UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

COMPARACIÓN DE TÉCNICAS DE DETECCIÓN DE VULNERABILIDADES DE ATAQUES DE CROSS SITE SCRIPTING EN APLICACIONES WEB DE MICROEMPRESAS.

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor(a) (es):

Bachiller, Bocanegra Chávez Cristian Alexander

ORCID: https://orcid.org/0000-0001-7787-4556

Asesor(a):

Ingeniero, Mejía Cabrera Heber Ivan

ORCID: https://orcid.org/0000-0002-0007-0928

Línea de Investigación:

Tecnologías de la información

Pimentel – Perú 2021

APROBACIÓN DEL JURADO

COMPARACIÓN DE TÉCNICAS DE DETECCIÓN DE VULNERABILIDADES DE ATAQUES DE CROSS SITE SCRIPTING EN APLICACIONES WEB DE MICROEMPRESAS.

_____

Bachiller, Bocanegra Chavez Cristian Alexander

Autor

_____

Mg. Mejía Cabrera Heber Iván

Asesor

_____ _____

Dr. Vásquez Leyva Oliver

Presidente de Jurado

Mg. Chirinos Mundaca Carlos Alberto

Secretario de Jurado

_____

Mg. Mejía Cabrera Heber Iván

Vocal de Jurado

DEDICATORIAS

El presente trabajo lo dedico principalmente a Dios, por darme fuerza para continuar en el proceso de obtener el título en Ingeniería de Sistemas.

A mi madre Shulixzar del Pilar Chavez Prieto que con su amor, paciencia y apoyo me ha permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo de esfuerzo y valentía, de no temer las adversidades.

A mi abuelita Clara Elena Prieto Cruzado por darme todo lo necesario en toda la etapa de la universidad, gracias por todo el amor, valores y sobre todo por darme una oportunidad de estudiar y ser alguien en la vida, te amo mucha mami clara.

A mi tío Marco Antonio Chávez Prieto por tomar el papel de padre en mí, tus consejos, amor y sobre todo por apoyarme en todas las decisiones que he tomado. Te considero mucho.

Finalmente, una dedicatoria especial a mis docentes, por toda la disciplina y enseñanza recibida durante toda la carrera universitaria.

AGRADECIMIENTOS

Agradecer a todos los docentes de la Universidad Señor de Sipán por inculcarme valores, conocimientos y brindarme su apoyo. Así mismo, a mis compañeros por el apoyo brindado y al asesor, gracias totales.

RESUMEN

En los últimos años las aplicaciones Web se convirtieron en el medio más común en Internet y son fácilmente accesibles usando software de navegación. Debido a la popularidad y facilidad de uso, ganó el interés de los atacantes. El problema viene desde el desarrollador de aplicaciones web, porque no todos aplican reglas de prevención de vulnerabilidades, y toman como requisito prioritario el funcionamiento de procesos generales de la aplicación web. Las aplicaciones web son propensos a todo tipo de ataques entre ellos los códigos maliciosos, inyecciones SQL, incorporación de ficheros maliciosos, entre otros. Cross Site Scripting está dentro de las cinco principales vulnerabilidades en aplicaciones web, haciendo que ocurran con frecuencia ataques. Cross Site Scripting permite al atacante ejecutar código malicioso en el navegador de otro usuario, una vez que el atacante gana el control él será capaz de realizar acciones como el secuestro de cookie, el malware-difusión y la redirección maliciosa mediante la incorporación de Cross Site Scripting que se ejecutan cada vez que se carga la página. La presente investigación se comparó dos técnicas de detección de vulnerabilidades, estudiando los procesos que emiten para detectar este tipo de vulnerabilidad, aclarando que detecta por el método de envió POST y se basa en el lenguaje HTML y PHP, entendiendo que se necesitó una herramienta por técnica para escanear el caso de estudio, utilizando python para escribir el código. Una vez concluidas las herramientas se realizaron pruebas en el caso de estudio vulnerable, ejecutando las herramientas en Kali Linux. Mostrando eficiencia de las herramientas para detectar vulnerabilidades de Cross Site Scripting y el poco tiempo en escanear al caso de estudio. Por lo que se recomienda utilizar dichas herramientas para rescribir el código de aplicaciones web y así prevenir ataques de Cross Site Scripting. Finalmente, la presente investigación se basa principalmente en comprar las dos técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting en las aplicaciones web, tomando como resultado final que la técnica Detección Dinámica de Vulnerabilidades (DDV) es mejor que la técnica Sumidero Detección de Vulnerabilidades (SDV).

Palabras Claves: Seguridad de Aplicación Web, Cross Site Scripting, detección de ataques, fuga de información, escáner web, análisis, vulnerabilidad de seguridad web.

ABSTRACT

In recent years, Web applications have become the most common medium on the Internet and are easily accessible using browser software. Due to the popularity and ease of use, it won the interest of attackers. The problem comes from the web application developer, because not all apply vulnerability prevention rules, and take as a priority requirement the operation of general processes of the web application. Web applications are prone to all kinds of attacks including malicious code, SQL injections, and embedding of malicious files, among others. Cross Site Scripting is among the top five vulnerabilities in web applications, causing them to occur with frequent attacks. Cross Site Scripting allows the attacker to execute malicious code in another user's browser, once the attacker gains control, he will be able to perform actions such as cookie hijacking, malware-spreading and malicious redirection by incorporating Cross Site Scripting that is executed every time the page is loaded. The present investigation compared two vulnerability detection techniques, studying the processes that they emit to detect this type of vulnerability, clarifying that it detects by the POST sending method and is based on the HTML and PHP language, understanding that a tool was needed for technique to scan the case study, using python to write the code. Once the tools were completed, tests were performed on the vulnerable case study, running the tools on Kali Linux. Showing efficiency of the tools to detect Cross Site Scripting vulnerabilities and the little time to scan the case study. Therefore, it is recommended to use these tools to rewrite the code of web applications and thus prevent Cross Site Scripting attacks. Finally, this research is mainly based on purchasing the two techniques for detecting vulnerabilities from Cross Site Scripting attacks in web applications, taking as a final result that the Dynamic Vulnerability Detection (DDV) technique is better than the Sink Detection technique. Vulnerabilities (SDV).

Keywords: Web Application Security, Cross Site Scripting, Attack Detection, Information Leak, Web Scanner, Scan, Web Security Vulnerability.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

I. INTRODUCCIÓN

La seguridad de la información en las microempresas está descuidada y pierde cierto criterio de importancia, creyendo conveniente en el no invertir en seguridad, es por ello que el índice de fraude es muy alto, permitiendo robo de data o información, pérdidas económicas, manipulaciones, entre otros, dichos fraudes son provocados por ciberataques. (ESET, 2018), señala como está la seguridad de la información en empresas de Latinoamérica, demostrando que en el año 2016 se reportó 6474 vulnerabilidades a los sistemas de empresas, mientras que en el año 2017 se duplico las vulnerabilidades con una cantidad de 14700. Así mismo (ESET & Welivesecurity, 2019) en su reporte demuestra que en el 2018 las vulnerabilidades llego a 15000, mientras (ESET & Welivesecurity, 2020) afirma que para el año 2019 disminuyo las vulnerabilidades a 12036, pero en el año 2020 incremento a 14090 las vulnerabilidades a los sistemas de empresa, demostrando que la gran parte de empresas en Latinoamérica se preocupan por dicho incidente, teniendo la variedad de amenazas que pueden emplearse para robar información valiosa, desde ataques externos hasta fraudes. ESET también muestra los incidentes de seguridad con un 60%, afirmando que empresas encuestadas en Latinoamérica son víctima de robo de datos (información), demostrando que Ecuador y Paraguay son los países que sufren mayor índice de infecciones con un 70% y Perú se encuentra con un 58% que es una cantidad muy elevada.

La riqueza de las aplicaciones web y la funcionalidad avanzada, así como la facilidad de acceso y la disponibilidad, han llevado a la mayoría de las empresas a confiar más en ellas. Desafortunadamente, por las mismas razones, se han convertido en objetivo principal de atacantes. Las principales deficiencias de seguridad han hecho que las aplicaciones sean vulnerables a numerosos ataques graves y exitosos. (Babiker, Karaarslan, & Hoscan 2018) confirma que los ataques de aplicaciones web son un área cada vez más importante en seguridad de la información. En el cual se observa que los atacantes desarrollan la capacidad de omitir los controles de seguridad y lanzan una gran cantidad de ataques sofisticados. Demostrando que Cross Site Scripting es el ataque más

usado en la actualidad y tiene resultados positivos para los atacantes, ellos utilizaron técnicas y soluciones para detectar ataques, como cortafuegos, sistemas de detección de intrusos, honeypots y técnicas forenses. Finalmente mostraron la capacidad que pose todas las técnicas y soluciones para prevenir los posibles ataques Cross Site Scripting a aplicaciones web.

Wang et al, (2015), presentan una técnica utilizando clasificadores y el modelo de N-gram mejorado para detectar vulnerabilidades de ataques de Cross Site Scripting, los resultados demuestran la eficacia de la técnica en detectar vulnerabilidades de ataques de Cross Site Scripting, pero realizaron las pruebas en aplicaciones web en línea como las redes sociales, deduciendo que no es apta para la presente investigación. Así mismo, (Huang et al, 2013) muestran una técnica, en la cual implementan un generador automático de exploits para problemas de seguridad web en aplicaciones web del mundo real. El socket simbólico se utiliza como entrada para ejecuciones simbólicas en las plataformas web. Se evalúan nueve aplicaciones web del mundo real, generando resultados experimentales que ponen de manifiesto la viabilidad de la aplicación, pero en la evaluación que se realizó para la presente investigación no obtuvo un puntaje apto para la necesidad de esta, descartándolo por completo. En la tercera técnica (Mohammadi et al, 2016) los autores proponen pruebas automáticas para detectar un tipo común de vulnerabilidad de ataques de Cross Site Scripting causada por una codificación incorrecta de datos que no son de confianza. Estudios empíricos sugieren que la técnica tiene una buena cobertura en detectar posibles ataques de Cross Site Scripting, mas no detectar vulnerabilidades, y en la evaluación de la presente investigación también obtuvo un puntaje no ideal, por lo que se descartó.

Las dos siguientes técnicas fueron escogidas para ser implementadas en la presente investigación, la primera técnica sumidero - detección de vulnerabilidad “SDV” (Gupta, Govil, & Singh, 2014) y la segunda detección dinámica de vulnerabilidades (DDV) (Hou and Chen, 2018). Dichas técnicas obtuvieron el puntaje de evaluación más alto, determinando aptas para la presente investigación. Cabe destacar que se evalúo para la selección de

técnicas los criterios de exactitud y flexibilidad. Pero en la presente investigación se consideró necesario una vez realizadas las pruebas, evaluar indicadores diferentes a los de la selección de técnicas, porque se quiere conocer nuevos contextos donde se exige mayor evaluación de las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting para determinar cuál de ellas es mejor que la otra, evaluando la eficiencia y utilizando métricas de precisión, sensibilidad, especificidad y exactitud.

Los programadores cuando crean aplicaciones web utilizan criterios simples de seguridad y otros no utilizan los criterios de seguridad porque solamente se encargan de escribir el código y ver el funcionamiento del sistema, y como las pruebas del funcionamiento lo realizan en un servidor local no tienen idea de que puede haber vulnerabilidades, y que posteriormente serán explotados por los atacantes cuando dicha aplicación web este en la web. Es recomendable poder usar las técnicas de detección de vulnerabilidades de ataques Cross Site Scripting seleccionadas para mejorar la seguridad de aplicaciones web, exactamente no va a garantizar la seguridad al 100% porque existen muchos ataques web como Inyecciones Sql, Cross Site Request Forgery, entre otros, pero ayudará a mejorar la seguridad de ataques de Cross Site Scripting. Dicho así, la presente investigación ayuda a los programadores que están creando aplicaciones web para vender a microempresas, ayudando a encontrar vulnerabilidades en dichas aplicaciones y así poder corregir el código y poder evitar ataques informáticos al servidor o al empresario.

1.1. Realidad Problemática.

Los problemas de la inseguridad de los sistemas de la información en la actualidad, tienen un alto incremento de inseguridad. El reporte de Background Check afirma que se detectó el 65% de incidentes en los sistemas información en los últimos años. (Cano, 2008) afirma que el esfuerzo y entendimiento de controlar la seguridad informática es muy complejo, llevando al punto de creer que lo tienen controlado, pero aparecen nuevas vulnerabilidades y complican todo. Esto lleva a la necesidad de desarrollar metodologías, técnicas y algoritmos

para reducir al mínimo los problemas de inseguridad de los sistemas de la información, debido a que muchos de estos llegan a ser desconocidos y traen consigo problemas. (Gómez V & Suárez R, 2006) comenta que existen variedades de formas de distinguir la seguridad, las cuales se deben evaluar según las necesidades de tales organizaciones, comprendiendo dinámicamente el negocio y limitar las características del producto.

Asimismo, en la actualidad la propagación del virus ransomware (secuestro de datos) es más eficiente por la influencia del internet, este ransomware actúa permaneciendo oculto en el ordenador sin ser detectado por el usuario, pero en realidad el ordenador se encuentra contagiado de virus. Mientras (ESET, 2018) afirma que el objetivo principal del virus ya no es infectar los ordenadores de los usuarios, sino robar información delicada como contraseñas, fotos, videos, información, etc. O también utilizando como una conexión de puerta abierta para realizar ataques a otros usuarios. De acuerdo con el autor mencionado, los virus han evolucionado y ahora también causan daño o perjuicio al comportamiento del sistema y por tanto de la organización. Para contrarrestar los virus se propusieron desde los comienzos de protección de sistemas los modelos, herramientas, técnicas, algoritmos y enfoques para hacerles frente a estas amenazas.

Cabe destacar que, “En 1977, el senador Ribicoff, propuso la iniciativa de Acta de Protección de los sistemas de cómputo federales, buscando definir ciberataques y recomendar sanciones por dicho delito, pero no prospero” (Voutssas, 2010). Esto quiere decir que la protección de los sistemas de cómputo se trabajaba de años posteriores. (Gupta et al, 2014), comenta que hoy en día los ciberataques se han convertido en un problema importante, esto a causa de la alta expansión y practica de explotación de sistemas informáticos, organizaciones y comunicaciones que dependen de tecnología. En la actualidad los ciberataques han tomado una medida fuerte en ataques a empresas para hacerse con la información de los usuarios, robo de dinero y entre otros, es por ello que Servicios, Soluciones y Consultoría de TI para Empresas (ICORP)

muestra una lista de ciberataques más sonados en la historia del hackeo informático

De acuerdo con los estudios presentados en servicios, soluciones y consultoría de TI para empresas 2017, muestran siete ciberataques más populares, el cual se robó información y dinero, demostrando que los ciberataques pueden llegar hacer muy peligrosos.

Tabla 1:

Ciberataques más famosos de la historia.

Víctima o Bug	Año de ataque	Objetivo de ataque y perdidas
Titan Rain *(Víctima)*	2004	- Infiltración a inteligencia militar y datos clasificados. - Espionaje e inhabilitación a diversas máquinas.
Google China *(Víctima)*	2009	- Robó propiedad intelectual de Google. - 30 compañía fueron objeto de este malware.
Epsilon *(Víctima)*	2011	- Tuvo pérdidas de 4 mil millones de dólares. - Su objetivo eran los correos electrónicos para hacer uso de éstas con fines criminales.
PlayStation Network *(Victima)*	2011	- Robo de información a 77 millones de cuentas de usuarios en 23 días. - Por tener un pésimo plan de seguridad se le multo ¼ de millones de libras. - Tuvo pérdidas de 140 millones.
Sony Pictures Entertainment *(Víctima)*	2014	- Información confidencial de empleados y guiones para próximas producciones fueron filtradas.

		- Invirtió 15 millones de dólares para hacerle frente a este ataque.
Yahoo *(Víctima)*	2012 – 2014	- Robo de información a 500 millones de sus usuarios.
Heartbleed *(Bug)*	2012 – 2014	- Tuvo un total del 17 por ciento de aplicaciones web que fueron afectados en todo el mundo.

Nota: Tomado de (Martínez, 2017)

La seguridad del sistema de información es una parte esencial de su concepción y desarrollo. (Voutssas, 2010), afirma que la seguridad de la información debe tomar medidas para prevenir a organizaciones que permitan proteger copias de respaldo de información teniendo en claro las la triada - CID. Sin embargo, el peligro de información siempre está presente en los sistemas de la información y más cuando se juntan las vulnerabilidades y amenazas. (Tarazona, 2006), da a conocer, las amenazas atacan desde distintas partes, externa o interna, siendo vinculadas con áreas de la organización, descubriendo vulnerabilidades que son debilidades en las tecnologías que manejan información confidencial.

Por otro lado (ESET & Welivesecurity, 2020), muestra el estado de empresas de Latinoamérica que emiten seguridad de información, el cual informa que la explotación de vulnerabilidades aumento a un 65% , esperándose dicho resultado, si analizamos que en el 2017 fue el año que hubo mayo incremento de vulnerabilidades, según a la información de common vulnerabilities and exposures. Anteriormente en el 2017 se reportó 14700 vulnerabilidades, contra las 6474 de 2016, esto significa que se duplico las vulnerabilidades; así mismo en el 2018 se reportó más de 15000 vulnerabilidades, pero en el 2019 disminuyo en un 12036, y en el 2020 aumento gravemente con 17090 vulnerabilidades, lo que indica que este año podría seguir

la misma línea. Dicho reporte demuestra que más de la mitad de empresas encuestadas se preocupan por este incidente, teniendo la variedad de amenazas que pueden emplearse para robar información valiosa, desde ataques externos hasta fraudes. Seguidamente se muestra los incidentes de seguridad infectados de códigos maliciosos en Latinoamérica, llegando a la conclusión que el 45% de personas utilizan códigos maliciosos para arrebatar información de organizaciones o empresas. Como se muestra en la figura 1, demostrando que Ecuador y Paraguay con 70% son los que tienen mayor porcentaje de infecciones, mientras que Panamá tiene el menor porcentaje de infecciones y Perú con 58%.

***Figura 1**:* Infecciones de robo de información de las empresas de Latinoamérica en el 2020. Fuente: (ESET & Welivesecurity, 2021)

En consecuencia, con el grafico anterior ESET da a conocer el porcentaje de organizaciones que tuvieron problemas por influencia de códigos maliciosos, mostrando que las pequeñas empresas tuvieron menos incidentes, mientras que las más grandes registraron más. Deduciendo que las empresas grandes llevan una preparación fuertemente alta en cuanto a detección de problemas de seguridad y la eficiencia de corregir a tiempo. En la siguiente figura se muestra los resultados del reporte de seguridad de incidentes al tipo de empresas mediante la utilización de códigos maliciosos.

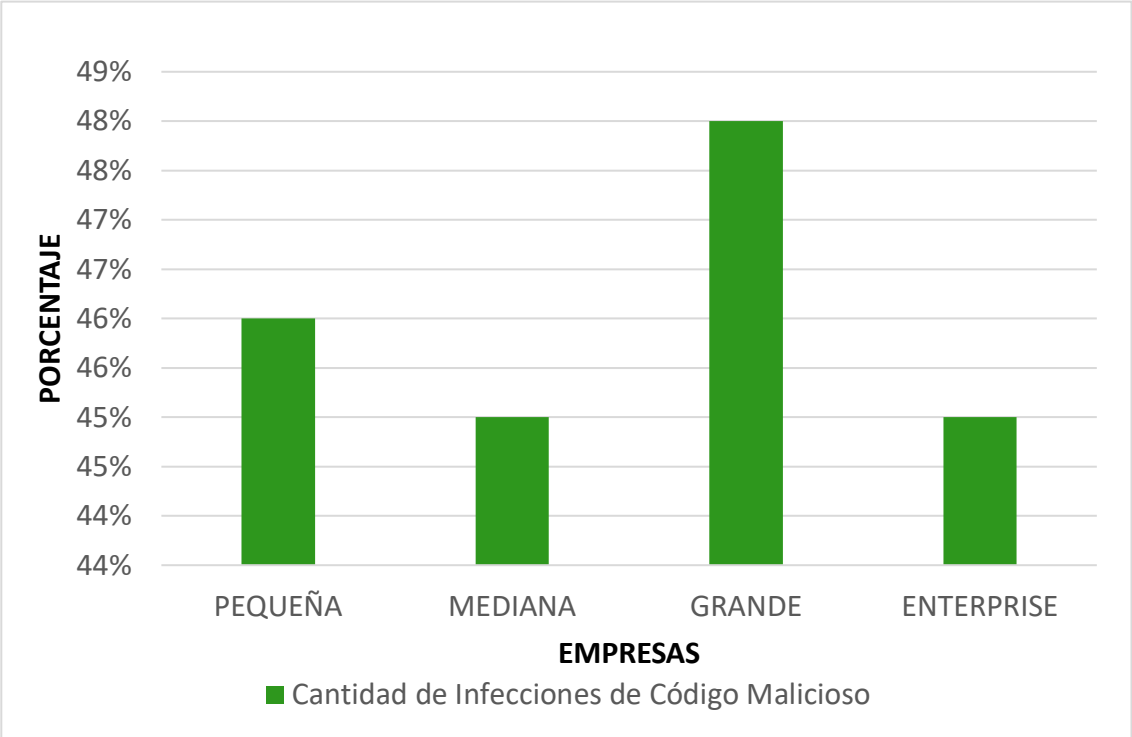

Figura 2: Porcentaje de incidentes al tipo de empresas por códigos maliciosos en el año 2019.

