportamiento en la medición tanto de la eficiencia como la eficacia. Gracias al análisis de las curvas ROC se pudo realizar las respectivas comparaciones en términos de detección de vulnerabilidades de ScanLynx con las diferentes herramientas que actualmente son las más apetecidas en el mercado. Gracias a este método estadístico, se puede decir que hasta el momento, la herramienta propuesta a lo largo de este trabajo de investigación **es eficiente**, lo que significa que el hallazgo de vulnerabilidades se considera confiable. Por otro lado, con respecto a la medición de la eficiencia, el producto desarrollado con las 37 pruebas de concepto implementadas, se mantiene en un **rango óptimo** términos de segundos con respecto a las demás herramientas al momento de realzar el pentest a cada aplicación web vulnerable, en otras palabras, hasta el momento ScanLynx puede realizar un pentest de tipo completo a una aplicación web dentro del intervalo 600 a 2400 segundos como tiempo estimado.

En la revisión bibliográfica de los estudios relacionados con el pentesting sobre aplicaciones web, la mayoría están enfocados hacia la evaluación de herramientas comparando unas con otras para determinar cuál es más eficaz a la hora de realizar un pentest, gracias a esto se pudo concretar rápidamente el tipo de análisis estadístico utilizar para obtener de forma cualitativa y cuantitativa la eficacia de nuestro prototipo hardware y software con respecto a otras herramientas que tienen una alta demanda en el mercado hoy en día. Los estudios restantes se centraron en la creación de herramientas con el fin de minimizar la cantidad de falsos positivos que detectan las herramientas que automatizan el pentest, siendo este uno de los grandes problemas que padecen dichas herramientas, algunos mencionan las metodologías

existentes que mejor encajan en el contexto de las aplicaciones web como otro que crearon una que incluían características extras, esto dio un gran espectro para poder seleccionar una metodología acorde a las necesidades de este proyecto de investigación.

La ausencia de conocimiento en áreas de la matemática como la estadística analítica para comprender el uso de las fórmulas y poder graficar las curvas ROC que permitieron determinar la eficacia del cluster, dificultaron la comprensión de algunos artículos e investigaciones relacionadas. Para darle solución a ese problema se buscó soporte en el programa de matemáticas de la Universidad del Cauca para la comprensión de los mismos y esto a su vez generó una sinergia con tal departamento para el desarrollo del trabajo y futuras investigaciones.

Con respecto a los estudios a los estudios tomados en el capítulo II, se encontró que la mayoría de proyectos sobre dispositivos SBC de bajo costo que estuvieran enfocados al pentesting sobre aplicaciones web, eran proyectos sin fines investigativos, sin embargo, se pudieron identificar las tecnologías que permiten el procesamiento paralelo y distribuido, así como el conocimiento de otros dispositivos SBC que pudieran ser una alternativa para un trabajo futuro.

Por otro lado, cabe resaltar que los datos obtenidos pudieron ser alterados al llevar a cabo el experimento controlado, debido a que no se contemplaron algunas características elementales que conforman un cluster Beowulf como es el uso de una red dedicada, debido a que el ambiente de pruebas se sitúo en establecimientos de la Universidad del Cauca donde el ancho de banda es compartido por otros equipos externos. Por otro lado, se tuvieron limitaciones con respecto al hardware en el dispositivo SBC seleccionado, ya que este posee una interfaz de red que proporciona una velocidad máxima de 100 Mbps (FastEthernet), lo cual, pudieron verse afectado algunos datos debido que por naturaleza un pentest sobre una aplicación web requiere un número elevado de peticiones HTTP, así como también en la descarga y transferencia de archivos.

Gracias a la revisión bibliográfica se pudo catalogar el nivel donde se encuentran las investigaciones y hacia donde apuntan los trabajos futuros. Se pudo notar que al revisar cada artículo, libro y página web no se ha establecido de manera formal la selección de una metodología enfocada hacia el pentesting sobre aplicaciones, asimismo, como la selección de los dispositivos SBC de bajo costo para el mismo fin. Por tanto, la propuesta de métricas que fueron mencionadas en las secciones 2.3 y 2.7.1 generaron un gran aporte para formalizar a organización del cluster y las actividades que se deben seguir para la construcción de una herramienta que permita automatizar el pentesting sobre aplicaciones web con sus respectivos reportes.

5.3 Trabajos futuros

Como trabajos futuros para esta investigación se sugiere realizar más estudios que contemplen el uso de otros dispositivos SBC de bajo costo como la Orange PI Prime o la Banana PI BPI-M2 Ultra para las aplicaciones en el área de la seguridad de la información, extendiendo este experimento para que se incluyan otras pruebas de

rendimiento para así tener conocimiento de otras soluciones en la computación de alto rendimiento para la solución de tareas complejas, como es el pentesting sobre aplicaciones web. Asimismo, se sugiere experimentar con otras tecnologías middleware que permita el procesamiento en paralelo y distribuido como lo es Python Remote Object (Pyro) que permite la comunicación entre objetos remotos a través de una red, esto con el fin de poder observar las ventajas y desventajas que tienen con respecto al protocolo MPI que fue usado a lo largo de este proyecto.

Como otro trabajo futuro, se propone implementar más pruebas de concepto sobre el cluster que permita alcanzar las 91 que propone la guía OWASP Testing Guide v4, para así poder realizar más experimentos y observar con mayor detalle las diferencias, ventajas, desventajas y sobre todo el análisis de la eficacia con respecto a las demás herramientas que actualmente lideran el mercado de la automatización del pentesting sobre las aplicaciones web.

BIBLIOGRAFÍA

[1] E. J. Kretowicz, “TEAM :HAKIN9,” *IT Secur. Mag.*, vol. 11, pp. 37–49, 2012.

[2] M. Meucci and A. Muller, “Testing Guide 4.0,” no. Cc, 2014.

[3] E. Fong, R. Gaucher, V. Okun, P. E. Black, and E. Dalci, “Building a test suite for web application scanners,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 1–8, 2008.

[4] Acunetix, “Web Application Vulnerability Report,” pp. 1–27, 2016.

[5] OWASP, “OWASP Top 10 - The Ten Most Critical Web Application Security Risks,” *Owasp*, p. 22, 2017.

[6] Kaspersky Lab, “33 ataques por segundo: Kaspersky Lab registra un aumento del 59% en ataques de malware en América Latina | Kaspersky Lab LATAM,” 2017. [Online]. Available: https://latam.kaspersky.com/about/press-releases/2017_33-attacks-per-second-increase-in-malware-attacks-in-latin-america. [Accessed: 01-Nov-2017].

[7] Juliana Peña, “Actualización: Las contraseñas del eCenso no están precisamente guardadas en texto plano, pero sigue siendo muy feo el asunto,” 2018. [Online]. Available: http://julip.co/2018/01/actualizacion-contrasenas-censo/. [Accessed: 17-Feb-2018].

[8] H. S. H. Shi, B. C. B. Chen, and L. Y. L. Yu, “Analysis of Web Security Comprehensive Evaluation Tools,” *Networks Secur. Wirel. Commun. Trust. Comput. (NSWCTC), 2010 Second Int. Conf.*, vol. 1, pp. 285–289, 2010.

[9] T. Vieira and C. Serrao, “Web security in the finance sector,” *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, no. July, pp. 255–259, 2017.

[10] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, “State of the art: Automated black-box web application vulnerability testing,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 332–345, 2010.

[11] F. J. Pino, M. Piattini, and G. H. Travassos, “Managing and developing distributed research projects in software engineering by means of action-research,” *Rev. Fac. Ing.*, no. 68, pp. 61–74, 2013.

[12] C. wohlin, P. Runeson, M. Host, M. Ohlson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, 1st ed. Massachusetts, USA: Academic Publishers, 2000.
[13] D. Dalalana Bertoglio and A. F. Zorzo, “Overview and open issues on penetration test,” *J. Brazilian Comput. Soc.*, vol. 23, no. 1, pp. 1–16, 2017.
[14] A. Sudhodanan, R. Carbone, L. Compagna, N. Dolgin, A. Armando, and U. Morelli, “Large-Scale Analysis & Detection of Authentication Cross-Site Request Forgeries,” *Proc. - 2nd IEEE Eur. Symp. Secur. Privacy, EuroS P 2017*, no. i, pp. 350–365, 2017.
[15] P. Zech, M. Felderer, and R. Breu, “Knowledge-based security testing of web applications by logic programming,” *Int. J. Softw. Tools Technol. Transf.*, pp. 1–26, 2017.
[16] A. O. W. Halfond, S. Choudhary, “Improving penetration testing through static and dynamic analysis,” *Softw. Test. Verif. Reliab.*, vol. 24, no. 8, pp. 591–592, 2014.
[17] J. Dawson and J. Todd McDonald, “Improving Penetration Testing Methodologies for Security-Based Risk Assessment,” *Proc. - 2016 Cybersecurity Symp. CYBERSEC 2016*, pp. 51–58, 2017.
[18] N. M. Vithanage and N. Jeyamohan, “WebGuardia - An integrated penetration testing system to detect web application vulnerabilities,” *Proc. 2016 IEEE Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2016*, pp. 221–227, 2016.
[19] F. R. Muñoz, E. A. A. Vega, and L. J. G. Villalba, “Analyzing the traffic of penetration testing tools with an IDS,” *J. Supercomput.*, pp. 1–16, 2016.
[20] S. Shah and B. M. Mehtre, “An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0,” *Proc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014*, no. 978, pp. 707–712, 2015.
[21] Z. DURIC, “WAPTT - Web Application Penetration Testing Tool,” *Adv. Electr. Comput. Eng.*, vol. 14, no. 1, pp. 93–102, 2014.
[22] M. Mirjalili, A. Nowroozi, and M. Alidoosti, “A survey on web penetration test,” *ACSIJ Adv. Comput. Sci.*, vol. 3, no. 6, pp. 107–121, 2014.
[23] K. P. Haubris and J. J. Pauli, “Improving the Efficiency and Effectiveness of Penetration Test Automation,” *2013 10th Int. Conf. Inf. Technol. New Gener.*, pp. 387–391, 2013.
[24] A. Austin, C. Holmgreen, and L. Williams, “A comparison of the efficiency and effectiveness of vulnerability discovery techniques,” *Inf. Softw. Technol.*, vol. 55, no. 7, pp. 1279–1288, 2013.
[25] D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu, “Automated security test generation with formal threat models,” *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 4, pp. 526–540, 2012.
[26] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, “An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection,” *IEEE Int. Conf. Soc. Comput.*, pp. 1095–1101, 2011.
[27] M. Prandini and M. Ramilli, “Towards a practical and effective security testing methodology,” *Proc. - IEEE Symp. Comput. Commun.*, pp. 320–325, 2010.
[28] S. J. Matthews, R. W. Blaine, and A. F. Brantly, “Evaluating single board computer clusters for cyber operations,” *2016 IEEE Int. Conf. Cyber Conflict, CyCon U.S. 2016*, vol. 2, no. OCTOBER, pp. 1–8, 2017.
[29] Y. Hu *et al.*, “Employing miniaturized computers for distributed vulnerability