Alzahrani et al (2017), afirman “La mayoría de las empresas dependen del poder de los sitios web para interactuar con sus clientes y vender productos (…). De este modo, las aplicaciones web se han utilizado cada vez más para proporcionar servicios de seguridad críticos” (p, 237). Las empresas para agilizar procesos hacen uso de aplicaciones web y esto implica que la seguridad sea mayor para la protección de sus procesos, pero no siempre la protección es buena, también corren riesgos que pueden ser los ataques informáticos. (López

& Jorgue, 2016), deduce que los ataques informáticos consisten en el aprovechamiento de vulnerabilidades en hardware, software y en los usuarios que se encuentran en un área de informática, con el beneficio de generar ingresos económicos hacia el atacante, causando la burla de la seguridad del sistema, que más adelante afectan a los activos de las empresas. (Alzahrani et al, 2017), hace referencia que las empresas implementan aplicaciones web para su el uso comercial, pero que todos los sistemas no son seguros porque siempre se crean nuevas formas de ataques, es por eso que dicha seguridad de aplicaciones web es de vital importancia para contrarrestar dichos ataques y también para hacerles frente.

Por lo que (Pranathi et al, 2018), manifiestan que la seguridad en aplicaciones web lleva a un problema principal en la actualidad para diferentes asociaciones y comercio electrónico. La mayoría de asociaciones que utilizan la web (…) aseguran su información mediante firewalls y algunas técnicas de control de acceso". Sin embargo, dichas técnicas no siempre son seguras, por lo que existen puntos ciegos como vulnerabilidades en los sistemas que son aprovechados por los atacantes informáticos, estas vulnerabilidades se dan por las malas prácticas de los desarrolladores web. (U. Autonoma Mexico, 2019), expresa que escribir líneas de código seguras para aplicaciones web no es tarea fácil, porque se necesita que los desarrolladores cumplan con objetivos no solo básicos de seguridad si no altos, llevando a ideas generales de riesgos que corre la aplicación web en cuanto a su información que procesa. Asimismo (Householder, Houle, & Dougherty, 2002), comenta que existen muchas formas de hacer que la seguridad en aplicaciones web sean vulnerables con virus, códigos maliciosos, entre otros. El autor hace referencia que los códigos maliciosos es una manera compleja de vulnerar el sistema. En definitiva, los códigos maliciosos están diseñados para detectar o inventar vulnerabilidades en sistemas a las cuales se tiene acceso mediante puertas traseras para ser puente de conexión con otros sistemas para el robar de información o dañar el mismo sistema o archivos. (Garcia Alfaro and Navarro Arribas 2007), afirma que Cross Site Scripting es un ataque peligroso a nivel de aplicaciones web, ya que el atacante si es que tiene acceso al navegador de una persona o usuario puede

ejecutar scripts maliciosos, provocando la autorización de forma pasiva o activa para tener control de la información que contengan los identificadores asociados con los sitios web, así como las cookies. (Pranathi et al, 2018), argumenta que “Los ataques de XSS no son difíciles de encontrar y detectar, pero son difíciles de distinguir y contrarrestar”.

Durante el proceso de desarrollo de aplicaciones web, los programadores solo tienen en cuenta la funcionalidad de las aplicaciones, mas no la seguridad de dichas aplicaciones. De manera que los atacantes (terroristas, criminales, hackers, espías corporativos, entre otros) encuentran vulnerabilidades de ataques web, para que después puedan explotarlos y así poder llevar acciones malas contra las personas perjudicadas, y la información que diariamente viaja en una compañía, típicamente sensible y confidencial se ve afectada continuamente, en este caso para los empresarios que usan una aplicación web para sus labores.

En el Perú las empresas medianas y grandes tienen una inversión moderada para contratar empresas dedicadas al desarrollo de sistemas de aplicaciones web, y así dichas empresas puedan brindar un aplicativo web con alta seguridad, respaldando y protegiendo la información y evitando ataques web. Invierten mucho en la seguridad porque su organización insiste que la seguridad debe tratarse con cuidado, además conocen que en los últimos años la evolución de los atacantes es muy elevado, es por eso que exigen seguridad en su aplicación web. Pero en las microempresas, no muchos de ellos tienen inversiones moderadas para poder contratar empresas de desarrollo de sistemas de aplicaciones web, para que les brinden lo requerido. Las personas que forman su empresa inician con procesos básicos como procesos estratégicos y operativos, omitiendo el proceso de soporte porque se dan cuenta que tener tal proceso es tener una inversión moderada con lo cual no cuentan, porque su inversión actual da prioridad a los dos primero procesos, invirtiendo una cantidad baja para el proceso de soporte. Es por ello que acuden a contratar a personas que se encargan de crear aplicaciones web “Programadores” con un cobro bajo. Tales programadores inician creando el aplicativo web sin pensar en la seguridad

web, se encargan de escribir el código teniendo en mente los procesos y sub procesos que debe cumplir la aplicación web según los requerimientos que se le han entregado y así dejan de lado a la seguridad de la aplicación web.

Se realizo una encuesta hacia los programadores sobre la seguridad de aplicaciones web al momento de escribir el código fuente, con el fin de saber cuántos programadores desarrollan aplicaciones web pensando en la seguridad de dichos aplicativos. Se realizo la encuesta a programadores que son estudiantes y egresados de la Universidad Señor de Sipán (sede Chiclayo), Universidad Nacional de Cajamarca, Universidad Tecnológica del Perú (sede Chiclayo) y Universidad Privada del Norte (sede Cajamarca). Teniendo un total de 59 programadores.

En el **ANEXO 1** se muestra el modelo de encuesta y en las siguientes figuras se muestran los datos recolectados que fueron promediadas de las encuestas realizadas hacía los programadores.

Afirmando que la 95% de programadores desarrollan aplicaciones web para microempresas y 70% toman como prioridad los procesos que debe cumplir la aplicación web al momento de escribir el código fuente. También que todos los programadores no aplican reglas de prevención de vulnerabilidad en el código fuente de la aplicación web, asumiendo que no hay ningún programador que siga reglas de vulnerabilidades de ataques de Cross Site Scripting. Finalmente se muestra que el 90% no aplica pruebas de seguridad en aplicaciones web desarrolladas y que el 95% no atacan a su aplicación web para detectar donde está la vulnerabilidad.

Llegando a entender que el 92.7% de los programadores encuestados no toman como importancias la seguridad de aplicaciones web que desarrollan para vender a microempresas. Entonces se asume que los programadores no se preocupan en la seguridad de las aplicaciones que desarrollan para las microempresas, escribiendo el código fuente priorizando las funciones que les pide el usuario o empresario.

Es por ello que las microempresas cuando compran la aplicación web y posteriormente hacen uso de estas, tiene problemas de seguridad, pero los usuarios de la microempresa no tienen idea de cómo les pueden atacar tanto al usuario como a la empresa. Sin embargo, los ataques informáticos no se hacen siempre de una forma parcial, sino que a veces se realizan de manera indiscriminada. Es entonces cuando una microempresa se ve vulnerable a un problema de seguridad informática, que puede amenazar su negocio. El primer mecanismo de seguridad informática sobre estos ataques de Cross Site Scripting es el de la prevención. Lástima que en el Perú no suelen preocuparse en la seguridad informática, se preocupan en otro tipo de seguridad y gran parte de microempresas laboran al crecimiento de su negocio. Dichas empresas deberían empezar por las buenas prácticas para evitar que un virus o un ciberdelincuente entre en la aplicación web. Por ejemplo, hay que evitar abrir correos de origen desconocido, sobre todo con archivos adjuntos o que soliciten hacer clic en un enlace o descargar un programa. Las contraseñas deberían cambiarse periódicamente y hacerlo mediante algoritmos no predecibles. Por supuesto, se recomienda no instalar software que no tenga una procedencia completamente segura y legal. Con ello no solo protegen su información digitalizada, sino que además evitan sobrecarga en la conexión y todo funcionará más rápido. Pero la mayoría de microempresas no aplican dichas buenas prácticas, comentando que los problemas de seguridad de las aplicaciones web de microempresas, comienza desde el código fuente, y así produce fallos en la seguridad de las microempresas. El segundo problema es pensar que a nadie le interesa los datos que genera a diario la microempresa; existen empresarios que piensan que los hackers sólo se dedican a tratar de buscar agujeros de seguridad en los servidores y sistemas de grandes multinacionales. Pero no suele ser así. De hecho, las microempresas suelen ser más vulnerables porque invierten menos en la seguridad de su negocio y suele ser más fácil perjudicarlas, aunque no salga en los periódicos. Las empresas deben mantener el máximo de confidencialidad a nivel interno. El tercer problema es olvidar la gestión de la red informática; en muchos negocios se utilizan redes Wifi para acceder a Internet, pero no se han comprobado los niveles de seguridad de esta red. Probablemente, personal ajeno a la empresa pueda conectarse a nuestra red o

incluso interceptar transferencias de datos si sabe cómo hacerlo. La gestión de la red informática y su nivel de seguridad es clave. El cuarto problema es pensar reparar, no en mantener; muchas empresas sólo piensan en reparar los problemas informáticos cuando estos se producen. Pero, ¿y todo el tiempo que sucede entre incidencia e incidencia? ¿No es ésta la causa de muchos problemas informáticos? El mantenimiento debe ser diario y hay que ajustarse también a un código de buenas prácticas. Si no disponemos de un departamento de informática, podemos contratar servicios profesionales en mantenimiento informático a precio más económico. Finalmente, el quinto problema es la confianza en tener un antivirus y un firewall; lamentablemente, no es tan sencillo. Un antivirus y un firewall nos protegen en cierta medida de los problemas de seguridad más comunes relacionados con software espía, intrusos y malware conocido. Pero es necesario actualizar los antivirus, comprobar si existen agujeros de seguridad, ver si se está cumpliendo el código de buenas prácticas informáticas, etc.

Basados en su investigación los autores presentan una técnica llamada: sumidero - detección de vulnerabilidad “SDV” (Gupta et al, 2014), construyendo una herramienta para la detección de vulnerabilidades de ataques de Cross Site Scripting, utilizada en Kali Linux y escrita en el lenguaje de programación python, realizando una sensibilidad del contexto HTML y PHP basado en el análisis de Taint y programación defensiva para la detección de vulnerabilidades de ataques de Cross Site Scripting a partir del código fuente de las aplicaciones web. También proporcionan sugerencias automáticas para mejorar el código fuente vulnerable y ayudar a los programadores a mejorar el código. Se basan en la debilidad de programadores novatos al escribir código web. Los autores en los experimentos preliminares obtuvieron resultados contundentes, deduciendo la eficiencia de la técnica y comparando que es mucho mejor que las técnicas existentes. Resaltando que se hacen cargo de los falsos negativos (FN) y falsos positivos (FP), mientras las técnicas comparadas por los autores, RIPS y Pixy no se hacen cargo de los FN y FP.

Fundamentando la investigación los autores presentan la técnica: Detección dinámica de vulnerabilidades “DDV” (Hou & Chen, 2018), para detectar vulnerabilidades de ataques de Cross Site Scripting, basada en teorías existentes de vulnerabilidades de detección de caja negra. El proceso de detección dinámica contiene cinco pasos: crawler, construcción de características, simulación de ataques, detección de resultados y generación de informes. De acuerdo con la técnica DDV, se realiza una herramienta de detección, utilizada en Kali Linux y utilizando lenguaje de programación python3 para detectar vulnerabilidades en aplicaciones web. También realiza reglas de prevención de dichas vulnerabilidades, sugiriendo ciertos pasos para poder atacar la aplicación web y luego mejorar el código con reglas científicas y de OWASP. Las pruebas realizadas verifican los resultados y los comparan con otros resultados de pruebas de herramientas existentes, por lo que analizaron la usabilidad, sus ventajas y desventajas de la detección, confirmando la viabilidad de aplicar o utilizar la técnica DDV.

Como se mostró anteriormente en las investigaciones de técnicas de detección de vulnerabilidades desarrolladas en el trascurso de los años, mostraron resultados en donde las vulnerabilidades de aplicaciones web de las microempresas tienen impactos negativos, que fueron ocasionados por vulnerabilidades de ataques Cross Site Scripting tanto a servidores web y usuarios, el ataque Cross Site Scripting fue seleccionado para la investigación, por el impacto en las aplicaciones web que tiene en la actualidad. Así mismo las técnicas son buenas, pero se debe seleccionar cuál es la mejor técnica para la detección de vulnerabilidad. Donde los programadores se beneficien para la toma de decisiones en cuanto a la seguridad que deberían implementar para contrarrestar y prevenir aquellos ataques informáticos más frecuentes.

1.2. Trabajos previos.

Guo, Jin, and Zhang, (2015), realizaron la investigación, XSS Vulnerability Detection Using Optimized Attack Vector Repertory, desarrollada en Instituto de Tecnología de la Computación y en la Academia China de Ciencias. Abordaron

el problema de vulnerabilidades de XSS en las aplicaciones web, en las que demuestran que la mayoría de aplicaciones web padecen de vulnerabilidades de tipo XSS. Los autores proponen un método de detección de vulnerabilidad XSS utilizando un repertorio óptimo de vectores de ataques. Este método genera un repertorio de vectores de ataque automáticamente, optimiza el repertorio de vectores de ataque utilizando un modelo de optimización, y detecta las vulnerabilidades de XSS en aplicaciones web dinámicamente. Para optimizar el repertorio de vectores de ataque, se construye un modelo de optimización con un algoritmo de aprendizaje automático, reduciendo el tamaño del repertorio de vectores de ataque y mejorando la eficiencia de la detección de vulnerabilidades XSS. El método propuesto por los investigadores obtuvo una detección de 848 vulnerabilidades de XSS en 24 aplicaciones web del mundo real, debido a que detecta las vulnerabilidades de XSS. Por lo que llegaron a la conclusión de que el método tiene un buen rendimiento en la optimización del repertorio de vectores de ataque.

Soleimani, Hadavi, & Bagherdaei, (2018), en su investigación: WAVE: Black Box Detection of XSS, CSRF and Information Leakage Vulnerabilities, desarrollada en la, Universidad de Tecnología Malek-e-Ashtar. Enfrentaron el problema de vulnerabilidades de aplicaciones web de tipo XSS y Cross Site Request Forgery (CSRF), estas vulnerabilidades son una de las 10 vulnerabilidades críticas más importantes, en el cual demuestran que los atacantes pueden infiltrar XSS por vulnerabilidades de desarrollo de web y CSRF por la vulnerabilidad de la URL, mostrando que la fuga de información es alta en las aplicaciones web. Los autores propusieron un método de caja negra, primero analizaron el flujo de información entre los usuarios de la aplicación web bajo evaluación para extraer flujos potencialmente vulnerables. Luego, examinaron los flujos mediante el envío de solicitudes especiales para descubrir vulnerabilidades de XSS y CSRF. Además, discuten cualitativamente sobre el riesgo de las vulnerabilidades descubiertas con respecto al análisis del flujo como el origen de la vulnerabilidad. El método propuesto por los investigadores se implementa como un prototipo para detectar vulnerabilidades de aplicaciones web, denominada “WAVE”, por lo que llegaron a la conclusión que WAVE tiene

una tasa baja de falsos negativos, es decir, puede usarse de manera confiable para el análisis de seguridad de aplicaciones web.

Liu et al. (2016), en su investigación, A XSS vulnerability detection approach based on simulating browser behavior, desarrollada en Universidad de Tecnología de Beijing. Enfrentaron el problema de uno de los diez principales riesgos de seguridad de aplicaciones web más importantes, la vulnerabilidad de XSS, en el cual demuestran muchas aplicaciones web carecen de seguridad adecuada debido a la omisión de la validación de entrada necesaria en el proceso de desarrollo. Los autores proponen un método de detección dinámico basado en simular el comportamiento del navegador y diseña un rastreador web basado en un navegador sin cabeza, que puede interpretar el código JavaScript y recuperar el contenido Ajax para encontrar los puntos de inyección ocultos en las páginas con total consideración de las aplicaciones web que contienen scripts complejos en un entorno Web 2.0. Además, proporcionan un método más preciso para identificar la vulnerabilidad de XSS con los vectores de ataque XSS mediante el examen del comportamiento en tiempo de ejecución de la aplicación web y decide si existe la vulnerabilidad XSS con la prueba de caja negra. El método propuesto por los investigadores demuestra que el método funciona a un 85%. Deduciendo que el presente método, encontrará puntos de inyección ocultos, lo que ampliará la cobertura de los puntos de inyección e identificará la vulnerabilidad XSS en los puntos de inyección con mayor precisión.

Marashdih & Zaaba, (2017), en su investigación, Detection and removing cross site scripting vulnerability in PHP web application, desarrollada en la Universidad Sains Malaysia. Abordaron el problema de la vulnerabilidad de los scripts entre sitios XSS en el cual actúa como principales problemas de seguridad generalizados en sitios web, en el cual demuestran que la detección de vulnerabilidades de XSS es el primer paso para hacerles frente a dichas vulnerabilidades. Los autores proponen un enfoque que detecte y elimine completamente las vulnerabilidades de XSS del código fuente de PHP. Llevan a cabo dos experimentos para detectar y eliminar las vulnerabilidades de XSS reflejadas y almacenadas. El enfoque está basado en la detección de la

vulnerabilidad de XSS utilizando el algoritmo genético (GA) y eliminando la vulnerabilidad detectada del código fuente de PHP mediante el uso de HTMLPurifier biblioteca. El enfoque propuesto por los investigadores muestra que es capaz de realizar la detección y eliminación de la vulnerabilidad de XSS reflejada y almacenada de ambos experimentos. Sin embargo, los investigadores consideran que la etapa de eliminación de vulnerabilidades aún no es segura. Por lo que llegaron a la conclusión que se debe de mejorar la etapa de eliminación de vulnerabilidades de XSS, ya que existen muchas tecnologías web.

Choi et al. (2018), en su investigación, HXD: Hybrid XSS detection by using a headless browser, desarrollada en la Universidad de Sejong. Abordaron el problema de las vulnerabilidades de aplicaciones web más frecuente que se produce cuando una aplicación web implementa una validación de entrada insuficiente o salida de saneamiento, en el cual demuestran que los adversarios pueden usar XSS para entregar un script malicioso que lleva al secuestro de sesiones, el robo de credenciales y la escalada de privilegios. Los autores proponen un sistema de detección de XSS híbrido (HXD), un enfoque de detención XSS basado en una caja negra utilizada tanto en el análisis de cadenas estáticos como la representación dinámica del navegador. Extraen las URL de los registros web y las refinan como URL de entrada adecuadas. Por lo que, HXD no necesita rastrear o borrar las entradas de URL. Utilizando PhantimJS, un navegador sin cabeza para ejecutar un JavaScript y detectar fallas de XSS para que pueda detectar vulnerabilidades de XSS en marcos de JavaScript. Y, por último, el analizador estático de HXD utiliza un enfoque basado en el análisis de cadenas para acelerar la velocidad de detección. El sistema y enfoques propuestos por los investigadores realizaron evaluaciones HXD en las principales aplicaciones web del portal de internet de Corea “Never”. Por lo que llegaron a la conclusión de que la evaluación HXD tiene bajo falsos negativos y detecta fallas de XSS que otros detectores basados en cajas negras no los detectan.

Mohammadi et al. (2016), en su investigación: Automatic web security unit testing, desarrollada en el taller Internacional IEEE / ACM en Automatización de Pruebas de Software. Abordaron el problema de la integración de las pruebas de seguridad en el flujo de trabajo de los desarrolladores de software, que no solamente pueden ahorrar recursos para realizar pruebas de seguridad por separado, sino que también reduce el costo de la solución de la seguridad, en el cual demuestran que varias vulnerabilidades se detectan temprano en el ciclo de desarrollo. Los autores proponen un enfoque de pruebas automáticas de para la detección de XSS, causadas por la codificación incorrecta de datos no confiables, extrayendo automáticamente las funciones de codificación utilizadas en una aplicación web para desinfectar las entradas que no son de confianza y luego evalúan su efectividad al generar automáticamente la banda de ataques XSS. El enfoque propuesto por los investigadores evalúa la técnica, que pueda detectar vulnerabilidades de XSS desde los primeros días y que cubra de manera eficiente un tipo común de vulnerabilidad XSS. Por lo que llegaron a la conclusión que el enfoque garantiza la validación de entrada frente a inyección de línea de comando.

Ruse & Basu, (2013), en su investigación: Detecting cross-site scripting vulnerability using concolic testing, desarrollada en el Departamento de Ciencias de la Computación, Universidad del Estado de Iowa. Abordaron el problema de ataques de XSS en la web, ejecutándose secuencia de comandos malintencionada (desde una inyección inmediata o desde una fuente almacenada), en el cual demuestran que lo utilizan para el robo de información, para obtener acceso no autorizado a los recursos del usuario y del sistema. Los autores proponen una técnica de dos fases para detectar las vulnerabilidades de XSS y prevenir los ataques de XSS. La primera fase, traducen la aplicación web a un idioma para el cual están disponibles herramientas de prueba concólica desarrolladas recientemente. La traducción también identifica las variables de inicio y fin que se utilizaran para generar casos de prueba para determinar las dependencias de entrada / salida en la aplicación. Las dependencias indican vulnerabilidades en la aplicación que pueden explotarse potencialmente cuando la aplicación está desplegada. En la segunda fase, en función de las

dependencias de entrada / salidas determinadas en la primera fase, instrumentamos (automáticamente) adecuadamente el código de la aplicación mediante la inclusión de monitores. Los monitores verifican la explotación de vulnerabilidades en tiempo de ejecución. El método propuesto por los investigadores, identifica las vulnerabilidades XSS que se produce debido a la copia condicional (de entradas a salidas) y la construcción de entradas de cadena maliciosas a partir de la concatenación de entradas singularmente benignas. Por lo que llegaron a la conclusión que la efectividad del prototipo de implantación del Framework es buena, utilizando aplicaciones web JSO no triviales.

Baojiang, Baolian, & Tingting, (2014), en su investigación, Reverse analysis method of static XSS defect detection technique based on database query language, desarrollada en la Universidad de Beijing de Correos y Telecomunicaciones. Abordaron el problema de seguridad más común en aplicaciones web la vulnerabilidad de XS, en el cual demuestran la gravedad de la perdida de información en los usuarios. Los autores proponen un método de análisis estático de la detección de defectos XSS de la aplicación web java mediante el análisis inverso del flujo de datos. El método primero convierte el archivo JSP en un archivo Servlet, y luego utiliza el método de prueba simulado para generar llamadas para todo el código Java automáticamente para un análisis exhaustivo. Se originó a partir de los métodos donde puede ocurrir un defecto de seguridad XSS, Analizaron el flujo de datos a la inversa para detectar el defecto XSS al evaluar si puede ser introducido por la entrada del usuario sin filtro. Este método inverso reduce efectivamente las tareas de análisis que son necesarias en formas avanzadas. El método propuesto por los investigadores se probó mediante experimentos en proyectos web Java creados artificialmente con fallas XSS y algunos proyectos web Java de código abiertos. Dicho así, se llegó a una conclusión, el método no solo mejoró la eficiencia de detección, sino que también mejoró la precisión de detección para el defecto XSS.

Gupta et al. (2014), en su investigación, context-sensitive approach for precise detection of cross-site scripting vulnerabilities, desarrollada en el Instituto

Nacional de Tecnología de Malviya. Abordaron el problema de vulnerabilidad de aplicaciones web, mostrando que el 63% de aplicaciones web evaluadas son vulnerables, cada uno con promedio de 6 fallas no resueltas y que XSS son las vulnerabilidades más graves en sitios web, por lo que demuestran que el código fuente es la razón de la debilidad de aplicaciones web, haciendo entender que la vulnerabilidades se dan por las debilidades del lenguaje de programación, las validaciones de entrada incorrectas o al desconocimiento de las pautas de seguridad por parte de los desarrolladores. Los autores proponen un enfoque sensible al contexto HTML basado en análisis defensivo y programación defensiva para la detección precisa de la vulnerabilidad XSS a partir del código fuente de las aplicaciones web de PHP. También proponen sugerencias automáticas para mejorar el código fuente vulnerable. El enfoque propuesto por los investigadores encontró que la ignorancia de los datos de entrada utiliza resultados de contexto en resultados de detección falsos. Por lo que llegaron a la conclusión de que los experimentos preliminares y los resultados en sujetos de prueba muestran que el enfoque propuesto es más eficiente que los existentes.

Nguyen & Hwang, (2017), en su investigación, Large-Scale Detection of DOM-Based XSS Based on Publisher and Subscriber Model, desarrollada en la Universidad de Hongik,Corea. Abordaron el problema de vulnerabilidades de XSS basado en DOM, que afectan al código de script que se ejecuta en el navegador de los clientes, en el cual demuestran que las vulnerabilidades XSS basadas en DOM son mucho más difíciles de detectar que las clásicas. Los autores proponen una técnica de escaneo distribuido para rastrear aplicaciones web a gran escala y detectar y validar vulnerabilidades XSS basadas en DOM. El sistema rastreara la aplicación web moderna, incluido el evento Java Script y las técnicas modernas, mediante el uso de la técnica de enganche en los pasos de rastreo y exploración. Detallan 3 componentes, el primero Crawler, el sistema debe mantener una estructura de datos de acceso de alta velocidad, en el sistema, el espacio de URL es mantenido por redes clusters, que se escala fácilmente cuando aumenta el número de URL. El segundo componente Scanner, obtendrá la fuente para crear una URL sospechosa y enviará al

componente verificador al escáner que inyectará un script y esperará la carga de la página. El resultado se marcará con una ID que corresponda a un objeto fuente / receptor. Esta información mostró el seguimiento de la pila directo al origen en el que se verificará la respuesta que se manejará en la aplicación web. Y, por último, el componente Exploit and verifier, el que leerá la información de origen según la ID de la base de datos. El analizador de contexto es responsable de encontrar el contexto de la inyección utilizando la información taintSource y taintSink que generara un árbol de análisis HTML basado en la entrada dada, usando este árbol de análisis y la información de coloración precisa del carácter, determinara el contexto exacto en el que tiene lugar la inyección. Para que un exploit funcione, la cadena contaminada se reemplaza con un vector de ataque. El vector de ataque que se va a inyectar debe ajustarse al contexto en el que se va a inyectar el vector de ataque en una página web. La técnica propuesta por los investigadores muestra una novedosa arquitectura de sistema flexible. Por lo que llegaron a la conclusión que la eficacia de rastreo de vulnerabilidades de aplicaciones web es buena y recomendable ya que muy pocos analizadores de vulnerabilidad web pueden realmente lograr esto.

1.3. Teorías relacionadas al tema.

Seguridad de la información.

Rodolfo Godoy, (2014), en su libro comenta que el objetivo principal de la seguridad en la información es protegerlas de interrupciones, accesos, destrucciones no autorizadas y divulgaciones. Se entiende que la seguridad es el estado actual de sistemas o información personal, indicando que están libres de riesgos, peligros o daños. Entendiendo que el peligro es aquel que puede afectar al funcionamiento de tal información o los resultados obtenidos. El termino seguridad de la información para los autores, no se debe confundir con el termino seguridad informática, porque tiene funciones en el mundo informático, pero la información puede encargarse de ambas áreas. De tal manera implica implementar tácticas que protejan procesos teniendo como objetivo primordial los activos. (Cano, 2008), dichas tácticas tienen la función

principal de establecer control de seguridad, tecnología, política y procesos que detecten riesgos o amenazas que aumentan la explotabilidad de vulnerabilidades colocando en peligro los activos, es decir, ayuda a la protección de la información y sistemas en donde se encuentra el almacenamiento y sus activos de administrativos.

El autor, menciona que existen preámbulos que son básicos para la seguridad de la información que son: integridad, disponibilidad y confidencialidad, mostrándose en la siguiente figura.

Figura 3: Preámbulos para la protección de la seguridad de la información llamada TRIADA. Fuente: (López, 2015)

Confidencialidad

Rodolfo Godoy, (2014), en su libro comenta que garantizar la seguridad de la información de usuarios o empresas protegiéndolas de no divulgarlas sin su consentimiento de estas, se le llama confidencialidad. Esta garantía es posible por ciertas normas que están limitadas en la restricción de la información a personas no autorizadas.

Integridad

Seguidamente (Baojiang et al, 2014), en su investigación afirma que integridad se le denomina a la búsqueda de información modificada sin autorización de un usuario, llevando a la diferencia total con respecto integridad contextual de la Base de Datos (BD). Deduciendo que la integridad en la información es el mantener tal cual fue creada sin tener que modificarla o manipularla por otros usuarios o virus informáticos. Violar la integridad es afirmar que un usuario, procesos o programas realizan modificaciones o elimina información delicada, mostrando confiabilidad en el contenido e tal información, a menos que sea modificada por usuarios autorizados, registrando dicho procesos de modificación, y así asegurar la precisión

Disponibilidad

Rodolfo Godoy, (2014), Finalmente define a la disponibilidad como cualidades, condiciones y características de la información de otorgar permisos a usuarios, aplicaciones o procesos que hagan uso de ello. Afirmando como acceso a información y sistemas controlados por usuarios autorizados en la circunstancia que sean requeridos. Deben funcionar correctamente en los sistemas informáticos que lo utilicen para el almacenamiento y procesos de información, teniendo el control de la seguridad con el fin de protegerlos, y que la comunicación utilizada para acceder a ello debe funcionar correctamente. La máxima disponibilidad de los sistemas se requiere que este apto en cualquier momento, para evitar interrupción con los servicios, como el corte de energía, actualización del sistema y mal funcionamiento de hardware. Se garantiza la disponibilidad implicando prevenciones de ataques de desestimación de servicios.

Sin embargo (Costas. J, 2003), menciona sobre el concepto de disponibilidad que es muy común en el contexto donde la seguridad se aplica para proteger a la información, ya sea en las normativas vigentes que relacionan a la protección de información personal, ámbitos de proteger los datos, también

códigos que sigan normas para tener una buena práctica o la recomendación de gestionar seguridad de la información y de certificados que son prestigiosos, estás dos últimas se relacionan con las auditorias en sistemas de información. En los tres preámbulos mencionados también se acoplan el no repudio y autenticación de los sistemas de información. Llamándose CIDAN, mencionándose así por las iniciales de cada palabra. En conclusión, son cinco: disponibilidad, confidencialidad, integridad, no repudio y autenticación.

Autenticación

Costas Jesus, (2003), define que la autenticación son situaciones en las cuales se verifica sobre documentos que han sido elaborados o pertenecientes a quienes los documentos dicen. Llevados a aplicar verificaciones de identidades de usuarios, este conlleva a que el usuario aporte o afirme que tal persona es quien ha creado tal documento, a partir de tal afirmación se le acredita autorización única de creador del documento. De tal manera influye en los sistemas informáticos solo que se le llama inicio de sesión o login, teniendo que ingresar los parámetros requeridos que son dos: usuario y contraseña. También se define como la capacidad de afirmar que una persona estableció su comprobación de tal contenido de mensajes.

No Repudio

Seguidamente (Costas. J, 2003), nos da a conocer que irrenunciabilidad o no repudio es el servicio estrecho de seguridad que se relaciona con autenticación, permitiendo la participación de fracciones en las comunicaciones. La disimilitud con la autenticación es la comunicación que se establece entre las partes y el no repudio se establece en un tercero, de esta forma existe el No repudio en destino, tratándose que el receptor no debe negarse al recibir un mensaje porque el que envía posee una prueba de recepción. Probando que el destinatario a sido correctamente enviando y recibido, y así evitando total negatividad del receptor. En líneas generales existe prueba de que lo ha creado el receptor y es recibido por el emisor. Tal principio garantiza la seguridad de

información con llevan a tener la información protegida y segura, pero en los sistemas informáticos que también hay información, es necesario conocer cómo se protegen los sistemas informáticos.

Seguridad Informática

Por ello (Rodolfo. G, 2014), afirma “La informática permite la permanente vulnerabilidad de la información digital, es por ello que se habla la seguridad informática”. Llevando a la protección del almacenamiento, procesamiento y transformación de la información digital, esto a pesar de todas las medidas correspondientes. Es importante realizar una auditoría de seguridad informática para estar conforme con la seguridad en nuestro sistema. No obstante recalcar que existe seguridad en los mecanismos adaptando a casos particulares, los activos se encuentran en los equipos, aplicaciones, datos y comunicaciones. Lo anterior se pone en relevancia la importancia de la seguridad física, lo tangible en la informática, que es todo relatado a equipos informáticos tales como servidores, equipos en red y propósitos generales. Por otro lado, es importante la seguridad lógica, la parte intangible es referirse a diferentes software o aplicaciones que ayudan a ejecutar las tareas en cada uno de estos equipos. El autor considero las amenazas para cada una de las seguridad física y lógica.

Tabla 2:

Amenazas contra la seguridad lógica y física.

Amenazas a la seguridad física	Amenazas contra la seguridad lógica
- Robos. - Catástrofes naturales. - Fallos de suministro	- Troyanos, virus y scripts maliciosos. - Extravió de datos. - Ataques a los servicios ofrecidos por servidores.

Nota: Tomado de (Campetella, 2015)

Para accionar en contra de estas amenazas se puede reaccionar o prevenir, es por ello que dentro de la seguridad informática se encuentran diferentes tipos de seguridad. El primero de ellos es la seguridad pasiva, que solamente los mecanismos con lo que se trabajan sufren un ataque, pero se tiene una recuperación razonable. Seguidamente, el segundo tipo es la seguridad activa, que protege ataques informáticos por medio de adaptar planes de protección de activos. (Mourad, Otrok, & Ayoubi, 2011), dice que las amenazas informáticas atacan a muchos vínculos informáticos, los más comunes son las aplicaciones web ya que a diario suben aplicaciones web a la internet. Los programadores que crean las aplicaciones web no conocen las vulnerabilidades que pueden tener y esto se produce por no saber sobre la seguridad de aplicaciones web, dejando a simple vista del hacker las vulnerabilidades y así poder atacar las aplicaciones web fácilmente, Permitiendo extravió de la información de la empresa, del personal y usuarios.

Seguridad en aplicaciones Web

Asimismo (Asensio, 2014), define unos de los puntos críticos de la seguridad de aplicación web, las cuales vienen hacer las herramientas que se interrelacionan de manera directa con las personas o usuario. Es habitual oír sobre la deficiencia de sistemas de seguridad en los lenguajes de programación o servidores. Por otra parte, la cantidad de problemas localizados en los servicios web, no son a causa de dichas partes, pero si son de las malas prácticas que realizan los programadores o desarrolladores web. Los programadores tienen la costumbre de solo escribir líneas de código sin restringir parámetros de entrada, tan solo conformándose con el funcionamiento de la aplicación web. Los autores indican que se debe tener como requisito primordial el conocimiento de protección en los sitios donde es la información es confidencial. Householder et al. (2002), afirma que hablar de aplicaciones web es conocer sobre la seguridad de estas. A diario se enfrentan a ataques informáticos como los phishing, baiting, ataque DDoS, spyware, ransomware y malware que son los más frecuentes en la actualidad, estos ataques son intentos de destruir, modificar o eliminar para tener acceso a información sin tener autorización de los usuarios.

Ataques informáticos

De acuerdo con (Guamán, 2011), afirma que dialogar de ataques informáticos es hablar sobre el aprovechamiento de fallas o debilidades de aplicaciones y procesos, con el fin de tener un beneficio propio de condición económica, haciendo que el sistemas sea vulnerable a la seguridad de un sistema, que después se puede alargar hasta ataques a empresas o organizaciones. Utilizando scripts maliciosos para despistan algoritmos de seguridad, haciendo débil a sus estrategias de protección de información y sistemas. Es por ello que (Leonardo, 2014), afirma como intentos de robo de información, actuando de manera silenciosa, generado una red de ataques que les permita entradas hacia otros sistemas y atacando cuando el acceso se vulnerable, sin dejar pista alguna. Así se deduce (Gupta, Ranjan Singh, & Dixit, 2018), que son transparentes al ojo humano, que ni se percatan de alguna script maliciosos que haya sido inyectado en el pasado o en el presente. Estos ataques crecen y evolucionan constantemente por el nivel de seguridad que desarrollan empresas como Microsoft, Avast, etc. Para no permitir que sean atacados por scripts maliciosos, ya que si lo permiten pierden prestigio y clientes. A esto se les llama parches de seguridad que son instalados en las actualizaciones del software.

Tabla 3:

Tipos de daños de ataque o virus informáticos

TIPOS DE DAÑOS			
MENORES	**MODERADOS**	**MAYORES**	**SEVEROS**
Este tipo de virus borra varios programas, deduciendo que se puede	Estos virus se producen cuando la amenaza elimina información de un disco duro y sobrescribiendo tal disco. Pero esto se puede restablecer con	Los virus están ocultos en el sistema generando daños a placer sin ser detectados a tiempo, realizando perdidas de	Este daño es el más leve dándose cambios menores, pero son progresivos con el tiempo. Dándose el caso

instalar de nuevo los programas porque los daños son leves.	una copia de seguridad creada anteriormente, tomando mínimo media hora en restablecer lo perdido.	información y daños al ordenador.	que el usuario no percibe cuando la información ha sido modificada

Nota: Tomado de (Leonardo, 2014)

Categorias del ataque

Los ataques informáticos se definen como el aprovechamiento de vulnerabilidades de los sistemas y/o servidores que se encuentran en la red de datos, así mismo estos ataques siguen una serie categorías para completar sus objetivos:

Amutio Gómez, (2012), interpreta que la interrupción es la detección del ataque viene a ser inmediata, puesto que el impacto causado por los ataques llega a dejan un pequeño daño en alguna parte del sistema, esta categoría llega a afectar y atentar directamente contra la disponibilidad. Algunos daños se encuentran en esta categoría, por ejemplo: destrucción del disco duro y borrado de información. Seguidamente, la intercepción es el daño que se puede causar en esta categoría, es el acceso a la diferente información por parte de personas ajenas a la empresa y no autorizadas. La detección llega a ser un tanto difícil, no suele dejar huellas, para cualquier rastreo. Estos tipos de ataques son los que van en contra de la confidencialidad causando diversos daños, por ejemplo: copias ilícitas de programas, escucha en línea de datos. Continuando con la suplantación es el daño que puede llegar a causar es la sustitución de datos o de información, ocasionando diversos problemas entre los usuarios que se encuentran dentro.

La detección de estos es más difícil que los anteriores, son tomados como delitos de falsificación de información. Así mismo los ataques llegan a violar la autenticación. Se encuentran algunos, por ejemplo: la adulteración de mensajes

en la red, registrar incorrectos en la base de datos, etc. Por último, la modificación, como su nombre bien lo dice, es cuando la información o los datos han sido modificados sin permiso alguno, esto lleva a ocasionar la alteración de los datos para el beneficio de los usuarios que lo realizaron. Se encuentra en el rango de detección difícil, ya que los ataques llegan a atentar sobre la integridad de los datos. Dicho autor menciona que en la actualidad se han implementado bastantes técnicas que ayudan a la detección de ataques para así encontrar las vulnerabilidades de los sistemas y poder corregirlos a tiempo. Sin embargo (Larrieu, 2015), comenta que pese a la gran cantidad de técnicas y herramientas de seguridad informáticas que se implementan en las empresas, puede llegar a ocurrir diferentes incidentes, clasificados en algún tipo de ataque, los cuales atenten contra los objetivos de la seguridad ya explicados. De acuerdo a la categorización de los ataques, se puede rescatar que existen tipos, tanto activos como pasivos, los cuales son capaces de interrumpir el funcionamiento de sistemas, servidores, computadores, etc. El autor explica que también existen fases para hacerles frente a los ataques. Dichos ataques informáticos ejecutan pruebas de penetración para ello necesitan realizarlas por fases. Estas fases son buenas para los programadores, porque aseguran el sistema de una organización, para ello es necesario que cada prueba que se realice quede documentada de esta manera y así decir exactamente donde falla el sistema y justificar el trabajo.

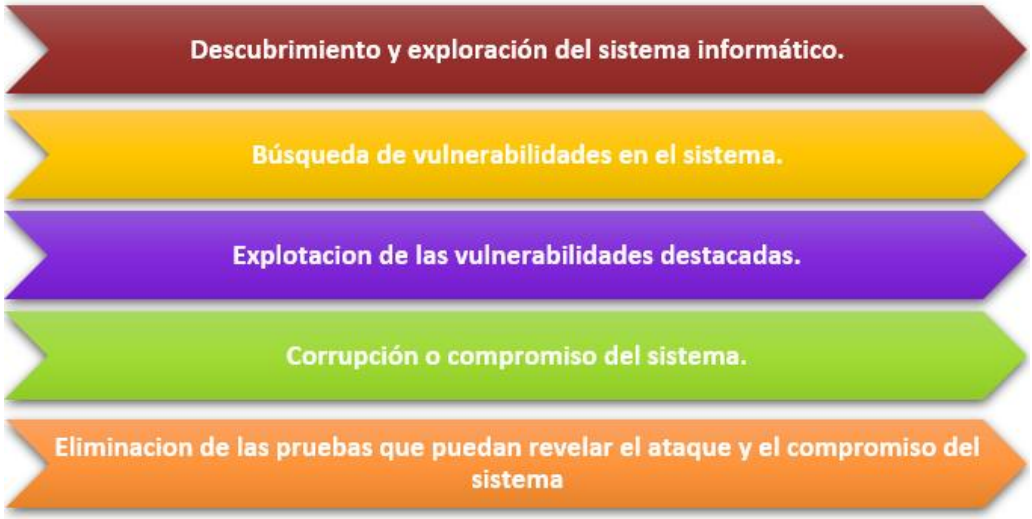

Figura 4: Cinco de las fases más comunes de los ataques informáticos. Enciclopedia de la Seguridad Informática. Fuente: Elaboración Propia

Ataques informáticos a Aplicaciones Web

En su articulo (Leonardo, 2014), deduce que las aplicaciones web proyectan una lista de dudas derivadas del desarreglo de mecanismos de autenticidad y la codificación inexacta. Afirmando que el producto de tales razones, hacen la creación de vulnerabilidades que son explotadas por lo hackers, ganando acceso a la parte de información (Base de Datos) que es controlada por la aplicación web que guarda información confidencial y valiosa, llegando a ser valoradas como punto de ataque para los hackers por los beneficios que obtienen al manejar la información que lo pueden utilizar como una amenaza.

Estos ataques se beneficiaron por el éxito que se tuvo gracias a los desarrolladores que se involucraron en crear a los botnets. (Prassi, 2018), Define una botnet como una red o conjunto de bots o robots piratas, que se pueden ejecutar de forma automática u autónoma. El ejecutor del botnet controla todos los servidores, aplicaciones web, ordenadores y celulares infectados de manera lejana o distante en el espacio o tiempo.

Aclarando que cualquier ordenador que tenga acceso a internet puede ser víctima de un botnet y así utilizarlos con el paso del tiempo sin ser detectados. El hacker tomará el control y podrá controlar acciones o hechos perversas a su beneficio. Así el hacker realiza también instalaciones de programas maliciosos como form grabbers, troyanos bancarios y keyloggers, para robar información o credenciales sensibles.

También el hacker podrá guiar ejecuciones de exploits o utilizarlos como red de conexión y así poder atacar a otros ordenadores. Y todo se produce a escondidas de los usuarios. Los autores mencionados muestran que se dan por la vulnerabilidad de aplicaciones web, es decir, contienen códigos de programación débiles, puertas traseras a la vista del atacante que son aprovechadas inmediatamente.

Vulnerabilidad de aplicaciones Web

Por lo que (Pranathi et al, 2018), explica que la aplicación web ejecuta códigos en el servidor del usuario, lo almacenan y acceden a él en la base de datos. Fuera de la oportunidad si su sitio tiene problemas de seguridad, las aplicaciones web son el lugar donde se descubrirán esos problemas. Estas son conocidas como vulnerabilidades o ataques. Estos ataques se interpretan como: inyecciones de SQL, Cross Site Scripting o XSS, inyección de XML, inyecciones XPath, inyecciones LDAP, inyección de C de byte nulo, Cross Site Request Forgery o CSRF y muchos otros problemas de infusión. Dichos ataques introducen códigos maliciosos.

Códigos Maliciosos

Chapple, Stewart, & Gibson, (2018), afirman que los códigos maliciosos son programas dañinos que ocasionan daños en los sistemas o en la información con el fin de acudir al beneficio de la persona que lo ha creado. Seguidamente (Inform & Mart, 2009), comenta que Darwin se llamó el primer juego informático siendo su función la creación de varios programas que disputaban la memoria de la PC, otorgando el premio ganador al que capturaría una gran cantidad de la memoria y así se podía eliminar a los otros oponentes. Deduciendo una clara competencia por ganar los recursos de la PC, se convirtió en una idea básica de ataque.