assessment," *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, pp. 57–61, 2017.
[30] A. Akkiraju, D. Gabay, H. B. Yesilyurt, H. Aksu, and S. Uluagac, "Cybergrenade: Automated Exploitation of Local Network Machines via Single Board Computers," *2017 IEEE 14th Int. Conf. Mob. Ad Hoc Sens. Syst.*, pp. 580–584, 2017.
[31] C. Baun, "Mobile clusters of single board computers: an option for providing resources to student projects and researchers," *Springerplus*, vol. 5, no. 1, 2016.
[32] A. Shipurkar, "Building Supercomputer With Raspberry Pi," *Int. J. Adv. Found. Res. Sci. Eng.*, vol. 1, pp. 1–9, 2015.
[33] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'Brien, "Iridis-pi: A low-cost, compact demonstration cluster," *Cluster Comput.*, vol. 17, no. 2, pp. 349–358, 2014.
[34] P. Abrahamsson *et al.*, "Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment," *Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, vol. 2, pp. 170–175, 2013.
[35] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," *IEEE Int. Conf. Electro Inf. Technol.*, pp. 362–366, 2014.
[36] M. I. P. Salas and E. Martins, "Security testing methodology for vulnerabilities detection of XSS in web services and WS-security," *Electron. Notes Theor. Comput. Sci.*, vol. 302, pp. 133–154, 2014.
[37] M. I. P. Salas, P. L. De Geus, and E. Martins, "Security Testing Methodology for Evaluation of Web Services Robustness - Case: XML Injection," *2015 IEEE World Congr. Serv.*, pp. 303–310, 2015.
[38] M. Meissner, "Linux Kernel < 4.10.13 - 'keyctl_set_reqkey_keyring' Local Denial of Service," 2017. [Online]. Available: https://www.exploit-db.com/exploits/42136/. [Accessed: 27-Feb-2018].
[39] M. Ascencio Mendoza and P. J. Moreno Patiño, "Desarrollo de una Propuesta Metodologica para Determinar la Seguridad de una Aplicación web," p. 82, 2011.
[40] S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review," *Int. J. Embed. Syst. Appl.*, vol. 2, no. 2, pp. 29–50, 2012.
[41] B. Garn, I. Kapsalis, D. E. Simos, and S. Winkler, "On the applicability of combinatorial testing to web application security testing: a case study," *Proc. 2014 Work. Join. Acad. Ind. Contrib. to Test Autom. Model. Test. - JAMAICA 2014*, pp. 16–21, 2014.
[42] N. A. Almubairik and G. Wills, "Automated penetration testing based on a threat model," *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, pp. 413–414, 2017.
[43] B. Rathore *et al.*, "Penetration Testing Framework (PTF)," *Inf. Syst. Secur. Assess. Framew.*, vol. 0.2.1B, p. 845, 2006.
[44] K. Scarfone and A. Orebaugh, "Technical Guide to Information Security Testing and Assessment Recommendations of the National Institute of Standards and Technology," *Nist Spec. Publ.*, vol. 800–115, pp. 1–80, 2008.
[45] P. Herzog, "OSSTMM 3 Lite Introduction and Sample to the Open Source Security Testing Methodology Manual," *Inst. Secur. Open Methodol. ISECOM*, vol. 3, 2008.
[46] B. Liu, L. Shi, Z. Cai, and M. Li, "Software vulnerability discovery techniques: A

survey," *Proc. - 2012 4th Int. Conf. Multimed. Secur. MINES 2012*, pp. 152–156, 2012.
[47] F. Alisherov and F. Sattarova, "Methodology for Penetration Testing," *Distrib. Comput.*, vol. 2, no. 2, pp. 43–50, 2009.
[48] Owasp, "OWASP Risk Rating Methodology," *Owasp*, pp. 1–5, 2013.
[49] OWASP, "OWASP Nettacker - OWASP," 2018. [Online]. Available: https://www.owasp.org/index.php/OWASP_Nettacker. [Accessed: 26-Feb-2018].
[50] Alharbi Mansour, "Writing a Penetration Testing Report," 2010.
[51] J. Muniz and A. Lakhani, *Penetration Testing with Raspberry Pi*. Packt publishing, 2015.
[52] B. Bullock, "How to Build Your Own Penetration Testing Drop Box - Black Hills Information Security," 2016. [Online]. Available: https://www.blackhillsinfosec.com/how-to-build-your-own-penetration-testing-drop-box/. [Accessed: 12-Sep-2017].
[53] L. Garc, P. Etsi, C. Etsi, R. M. Garc, and B. Etsi, "Propuesta de docencia en computación de alto rendimiento utilizando equipos de bajo coste," pp. 1–11, 2013.
[54] E. Brown, "Catalog of hacker-friendly SBCs," 2018. [Online]. Available: http://linuxgizmos.com/january-2018-catalog-of-hacker-friendly-sbcs/. [Accessed: 12-Mar-2018].
[55] P. Wurst and C. Dupré, "Making A Beowulf Cluster."
[56] L. O. Salvador, "Peña Cabarga : Construcción de un clúster de bajo consumo empleando tecnología SBC (Building a low consumption cluster using SBC," 2016.
[57] L. Dalcín, M. Storti, and R. Paz, "Desarrollo De Aplicaciones Paralelas En Python," vol. XXIII, no. November, pp. 3153–3163, 2004.
[58] I. de Jong, "Pyro - Python Remote Objects - 4.60 - Pyro 4.60 documentation," 2017. [Online]. Available: https://pyro4.readthedocs.io/en/stable/. [Accessed: 20-Mar-2018].
[59] T. A. S. Foundation, "Welcome to Apache™ Hadoop®!," *The Apache Software Foundation.*, 2014. [Online]. Available: http://hadoop.apache.org/. [Accessed: 21-Mar-2018].
[60] A. B. Al Geist, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, "PVM: Parallel virtual machine," *MIT Press Cambridge Massachusetts*, vol. 79, p. 80, 1994.
[61] K. Doucet and J. Zhang, "Learning Cluster Computing by Creating a Raspberry Pi Cluster," *Proc. SouthEast Conf. - ACM SE '17*, pp. 191–194, 2017.
[62] A. Mappuji, N. Effendy, M. Mustaghfirin, F. Sondok, R. P. Yuniar, and S. P. Pangesti, "Study of Raspberry Pi 2 quad-core Cortex-A7 CPU cluster as a mini supercomputer," *Proc. 2016 8th Int. Conf. Inf. Technol. Electr. Eng. Empower. Technol. Better Futur. ICITEE 2016*, pp. 7–10, 2017.
[63] "How to Make a Raspberry Pi SuperComputer!: 9 Steps (with Pictures)." [Online]. Available: https://www.instructables.com/id/How-to-Make-a-Raspberry-Pi-SuperComputer/. [Accessed: 29-Jul-2018].
[64] "Learn To Build Your Own Supercomputer With Raspberry Pi 3 Cluster » TechWorm." [Online]. Available: https://www.techworm.net/2018/03/learn-build-supercomputer-raspberry-pi-3-cluster.html. [Accessed: 29-Jul-2018].
[65] "Nmap: the Network Mapper - Free Security Scanner." [Online]. Available:

https://nmap.org/. [Accessed: 29-Jul-2018].
[66] “WhatWeb.” [Online]. Available: https://www.morningstarsecurity.com/research/whatweb. [Accessed: 29-Jul-2018].
[67] “GitHub - EnableSecurity/wafw00f: WAFW00F allows one to identify and fingerprint Web Application Firewall (WAF) products protecting a website.” [Online]. Available: https://github.com/EnableSecurity/wafw00f. [Accessed: 29-Jul-2018].
[68] “GitHub - x90skysn3k/brutespray: Brute-Forcing from Nmap output - Automatically attempts default creds on found services.” [Online]. Available: https://github.com/x90skysn3k/brutespray. [Accessed: 29-Jul-2018].
[69] “GitHub - laramies/theHarvester: E-mails, subdomains and names Harvester - OSINT.” [Online]. Available: https://github.com/laramies/theHarvester. [Accessed: 29-Jul-2018].
[70] “GitHub - rek7/python-dirbuster: a dir buster replica written in python that relies on request codes.” [Online]. Available: https://github.com/rek7/python-dirbuster. [Accessed: 29-Jul-2018].
[71] “Clusters and parallel programming with MPI and Raspberry Pi – Meccanismo Complesso.” [Online]. Available: https://www.meccanismocomplesso.org/en/cluster-e-programmazione-in-parallelo-con-mpi-e-raspberry-pi/. [Accessed: 29-Jul-2018].
[72] H. Holm, T. Sommestad, J. Almroth, and M. Persson, “A quantitative evaluation of vulnerability scanning,” *Inf. Manag. Comput. Secur.*, vol. 19, no. 4, pp. 231–247, 2011.