También comentan que en los inicios de estos virus cabía la posibilidad de darse cuenta que se sabía cuándo la PC era infectada porque los recursos disminuían contundentemente. (Pacheco, 2009), afirma que los desarrolladores de códigos maliciosos tienen motivaciones muy lejos de contemplar algo personal, ya que si se crea dichos ataques se obtiene beneficios altos, pero con delincuencia informática como secuestro de información, extorciones, etc. Y así se empezó a crear grupos que se dedican especialmente a introducir malware en las computadoras. Con el transcurso del tiempo se han creado muchos códigos maliciosos, clasificándolos en la siguiente tabla.

Tabla 4:

Variedad de códigos maliciosos identificados

Código malicioso	Definición
Virus	Vienen hacer una secuencia de códigos que intentan meterse en los sistemas sin el consentimiento de los usuarios, realizando acciones maliciosas.
Caballo de troya (troyano)	Software de computadora que simula una funcionalidad validad para el usuario, pero contiene código malicioso que evade la protección de los sistemas y alces explota acceso legítimo de un sistema.
Gusanos	Se propagan rápidamente con la finalidad de infectar a muchos equipos, llevando a consecuencias de interrupciones entre la comunicación de la red.
Bot	Scripts maliciosos que realizan manualmente. Refiriéndose a una computadora que ya fue afectada y así poder ejecutar instrucciones que el hacker desee.
Puertas traseras	Su función principal es modificar alguna función para permitir a alguien a tener acceso al sistema o equipo, pudiendo robar todo tipo de datos.
Riskware	Virus únicos, como prototipos que son administrados remotamente, conteniendo agujeros que son utilizados por los hackers para causa acciones que perjudican a un usuario.
Ransomware	Script malicioso que descifra información de una computadora, ingresando en ello para realizar advertencias a usuarios para que puedan recuperar lo hackeado. Tal usuario deberá cumplir con un monto de dinero si desea obtener la información

Phishing	Se centra en robar información de los usuarios, en ellos incluye cuentas bancarias, a través aplicaciones creadas por el hacker que hacen creer al usuario que son las verdaderas fuentes. En si hace el hacker clona aplicaciones que son utilizadas para personas que solamente ingresan sus datos sin ver la fuente de confianza.
Keylogger	Scripts que se encargan de guardar información que los usuarios ingresan desde su teclado, capturando todo tipo de actividad. Pero tales ataques tienen que estar instalados en los ordenadores de los usuarios.
Hijacker	Secuestran funciones del browser, permitiendo modificar la página principal y la de búsqueda mediante la red maliciosa, impidiendo que sean restauradas por un usuario.

Nota: Recuperado de (Pacheco Ortega, 2009)

Cabe señalar que los creadores de virus realizan una mezcla de clasificadores siendo difícil detectar tipos de software maliciosos comportándose de igual manera que los gusanos, pero con características similares a los virus. (Dunham & Honors, 2007), menciona cuales son las funciones más maliciosas; el robo de información, llegando hacerse con la información de las personas vulnerables a dichos ataques y optando por el manejo inapropiado del sistema para el beneficio del hacker. Seguidamente la función de correo pirateados o llamados spam son una forma de introducir malware por autorización del usuario si es que hace caso al correo que recibió, esta manera es la más fácil ya que solo se envía correo y se espera una respuesta de la víctima, bien sea ignorado o reenviado. (Kulkarni & Kaduskar, 2010), los problemas que ocasionan los virus pueden ser contrarrestados con antivirus o configuraciones expertas en el sistema. Po tal razón es necesario realizar un escaneo de la computadora para ver si ha se ha infectado, haciendo que los hackers realicen mejoras en cuanto a los virus para ocultarles bien en los sistemas o la red, pero aún existen algunas

singularidades que detectan si las computadoras están dañadas por dichos virus, las cuales se presentan a continuación.

Tabla 5:

Características para la identificar si las computadoras están infectadas de códigos maliciosos

Características	**Definición**
Disminución del rendimiento del equipo	Si un código malicioso se empieza a ejecutar utilizando ciertos recursos de la computadora como memoria y un procesamiento para identificar la existencia de malware si se visualiza que la computadora va más lenta.
Problemas en la red	La conexión de red se vuelve vulnerable y lento debido que los códigos maliciosos atacan por medio de ello o si transfieren datos los cuales demoran.
Aparición inesperada de publicidad	Los hackers envían anuncios mediante correos o links en las redes sociales, que sin necesidad de hacer clic aparecen publicidad en las computadoras, estas publicidades maliciosas son ocultadas en internet.
Pérdida inesperada de información	Las computadoras pueden ser controladas a la voluntad por el hacker haciendo que la información se mueva o elimine sin consentimiento de la persona dueña de la computadora.
Aparición de nuevos archivos no creados por el usuario	Los hackers crean archivos maliciosos en las computadoras como archivos trashes, viendo de tal manera un control de alteración de los datos de la computadora, siendo una infección de virus, llegando hasta perder información.
Desactivación del antivirus y otro software de seguridad	Los virus pueden meterse bien adentro del ordenador, ocultándose donde los antivirus no pueden detectarlos, llegando a la función y desactivar funciones de la seguridad de las computadoras permitiendo engallar al usuario que está bien seguro, pero en realidad está desprotegido.

Nota: Recuperado de (Dunham & Honors, 2007)

Vulnerabilidad Cross Site Scripting (XSS)

Alonso Cebrián et al. (2014), menciona que se le denomina vulnerable un sitio web de ataque de Cross Site Scripting cuando se envía al servidor web búsquedas, comentarios, actualizaciones, entre otro, las cuales se ve reflejadas como respuesta en otra web. Seguidamente para los autores el ataque es producto de la mala seguridad del sistema que se refleja claramente en sus vulnerabilidades comprometiendo la seguridad de sus clientes. Su función es inyectar scripts maliciosos de tipo JavaScript o HTML en un sitio, aplicación o sistema web, con el objetivo principal que la persona tenga que ejecutar el script inyectado en el preciso momento que la aplicación se ejecute. Esto funciona cuando se inyecta un script a tal aplicación web. Se puede inyectar script malicioso a través de cajas de texto, URL, botones o aceptando notificaciones de la web. Pero el usuario no se percata si hay un ataque porque se ejecuta de forma simultánea cuando se carga la interfaz de la aplicación web. Otros factores apuntan que Cross Site Scripting realice funciones en el navegador de forma ilegal llegando hasta generar fallos en los servidores. Pero se dice que lo último es difícil porque los scripts están escritos n JavaScript y HTML haciéndoles inmunes al lenguaje del servidor que es el SQL. (Pranathi et al. 2018), muestra que la mayoría de casos de ataques Cross Site Scripting toman la siguiente vista.

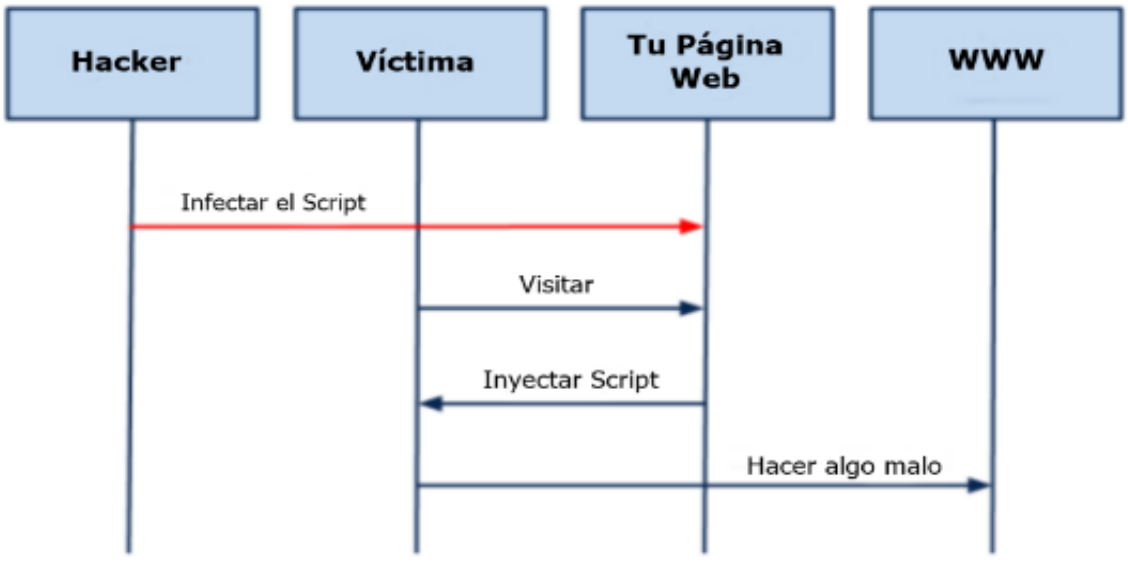

Figura 5: Una vista de alto nivel de un ataque XSS típico. Fuente: (Alzahrani et al. 2017)

También (Subramaniyaswamy, Gopireddy Venkata, & Naladala 2018), deduce que, con el SQL Inyección, el objetivo principal del hacker son las bases de datos, comúnmente sobre información empresarial que controla la aplicación web. Mientras que Cross Site Scripting atacan a otros usuarios que utilicen los servicios de dichas aplicaciones web, atacando por búsqueda de información que considera interesante. Puede ser en casos relevantes el usuario se dirija a una aplicación maliciosa o que pueda instalar programas secundarios en la computadora del usuario como ejemplo la actividad recolectada de toda la función que realice el maus durante un periodo de tiempo.

Cross Site Scripting de tipo Persistente (Almacenado)

Alonso Cebrián et al. (2014), distingue dos grandes categorías sobre los posibles fallos de Cross Site Scripting, el primero son los Cross Site Scripting persistentes (almacenados). Consisten en inyectar scripts de JavaScript o HTML peligrosos en aplicaciones web que poseen vulnerabilidades; de tal manera queda abierta a intentos de hacking y que son llamadas webs modificados. (Li, 2009), afirma que, si el script se encuentra con la funcionalidad de quedarse almacenado en un servidor web entonces se le llama persistentes. Funciona de manera que el usuario debe permitir el ingreso del ataque, ya sea un texto simple, alerte y notificación, con tan solo un clic se evita la seguridad y el hacker está en el navegador del usuario sin que se cuenta.

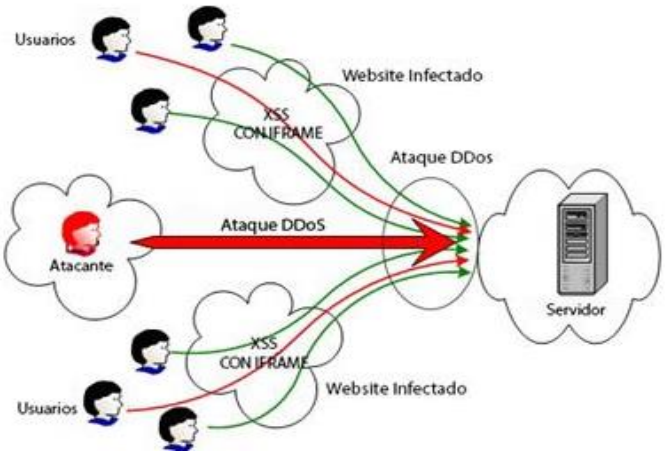

Figura 6: Efecto de ataque DDoS masivo con Cross Site Scripting en la red. Fuente: Fogie et al. (2007)

Muchos ataques DDoS están siendo generados desde páginas con vulnerabilidades de tipo Cross Site Scripting Persistentes, será ejecutado por todos los visitantes a través de su navegador. De esta manera se puede generar una gran cantidad de peticiones por segundo hacia el sitio atacado, incluido en el código JavaScript, y dependiendo del tráfico de visitas de la web que alberga el ataque Cross Site Scripting persistente, genera el efecto de DDoS masivo. En la siguiente figura se muestra como es el efecto de DDoS masivo al atacar con Cross Site Scripting.

Fogie et al. (2007), aprecia que el tipo de ataque puede ser muy dañino, ya que solo se necesita encontrar un sitio vulnerable a Cross Site Scripting persistentes con bastante tráfico y visitantes que mantengan la página abierta, para mantener el código JavaScript ejecutándose en su navegador. Los atacantes también tienen en cuenta, el tipo de petición que se realizará el código JavaScript insertado en a través del ataque Cross Site Scripting. En general, son ataques simples, pero eficaz cuando los atacantes se toman su tiempo y ponen empeño en conseguirlo. También tienen una gran ventaja para el atacante, ya que no requieren ningún tipo de infección en un ordenador de los usuarios, sino tan sólo código JavaScript en sitios vulnerables. A continuación, se nuestra cómo funciona el ataque de Cross Site Scripting Persistente (Almacenado).

Figura 7: XSS explotado de manera Persistente. Fuente: (Amutio Gómez, 2012)

Seguidamente (Alonso Cebrián et al. 2014), nos dicen que la segunda categoría son los No persistente (Reflejado), el cual radica en la modificación de datos que la aplicación web utiliza al enviar parámetros a una página de la misma. Un claro ejemplo es realizar funciones simples a las aplicaciones web, como utilizar el buscador para ejecutar el mensaje clásico del ataque, que se refleja en un cuadro de JavaScript. Con Cross Site Scripting reflejado el hacker puede hacerse con el uso de cookies, y así robar contraseñas, pero para esto la victima tiene que ejecutar unos comandos determinados dentro de la dirección de la web. (Mewara et al. 2015), comenta que los atacantes realizan envíos de correos que aparentan ser 100% confiables y así la víctima con tal solo hacer clic en el enlace del correo abre la puerta para que el hacker entre.

Este enlace engaña al usuario llevándole a un sitio confiable que normalmente el usuario tiene una cuenta abierta y en ese preciso momento el hacker ejecuta un script malicioso, intentando robar la información introducida en tal formulario de inicio de sesión o en las cookies de la sesión. Pero la diferencia abismal con Cross Site Scripting persistentes es que no se almacena nada en el servidor. Además, se construye la URL a voluntad para que pueda darse cuenta de no levantar alguna sospecha.

Figura 8: Tipo de vulnerabilidad Cross Site Scripting (XSS), explotado de manera No persistente (reflejado). Fuente: (Larrieu 2015)

Shrivastava, Choudhary, and Kumar, (2017), dice que es necesario “utilizar una herramienta de prueba de seguridad para detectar Cross Site Scripting”. Ellos utilizan Burp Suit, que viene hacer un software grafico para acreditar la seguridad de la aplicación web.

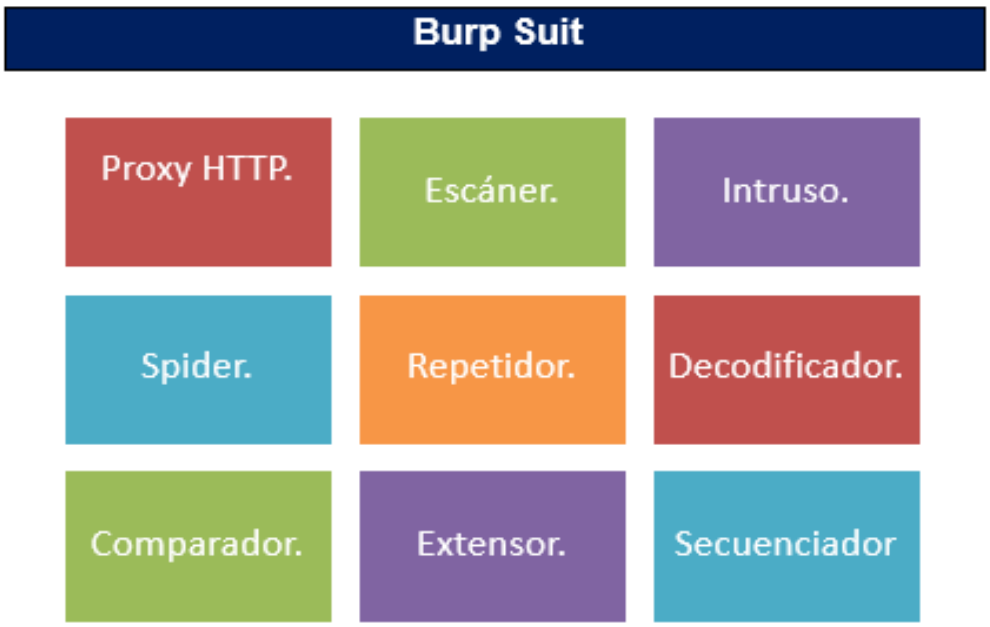

Figura 9: Funciones que proporciona la herramienta de prueba de seguridad Burp Suit. Fuente: Elaboración Propia

Navegador sin cabeza

Choi et al. (2018), para la ejecución de JavaScript para que pueda detectar vulnerabilidades de XSS en marcos de JavaScript, es necesario utilizar navegador sin cabeza. Es la navegación automatizada de Internet y dominios individuales utilizando un navegador sin cabeza, que es un navegador web sin interfaz gráfica de usuario. Incluye muchos enfoques y métodos para extraer, almacenar, analizar y procesar datos. Los sitios web, las aplicaciones web y las características individuales de las webs también pueden probarse y comprobarse automáticamente. El navegador sin cabeza incluye superposiciones temáticas con temas como la recuperación de información, data mining, el scraping y la automatización de pruebas. Los motores de búsqueda

pueden usar navegador sin cabeza para evaluar sitios web. En la medida en que el crawler simula una llamada a un sitio web con una interfaz no gráfica, los motores de búsqueda pueden sacar conclusiones de esta información y calificar los sitios web en función de su comportamiento en el navegador sin cabeza. Esta herramienta es muy útil para pruebas de ataques, ayudando de una manera muy eficaz al programador.

Pruebas de Software

Así mismo las pruebas ayudan mucho a las técnicas, enfoques y métodos a medir la complejidad de estas. (Sommerville, 2011), menciona que se llama caja blanca a pruebas de software realizadas en función interna de muchos módulos, que se encuentran coordinadas con sus funciones. Existen técnicas que realizan tal función interna son: pruebas de camino de datos, constatación de bucles, expresión lógica y aritmética. Siguen pasos para concretar su funcionalidad, primero se realiza las pruebas en los módulos concretos, después ingresa la prueba de caja negra, que ingresa a realizar sus funciones en la integración del software. Es por esto que las pruebas de caja blanca están intensamente enlazadas en una implementación en específica, por lo que, si se modifica, entonces las pruebas se modificarán o se rediseñarán. En la imagen que se encuentra a continuación, las personas que realizan test tendrán la oportunidad de contemplar la actividad de varios elementos, y se puede verificar la interfaz y los flujos con los que se comunican entre ellos.

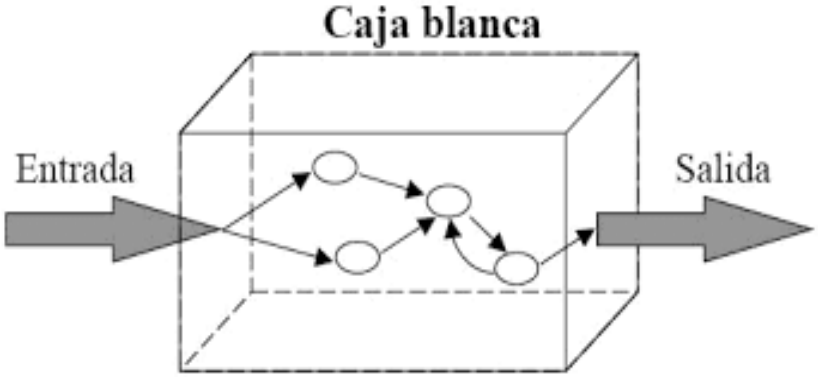

Figura 10: Funcionamiento de Prueba de Caja Blanca. Fuente: (Sommerville, 2011)

(Santos, Ordinez, and Eggly 2019), afirma que existe gran cantidad de investigaciones que amparan el trabajo de un analista de pruebas, que se incluye la solución en paquetes ya establecidas como JUnit y entre otras. Dejando en claro que depende del lenguaje de programación que se utilicé, además al desarrollar las pruebas, elaborar casos de pruebas, etc., con las herramientas facilitan un montón. De tal manera que se lleve los procesos ordenados y que sea lo más completo posible. Pero se sabe a qué nivel se puede aplicar. (Blanco Bueno 2015) dice que cada nivel es diferente en cuanto a criterios, ya que se aplica a cualquier nivel. Pero usualmente se aplica articular mete a una o muchas unidades funcionales, siendo su objetivo la verificación e inspección del comportamiento de los elementos en su inicio de integración del programa como tal. Después que se prueba dichos elementos independientemente, se realiza la prueba general, sabiendo que se ha pasado por la fase de integración, donde se comprobó la existencia de flujos de diferentes unidades funcionales. Siendo válido dicho criterio para un sistema completo buscando comprobar la comunicación del código con todos los subsistemas del proyecto.

Por su parte (Grabiel 2014), afirma que las pruebas de caja negra tienen como funcionalidad verificar sin tener en cuenta la disposición del código, posibles campos donde se ejecutan la funciones del software y el detalle de la creación. Se enfoca principalmente en la entrada y salida (as) sin la precaución de conocimientos de las funcionalidades internar del software, basándose en las especificaciones de función y los requerimientos del software.