ANEXOS

Anexo A: Tabla con la caracterización de los 40 dispositivos SBC y su calificación cuantitativa

Dispositivo	Característica		Puntaje		Total
			Métrica	Métrica x Porcentaje	Cuantitativo
Banana Pi BPI-M2 Ultra	Precio (COP):	$138.000	1	0,4	0,885
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
Orange Pi Prime	Precio (COP):	$95.865	1	0,4	0,8125
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
NanoPi M2A	Precio (COP):	$137.142	1	0,4	0,81
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,4	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
RaspBerry PI 3	Precio (COP):	$128.706	1	0,4	0,75

Model B	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
Firefly-ROC-RK3328-CC	Precio (COP):	$114.285	1	0,4	0,7125
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Le Potato	Precio (COP):	$97.222	1	0,4	0,675
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,6	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	No	0	0	
pcDuino8 Uno	Precio (COP):	$142.857	0,5	0,2	0,625
	Núcleos:	8	1	0,09	
	Frecuencia (GHZ):	2	1	0,09	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Tinker Board	Precio (COP):	$142.857	0,5	0,2	0,6225

	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,8	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	n	0,25	0,0125	
Rockchip RK3399 Sapphire	Precio (COP):	$214.285	0,5	0,2	0,61
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,43	0,5	0,045	
	RAM (Gb):	4	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Wandboard	Precio (COP):	$214.285	0,5	0,2	0,6
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	n	0,25	0,0125	
Pine A64 / A64-LTS	Precio (COP):	$140.000	0,5	0,2	0,56
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Android	0	0	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Khadas Vim	Precio (COP):	$157.142	0,5	0,2	0,54

	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	n	0,25	0,0125	
Raspberry Pi Zero W	Precio (COP):	$98.500	1	0,4	0,5225
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	n	0,25	0,0125	
Wand-Pi-8M	Precio (COP):	$257.142	0,25	0,1	0,5225
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,3	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	ac	0,25	0,0125	
HummingBoard -Base	Precio (COP):	$245.714	0,25	0,1	0,51
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
PocketBeagle	Precio (COP):	$85.714	1	0,4	0,51

	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	No	0	0	
ODROID-U2	Precio (COP):	$256.491	0,25	0,1	0,5
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,7	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	No	0	0	
VoltaStream Zero	Precio (COP):	$111.428	1	0,4	0,4875
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	0,996	0	0	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	No	0	0	
Cubieboard4	Precio (COP):	$356.238	0	0	0,4675
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	2	1	0,09	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	b/g/a	0,25	0,0125	
UP board	Precio (COP):	$254.285	0,25	0,1	0,435

	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,44	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
UP Core	Precio (COP):	$257.142	0,25	0,1	0,435
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,44	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Udoo Neo	Precio (COP):	$171.428	0,5	0,2	0,4325
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
Z-turn Lite	Precio (COP):	$197.142	0,5	0,2	0,43
	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	0,667	0	0	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
DragonBoard	Precio (COP):	$218.743	0,5	0,2	0,4275

410c	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	b/g/n	0,25	0,0125	
LeMaker Guitar	Precio (COP):	$168.571	0,5	0,2	0,4275
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,6	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	Fast	0	0	
	Inalámbrica (802.11):	b/g/n	0,25	0,0125	
Seeeduino Cloud	Precio (COP):	$200.000	0,5	0,2	0,3725
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	0,4	0	0	
	RAM (Gb):	0,64	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
Rico Board	Precio (COP):	$282.857	0,25	0,1	0,3675
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Parallella	Precio (COP):	$282.857	0,25	0,1	0,3675

	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	0,667	0	0	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
Z-turn Board	Precio (COP):	$285.714	0,25	0,1	0,3675
	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	0,667	0	0	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
DE0-Nano-SoC Dev Kit	Precio (COP):	$257.151	0,25	0,1	0,3675
	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	0,952	0	0	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
HiKey	Precio (COP):	$342.857	0	0	0,3475
	Núcleos:	8	1	0,09	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	g/n	0,25	0,0125	
UP Squared	Precio (COP):	$414.285	0	0	0,3425

	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	1,1	0	0	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	GbE	1	0,12	
	Inalámbrica (802.11):	No	0	0	
ChipPro Dev Kit	Precio (COP):	$142.857	0,5	0,2	0,335
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	0,256	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
PICO-PI-IMX6UL	Precio (COP):	$197.142	0,5	0,2	0,3225
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	0,528	0	0	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Android Things	0	0	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
BeagleBone Green	Precio (COP):	$282.857	0,25	0,1	0,32
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	No	0	0	
CloudBit	Precio (COP):	$166.666	0,5	0,2	0,3

	Núcleos:	1	0	0	
	Frecuencia (GHZ):	0,454	0	0	
	RAM (Gb):	0,64	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	b/g	0,25	0,0125	
MediaTek X20 Dev Board	Precio (COP):	$551.428	0	0	0,265
	Núcleos:	4	0,5	0,045	
	Frecuencia (GHZ):	1,4	0,5	0,045	
	RAM (Gb):	2	1	0,15	
	Sistemas operativos:	Android	0	0	
	LAN:	No	0	0	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
PandaBoard	Precio (COP):	$512.983	0	0	0,1925
	Núcleos:	1	0	0	
	Frecuencia (GHZ):	1,2	0,5	0,045	
	RAM (Gb):	1	0,5	0,075	
	Sistemas operativos:	Android	0	0	
	LAN:	Fast	0,5	0,06	
	Inalámbrica (802.11):	b/g/n	0,25	0,0125	
Chameleon96	Precio (COP):	$368.571	0	0	0,1575
	Núcleos:	2	0,25	0,0225	
	Frecuencia (GHZ):	1	0,25	0,0225	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Linux	0,5	0,05	
	LAN:	No	0	0	
	Inalámbrica (802.11):	b/g/n	0,5	0,025	
USB Armory	Precio (COP):	$400.000	0	0	0,1375

	Núcleos:	1	0	0	
	Frecuencia (GHZ):	0,8	0	0	
	RAM (Gb):	0,512	0,25	0,0375	
	Sistemas operativos:	Kali Linux ARM	1	0,1	
	LAN:	No	0	0	
	Inalámbrica (802.11):	No	0	0	

Anexo B: Solucionar modo emergency en kali Linux

Se despliegan las particiones de la tarjeta SD mediante el comando *fdisk*
sudo fdisk -l

sudo fsck /dev/mmcblk1p1 -y
sudo fsck /dev/mmcblk1p2 -y

Deshabilitar interface GUI
systemctl set-default multi-user.target

Habilitar GUI
systemctl set-default graphical.target

Configuración de red estática

```
auto lo
iface lo inet loopback

# Statica
auto eth0
iface eth0 inet static
address 192.168.120.31/24
netmask 255.255.255.0
gateway 192.168.120.1
```

Figura 36. Configuración de red. Fuente: Propia

Anexo C: Instalación de MPICH en la RaspBerry PI

Para la instalación de MPICH en su versión 3.1 se deben seguir los siguientes pasos:

sudo apt-get update
mkdir mpich2
cd ~/mpich2

wget http://www.mpich.org/static/downloads/3.1/mpich-3.1.tar.gz
tar xfz mpich-3.1.tar.gz
sudo mkdir /home/rpimpi/
sudo mkdir /home/rpimpi/mpi-install
mkdir /home/pi/mpi-build
cd /home/pi/mpi-build
sudo apt-get install gfortran
sudo /home/pi/mpich2/mpich-3.1/configure -prefix=/home/rpimpi/mpi-install
sudo make
make install
nano .bashrc
#Agregar la siguiente línea al final del archivo
PATH=$PATH:/home/rpimpi/mpi-install/bin
sudo reboot
mpiexec -n 1 hostname

Una vez realizado todos los pasos anteriores, se debe visualizar el nombre del host de la raspberry pi, como se indica en la figura 37 para verificar que los procesos de instalación fue satisfactorio.

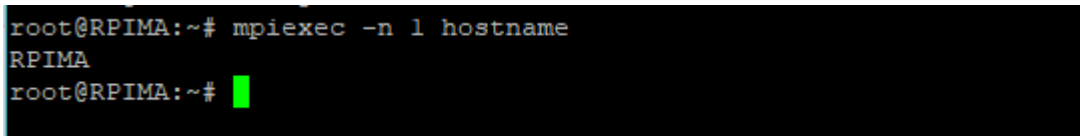


Figura 37. Prueba exitosa de MPICH. Fuente: Propia.

Instalación de MPI4PY

descargar mpi4py
wget https://bitbucket.org/mpi4py/mpi4py/downloads/mpi4py-2.0.0.tar.gz

#descomprimir el archivo
sudo tar -zxf mpi4py-2.0.0.tar.gz

ir al directorio
cd mpi4py-2.0.0

instalar el paquete python-dev
sudo apt install python-dev

Ejecutar la configuración
python setup.py build
sudo python setup.py install

Configurar la ruta para python
export PYTHONPATH=/home/pi/mpi4py-2.0.0

Probar el funcionamiento de MPI en el dispositivo
mpiexec -n 5 python demo/helloworld.py

Una vez ejecutado el último comando aparece un error (ver figura 38) con el hostname del dispositivo. La solución es cambiar el nombre del host a *RPIMA* en el archivo */etc/hosts/*

```
root@RPIMA:/home/pi/mpi4py-2.0.0# mpiexec -n 5 python demo/helloworld.py
Fatal error in PMPI_Init_thread: Other MPI error, error stack:
MPIR_Init_thread(467)..............:
MPID_Init(177).....................: channel initialization failed
MPIDI_CH3_Init(70).................:
MPID_nem_init(319).................:
MPID_nem_tcp_init(171).............:
MPID_nem_tcp_get_business_card(418):
MPID_nem_tcp_init(377).............: gethostbyname failed, RPIMA (errno 2)
Fatal error in PMPI_Init_thread: Other MPI error, error stack:
MPIR_Init_thread(467)..............:
MPID_Init(177).....................: channel initialization failed
MPIDI_CH3_Init(70).................:
MPID_nem_init(319).................:
MPID_nem_tcp_init(171).............:
MPID_nem_tcp_get_business_card(418):
MPID_nem_tcp_init(377).............: gethostbyname failed, RPIMA (errno 2)
```

Figura 38. Error del hostname en el archivo /etc/host. Fuente: Propia.

Después de solucionar el error, se puede apreciar en la figura 39 el correcto funcionamiento de MPI

```
root@RPIMA:/home/pi/mpi4py-2.0.0# mpiexec -n 5 python demo/helloworld.py
Hello, World! I am process 0 of 5 on RPIMA.
Hello, World! I am process 1 of 5 on RPIMA.
Hello, World! I am process 2 of 5 on RPIMA.
Hello, World! I am process 3 of 5 on RPIMA.
Hello, World! I am process 4 of 5 on RPIMA.
```

Figura 39. Prueba exitosa de MPI4PY. Fuente: Propia

Anexo D: Configurar SSH en cada dispositivo

RPIMA (192.168.120.31)
ssh-keygen
cd ~
cd .ssh
cp id_rsa.pub RPIMA

RPIESC01 (192.168.120.32)

ssh root@192.168.120.32
ssh-keygen
cd .ssh
cp id_rsa.pub RPIESC01
scp 192.168.120.31:/root/.ssh/RPIMA .
cat RPIMA >> authorized_keys
exit

RPIESC02 (192.168.120.33)

ssh root@192.168.120.33
ssh-keygen
cd .ssh
cp id_rsa.pub RPIESC02
scp 192.168.120.31:/root/.ssh/RPIMA .
cat RPIMA >> authorized_keys
exit

RPIESC03 (192.168.120.34)

ssh root@192.168.120.34
ssh-keygen
cd .ssh
cp id_rsa.pub RPIESC03
scp 192.168.120.31:/root/.ssh/RPIMA .
cat RPIMA >> authorized_keys
exit

RPIESC04 (192.168.120.35)

ssh root@192.168.120.35
ssh-keygen
cd .ssh
cp id_rsa.pub RPIESC04
scp 192.168.120.31:/root/.ssh/RPIMA .
cat RPIMA >> authorized_keys
exit

Ahora, el enlace entre el primer dispositivo maestro RPIMA y cada dispositivo ha sido configurado, sin embargo, aún se necesita configurar las llaves RSA de los dispositivos esclavos. Por lo tanto, se deben ejecutar los siguientes comandos desde cada dispositivo individual:

Ejecutar esto desde RPIMA mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.32:/root/.ssh/RPIESC01 .
cat RPIESC01 >> authorized_keys
scp 192.168.120.33:/root/.ssh/RPIESC02 .
cat RPIESC02 >> authorized_keys
scp 192.168.120.34:/root/.ssh/RPIESC03 .
cat RPIESC03 >> authorized_keys
scp 192.168.120.35:/root/.ssh/RPIESC04 .
cat RPIESC04 >> authorized_keys

```
# Ejecutar esto desde RPIESC01 mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.33:/root/.ssh/RPIESC02 .
cat RPIESC02 >> authorized_keys
scp 192.168.120.34:/root/.ssh/RPIESC03 .
cat RPIESC03 >> authorized_keys
scp 192.168.120.35:/root/.ssh/RPIESC04 .
cat RPIESC04 >> authorized_keys



# Ejecutar esto desde RPIESC02 mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.32:/root/.ssh/RPIESC01 .
cat RPIESC01 >> authorized_keys
scp 192.168.120.34:/root/.ssh/RPIESC03 .
cat RPIESC03 >> authorized_keys
scp 192.168.120.35:/root/.ssh/RPIESC04 .
cat RPIESC04 >> authorized_keys

# Ejecutar esto desde RPIESC03 mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.32:/root/.ssh/RPIESC01 .
cat RPIESC01 >> authorized_keys
scp 192.168.120.33:/root/.ssh/RPIESC02 .
cat RPIESC02 >> authorized_keysn
scp 192.168.120.35:/root/.ssh/RPIESC04 .
cat RPIESC04 >> authorized_keys

# Ejecutar esto desde RPIESC04 mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.32:/root/.ssh/RPIESC01 .
cat RPIESC01 >> authorized_keys
scp 192.168.120.33:/root/.ssh/RPIESC02 .
cat RPIESC02 >> authorized_keys
scp 192.168.120.34:/root/.ssh/RPIESC03 .
cat RPIESC03 >> authorized_keys

# Ejecutar esto desde RPIESC05 mediante PuTTY
cd ~
cd .ssh
scp 192.168.120.32:/root/.ssh/RPIESC01 .
cat RPIESC01 >> authorized_keys
scp 192.168.120.33:/root/.ssh/RPIESC02 .
cat RPIESC02 >> authorized_keys
scp 192.168.120.34:/root/.ssh/RPIESC03 .
```

cat RPIESC03 >> authorized_keys
scp 192.168.120.35:/root/.ssh/RPIESC04 .
cat RPIESC04 >> authorized_keys

Para comprobar si el cluster funciona como se espera, si todo está configurado correctamente, se debe desplegar una imagen como se indica en la figura 40

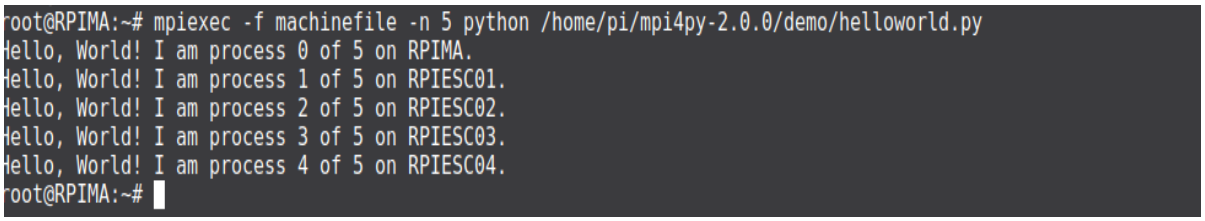
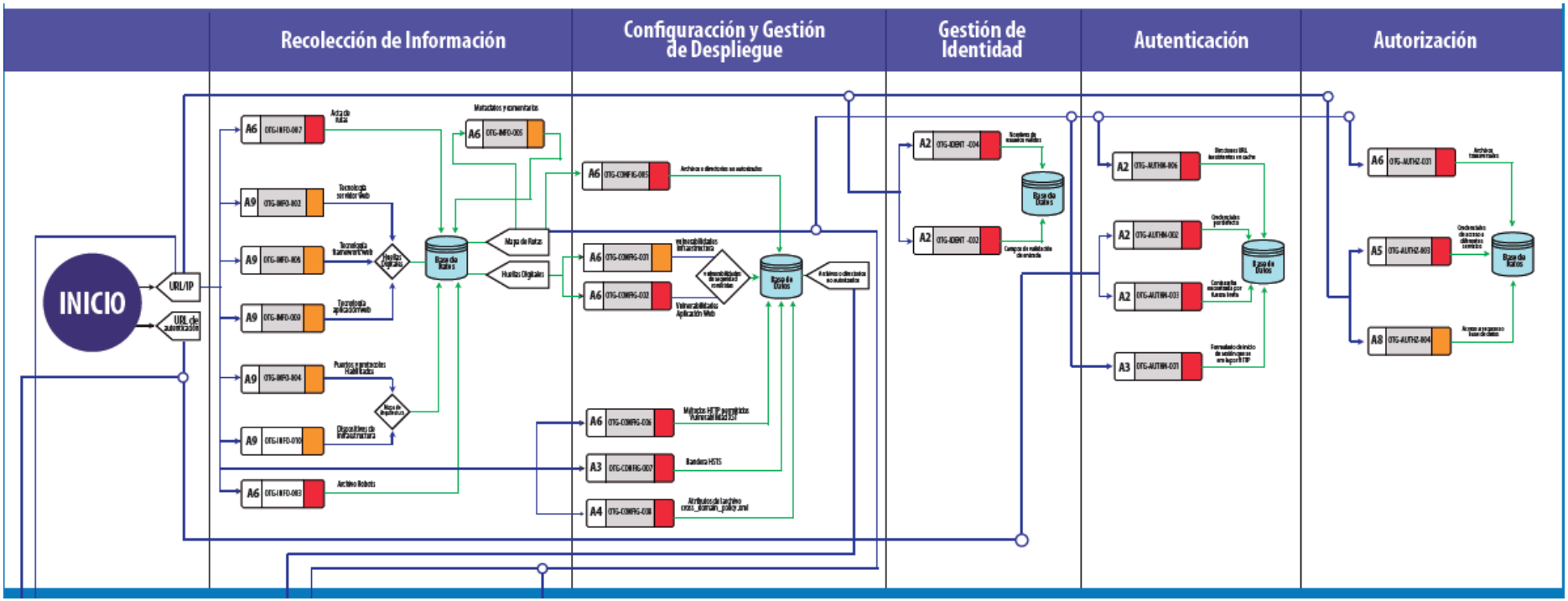
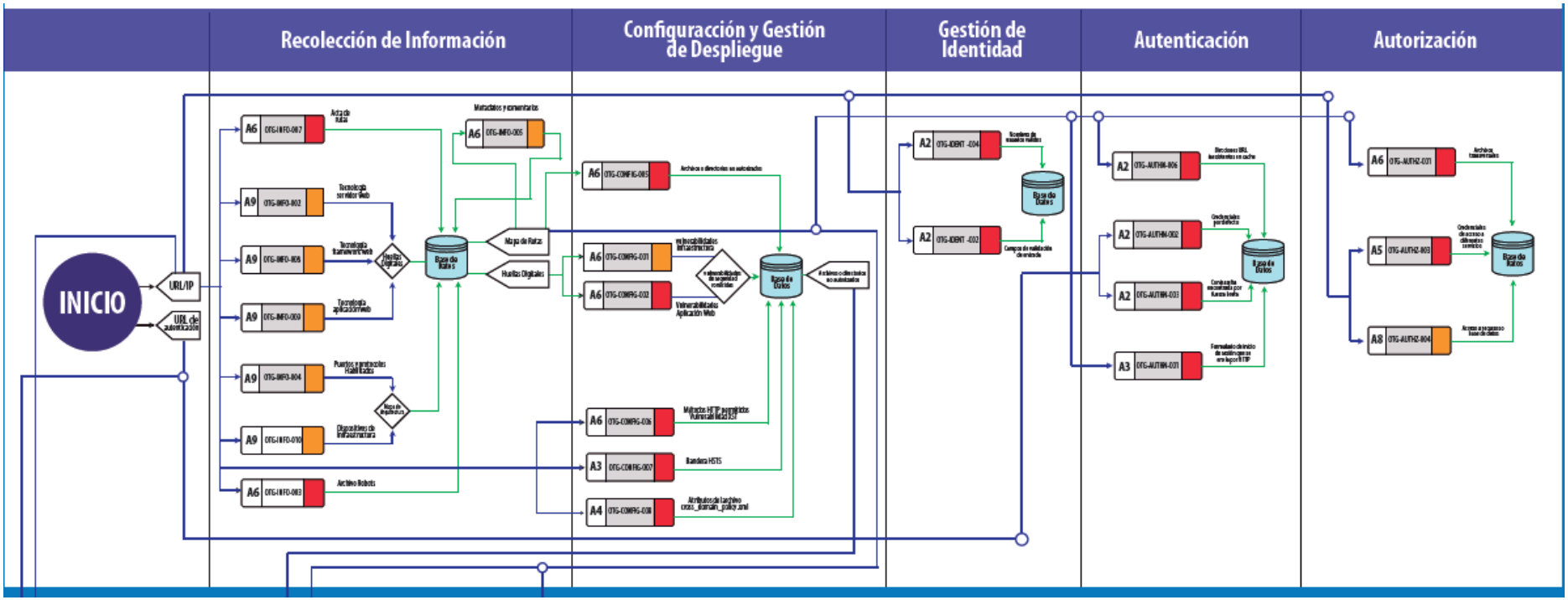