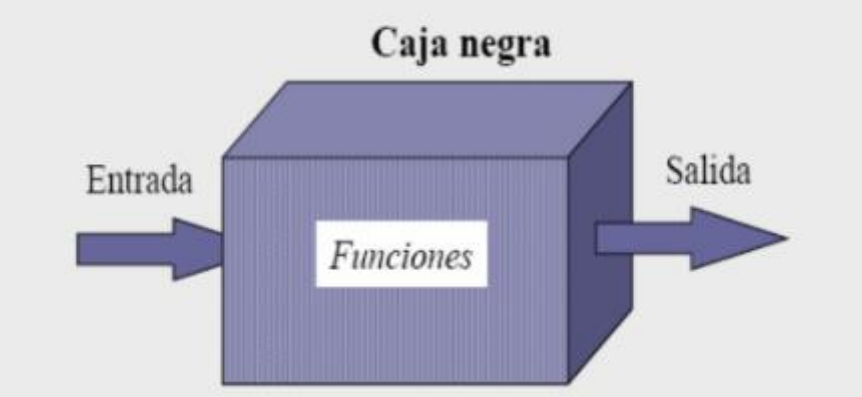

Figura 11: Funcionamiento de Prueba de Caja Negra. Fuente: (Santos et al. 2019)

Santos et al. (2019), asegura que, conociendo las funciones específicas, entonces se fuerza al diseño de pruebas demostrando que tal función se realiza de una buena manera a través de un panel de ejecución y la interfaz del software. Deduciendo que la función que más destaca la aplicación en base a la caja negra, proporciona entradas y salidas para revisar si coinciden con las pruebas esperadas. Mencionando que las fábricas realizan pruebas con empresas que se especializan en la confección de conjuntos de 'test' con el objetivo de que se evalué la calidad de las aplicaciones que se están probando. En gran parte de casos las empresas tienen solo aprobación de visualización del producto final, siendo así la inspección desde afuera. El interés de dicho punto de vista se asemeja al ámbito funcional, entendiendo la búsqueda de principios de mal funcionamiento en base a la caja blanca, deduciendo malas prácticas en el diseño. Si se inspecciona la conducta que la aplicación sea responsive, entonces se evalúa la salida de entradas en el campo de formularios de la aplicación web.

Crawler (Rastreador web)

El Crawler son normalmente robots, que realizan la función de inspeccionar aplicaciones web de manera metódica y automatizada. (Singh Ahuja, Singh Bal & Vernica 2014), afirman que muchos investigadores utilizan rastreadores web para obtener datos web. El rastreo web se puede utilizar en la web. campo minero para descubrir y extraer información automáticamente de la WWW. Así mismo comenta que un rastreador web es uno de los componentes principales de la búsqueda de motores web. El crecimiento del rastreador web está aumentando en el mismo a medida que crece la web. Una lista de URL está disponible con el rastreador web y cada URL se denomina semilla. Cada URL es visitado por el rastreador web. Identifica los diferentes hipervínculos de la página y los agrega a la lista de URL para visitar. Esta lista se denomina frontera de rastreo. Usando un conjunto de reglas y políticas, las URL en la frontera se visitan individualmente. Diferentes páginas de Internet son descargados por el analizador y el generador y almacenado en el sistema de base de datos de la búsqueda motor. Luego, las URL se colocan en la cola y luego se programa por el programador y se puede acceder uno a uno por el motor de búsqueda uno por

uno cuando sea necesario. Los enlaces y los archivos relacionados que se están buscando pueden estar disponibles siempre que sea necesario en un momento posterior de acuerdo con los requisitos. Con la ayuda de algoritmos adecuados, los rastreadores web encuentran los enlaces relevantes para los motores de búsqueda y utilícelos más. A continuación, se muestra el cómo funciona el Crawler.

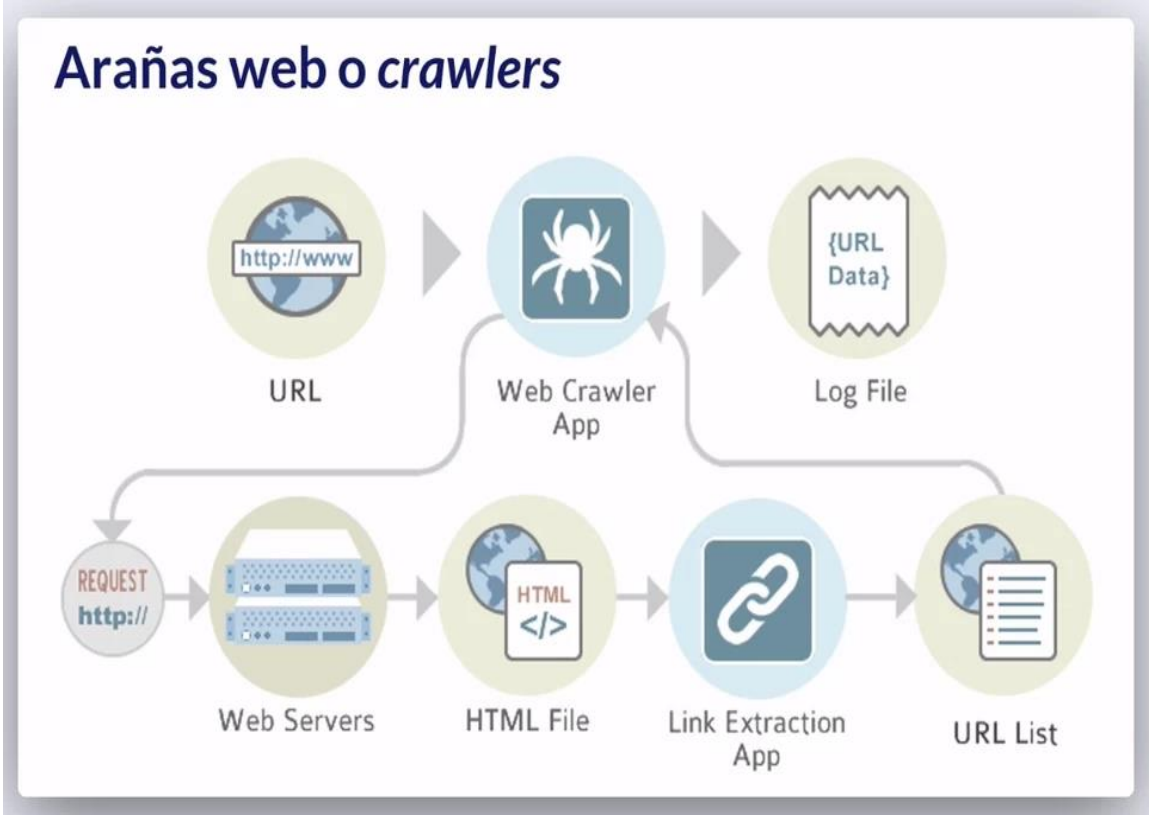

Figura 12: Esquema del funcionamiento del método Crawler para la búsqueda de URL en la herramienta DDV.

El Crawler cumple ciertos desafíos uno de ellos es Scale, el rastreador para lograr una amplia cobertura y un buen rendimiento, necesita dar un rendimiento muy alto. Esto llevó a la creación de un gran número de problemas basados en la ingeniería. Para superar estos problemas, las empresas necesitan emplear un gran número de computadoras que pueden contar hasta mil y casi una docena de enlaces de red de alta velocidad. Otro desafío bien hacer la compensación de selección de contenido, hay varios rastreadores que proporcionan un alto

rendimiento, pero no pueden rastrear toda la web y hacer frente a los cambios. El objetivo del rastreo es adquirir contenido con mayor valorar más rápidamente y recopilar información que contenga todo el contenido razonable. Los rastreadores deben ignorar todos los contenido irrelevante, redundante y malicioso.

Seguidamente también esta las obligaciones sociales, dicho rastreador debe seguir los mecanismos de seguridad para evitar un ataque de denegación de servicio. Los rastreadores deben establecer una buena coordinación con los diferentes sitios web para los que trabajan. Luego están los adversarios, existen proveedores de contenido que intentan inyectar inútiles contenido en el corpus del rastreador. Este tipo de actividades están motivados por incentivos financieros como desviar el tráfico a sitios web comerciales. Continuando con los desafíos está el Copyrightde, que viene hacer un rastreador aparentemente hacen algo ilegal: hacen permanente copias de material protegido por derechos de autor (páginas web) sin el propietario permiso. Los derechos de autor son quizás el problema legal más importante para motores de búsqueda. Este es un problema particular para Internet. Archive (http://www.archive.org/), que ha asumido el papel de almacenar y poner a disposición gratuitamente tantas páginas web como posible. También desafía la intimidad, para los rastreadores, el problema de la privacidad parece claro porque todo lo que hay en la web es de dominio público. Web La información aún puede invadir la privacidad si se usa en ciertas formas, principalmente cuando la información se agrega en un gran escalar en muchas páginas web.

Y, por último, el costo es un desafío grande, los rastreadores web pueden incurrir en costos para los propietarios de los sitios web. rastreadas utilizando su asignación de ancho de banda. Hay muchos diferentes hosts web, que proporcionan diferentes instalaciones de servidor, y cargando de diferentes formas. Diferentes hosts permiten una amplia variedad de ancho de banda. Las consecuencias de exceder el ancho de banda resultan en un exceso del costo a pagar para desactivar el sitio web

1.4. Formulación del Problema.

¿Qué técnica de detección de vulnerabilidad de tipo Cross Site Scripting, es más eficiente para la detección de vulnerabilidades en Aplicaciones Web de microempresas?

1.5. Justificación e importancia del estudio.

Las técnicas de detección de vulnerabilidades en aplicaciones web es un enfoque que aumentado su desarrollo en los últimos tiempos y probándose en el mundo, los científicos crean técnicas para la prevención de ataques y los programadores las emplean comprobando su validez y aprovechando sus beneficios. Las técnicas de detección mejoran las brechas de seguridad en aplicaciones web, haciendo que disminuya el robo de información, convirtiéndola en una de las medidas fuertes de seguridad de la empresa. Sin embargo, las técnicas brindadas no son implementadas por los programadores, llevando a que las aplicaciones web de microempresas sean más vulnerables al robo de información, por lo que la investigación resulta importante para comprobar si los beneficios propuestos por las técnicas de detención pueden menorar el robo de la información. El desarrollo científico debe ser aprovechado para el beneficio del ser humano, en este caso la seguridad de información de microempresas. Como se ha explicado hasta ahora las técnicas de detección prometen beneficios como protección y prevención. En este trabajo se propone utilizar las técnicas de detección desarrolladas anteriormente por científicos, para ser aprovechadas por las microempresas, generando la comprobación de la mejor técnica que puede ser simple pero muy eficaz para que los programadores puedan utilizarlas como mejora de seguridad de aplicación web.

1.6. Hipótesis.

La técnica dinámica de detección de vulnerabilidades (DDV) es más eficiente para detectar vulnerabilidades de ataque de Cross Site Scripting en aplicación web de microempresa.

1.7. Objetivos.

1.7.1. Objetivo general.

Comparar técnicas de detección de vulnerabilidades de ataques Cross Site Scripting en aplicaciones web de microempresas.

1.7.2. Objetivos específicos.

a) Identificar técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting a aplicaciones web de microempresas.
b) Implementar las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting a aplicaciones web de microempresas en el caso de estudio.
c) Evaluar las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting a aplicaciones web de microempresas.

c) Evaluar las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting a aplicaciones web de microempresas

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación.

2.1.1. Tipo de Estudio

El tipo de estudio para la presente investigación es Cuantitativo, dado que en esta investigación se realizará la comparación de técnicas de detección de vulnerabilidades de ataques Cross Site Scripting, con el fin de evaluar cuál de las técnicas es mejor detectando de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas.

2.1.2. Diseño de Investigación

En la presente investigación toma como experimental al diseño de investigación, porque utiliza establecer relación de efecto y causa de una situación, permitiendo contemplar el efecto ocasionado por la variable dependiente.

2.2. Población y muestra.

2.2.1. Población

Para la presente investigación se tomó cinco técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting, dichas técnicas fueron seleccionadas de los últimos años y han sido investigadas en artículos científicos, para ver la efectividad que tienen estas en la detección de vulnerabilidades de ataques Cross Site Scripting.

Cabe destacar que las técnicas seleccionadas encuentran vulnerabilidades en lenguaje de programación HTML y PHP, por lo que no encontrará vulnerabilidades en otros lenguajes de programación como JSP, JavaScript, entre otros.

Tabla 6:

Cinco técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting

	PAPER	CITA	TÉCNICA
Técnica N° 1	Improved N-gram approach for cross-site scripting detection in Online Social Network	(Wang et al. 2015)	N-gram mejorado modelo con ADTree clasificador
Técnica N° 2	CRAXweb: Automatic web application testing and attack generation	(Huang et al. 2013)	CRAXweb
Técnica N° 3	Automatic Web Security Unit Testing: XSS Vulnerability Detection	(Mohammadi et al. 2016a)	Unidad de Seguridad Web automática de pruebas
Técnica N° 4	A context-sensitive approach for precise detection of cross-site scripting vulnerabilities	(Gupta et al. 2014)	Sumidero - Detección de vulnerabilidad (SDV)
Técnica N° 5	A Dynamic Detection Technique for XSS Vulnerabilities	(Hou and Chen 2018)	Detección dinámica de vulnerabilidades (DDV)

Nota: Elaboración Propia

2.2.2. Muestra

Se estableció por conveniencia la muestra no estadística, con el fin de seleccionar las técnicas de detección de vulnerabilidades de ataques Cross Site Scripting en aplicaciones web de microempresas, estos estarán dentro

de la población previamente señalada, de los cuales tomaremos dos técnicas que han sido previamente estudiadas y validadas en los artículos, por lo que esto servirá para el desarrollo del proyecto de investigación.

Tabla 7:

Técnicas escogidas para la presente investigación de investigación.

N°	PAPER	CITA	TÉCNICA
Técnica N° 1	A context-sensitive approach for precise detection of cross-site scripting vulnerabilities.	(Gupta et al. 2014)	Sumidero - Detección de vulnerabilidad (SDV)
Técnica N° 2	A Dynamic Detection Technique for XSS Vulnerabilities	(Mokbal et al. 2019)	Detección dinámica de vulnerabilidades (DDV)

Nota: Elaboración Propia

2.3. Variables, Operacionalización.

La variable Dependiente viene hacer solamente una “Técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas” porque es muy limitada a toda la problemática que se ha expresado en la situación problemática y debido a que por tratarse de una comparación no existiría independiente, si no una sola para solucionar el problema. En las dimensiones se encuentra la “Aplicación Web de Microempresa” y su indicador es la “Características” porque se debe comparar que las características del caso de estudio estén alineadas a las características de las aplicaciones web de microempresas y así saber que las aplicaciones web que usan las técnicas seleccionadas en la presente investigación, son aplicaciones web de microempresas. La dimensión “Tiempo” y “Eficiencia” se utilizarán para medir a las técnicas seleccionadas.

Tabla 8:

Indicadores de la variable Dependiente

Variable Dependiente	**Dimensiones**	**Indicadores**	**Formula**	**Técnica e instrumento de recolección de datos**
Técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas	Aplicación Web de Microempresas	Características	Lenguaje de programación Funciones Procesos	Escala de licker / Ficha Técnica
	Tiempo	Tiempo disponible total de la herramienta	$Td = \frac{\sum_{i=1}^{n} Ci_1}{número\ de\ pruebas}$	Observación Científica / Ficha Técnica
		Tiempo de ejecución del proceso escáner	$T = \frac{\sum_{i=1}^{n} Ci_1}{número\ de\ pruebas}$	
	Eficiencia	Precisión	$PR = \frac{VP}{VP + FP}$	
		Sensibilidad	$SE = \frac{VP}{VP + FN}$	
		Especificidad	$ES = \frac{VN}{VN + FP}$	
		Exactitud	$EX = \frac{VP + VN}{VP + FP + FN + VN}$	

Nota: Elaboración Propia

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Las técnicas que se seleccionaron en la presente investigación para la recolección de los datos son:

Escala de Likert

Esta técnica de medición es utilizada por los investigadores con el objetivo de evaluar las características, opiniones y actitudes. Siendo una escala de calificación que se utiliza para cuestionar el nivel de acuerdo o desacuerdo con una declaración. Dicho método servirá para calificar si las características del caso de estudio se encuentran alineado con las características de las aplicaciones web de microempresas.

Observación Científica

Esta técnica consta de observar cómo se realizan las tareas y conocer con precisión las acciones que se realizan, además de capturar cada tarea, las actividades en el contexto de su realización. Dicha técnica se utilizará para observar el tiempo de indisponibilidad, precisión, sensibilidad, especificidad y exactitud que detectan las técnicas.

Observaremos cada técnica implementada al caso de estudio, sobre todo en el momento de revisar los resultados dados por cada técnica.

Ficha técnica

Este instrumento consta de un documento o ficha que sintetiza las características y funcionamiento de un subsistema o componente que tienen adecuada cantidad de datos que son utilizados por personas que diseñan componentes de un sistema. Dicho instrumento nos permitirá apuntar los resultados que se obtuvieron por cada técnica implementada en el caso de estudio.

A continuación, se muestra en que fases se utilizaran los instrumentos (Observación y ficha técnica).

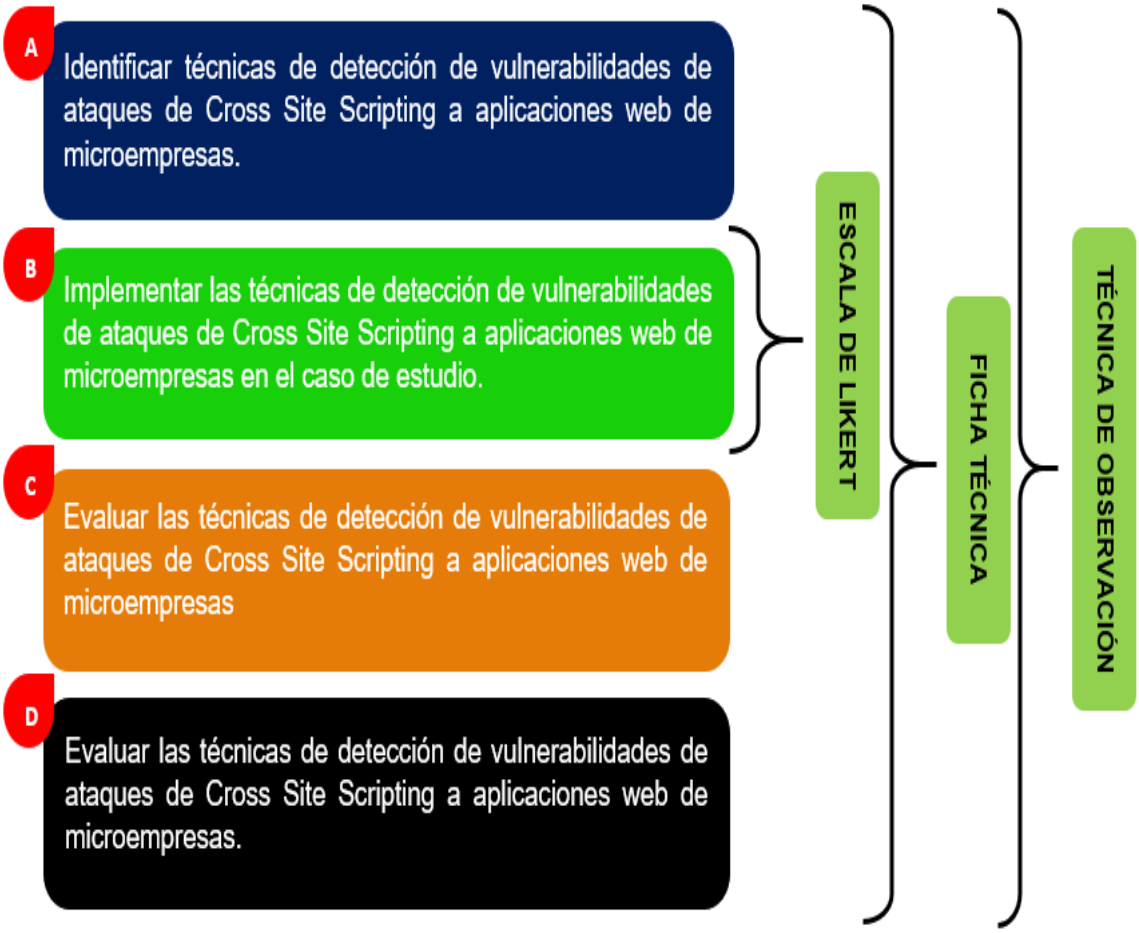

Figura 13: Señalización de donde se aplicarán las Técnicas e instrumentos de recolección de datos, validez y confiabilidad en los objetivos específicos; la técnica de escala de Likert, técnica de Observación Científica y Ficha Técnica. Fuente: Elaboración Propia

2.5. Procedimiento de análisis de datos.

Para analizar cada técnica de detección de vulnerabilidades de ataques Cross Site Scripting, se calculó los porcentajes de la disponibilidad, tiempo y eficiencia en la detección de dichos ataques. Los resultados de las pruebas que se realizaran se evaluaran utilizando las siguientes formulas:

Variable Dependiente: Técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas

Tiempo

1. Tiempo disponible total de la herramienta.

Tabla 9:

Análisis estadístico de Tiempo disponible total de la herramienta de las técnicas de detección de vulnerabilidades de ataques Cross Site Scripting.

$$\boldsymbol{Td = \frac{\sum_{i=1}^{n} Ci_1}{número\ de\ pruebas}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?
Td	Es el tiempo que la técnica de detección de vulnerabilidades demora en detectar vulnerabilidades de ataques de Cross Site Scripting en la aplicación web	Se obtiene del tiempo total que la técnica demora en encontrar las vulnerabilidades de aplicaciones web.
i = 1	Valor inicial o límite inferior.	Se obtiene la primera posición de la celda de los resultados del tiempo de ejecución
n	Valor final o límite superior.	Se obtiene la última posición de la celda de los resultados del tiempo de ejecución
Ci_1	Sumatoria, donde i toma los valores desde 1 hasta los valores n	Se obtiene de la suma de todos los valores de todas las posiciones de los resultados del tiempo de ejecución.

Nota: Elaboración Propia

2. Tiempo de ejecución del proceso escáner

Tabla 10:

Análisis estadístico de tiempo de ejecución del proceso escáner de las técnicas de detección de vulnerabilidades de ataques Cross Site Scripting.

$$\boldsymbol{Tr = \frac{\sum_{i=1}^{n} Ci_1}{número\ de\ pruebas}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?
T	Es el tiempo que la técnica de detección de vulnerabilidades demora en escanear a la aplicación web para poder detectar vulnerabilidades de ataques de Cross Site Scripting.	Se obtiene del tiempo que la técnica demora en escanear el código de la aplicación web para detectar vulnerabilidades en la aplicación web.
i = 1	Valor inicial o límite inferior.	Se obtiene la primera posición de la celda de los resultados del tiempo de respuesta
n	Valor final o límite superior.	Se obtiene la última posición de la celda de los resultados del tiempo de respuesta
Ci_1	Sumatoria, donde i toma los valores desde 1 hasta los valores n	Se obtiene de la suma de todos los valores de todas las posiciones de los resultados del tiempo de respuesta.

Nota: Elaboración Propia

Eficiencia

1. Precisión

Tabla 11:

Análisis estadístico de precisión.

$$\boldsymbol{PR = \frac{VP}{VP + FP}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?
VP	Son los verdaderos positivos del paquete de ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que son detectados como ataques por las técnicas de detección.
FP	Son los falsos positivos de los paquetes del ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que no son detectados como ataques por las técnicas de detección.
PR	Es la precisión con la que actúa la técnica de detección	Se obtiene de los verdaderos positivos (VP) sobre los verdaderos positivos (VP) más los falsos positivos (FP)

Nota: Elaboración Propia

2. Sensibilidad

Tabla 12:

Análisis estadístico de sensibilidad.

$$\boldsymbol{SE = \frac{VP}{VP + FN}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?

VP	Son los verdaderos positivos del paquete de ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que son detectados como ataques por las técnicas de detección.
FN	Son los falsos negativos de los paquetes del ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que son detectados como ataques por las técnicas de detección.
SE	Es la capacidad de las técnicas de detección para identificar resultados verdaderos positivos.	Se obtiene de los verdaderos positivos (VP) sobre los verdaderos positivos (VP) más los falsos negativos (FN)

Nota: Elaboración Propia

3. Especificidad

Tabla 13:

Análisis estadístico de especificidad.

$$\boldsymbol{ES = \frac{VN}{VN + FP}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?
VN	Son verdaderos negativos de los paquetes del ataque seleccionado, enviados por el tráfico normal.	Se obtienen de los paquetes que no son detectados como ataques por las técnicas de detección.
FP	Son los falsos positivos de los paquetes del ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que no son detectados como ataques por las técnicas de detección.

ES	Es la capacidad de los ataques de detección para identificar los resultados negativos	Se obtiene de los verdaderos negativos (VN) sobre los verdaderos negativos (VN) más los falsos positivos. (FN)

Nota: Elaboración Propia

4. Exactitud

Tabla 14:

Análisis estadístico de exactitud.

$$\boldsymbol{EX = \frac{VP + VN}{VP + FP + FN + VN}}$$

VARIABLE	¿Qué es?	¿Cómo se obtiene?
VP	Son verdaderos negativos de los paquetes del ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que son detectados como ataques por las técnicas de detección.
VN	Son los verdaderos negativos de los paquetes del ataque seleccionado, enviados por el tráfico normal.	Se obtienen de los paquetes que no son detectados como ataques por las técnicas de detección.
EP	Son los falsos positivos de los paquetes del ataque seleccionado, enviados por el tráfico malicioso.	Se obtienen de los paquetes que no son detectados como ataques por las técnicas de detección.
FN	Son falsos negativos de los paquetes del ataque seleccionado.	Se obtienen de los paquetes que son detectados como ataques por las técnicas de detección.

EX	Es el grado de cercanía de las mediciones de una cantidad al valor de la magnitud real.	Se obtiene de los verdaderos positivos (VP) más los verdaderos negativos (VN) sobre la suma de los falsos positivos (FP), verdaderos negativos, falsos negativos (FN) (VN) y verdaderos positivos (VP),

Nota: Elaboración Propia

2.6. Criterios éticos.

a) Veracidad: El ambiente en que se realizaron las pruebas sea objetivo y no subjetivo, donde los resultados fueron concordes a lo establecido.

b) Confidencialidad: La investigación guardó ética profesional al momento de la realización de pruebas, mantuvo en privacidad la información que contenían los servidores.

2.7. Criterios de Rigor Científico.

a. **Validez:** La Operacionalización de las variables tanto dependiente como independiente y sus dimensiones descritas se evaluaron mediante indicadores establecidos y las técnicas para recolectar datos.

b. **Fiabilidad:** La configuración de ambos servidores, las pruebas realizadas y de los resultados obtenidos se toman como referencia

III. RESULTADOS.

3.1. Resultados en Tablas y Figuras.

Se realizaron un total de seis pruebas, los resultados se muestran el tiempo que demora en detectar vulnerabilidades de ataques de Cross Site Scripting del caso de estudio, cuyos resultados fueron promediados y se clasificaron por medio de la matriz de confusión.

Resultados de la técnica de Detección Dinámica de Vulnerabilidades (DDV)

Se realizaron 3 pruebas para el caso de estudio, los cuales se obtuvieron resultados para los indicadores tiempo de ejecución del proceso escáner y el tiempo disponible total de la herramienta, interpretados en segundos. Se realizó la operación de cada indicador, que es realizar la sumatoria de cada dato y al total se le saco el promedio.

Tabla 15:

Promedio de tiempo de ejecución y respuesta en la detección de vulnerabilidades en el caso de estudio con la herramienta de la técnica DDV.

Indicador	Datos	Interpretación
Tiempo de ejecución del proceso escáner	51	ciencuenta y un segundos
Tiempo disponible total de la herramienta	53	cincuenta y tres segundos

Nota: De acuerdo con los datos obtenidos se lograron adecuar a las métricas de clasificador para la matriz de confusión. Fuente: Elaboración Propia

Tabla 16:

Matriz de confusión con los datos promediados de vulnerabilidades de ataques de Cross Site Scripting detectados con la herramienta de la técnica Detección Dinámica de Vulnerabilidades (DDV) en el caso de estudio.

		Datos de Herramienta de la técnica DDV	
		Detectado como vulnerabilidad	Detectado como no Vulnerabilidad
Datos reales	Vulnerabilidad	(VP)14	(FN)1
	No Vulnerabilidad	(FP)1	(VN)15

Nota: Como resultados del rendimiento se obtienen los porcentajes de las métricas de evaluación. Mostrando cada indicador con su operación y obteniendo el resultado en número entero. Fuente: Elaboración Propia

Se obtiene 14 vulnerabilidades como verdaderos positivos (VP) el número de predicción correcta de clase positiva, 1 vulnerabilidad como falsos positivos (FP) el número de predicciones incorrectas de clase positiva, 1 vulnerabilidad como falsos negativos (FN) número de predicciones incorrectas de clase negativa y 15 vulnerabilidades como verdaderos negativos (VN) número de predicciones correcta de clase negativa.

Dicho así, la técnica Detección Dinámica de Vulnerabilidades (DDV) con su herramienta para detectar vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas, encontró 14 vulnerabilidades (datos de la herramienta) en el caso de estudio, pero en realidad se encontró de manera manual (datos reales) 15 vulnerabilidades, es decir no detecto 1 vulnerabilidad. La herramienta de la técnica tiene un promedio alto para detectar vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas.

Tabla 17:

Métricas de evaluación de la eficiencia en la detección de vulnerabilidades de la técnica DDV

Indicador	Operación	Resultado
Precisión	$\frac{14}{14+1}$	93.33%
Sensibilidad	$\frac{14}{14+1}$	93.33%
Especificidad	$\frac{15}{15+1}$	93.75%
Exactitud	$\frac{14+15}{14+1+1+15}$	93.54%

Nota: Muestra de resultados efectivos por parte de la técnica DDV. Fuente: Elaboración Propia.

Resultados de la técnica Sumidero - Detección de Vulnerabilidades (SDV)

Se realizaron tres pruebas para el caso de estudio, los cuales se obtuvieron resultados para los indicadores tiempo de ejecución del proceso escáner y el tiempo disponible total de la herramienta, interpretados en segundos. Se realizó la operación de cada indicador, que es realizar la sumatoria de cada dato y al total se le saco el promedio.

Tabla 18:

Promedio de tiempo de ejecución en la detección de vulnerabilidades en el caso de estudio con la herramienta de la técnica SDV.

Indicador	Datos	Interpretación
Tiempo de ejecución del proceso escáner	28.3	veinte y ocho punto tres segundos
Tiempo disponible total de la herramienta	30.33	treinta punto treinta y tres segundos

Nota: De acuerdo con los datos obtenidos se lograron adecuar a las métricas de clasificador para la matriz de confusión. Fuente: Elaboración Propia.

Tabla 19:

Matriz de confusión con los datos promediados de vulnerabilidades de ataques de Cross Site Scripting detectados con la herramienta de la técnica Sumidero Detección de Vulnerabilidad (SDV) en el caso de estudio.

		Datos de Herramienta de la técnica SDV	
		Detectado como vulnerabilidad	Detectado como no Vulnerabilidad
Datos reales	Vulnerabilidad	(VP)3	(FN)1
	No Vulnerabilidad	(FP)12	(VN)15

Nota: Como resultados del rendimiento se obtienen los porcentajes de las métricas de evaluación. Mostrando cada indicador con su operación y obteniendo el resultado en número entero. Fuente: Elaboración Propia.

Se obtiene 3 vulnerabilidades como verdaderos positivos (VP) el número de predicción correcta de clase positiva, 12 vulnerabilidades como falsos positivos (FP) el número de predicciones incorrectas de clase positiva, 1 vulnerabilidad como falsos negativos (FN) número de predicciones incorrectas de clase negativa y 15 vulnerabilidades como verdaderos negativos (VN) número de predicciones correcta de clase negativa.

Dicho así, la técnica Sumidero Detección de Vulnerabilidad (SDV) con su herramienta para detectar vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas, encontró 3 vulnerabilidades (datos de la herramienta) en el caso de estudio, pero en realidad se encontró de manera manual (datos reales) 15 vulnerabilidades, es decir no detecto 12 vulnerabilidades. La herramienta de la técnica tiene un promedio bajo para detectar vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas.

Tabla 20:

Métricas de evaluación de la eficiencia en la detección de vulnerabilidades de la técnica SDV

Indicador	Operación	Resultado
Precisión	$\frac{3}{3+12}$	20.00%
Sensibilidad	$\frac{3}{3+1}$	75.00%
Especificidad	$\frac{15}{15+12}$	55.55%
Exactitud	$\frac{3+15}{3+12+1+15}$	58.06%

Nota: Muestra de resultados efectivos por parte de la técnica SDV. Fuente: Elaboración Propia.

3.2. Discusión de resultados.

Tiempo de ejecución del proceso escáner y tiempo de respuesta total de la herramienta

El tiempo de ejecución del proceso escáner de las herramientas de las técnicas DDV y SDV están relacionadas con el tiempo de respuesta total de la herramienta de detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web. Los resultados se obtuvieron del caso de estudio seleccionado. La herramienta de la técnica DDV se ejecutó con un tiempo mayor, utilizando una estrategia de rastreo Crawler y vectores de ataques.

El tiempo que la herramienta responde una vez detectada las vulnerabilidades se le llamó *tiempo disponible total de la herramienta*. Las pruebas se desarrollaron en el sistema operativo Kali Linux, para tener un mejor rendimiento y rapidez de detección de vulnerabilidades.

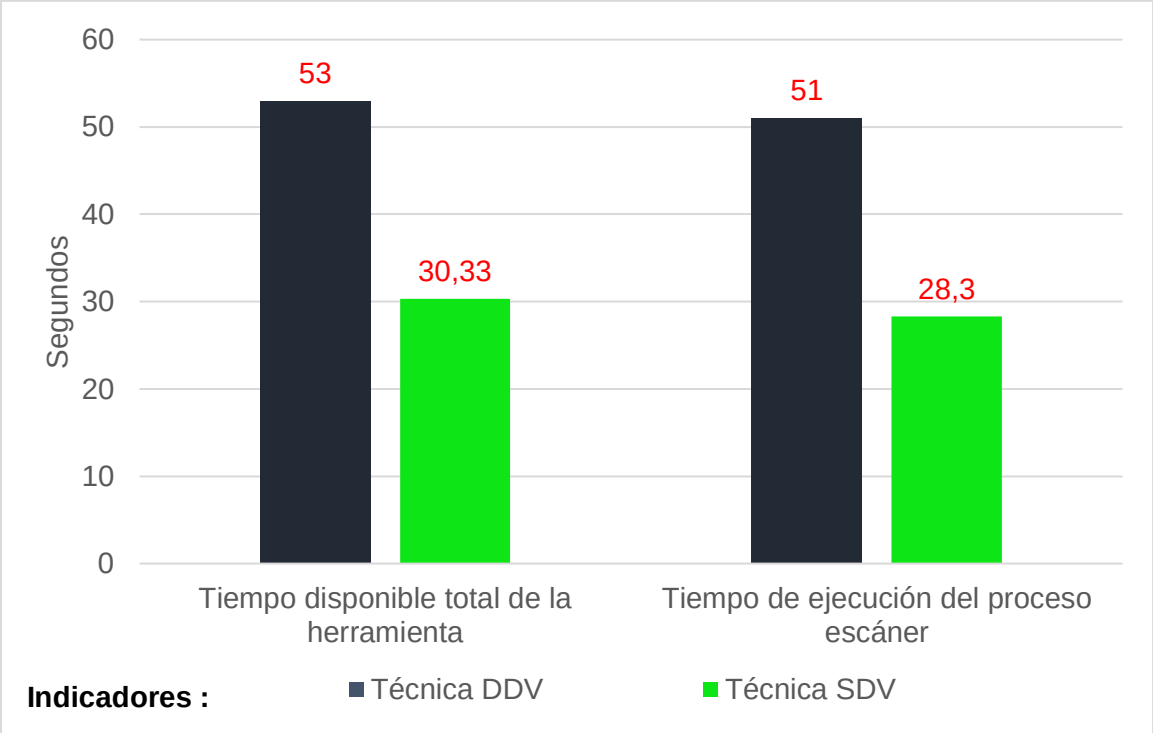

Figura 14: Resultado del tiempo en segundos según el escenario: tiempo de ejecución y tiempo de respuesta por la ejecución de las herramientas de las técnicas DDV y SDV en el caso de estudio. Fuente: Elaboración Propia.

Los resultados se obtuvieron en segundos por dos escenarios, el tiempo de ejecución del proceso escáner y tiempo de respuesta total de la herramienta ejecutada para cada aplicación web. Los datos muestran que el tiempo de ejecución del proceso escáner de la herramienta de la técnica SDV es menor que la herramienta de la técnica DDV, por 22.7 segundos de diferencia. Esto demuestra que la herramienta de la técnica SDV es más rápida en ejecutar sus procesos para detectar vulnerabilidades de ataques de Cross Site Scripting. En el tiempo de respuesta total de la herramienta también hay 22.7 segundos de diferencia, demostrando que la técnica DDV demora más en todo su proceso que la técnica SDV.

Promedio de vulnerabilidades

Los datos muestran que el tiempo de ejecución del proceso escáner de la herramienta de la técnica DDV es mayor que la herramienta de la técnica SDV.

Esto demuestra que la herramienta de la técnica DDV es más lenta en ejecutar sus procesos para detectar vulnerabilidades de ataques de Cross Site Scripting, pero detecta más vulnerabilidades que la técnica SDV.

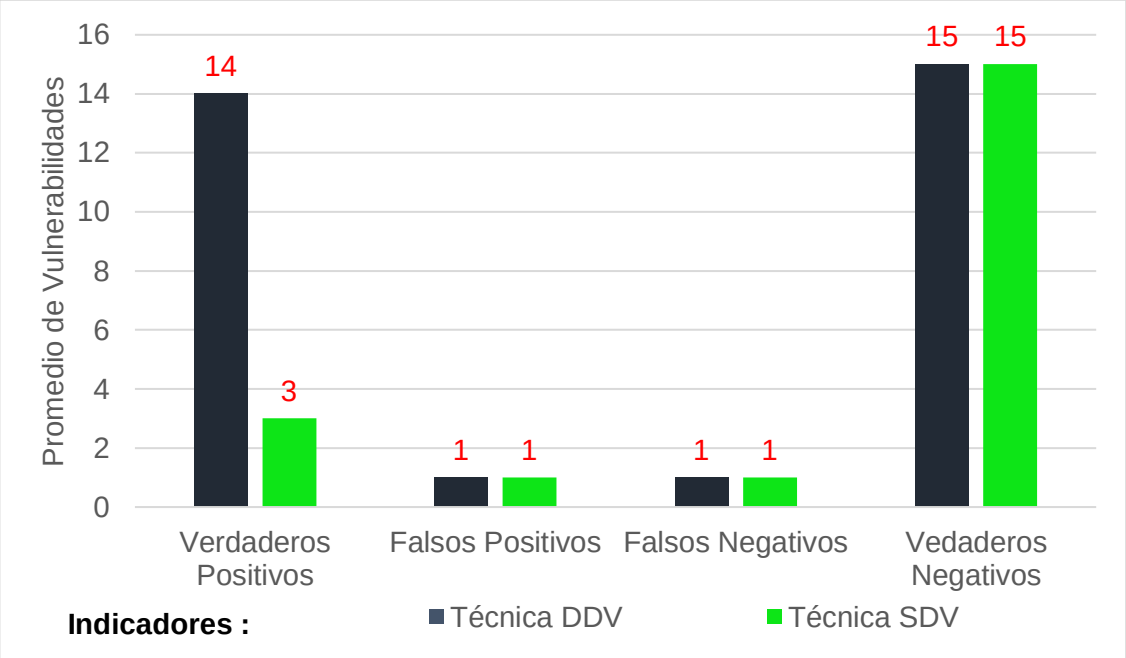

Figura 15: Resultado de los promedios de vulnerabilidades según las métricas de clasificación por la detección de vulnerabilidades de las herramientas de las técnicas DDV y SDV en el caso de estudio. Fuente: Elaboración Propia

Los datos demuestran que la herramienta de la técnica DDV detecta más vulnerabilidades que la herramienta de la técnica SDV por una diferencia de 12.06 en VP, también muestra una igualdad en cuanto a los VN, pero en los FP se lleva un 0.094 de diferencia y FN muestra una igualdad.

Indicadores de Eficiencia

Los resultados obtenidos fueron a partir de cuatro indicadores de eficiencia: precisión, sensibilidad, especificidad y exactitud. Con los datos obtenidos de detección de vulnerabilidades de las aplicaciones web seleccionadas. Con base

en ello se calculó la eficiencia de detección de vulnerabilidades de ataques de Cross Site Scripting

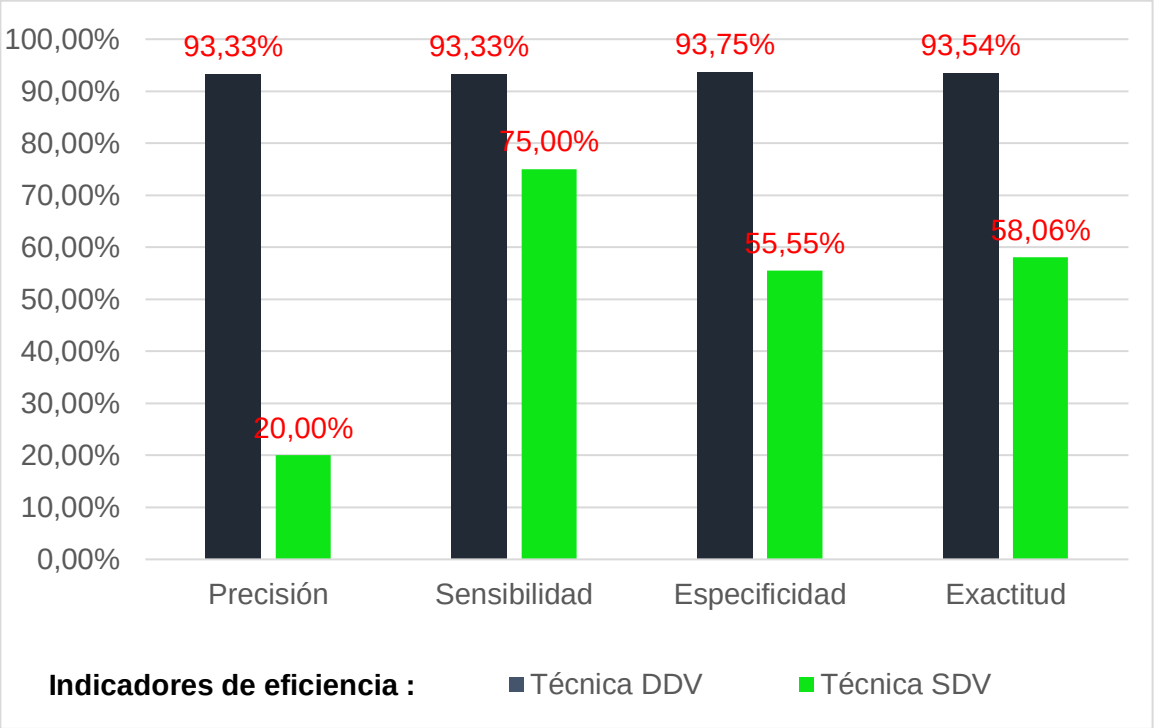

Figura 16: Resultado de los indicadores de eficiencia de la detección de vulnerabilidades de ataques Cross Site Scripting aplicados en el caso de estudio por medio de las técnicas DDV y SDV. Fuente: Elaboración Propia

Los datos demuestran que la técnica DDV es más precisa y exacta en detectar vulnerabilidades de ataques de Cross Site Scripting. Mientras que la capacidad de clasificar correctamente en la sensibilidad también es mejor en cuanto a la técnica SDV y de igual manera en la especificidad. Demostrando que la técnica DDV obtiene mejores resultados en detectar vulnerabilidades de ataques de Cross Site Scripting.

3.3. Aporte práctico.

Se completarán ciertos pasos para completar la presente investigación. Los pasos se detallarán en la siguiente arquitectura mostrando el primer objetivo específico hasta el último.