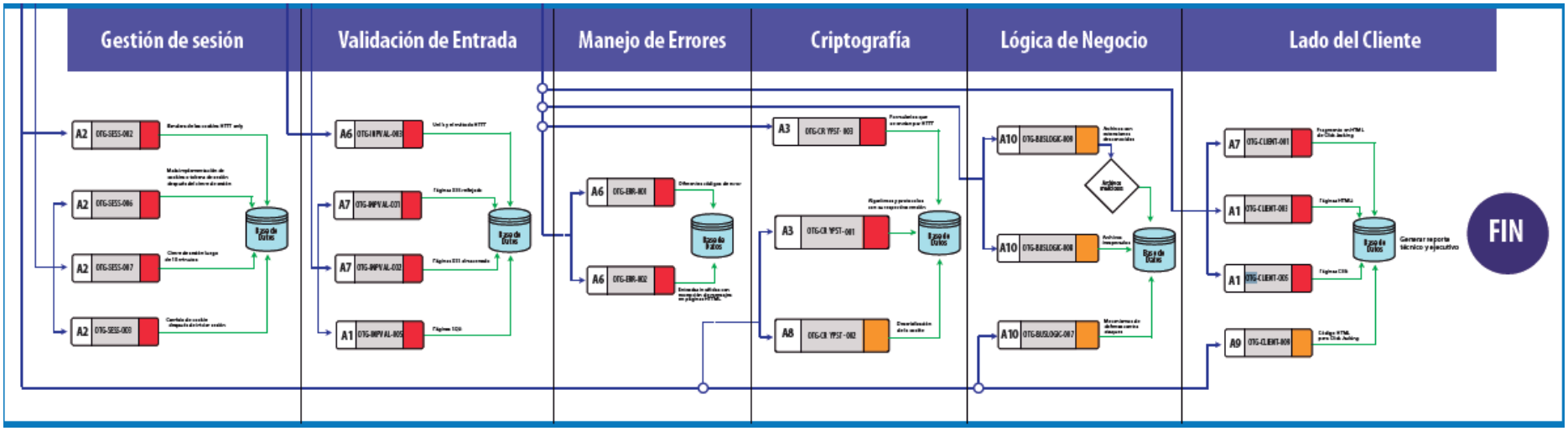
Figura 40. Prueba exitosa de la configuración del cluster. Fuente: Propia.

Ahora, el sistema está listo para desarrollar la aplicación paralela y distribuida.

Anexo E : Diagrama de secuencia de entradas y salidas de las pruebas implementadas




Recolección de Información
Configuración y Gestión de Despliegue
Gestión de Identidad
Autenticación
Autorización
INICIO
URL/IP
URL de autenticación
Base de Datos
Base de Datos
Base de Datos
Base de Datos
Base de Datos



Esta imagen podrá ser visualizado el día de la sustentación como poster, debido a que no es posible observarlo de mandera detallada en este anexo

Anexo F: Tabla de las pruebas implementadas siguiendo la metodología OWASP

A Top 10	Test ID	Test Name	Test Description	How to test
Information Gathering				
A9	OTG-INFO-002	Fingerprint Web Server	Hay varios proveedores y versiones diferentes de servidores web en el mercado hoy en día. Conocer el tipo de servidor web que se está probando ayuda significativamente en el proceso de recolección de información y también puede cambiar el curso del pentesting.	Conocer la la tecnología y versión de servidor web, para determinar vulnerabilidades conocidas.
A6	OTG-INFO-003	Review Webserver Metafiles for Information Leakage	En esta sección se describe cómo buscar el archivo robots.txt para detectar la fuga de información en el directorio raíz de la aplicación web o a través de las diferentes rutas. Además, analizar la lista de directorios que deben ser evitados por robots o rastreadores conformados por motores de búsqueda.	Identificar fuga de información de la ruta o rutas al directorio o carpeta de la aplicación web.

A9	OTG-INFO-004	Enumerate Applications on Webserver	En esta prueba se podrá visualizar un listado de los puertos que están habilitados en el servidor web, además de su respectiva información como el estado, nombre de la aplicación y el sevicio o protocolo asignado a cada puerto	Escanear los puertos que están habilitados en el servidor web, para poder observar información sensible.
A6	OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	Es muy común e incluso recomendable para los programadores incluir comentarios detallados y metadatos en su código fuente. Sin embargo, los comentarios y metadatos incluidos en el código HTML podrían revelar información interna que no debería estar disponible a intrusos potenciales. Los comentarios y metadatos deben ser revisados con el fin de determinar si hay fugas de información.	Revisar los comentarios y metadatos de la página web para entender mejor la aplicación y encontrar cualquier fuga de información.
A6	OTG-INFO-007	Map execution paths through application	Antes de comenzar las pruebas de seguridad, es de suma importancia entender la estructura de la aplicación. Sin una comprensión profunda de la distribución de la aplicación, es poco probable que sea probada exhaustivamente y obtener los mejores resultados para otra pruebas.	Crear mapas de la aplicación de destino y comprender los principales flujos de trabajo.
A9	OTG-INFO-008	Fingerprint Web Application Framework	En esta sección se describe cómo se puede obtener las huellas digitales (nombre y versión del framework web) que se utilizó ara desarrollar la aplicación web alojada en el servidor.	Identificar el tipo de framework web utilizado, para poder comprender mejor el entorno de la aplicación web.

A9	OTG-INFO-009	Fingerprint Web Application	Conocer los componentes de las aplicaciones web que se están probando, ayuda significativamente en el proceso de prueba y se reduce drásticamente el esfuerzo requerido durante la prueba. Estas aplicaciones web conocidas tienen diferentes tecnologías cómo lenguaje de programación, extensiones de criptografía, cookies, encabezados HTML que se pueden enumerar para identificar la aplicación.	Identificar las tecnologías de aplicación web y la versión para posteriormente determinar vulnerabilidades conocidas y la formas de explotarlas apropiadamente.
A6	OTG-INFO-010	Map Application Architecture	Esta sección se enfoca principalmente en mapear la arquitectura de la aplicación que se encuentra desplegada para identificar los dispositivos intermedios que se encuentra entre el cliente y el servidor, como por ejemplo servidores o balanceadores de carga, así también de otros dispositivos con los que posiblemente se comunica la aplicación	Identificar diferentes dispositivos en la infraestructura de rec, como un proxy, balanceador de carga o un sistema de detección de intrusos.
	Configuration and Deploy Management Testing			
A6	OTG-CONFIG-001	Test Network/Infrastructure Configuration	Una gestión apropiada de la configuración e infraestructura del servidor de web es muy importante para preservar la seguridad de la propia aplicación. Si elementos como el software del servidor web, la base de datos o los servidores de autenticación no están segguros, podrían presentar riesgos no deseados o introducir	Elaborar un mapa de los diferentes elementos que conforman la infraestructura para revisar la configuración de cada elemento encontrado y probarlos en busca de cualquier vulnerabilidad conocida.

			nuevas vulnerabilidades que podrían comprometer a la propia aplicación.	
A6	OTG-CONFIG-002	Test Application Platform Configuration	Durante esta fase se procede a identificar y probar la configuración de los elementos individuales que conforman la arquitectura del servidor web y/o aplicación web.	Identificar los elementos que genera por defecto el servidor web y/o aplicación web al momento de su instalación, para determinar si existe alguna vulnerabilidad que los pueda afectar.
A6	OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	Las interfaces del administrador se pueden presentar en la aplicación o en el servidor de aplicaciones para permitir que determinados usuarios puedan llevar a cabo actividades privilegiadas en el sitio, con el fin de obtener los nombres de directorios y archivos en el servidor web.	Descubrir interfaces de administrador y acceder a funcionalidades o directorios no autorizados.
A6	OTG-CONFIG-006	Test HTTP Methods	En esta prueba se procede a identificar los métodos HTTP disponibles y determinar si el servidor presenta vulnerabilidades contra XST (Cross Site Tracing).	Identificar los métodos HTTP disponibles sobre el servidor web, para conocer cómo se comunica un cliente con el servidor web. Además, para verificar si existe alguna vulnerabilidad que pueda explotarse.

A3	OTG-CONFIG-007	Test HTTP Strict Transport Security	El encabezado HTTPS Strict Transport Security (HSTS) es un mecanismo que tienen los sitios web para comunicarse con los navegadores web. Todo el tráfico intercambiado con un dominio debe siempre ser enviado mediante https; esto ayudará a proteger la información de que se envié mediante peticiones no cifradas.	Examinar la presencia del encabezado HSTS por medio de las cabecera de respuesta por parte del servidor web.
A4	OTG-CONFIG-008	Test RIA cross domain policy	En esta sección se procede a identificar la existencia del archivo de dominio cruzado (crossdomain.xml), sobre el servidor web y posteriormente observar el contenido de sus políticas o directivas que permiten el acceso remoto desde otros dominios diferentes.	Identificar el archivo crossdomain.xml en el servidor web y poder verificar sus directivas.
	Identity Management Testing			
A2	OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	Esta fase es útil para realizar ataques de fuerza bruta, al encontrar posibles usuarios que esten registrados en la aplicación web, en la que el se verifica si dado un nombre de usuario válido para es posible encontrar la contraseña correspondiente	Verificar si es posible recolectar un conjunto de nombres de usuario válidos para interactuar con el mecanismo de autenticación
	Authentication Testing			

A3	OTG-AUTHN-001	Testing for Transported over an Encrypted Channel	Probar el transporte de credenciales significa comprobar que los datos de autenticación del usuario se transfieren a través de un canal cifrado para evitar ser interceptados por usuarios maliciosos. El análisis se centra simplemente en tratar de entender si los datos viajan sin cifrar desde el navegador web al servidor, o si la aplicación web toma las medidas de seguridad apropiadas al usar el protocolo HTTPS.	Comprobar si las credenciales del usuario son transmitidas a través de un canal cifrado.
A2	OTG-AUTHN-002	Testing for default credentials	Para esta prueba se procede a verificar si la aplicación tiene un usuario o contraseña por defecto que los desarrolladores hayan dejado por algún motivo en algún formulario de acceso.	Identificar las credenciales por defecto que tiene la aplicación en el formulario de inicio de sesión, para poder comprobar un acceso exitoso.
A2	OTG-AUTHN-003	Testing for Weak lock out mechanism	Los mecanismos de protección se utilizan para mitigar los ataques de fuerza bruta, estos mecanismos requieren un equilibrio entre la protección de cuentas para el acceso no autorizado y evitar la negación del usuario a los servicios de la aplicación web.	Evaluar la capacidad del mecanismo de protección al intentar encontrar la contraseña por un ataque de fuerza bruta, con el fin de mitigar el ingreso hacia la aplicación web.
	Authorization Testing			
A2	OTG-AUTHZ-002	Testing for bypassing authorization schema		

A5	OTG-AUTHZ-003	Testing for Privilege Escalation	Esta prueba describe el problema del escalamiento de privilegios de una etapa a otra. Durante esta fase se debe verificar que no es posible para un usuario modificar sus privilegios o roles dentro de la aplicación.	Determinar el grado de escalamiento de los privilegios de un usuario, con el fin de obsevar si logra acceder a otras cuentas no autorizadas.
A8	OTG-AUTHZ-004	Testing for Insecure Direct Object References	En esta fase se procede a identificar en qué lugar de la aplicación web se presentan las referencias de objetos inseguros, lo cual consiste en dar acceso directo a los objetos que se comunican con la base de datos, normalmente este tipo de vulnerabilidad se logra descubrir en campos de entrada suministrada por el usuario.	Identificar el lugar donde se presente la referencia de objetos inseguros para poder evadir alguna autorización o acceder y modificar recursos de un parámetro directamente
	Session Management Testing			
A2	OTG-SESS-002	Testing for Cookies attributes	Estas funciones son usadas como una autorización de sesión y un token de autenticación. Por lo tanto si un atacante fuera capaz de adquirir un token de sesión podría usar esta cookie para secuestrar una sesión válida	Analizar cómo una aplicación puede tomar las precauciones necesarias al asignar cookies y cómo probar que los párametros se hayan configurado correctamente.
A2	OTG-SESS-003	Testing for Session Fixation	Cuando una aplicación no renueva las cookies de sesión después de una autenticación exitosa, podría ser posible encontrar una vulnerabilidad de fijación de sesión y forzar a un usuario a utilizar una cookie conocida por el atacante. En ese caso, un atacante podría robar la sesión	Verificar si el identificador de la sesión se vuelve a re asignar después de una autenticación exitosa.

			del usuario.	
A2	OTG-SESS-007	Test Session Timeout	Todas las aplicaciones deben implementar un tiempo de inactividad para las sesiones. Este tiempo de espera define la cantidad de tiempo que una sesión permanecerá activa en caso de que el usuario no interactue con la aplicación, entonces se deberá cerrar e invalidar la sesión, definida desde la última solicitud HTTP recibida por la aplicación web para un ID de sesión determinado.	Verificar que la aplicación desconecta automáticamente a un usuario cuando este usuario estuvo inactivo durante un período de tiempo, asegurando que no sea posible reutilizar la misma sesión y que no permanezcan datos confidenciales en la memoria caché del navegador ..
	Input Validation Testing			
A7	OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	Durante esta prueba se procede a evaluar mediante scripts o payloads las entradas que interactuan con el usuario como formularios de registro o campos de búsqueda que posee la aplicación, para verificar si han utilizado mecanismos que permitan sanitizar los datos ingresados por el usuario.	Evaluar las entradas (inputs) y las peticiones HTTP de la aplicación web, para poder identificar los puntos donde se pueda realizar un ataque de Cross Site Scripting reflejado.
A7	OTG-INPVAL-002	Testing for Stored Cross Site Scripting	En esta prueba se procede a evaluar mediante scripts o payloads si las entradas que tiene la aplicación se han utilizado mecanismos que permitan sanitizar los datos ingresados por el usuario, con el fin de no alterar la base de	Evaluar los campos de entrada de la aplicación web mediante para lograr identificar los puntos donde se pueda realizar un XSS almacenado y obtener información de la base de datos.

			datos de manera permanente.	
A1	OTG-INPVAL-005	Testing for SQL Injection	Un ataque de inyección SQL consiste en la inserción o "inyección" de una consulta SQL parcial o completa a través de los datos de entrada o transmitidos desde el cliente (navegador) hacia la aplicación web. Un ataque exitoso de inyección SQL puede leer o modificar (insertar/actualizar/eliminar) datos sensibles de la base de datos y hasta ejecutar operaciones administrativas.	Determinar si la aplicación posee mecanismos de control en los campos de entrada y URLs para evitar ser vulnerado mediante un ataque de inyección SQL que puede utilizar diferentes scripts o payloads
		Oracle Testing		
		MySQL Testing		
		SQL Server Testing		
		Testing PostgreSQL		
	Error Handling			
A6	OTG-ERR-001	Analysis of Error Codes	A menudo, durante una prueba de penetración en las aplicaciones web, se encuentran con muchos códigos de error generados desde aplicaciones o servidores web. Es posible hacer que estos errores se muestren mediante el uso de solicitudes particulares, especialmente	Encontrar los diferentes códigos de error, ya que revelan mucha información sobre bases de datos, errores y otros componentes tecnológicos directamente relacionados con las aplicaciones web.

			diseñadas con herramientas especializadas o generados manualmente.	
	Cryptography			
A3	OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	Los datos sensibles deben estar protegidos cuando se transmiten a través de la red. Tales datos pueden incluir credenciales de usuario y tarjetas de crédito. Como regla general, si los datos deben protegerse cuando se almacenan, también se deben proteger durante la transmisión.	Analizar los puertos que utilizan SSL/TLS para realizar una auditoria acerca de la vulnerabilidad de los protocolo de cifrado (SSLv2, SSLv3, TLS 1.0,1.1 o 1.2) en razón de su versión, configuración y explotabilidad.
A3	OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels	Si la aplicación transmite información confidencial a través de canales no cifrados por el protocolo HTTP se considera un riesgo de seguridad. Un ejemplo es la autenticación básica que envía credenciales de autenticación en texto plano.	Verificar si la información se transmite a través del protocolo HTTP en lugar de HTTPS o si se utilizan protolocos de de cifrado débiles.
	Business Logic Testing			