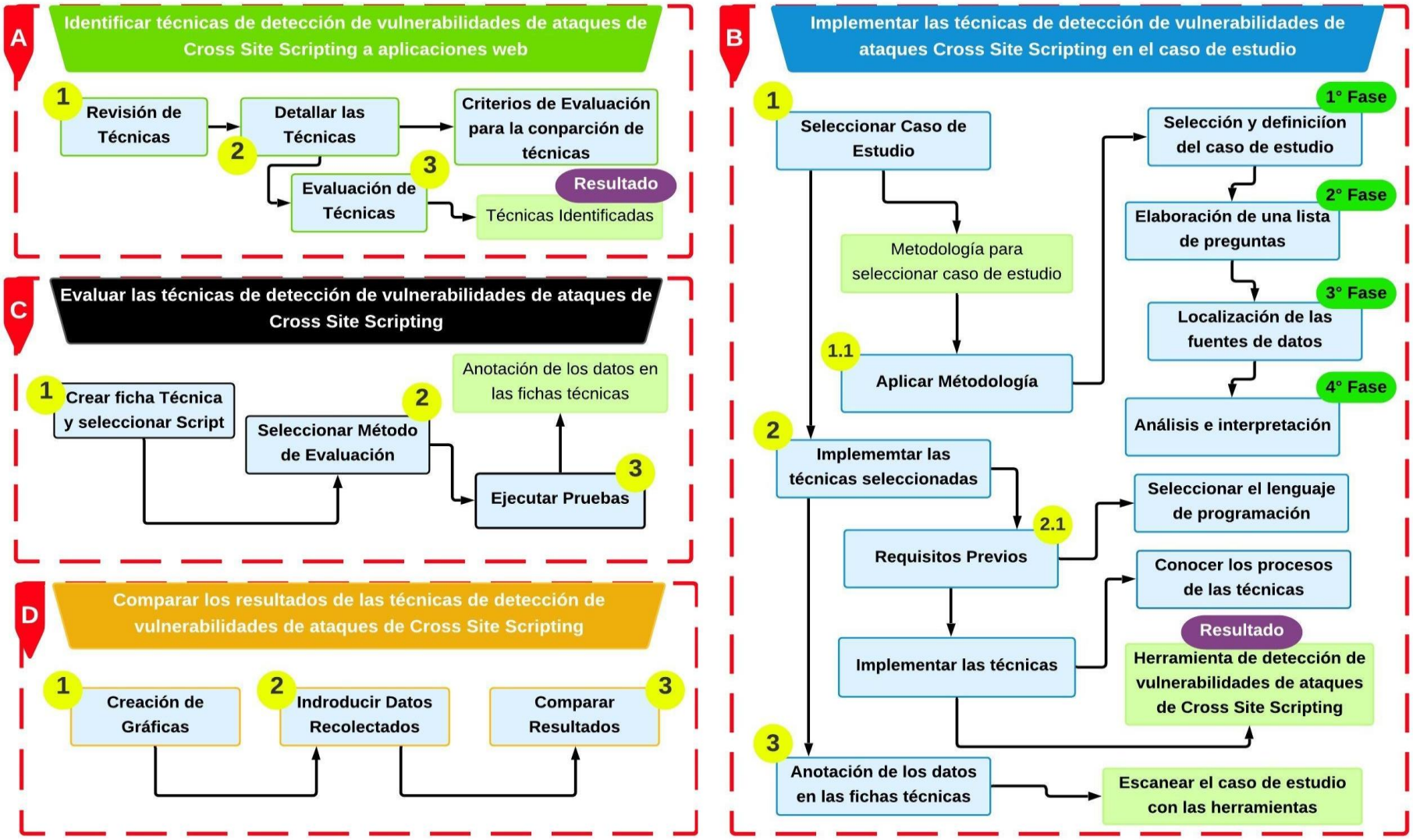

Figura 17: Proceso que se realizó para completar con los objetivos en la presente investigación. Fuente: Elaboración Propia

i. Identificar técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting a aplicaciones web.

1. Revisión de técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting.

Se realizó una revisión de investigaciones para encontrar técnicas de detección de vulnerabilidades de tipo Cross Site Scripting. La investigación se efectuó desde el año 2004 hacia el año 2019, se tomó recurrente el rango de dichas fechas por lo que en el año 2004 recién aparecieron propuestas de detección de vulnerabilidades de ataques Cross Site Scripting (Nirmal, Janet, & Kumar, 2018). Para realizar la revisión de técnicas se clasifico una base de datos en la cual se realizó búsquedas de las técnicas detección de vulnerabilidades de ataques de Cross Site Scripting, para ello se hace uso de la plataforma SCImago Journal & Country Rank (SJR), mostrándose en el **ANEXO *3*.** La segunda y tercera figura se muestra la búsqueda y resultados en la base de datos de IEEE.

2. Detallar técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting

Se encontraron cinco técnicas durante el periodo de búsqueda. En la siguiente tabla se detalla los datos principales de cada técnica.

Tabla 21:

Detalle de cada técnica de detección de vulnerabilidades de ataques de Cross Site Scripting.

N°	Técnica	Descripción	Año publicación	Autores
1	N-gram mejorado modelo con ADTree clasificador [a]	Técnica novedosa utilizando clasificadores y el modelo de N-gram	2015	- Rui Wang - Xiaoqi Jia - Qinlei Li - Daojuan Zhang

		mejorado para realizar la detección.		
2	CRAXweb [b]	Técnica para probar aplicaciones web y generar automáticamente los exploits factibles, incluidos ataques Cross Site Scripting.	2013	- Shih-Kun Huang - Han-Lin Lu - Wai-Meng Leong - Huan Liu
3	Unidad de Seguridad Web automática de pruebas [c]	Técnica de prueba automática para detectar un tipo común de vulnerabilidad Cross Site Scripting.	2016	- Mahmoud Mohammadi - Bill Chu - Heather Richter Lipford - Emerson Murphy-Hill
4	Sumidero - Detección de vulnerabilidad (SDV). [d]	Planteamiento defensivo de programación basada en HTML contexto sensible para la detección sensible para la detección precisa de la vulnerabilidad Cross Site Scripting a partir de la fuente (código) de las aplicaciones web.	2014	- Mukesh Kumar Gupta - Mahesh Chand Govil - Girdhari Singh
5	Detección dinámica de vulnerabilidades (DDV) [e]	Técnica dinámica de detección de vulnerabilidades de scripting entre sitios	2018	- Xin-Yu Hou - Mei-Jing Wu - Xiao-Lin Zhao - Rui Ma

	basada en las teorías existentes de la detección de vulnerabilidades de caja negra.	

Nota: Tomado de [a]Wang et al. (2015). [b]Huang et al. (2013). [c]Mohammadi et al. (2016). [d]Gupta et al. (2014). [e]Hou y Chen (2018)

3. Evaluación de técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting.

Para la evaluación de las técnicas se establecieron ciertos criterios de evaluación para obtener el mejor resultado y así poder trabajar con las técnicas mejor evaluadas. Para ello se necesitó ciertos criterios de evaluación, dando por hecho que en la investigación (Nguyen & Hwang, 2017), han probado la eficiencia de su evaluación proponiendo dos criterios.

Tabla 22:

Criterios de Evaluación para la comparación de técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting

	Criterios de evaluación	
	Exactitud	**Flexibilidad.**
Descripción	Establecer la ejecución de una acción de un software, tolerando como resultado la exactitud.	Amplitud o facilidad que las herramientas y prototipos puedan ser adoptados en otros servicios o productos.

Objetivo	Confirmación de la inexistencia del error o fallo.	El poder emplearse en distintos ámbitos o áreas de sistemas.

Nota: Elaboración (Nguyen & Hwang, 2017)

Seguidamente, al aplicarse los criterios de evaluación para las cinco técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting, se debe obtener resultados por cada técnica, pero antes se debe evaluar mediante la escala de cumplimiento que establece COBIT 5 (ISACA, 2013), estableciendo seis parámetros de cumplimiento mostrando ‘0’ como menor puntaje de cumplimiento y ‘5’ como máximo puntaje de cumplimiento. En la siguiente tabla se muestra con detalle.

Tabla 23:

Escala de cumplimiento planteado por COBTI 5

Número de escala	Definición de escala
0	No cumple
1	Cumple nivel muy bajo
2	Cumple en nivel medio bajo
3	Cumple agradablemente
4	Cumple en alta eficiencia
5	Cumple absolutamente

Nota: Elaboración (ISACA, 2013)

Dando a conocer los resultados dados por las escalas de COBIT 5, obteniendo que la técnica ‘Detección dinámica de vulnerabilidades (DDV)’ y la técnica ‘Sumidero – Detección de vulnerabilidad (SDV)’ tienen los resultados más altos con un puntaje de 8, demostrando que se puede trabajar con las 2 técnicas que superan el puntaje de 7. Así mismo, la técnica ‘N-gram mejorado modelo con ADTree’ obtiene el puntaje más bajo 5 y por tal motivo no se puede trabajar con

ello, la técnica ‘CRAXweb’ tuvo un puntaje de 6 y la técnica ‘Unidad de Seguridad Web automática de pruebas’ tuvo un puntaje de 7, demostrando que tampoco superan el rango establecido por las demás técnicas de alto puntaje y por tal motivo no se trabajará con ellas.

Finalmente se trabajarán con las 2 técnicas de mayor puntaje en el resultado para la detección de vulnerabilidades de ataques de Cross Site Scripting.

Tabla 24:

Evaluación y resultados de las técnicas de detección de vulnerabilidades de ataque de Cross Site Scripting.

N°	**Técnica**	**Criterios de evaluación**												**Resultado**
		Exactitud						**Flexibilidad**						
		0	1	2	3	4	5	0	1	2	3	4	5	
1	N-gram mejorado modelo con ADTree clasificador [a]					**X**			**X**					5
2	CRAXweb [b]				**X**						**X**			6
3	Unidad de Seguridad Web automática de pruebas [c]					**X**					**X**			7
4	Sumidero - Detección de vulnerabilidad (SDV) [d]					**X**						**X**		8
5	Detección dinámica de					**X**						**X**		8

	vulnerabilidades (DDV) [e]

Nota: Tomado de [a]Wang et al. (2015). [b]Huang et al. (2013). [c]Mohammadi et al. (2016). [d]Gupta et al. (2014). [e]Hou y Chen (2018)

ii. Implementar las técnicas de detención de vulnerabilidades de ataques Cross Site Scripting en el caso de estudio

1. Seleccionar caso de estudio

En la presente investigación se optó por utilizar un caso de estudio, basándose en la metodología de (Jiménez Chaves, 2012), brindando la información necesaria para trabajar con su propuesta de selección de caso de estudio para la investigación. La metodología consta de dos partes, la primera trata de utilizar pautas para la selección de estudio de caso, brindando tres argumentos para deducir que el caso de estudio brinda ayuda viable: se estudia el objetivo contextualizado a la investigación, realiza preguntas las cuales responde con: el porqué, el cómo.

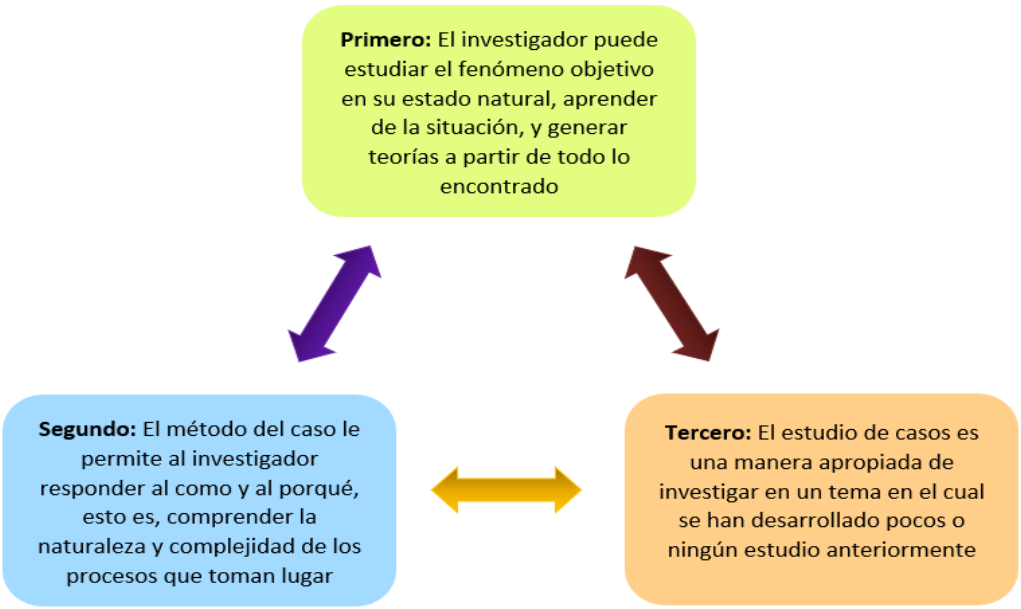

Figura 18: Criterios que se utiliza para entender con claridad por qué seleccionar caso de estudio como parte de la investigación. Fuente: (Jiménez Chaves, 2012)

Seguidamente la segunda parte, apunta a las características del estudio del caso de investigación, deduciendo que se estructura con pasos limitados haciéndole difícil de desarrollar, pero presenta una propuesta de otro actor (Montero, 2002), la investigación, Classification and description of research methodologies, en la cual desarrolla un método fácil de analizar, pero muy fuerte en su análisis de selección y desarrollo, dividida en cinco fases.

La primera fase es definir y escoger el caso de estudio; empezando por escoger el caso de estudio correcto y sobre todo definirlo, identificado el ámbito relevante que se necesita para la presente investigación, los resultados de la información recolectada y sobre todo que cumpla con los objetivos de la investigación. La segunda fase es plantear una pequeña lista de interrogantes; previamente indicado el problema, es primordial realizar un conjunto de interrogantes para ayudar al investigador.

Concluyendo con las primeras fases, es fundamental plantear la interrogante general y así desglosarla en interrogantes más variadas. Seguidamente la tercera fase, es localizar fuente de datos; en dicho apartado se escogerán algunas estrategias que será útiles para capturar dato, es decir, las observaciones, entrevistas, el caso que se examinaran, entre otras. Continuando con la cuarta fase, es en análisis e implementación; se basa en la etapa que tiene como objetivo principal la sensibilidad del caso de estudio.

El deduce que el objetivo debe corresponder a la información que ha capturado en el transcurso de la fase de terreno y también se debe establecer relación causa y efecto en las posibles observaciones. Siendo contrario a las fases de recolección de datos y la de diseño, dicho análisis es muy poco sujeto a las metodologías de trabajo, siendo relativamente difícil. Luego de definir la correlación de los personajes, situaciones y tareas, entre otros, del análisis por el presente investigador; con posibilidad de adquirir la propuesta hacía el caso de estudio.

Finalmente, la última fase es la elaboración del informe: que básicamente es el resultado de las fases anteriores, mostrando de manera ordenada y describiendo los eventos realizados y destacando los más relevantes. Por lo tanto, se debe explicar la obtención de los datos de información, las interrogantes, entre otros. Demostrando hacía los lectores que las fases son correctas y de bueno uso para la selección del caso de estudio.

Figura 19: Fases para la selección del caso de estudio para la investigación. Fuente: (Sommerville 2011)

1.1. Primera Fase: La selección y definición del caso.

Para la investigación presente se requiere de un espacio donde se pueda realizar pruebas, en este caso una aplicación web, para poder implementar las dos técnicas de detección de vulnerabilidades de ataques Cross Site Scripting. Se utilizará la metodología anteriormente mencionada para llevar a cabo la selección del caso de estudio. El entorno que es notable en la investigación es la necesidad de contar con una aplicación web que contenga muchas vulnerabilidades de ataques de Cross Site Scripting para realizar pruebas necesarias y así poder corroborar que las técnicas son útiles. Los autores de las técnicas que se compararan en la presente investigación, utilizan para sus

pruebas una aplicación web vulnerable para sus pruebas. De tal manera, se buscó una aplicación web vulnerable que este activa en el presente año 2021, encontrando la aplicación web “testphp.vulnweb.com/”. A continuación, se muestra en la tabla la aplicación web que se utilizará.

Tabla 25:

Caso de estudio seleccionado para la implementación de las técnicas de detección de vulnerabilidades de ataque Cross Site Scripting.

N°	Nombre	Dirección
1	Acunetix – test php	testphp.vulnweb.com/

Nota: Elaboración Propia

Es necesario la identificación del problema del caso de estudio. Se comentó que el caso de estudio seleccionado contiene tantas vulnerabilidades de ataques de Cross Site Scripting y sobre todo asegura ayudar a la presente investigación.

1.2. Segunda Fase:

La elaboración de preguntas se realizará en cuanto a la identificación del problema de las aplicaciones web mencionada anteriormente en la investigación, que son: ¿El caso de estudio seleccionado es lo bastante vulnerable en cuanto a ataques de Cross Site Scripting? Y ¿La aplicación web seleccionada ayudara a la investigación como caso de estudio para la implementación correcta del funcionamiento de las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting?

1.3. Tercer Fase

Investigando las fuentes de la selección del caso de estudio, se investigó una empresa que nos referencio hacía otra empresa que tiene muchos beneficios de seguridad informática, una de ellas es brindar una aplicación web vulnerable. La

primera empresa dedicada a la ciberseguridad, es “Seguridad América Ssl Limitada”, dicha empresa es reconocida en el año 2015 por GMO GlobalSign como compañero preferido (Prefered Partner), nivel de alianza que solo poseen 5 empresas en el mundo, siendo Seguridad América la única compañía en Latinoamérica en la categoría de ciberseguridad. Actualmente, Seguridad América maneja un catálogo que supera las 12 soluciones digitales pertenecientes a las marcas líderes en el mundo en ciberseguridad y gestión operativa, cuenta con oficinas físicas en Chile, Perú, Colombia, México y Argentina. Tal empresa afirma que existe una empresa llamada “Acunetix de Invicti Security”, que es la solución de seguridad en aplicaciones web todo en uno totalmente automatizada que le permite realizar el análisis de aplicaciones web para poder proteger de los atacantes. Y que dicha empresa de seguridad web cuenta con aplicaciones web vulnerables a ataques Cross Site Scripting, inyecciones sql, Cross Site Request Forgery, entre otros. Tienen aplicaciones web vulnerables con el fin de comprender como los errores de programación y configuración conducen a violaciones de seguridad, también para poder realizar pruebas de penetración manual o con fines educativos. Dicho así, se afirma que se encontró el caso de estudio para poder utilizar en las pruebas de las dos técnicas.

1.4. Cuarta Fase:

Las preguntas a resolver del caso de estudio seleccionado se responden con la presente investigación, porque se necesita una aplicación web vulnerable como objetivo principal para hacer las pruebas y así poder anotar los resultados que se darán al momento de implementar las dos técnicas en el caso de estudio. Claramente será de gran ayuda poder utilizar el caso de estudio en la presente investigación. Además, su objetivo de “Acunetix de Invicti Security” es apoyar a investigadores que se encuentren en la rama de seguridad informática, para poder facilitar lo que quieren demostrar, brindando el instrumento de forma legal. Entonces se puede decir que la aplicación web “testphp.vulnweb.com/” es lo bastante útil para la investigación presente investigación, tomando el papel del caso de estudio y así poder realizar las pruebas.

1.5. Quinta Fase

Esta fase no es necesario para la presente investigación por el motivo que se presentan a continuación: no se necesitara un informe para terminar con el método de la forma que lo indican los autores porque solo es necesario la selección del caso de estudio.

Luego de completar todas las fases de la metodología se obtuvo el caso de estudio para la presente investigación, pero también se consideró necesario especificar sus características del caso de estudio, para que posteriormente se pueda comparar si una aplicación web de microempresa cumple con la mayoría de características del caso de estudio. Con el fin de entender si el caso de estudio es apta para la presente investigación.

Tabla 26:

Características del Caso de Estudio de la presente Investigación

CARACTERÍSTICAS	CASO DE ESTUDIO
Lenguaje de programación en el que está creado	HTML
	PHP
	JAVASCRIPT
	CSS3
Funciones	Registro de Datos
	Visualización de Información
	Acceso a Datos
Procesos	Login
	Formulario
	Comentarios
	Hipervínculos
	Reportes
	Carrito de Compras
	Buscador

Nota: Recuperado de (testphp.vulnweb.com/)

En la **Tabla 26** se puede visualizar las características del caso de estudio, y básicamente una aplicación web de microempresa tiene todas esas características, una que otra tiene más características, pero lo principal son las características que se muestran.

Además, funciona con un servidor en la web de Acunetix enviando, recuperando y almacenando datos en el servidor. También tiene una interfaz apta para el usuario, afirmando que el caso de estudio sostiene una presencia agradable para el usuario que lo maneja. Finalmente muestra varias pestañas con diferentes funcionalidades como login, formularios, comentarios, entre otros, confirmando que sostiene múltiples funcionalidades como debería ser una aplicación web de microempresa.

Seguidamente se califica el caso de estudio con la técnica de medición la escala de Likert, teniendo como “1” el mínimo puntaje y “5 como máximo puntajes”.

Tabla 27:

Escala de medición planteada por la escala de Likert

Número de escala	Definición de escala
1	Seguridad nivel bajo
2	Seguridad nivel medio bajo
3	Seguridad nivel mediana
4	Seguridad nivel alto
5	Seguridad nivel extremo alto

Nota: Escala Likert.

En la **Tabla 27** se muestra el número de escala y su definición de Likert, para calificar el nivel en el que se encuentras las características del caso de estudio y si es apta para la presente investigación. En la siguiente tabla se muestra los resultados que se obtienen al momento de calificar el caso de estudio.

Tabla 28:

Evaluación de las características del caso de estudio de la presente investigación, alienándose a las características de una microempresa cuando usa tecnología web.