A10	OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use	La falta de pruebas de funcionalidad y validación en la etapa de desarrollo e implementación de la aplicación web, se tendrá que configurar o implementar dispositivos o mecanismos de defensa en la capa de aplicación, para evitar ser vulnerados.	Identificar posibles mecanimos de defensa o control como cambiar los códigos de estado, bloquear peticiones para evitar la denegación de servicio o tener un registro de control para el usuario.
	Client Side Testing			
A7	OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	Cross site scripting basado en DOM es el nombre para errores XSS que son el resultado del contenido activo del lado del navegador en una página, generalmente JavaScript que obtiene ingresos del usuario lo que lleva a la ejecución de código inyectado.	Modificar el contenido activo, como una función de JavaScript, la culase se modifica con una solicitud especialmente diseñada, tal como un elemento DOM que puede ser controlado por un atacante.
A1	OTG-CLIENT-003	Testing for HTML Injection	En esta prueba se verifica si los formularios de la aplicación son vulnerables a inyección HTML, esta puede utilizar para engañar a usuarios legítimos re direccionándolos a sitios maliciosos renderizados dentro de la misma aplicación web vulnerable.	Analizar si las entradas de usuario se sanatiza correctamente y la salida de la aplicación web están codificadas. El objetivo es lograr ejecutar todo tipo de inyección HTML en el contexto de la víctima

A9	OTG-CLIENT-009	Testing for Clickjacking	Es una técnica maliciosa que consiste en engañar a un usuario web para que interactúe (en la mayoría de los casos haciendo click) con un componente diferente en la aplicación web, pero que es transparente para el usuario. Este tipo de ataque podría enviar comandos no autorizados o revelar información confidencial mientras la víctima interactúa en aplicaciones web aparentemente inofensivas.	Descubrir si la aplicación web tiene protección contra los ataques de clickjacking o, si los desarrolladores han implementado alguna forma de protección, si estas técnicas se pueden pasar por alto. Se puede crear una prueba de concepto para explotar la vulnerabilidad.

Anexo G

Tabla del cálculo de riesgo para el agente de amenaza

Agente de Amenaza							
Parámetros de la prueba		Factores				Probabilidad	
A TOP 10	ID	Nivel de habilidad	Motivación	Oportunidad y Recursos	Tamaño	Cuantitativo	Cualitativo
A9	OTG-INFO-002	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A6	OTG-INFO-003	1 - Algunas habilidades técnicas	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,2	LOW
A6	OTG-INFO-004	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A6	OTG-INFO-005	1 - Algunas habilidades técnicas	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,2	LOW

A6	OTG-INFO-007	1 - Algunas habilidades técnicas	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,2	LOW
A9	OTG-INFO-008	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A9	OTG-INFO-009	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A6	OTG-INFO-010	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A6	OTG-CONFIG-001	1 - Algunas habilidades técnicas	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,2	LOW
A6	OTG-CONFIG-002	1 - Algunas habilidades técnicas	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,2	LOW

A6	OTG-CONFIG-005	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A6	OTG-CONFIG-006	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A3	OTG-CONFIG-007	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A4	OTG-CONFIG-008	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A2	OTG-IDENT-004	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A3	OTG-AUTHN-001	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW

A2	OTG-AUTHN-002	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A2	OTG-AUTHN-003	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A5	OTG-AUTHZ-003	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A8	OTG-AUTHZ-004	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A2	OTG-SESS-002	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A2	OTG-SESS-003	2 - Habilidades de red y programación	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,5	LOW

A2	OTG-SESS-007	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A7	OTG-INPVAL-001	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	3 - Algunos accesos o requiere algunos recursos	1 - Desarrolladores	2,5	LOW
A7	OTG-INPVAL-002	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	3 - Algunos accesos o requiere algunos recursos	1 - Desarrolladores	2,5	LOW
A1	OTG-INPVAL-005	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 -	1 - Desarrolladores	2,2	LOW
A1	OTG-INPVAL-005	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	3 - Algunos accesos o requiere algunos recursos	1 - Desarrolladores	2,5	LOW
A1	OTG-INPVAL-005	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	3 - Algunos accesos o requiere algunos recursos	1 - Desarrolladores	2,5	LOW

A1	OTG-INPVAL-005	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	3 - Algunos accesos o requiere algunos recursos	1 - Desarrolladores	2,5	LOW
A1	OTG-INPVAL-005	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A6	OTG-ERR-001	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW
A3	OTG-CRYPST-001	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A3	OTG-CRYPST-003	3 - Habilidades de penetración en seguridad	2 - Posible recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,0	LOW
A10	OTG-BUSLOGIC-007	3 - Habilidades de penetración en seguridad	1 - Bajo o sin recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	1,7	LOW

A7	OTG-CLIENT-001	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A1	OTG-CLIENT-003	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW
A9	OTG-CLIENT-009	3 - Habilidades de penetración en seguridad	3 - Alta recompensa	2 - Acceso especial o requiere recursos	1 - Desarrolladores	2,2	LOW

Anexo H

Reporte técnico y ejecutivo

Anexo I

Tablas comparativas de vulnerabilidades para calcular la medición de evaluación en las curvas ROC para la aplicación web **http://demo.testfire.com**

ScanLynx contra Netsparker

	ScanLynx	**Netsparker**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection			
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	0	1	FN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy			
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	0	1	VN
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			

Infrastructure y platform Configuration			
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces			
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Vulnerabilidades de componentes conocidos)			
Microsoft - IIS	1	1	VP
ASP_NET	1	1	VP
Operative System	0	0	VN
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra OWASP ZAP

	ScanLynx	**OWASP ZAP**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection			
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	0	1	FN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	0	FP
Insufficient Transport Layer Protection	1	0	FP

Sensitive information sent via unencrypted channels	1	0	FP
A4 (XML)			
Test RIA cross domain policy			
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	0	0	VN
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration			
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces			
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Vulnerabilidades de componentes conocidos)			
Microsoft - IIS	1	0	FP
ASP_NET	1	0	FP
Operative System	0	0	VN
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Nessus

	ScanLynx	**Nessus**	**Resultado cualitativo**

A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection			
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	0	1	FN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy			
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	0	FP
HTTP Methods	0	1	FN
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration			
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	0	0	VN
A7 Cross Site Scripting			
Cross Site Scripting	1	0	FP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Vulnerabilidades de componentes conocidos)			

Microsoft - IIS	1	1	VP
ASP_NET	1	1	VP
Operative System	0	0	VN
open ports	1	1	VP
Application services	1	1	VP
ClickJacking	1	0	FP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Acunetix

	ScanLynx	**Acunetix**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection			
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	VP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	0	1	FN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy			
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	0	1	FN
Review Webserver Metafiles (robots.txt)	1	1	VP

Analysis of Error Codes	1	0	FP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration			
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	0	1	FN
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Vulnerabilidades de componentes conocidos)			
Microsoft - IIS	1	1	VP
ASP_NET	1	1	VP
Operative System	0	0	VN
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

Tablas comparativas de vulnerabilidades para calcular la medición de evaluación en las curvas ROC para la aplicación web **http://www.hackazon.webscantest.com**

ScanLynx contra Netsparker

	ScanLynx	**Netsparker**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection			
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP

Cookies attributes	1	1	VP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	1	1	VP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	0	1	FN
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP
JQuery	1	1	VP
Operative System	1	0	FP
open ports	1	0	VP
Application services	1	1	VP
ClickJacking	1	1	VP

A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra OWASP ZAP

	ScanLynx	OWASP ZAP	Resultado cualitativo
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection	0	0	VN
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	1	VP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	0	FP
Insufficient Transport Layer Protection	1	0	FP
Sensitive information sent via unencrypted channels	1	0	FP
A4 (XML)			
Test RIA cross domain policy	1	0	FP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	1	FN
A6(Mala configuración)			
Map execution paths through application	1	0	FP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	0	FP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP

A7 Cross Site Scripting			
Cross Site Scripting	0	1	FN
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	0	VN
PHP programming language	1	1	VP
JQuery	1	0	FP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Nessus

	ScanLynx	**Nessus**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection	0	0	VN
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	0	FP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	1	1	VP

A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	1	1	VP
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	0	0	VN
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP
JQuery	1	1	VP
Operative System	1	1	VP
open ports	1	1	VP
Application services	1	1	VP
ClickJacking	1	0	FP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Acunetix

	ScanLynx	**Acunetix**	**Resultado cualitativo**
A1 (Injection)			

SQLinjection	0	1	FN
Testing for HTML Injection	1	1	VP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	0	FP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	0	1	FN
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	1	VP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	1	VP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP

JQuery	0	0	VN
Operative System	1	0	VP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

Tablas comparativas de vulnerabilidades para calcular la medición de evaluación en las curvas ROC para la aplicación web **http://testphp.vulweb.com**

ScanLynx contra Netsparker

	ScanLynx	**Netsparker**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	1	1	VP
Testing for HTML Injection	1	0	FP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation			
Cierre de sesión automático			
Test Session Timeout			
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	1	1	VP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods			

Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	0	FP
Review Webpage Comments and Metadata for Information Leakage	0	0	VN
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information	0	1	FN
Enumerate Infrastructure and Application Admin Interfaces	1	1	VP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Vulnerabilidades de componentes conocidos)			
Web Server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra OWASP ZAP

	ScanLynx	**OWASP ZAP**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	1	1	VP
Testing for HTML Injection	1	0	VP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	VP
Session Fixation			
Cierre de sesión automático			
Test Session Timeout			

A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	0	FP
Insufficient Transport Layer Protection	1	0	FP
Sensitive information sent via unencrypted channels	1		
A4 (XML)			
Test RIA cross domain policy	1	0	FP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	0	FP
HTTP Methods			
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage	0	0	VN
Infrastructure y platform Configuration	1	0	FP
Test File Extensions Handling for Sensitive Information	0	0	VN
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Vulnerabilidades de componentes conocidos)			
Web Server	1	0	FP
PHP programming language	1	0	FP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	0	FP
ClickJacking	1	0	FP
A10 (Registro y monitoreo insuficientes)			

Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra Nessus

	ScanLynx	**Nessus**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	1	0	FP
Testing for HTML Injection	1	0	FP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation			
Cierre de sesión automático			
Test Session Timeout			
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	1	1	VP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include			
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods			
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage	0	0	VN
Infrastructure y platform Configuration	1	1	P
Test File Extensions Handling for Sensitive Information	0	0	VN

Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	1	0	FP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Vulnerabilidades de componentes conocidos)			
Web Server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	1	VP
open ports	1	1	VP
Application services	1	1	VP
ClickJacking	1	0	FP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra Acunetix

	ScanLynx	**Acunetix**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	1	1	VP
Testing for HTML Injection	1	1	VP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	1	VP
Session Fixation			
Cierre de sesión automático			
Test Session Timeout			
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP

A4 (XML)			
Test RIA cross domain policy	1	1	VP
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	1	FN
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods			
Review Webserver Metafiles (robots.txt)			
Analysis of Error Codes	1	0	FP
Review Webpage Comments and Metadata for Information Leakage	0	0	VN
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information	0	1	FN
Enumerate Infrastructure and Application Admin Interfaces	1	1	VP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Vulnerabilidades de componentes conocidos)			
Web Server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

Tablas comparativas de vulnerabilidades para calcular la medición de evaluación en las curvas ROC para la aplicación web **http://php.testsparker.com**

ScanLynx contra Netsparker

	ScanLynx	Netsparker	Resultado cualitativo
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection	0	0	VN
A2 (Broken Authentication and Session Management)			
Weak lock mechanism	1	1	VP
Session Fixation	1	0	FP
Cierre de sesión automático	1	0	FP
Test Session Timeout	0	0	VN
A3 (Sensitive Data Exposure)			
HTP HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	0	FP
A4 (XML EXternal Entity)			
Test RIA cross domain policy	0	1	FN
A5 (Broquen Access Control)			
Testing Directory traversal/file include	0	0	VN
A6(Security Misconfiguration)			
Map execution paths through application	1	1	VP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	0	FP
Revisar comentarios y metadatos	0	0	VN
Infrastructure y platform Configuration	0	1	FN
Test File Extensions Handling for Sensitive Information	0	1	FN
A7 (XSS)			
Cross Site Scripting	0	1	FN
A8 (Insecure Deserialization)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Using Components with Known Vulnerabilities)			

Apache server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Insufficient LogIn and Monitoring)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra OWASP ZAP

	ScanLynx	**OWASP ZAP**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection	0	0	VN
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Cierre de sesión automático	1	0	FP
Test Session Timeout	0	0	VN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	0	FP
Insufficient Transport Layer Protection	1	0	FP
A4 (XML)			
Test RIA cross domain policy	0	0	VN
A5 (Broquen Access Control)			
Testing Directory traversal/file include	0	0	VN
A6(Security Misconfiguration)			
Map execution paths through application	1	0	FP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP

Revisar comentarios y metadatos	0	0	VN
Infrastructure y platform Configuration	0	0	VN
Test File Extensions Handling for Sensitive Information	0	0	VN
A7 Cross Site Scripting			
Cross Site Scripting	0	0	VN
A8 (Insecure Deserialization)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Using Components with Known Vulnerabilities)			
Apache server	1	0	FP
PHP programming language	1	0	FP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	0	FP
ClickJacking	1	1	VP
A10 (Insufficient LogIn and Monitoring)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Nessus

	ScanLynx	**Nessus**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection	0	0	VN
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Cierre de sesión automático	1	0	FP
Test Session Timeout	0	0	VN

A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
A4 (XML)			
Test RIA cross domain policy	0	0	VN
A5 (Broquen Access Control)			
Testing Directory traversal/file include	0	0	VN
A6(Security Misconfiguration)			
Map execution paths through application	1	0	FP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	0	VP
Revisar comentarios y metadatos	0	0	VN
Infrastructure y platform Configuration	0	1	FN
Test File Extensions Handling for Sensitive Information	0	0	VN
A7 Cross Site Scripting			
Cross Site Scripting	0	0	VN
A8 (Insecure Deserialization)			
Testing for Insecure Direct Object References	0	0	VN
A9 (Using Components with Known Vulnerabilities)			
Apache server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	1	VP
open ports	1	1	VP
Application services	1	1	VP
ClickJacking	1	0	FP
A10 (Insufficient LogIn and Monitoring)			
Test Defenses Against Application Mis-use	1	0	FP

ScanLynx contra Acunetix

	ScanLynx	Acunetix	Resultado cualitativo
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection	0	0	VN
A2 (Perdida de autenticación)			
Weak lock mechanism	1	1	VP
Session Fixation	1	1	VP
Cierre de sesión automático	1	0	VP
Test Session Timeout	0	0	VN
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
A4 (XML)			
Test RIA cross domain policy	0	1	FN
A5 (Broquen Acces Control)			
Testing Directory traversal/file include	0	0	VN
A6(Security Misconfiguration)			
Map execution paths through application	1	1	VP
HTTP Methods	1	1	VP
Review Webserver Metafiles (robots.txt)	1	1	VP
Analysis of Error Codes	1	1	VP
Revisar comentarios y metadatos	0	0	VN
Infrastructure y platform Configuration	0	1	FN
Test File Extensions Handling for Sensitive Information	0	1	FN
A7 Cross Site Scripting			
Cross Site Scripting	0	1	FN
A8 (Insecure Deserialization)			
Testing for Insecure Direct Object References	0	1	FN
A9 (Using Components with Known Vulnerabilities)			

Apache server	1	1	VP
PHP programming language	1	1	VP
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Insufficient LogIn and Monitoring)			
Test Defenses Against Application Mis-use	1	0	FP

Tablas comparativas de vulnerabilidades para calcular la medición de evaluación en las curvas ROC para la aplicación web **http://www.webscantest.com**

ScanLynx contra Netsparker

	ScanLynx	**Netsparker**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection	1	0	FP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	1	VP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	0	1	FN
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	1	VP

HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP
JQuery	0	1	FN
Operative System	1	0	FP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra OWASP ZAP

	ScanLynx	**OWASP ZAP**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection	1	0	FP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP

Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	1	VP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	0	FP
Sensitive information sent via unencrypted channels	1	0	FP
A4 (XML)			
Test RIA cross domain policy	0	0	VN
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	1	FN
A6(Mala configuración)			
Map execution paths through application	1	0	FP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	0	FP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	0	FP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	0	FP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	0	VN
PHP programming language	1	1	VP
JQuery	0	0	VN
Operative System	1	0	FP
open ports	1	0	FP

Application services	1	0	FP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra Nessus

	ScanLynx	**Nessus**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	0	VN
Testing for HTML Injection	1	0	FP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	0	FP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP
A4 (XML)			
Test RIA cross domain policy	0	0	VN
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	0	FP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	1	VP
Analysis of Error Codes	1	0	FP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive			

Information			
Enumerate Infrastructure and Application Admin Interfaces	1	1	VP
A7 Cross Site Scripting			
Cross Site Scripting	1	0	FP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP
JQuery	0	0	VN
Operative System	1	1	VP
open ports	1	1	VP
Application services	1	1	VP
ClickJacking	1	0	FP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN

ScanLynx contra Acunetix

	ScanLynx	**Acunetix**	**Resultado cualitativo**
A1 (Injection)			
SQLinjection	0	1	FN
Testing for HTML Injection	1	1	VP
A2 (Perdida de autenticación)			
Weak lock mechanism	1	0	FP
Session Fixation	1	0	FP
Test Session Timeout	1	0	FP
Cookies attributes	1	0	FP
A3 (Exposición de datos sensibles)			
HTTP Strict Transport Security	1	1	VP
Insufficient Transport Layer Protection	1	1	VP
Sensitive information sent via unencrypted channels	1	1	VP

A4 (XML)			
Test RIA cross domain policy	0	0	VN
A5 (Perdida de control de acceso)			
Testing Directory traversal/file include	0	0	VN
A6(Mala configuración)			
Map execution paths through application	1	1	VP
HTTP Methods	1	0	FP
Review Webserver Metafiles (robots.txt)	1	1	VP
Analysis of Error Codes	1	1	VP
Review Webpage Comments and Metadata for Information Leakage			
Infrastructure y platform Configuration	1	1	VP
Test File Extensions Handling for Sensitive Information			
Enumerate Infrastructure and Application Admin Interfaces	1	1	VP
A7 Cross Site Scripting			
Cross Site Scripting	1	1	VP
A8 (Deserialización insegura)			
Testing for Insecure Direct Object References			
A9 (Vulnerabilidades de componentes conocidos)			
Apache Server	0	1	FN
PHP programming language	1	1	VP
JQuery	0	0	VN
Operative System	1	0	VP
open ports	1	0	FP
Application services	1	1	VP
ClickJacking	1	1	VP
A10 (Registro y monitoreo insuficientes)			
Test Defenses Against Application Mis-use	0	0	VN