CARACTERÍSTICAS	CASO DE ESTUDIO	ESCALA DE LIKERT				
		1	2	3	4	5
Lenguaje de programación	HTML	X				
	PHP		X			
	JAVASCRIPT	X				
	CSS3		X			
Funciones	Registro de Datos	X				
	Visualización de Información	X				
	Acceso a Datos		X			
Procesos	Login	X				
	Formulario	X				
	Comentarios	X				
	Hipervínculos		X			
	Reportes		X			
	Carrito de Compras	X				
	Buscador	X				
	Imágenes	X				

APLICACIÓN WEB DE MICROEMPRESA

Nota: Recuperado de (testphp.vulnweb.com/)

Los resultados obtenidos dan a conocer que la seguridad de las características del caso de estudio es baja, concluyendo que el caso de estudio pertenece a una aplicación web de microempresa, porque si tuviera resultados moderados en cuanto a la seguridad de sus características perteneciera a una aplicación web de empresa mediana y si tuviera puntaje alto en la seguridad de sus características perteneciera a empresa grande.

2. Implementar las técnicas seleccionadas

Para la presente investigación se implementó las dos técnicas seleccionadas anteriormente, siguiendo con los procesos otorgados de cada investigación. Antes de desarrollar y ejecutar los procesos de cada técnica, se necesitó ciertos requisitos previos.

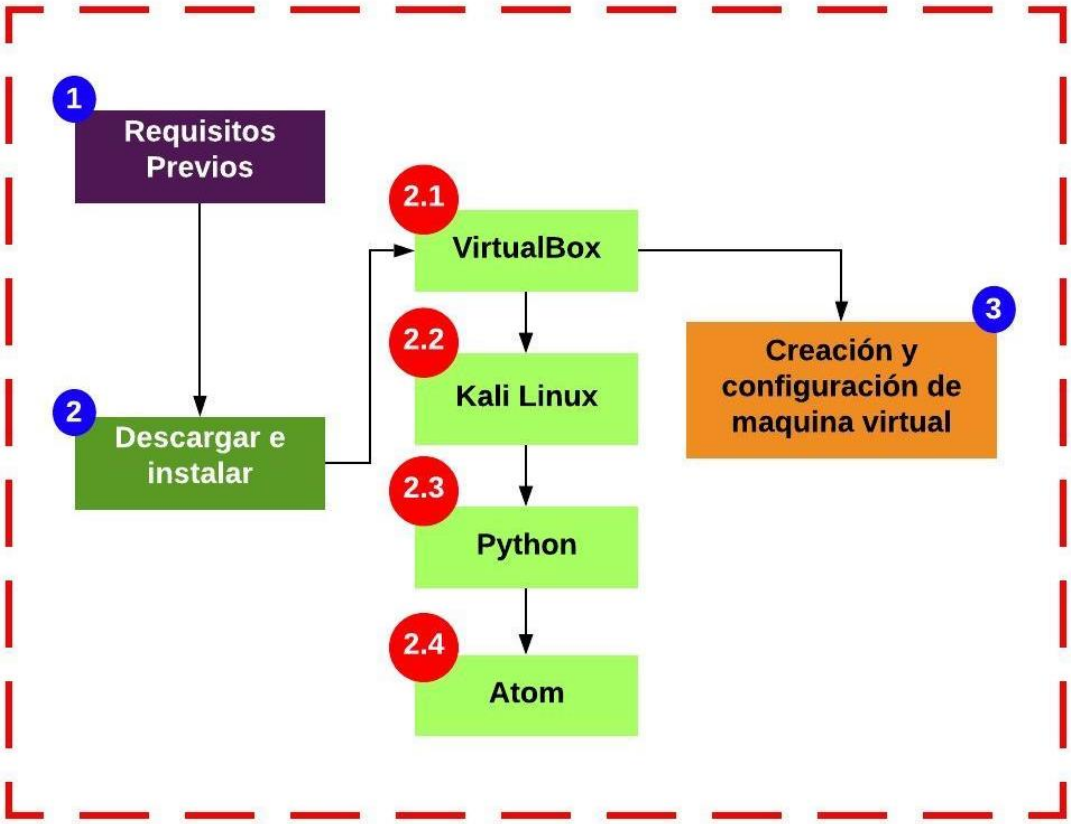

Figura 20: Requisitos previos para la implementación de las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting. Fuente: Elaboración Propia

El primer requisito es descargar e instalar el software de virtualización (VirtualBox). En el **ANEXO 4**, se muestra detalladamente como se logró todo el proceso para que funcione en el ordenador. Mencionando que se utilizó la versión 6.0.14 de VirtualBox. Básicamente es una herramienta que virtualiza sistemas operativos con una arquitectura de 86 y 64 bits. Fue y sigue perteneciendo a la empresa Oracle.

Figura 21: Sitio Web del software de VirtualBox. Fuente: virtualbox.org

Luego se continuo con el segundo requisito que es descargar el Sistema Operativo Kali Linux, siendo un producto de Linux que se encuentra basada en Debian. Siendo su objetivo primordial incluir tantas herramientas que sean seguros y difícil de penetrar por hackers. También distribuye herramientas abiertas a código y así poder crear pruebas de forma segura. Esta desarrollado por Offensive Security. Está pensando principalmente para la auditoría y seguridad informática en general. Por lo cual es recomendable utilizarlo en la presente investigación. En el **ANEXO *5***, se puede ver detalladamente el proceso que se aplicó.

Figura 22: Sitio Web del Sistema Operativo Kali Linux. Fuente: kali.org

Una vez descargada Kali Linux se creó y configuro una máquina virtual, y luego se instaló Kali Linux dentro de la máquina virtual que se puede ver los procesos en el **ANEXO *6*,** concluyendo con la fase de virtualización. Lo siguiente es descargar e instalar Python en Kali Linux.

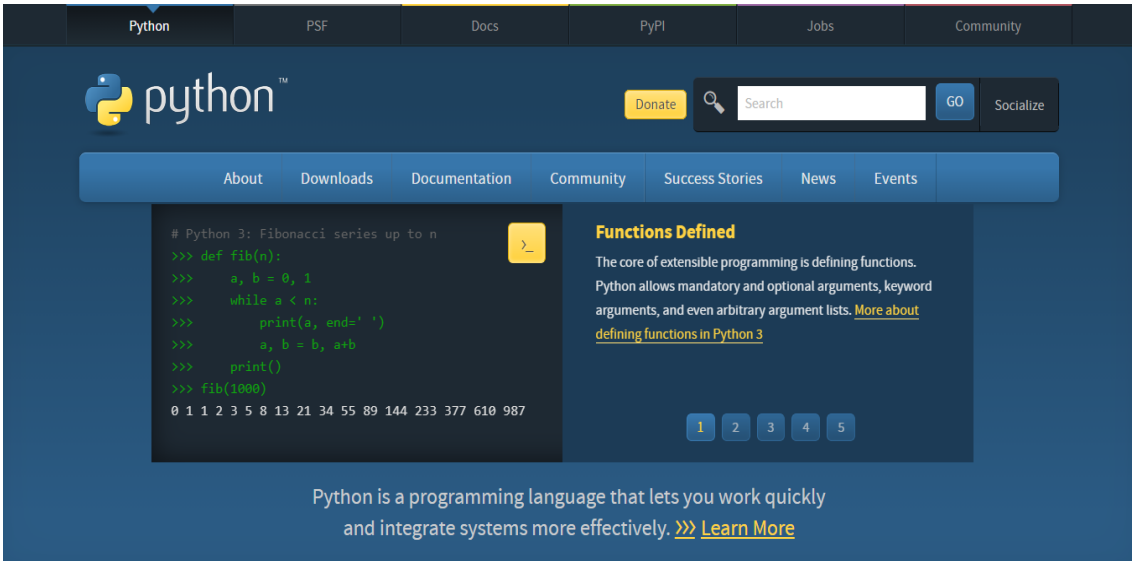

Figura 23: Sitio Web del lenguaje de programación Python. Fuente: python.org

Finalmente, para escribir el código se trabajó con un editor de código llamado Atom. Utiliza soporte para multiplataformas, con ciertas formas de plugin que están desarrollados en Node, desarrollado por GitHub. Atom es una aplicación que esta creada haciendo uso de tecnologías web. El proceso se muestra ver en el **ANEXO *7***.

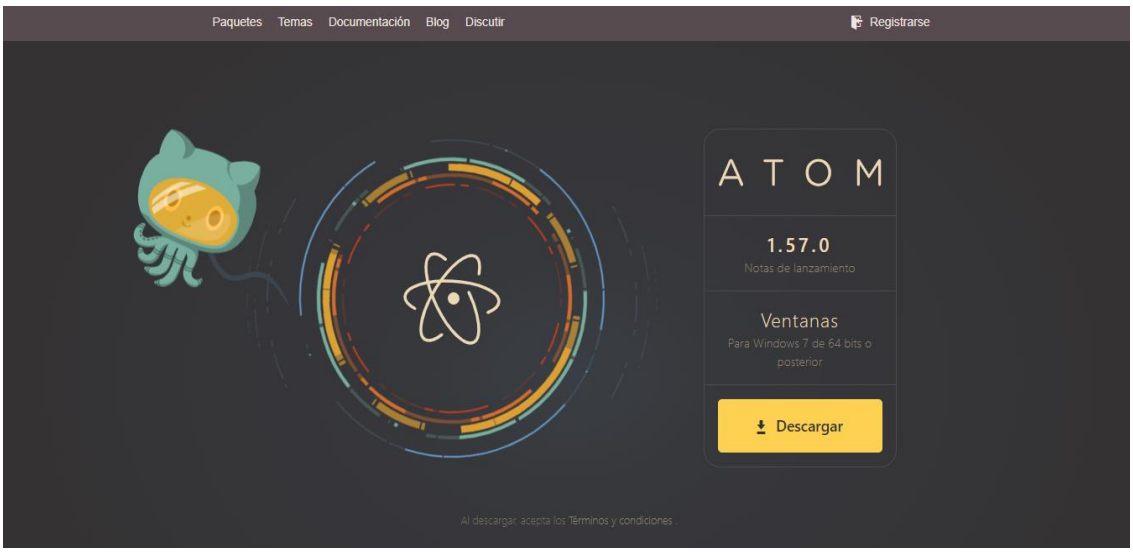

Figura 24: Sitio web del editor de código - Atom. Fuente: atom.io

Cumpliendo con todos los requisitos previos, se empezó a implementar las técnicas escritas en lenguaje de programación python. Cabe destacar que las dos técnicas están basadas en análisis en código HTML y PHP, por lo tanto, no puede detectar vulnerabilidades en otro tipo de códigos como JavaScript, Ruby, Java, entre otros. También utilizaron librerías iguales “User_agent” y “argparse” para el proceso de análisis, dichas librerías son fundamentales para poder realizar el escaneo. La primera permite realizar los procesos de manera libre y sin tener que vulnerar el navegador, sin restricciones de seguridad, la segunda sirve para obtener el inicio de la herramienta especificado un comando.

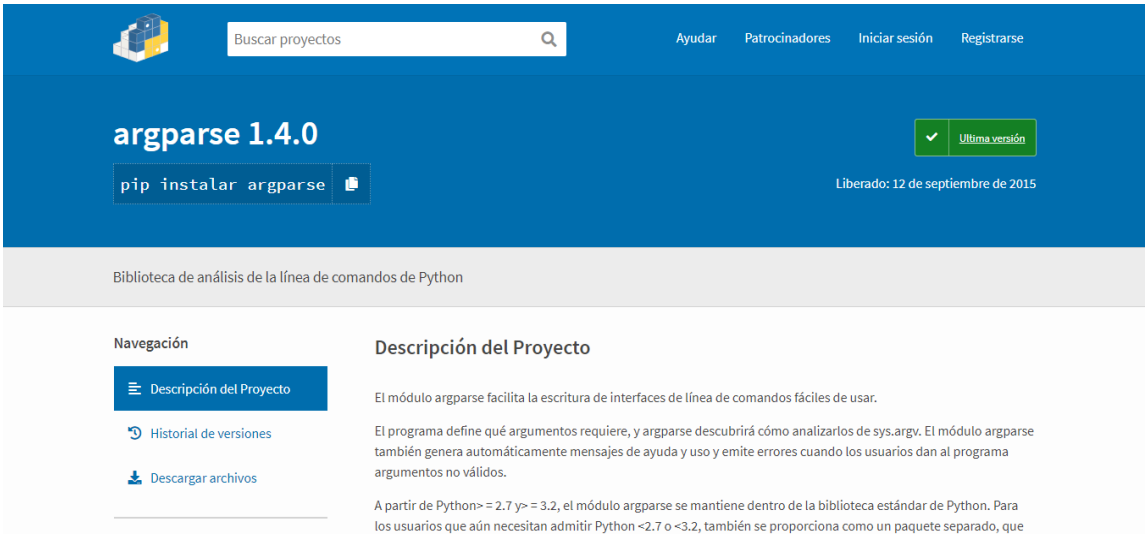

Figura 25: Sitio Web de la librería argparse. Fuente: pypi.org/project/ argparse/

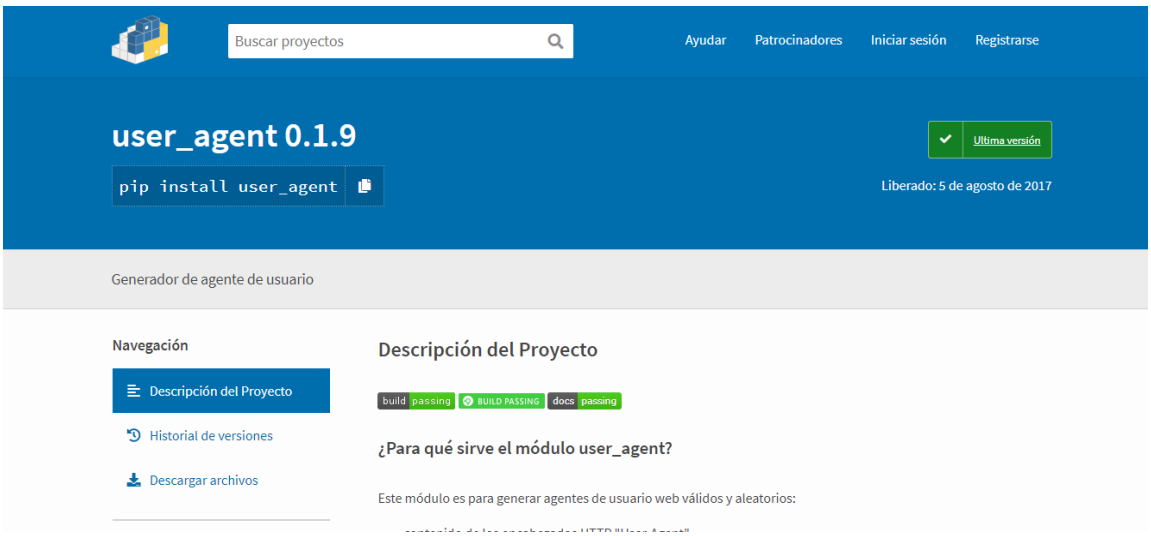

Figura 26: Sitio Web de la librería user_agent. Fuente: pypi.org/project/user_agent/

Cada técnica utiliza diferentes tipos de pruebas, la técnica DDV utiliza la prueba caja negra, mientras que la técnica SDV utiliza la caja blanca, A continuación, se presente la implementación de las dos técnicas, empezando por la técnica DDV.

a) Técnica Detección Dinámica de Vulnerabilidades (DDV)

En dicha técnica se planteó dos fases, en la primera fase trata del tema que aborda la presente investigación, la detección de vulnerabilidades de ataques de Cross Site Scripting y la segunda fase trata de prevenir los ataques de Cross Site Scripting. Por lo tanto, se trabajó solamente con la primera fase. En la siguiente figura se muestra las dos fases, y los pasos a seguir de la 'FASE N°1' para desarrollar la herramienta de detección.

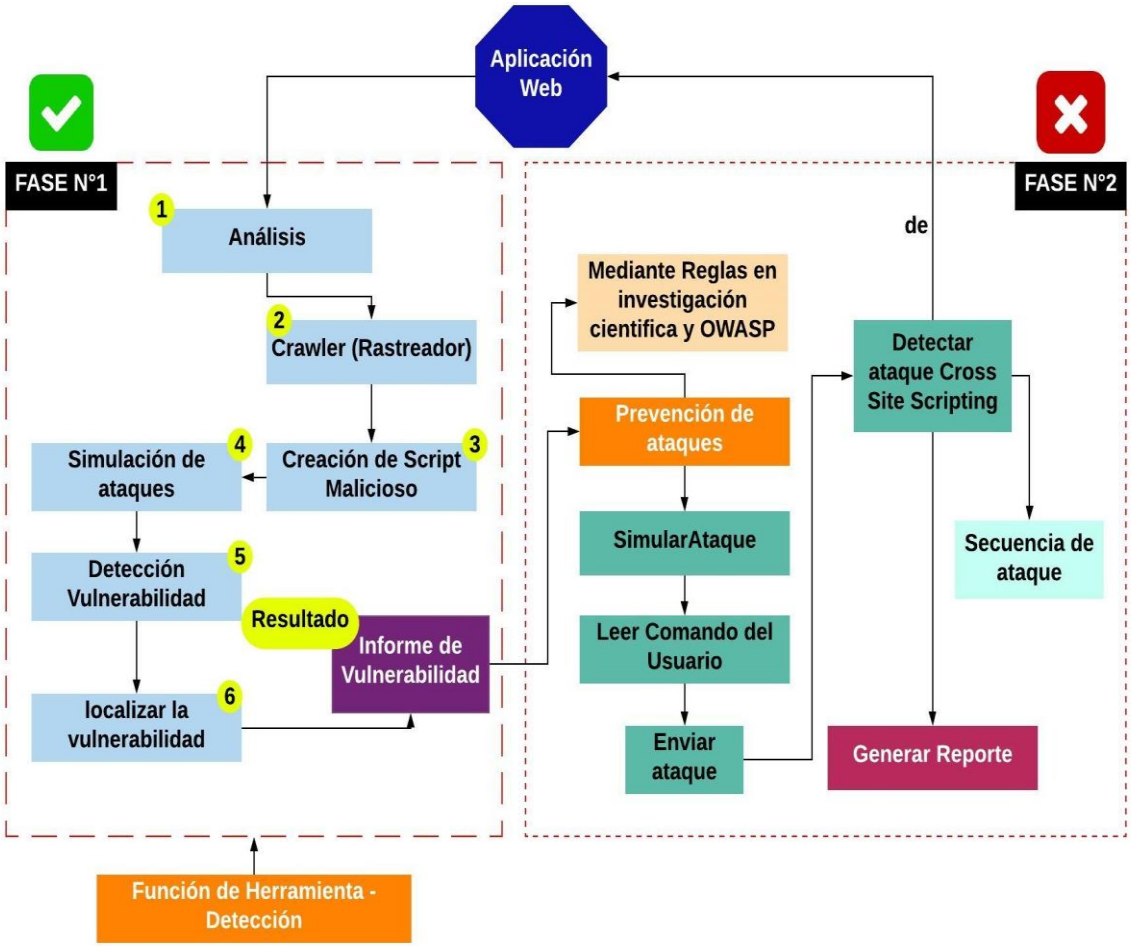

Figura 27: Arquitectura de desarrollo de la técnica Detección Dinámica de Vulnerabilidades (DDV). Fuente: Elaboración Propia

En la fase de análisis se implementó una serie de pasos obtenidos de la técnica DDV, en el siguiente diagrama de flujo se muestra los procesos generales para la detección de vulnerabilidades de ataque Cross Site Scripting.

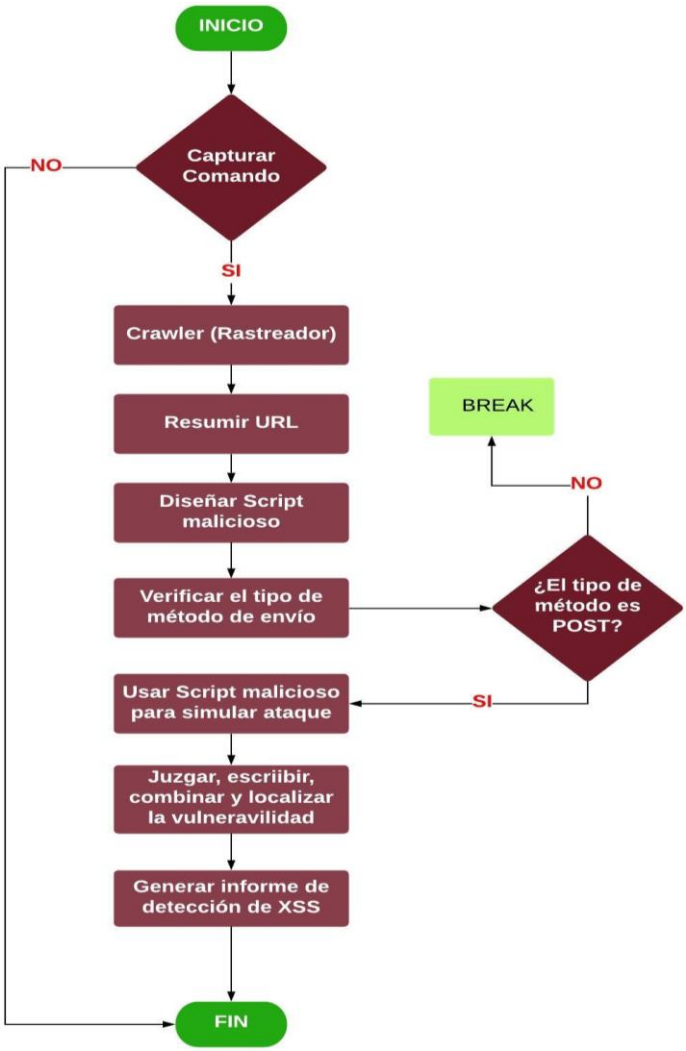

Figura 28: Diagrama de Flujo del proceso general de la fase de implementación de la herramienta DDV.

En todo el proceso se utilizarán bibliotecas, con el paso de la explicación se irá mencionando cada una de ellas y su respectivo funcionamiento. Primero se tiene que capturar la URL ingresada por el usuario en el terminal de Kali Linux, el comando tendrá parámetros de función de la herramienta. Colocando primero el tipo de lenguaje de programación de la herramienta, luego se escribe el archivo ejecutable de la herramienta “xss-ddv.py”, seguido el parámetro “-u” haciendo

referencia que vamos a examinar en la URL que se escribe dentro de las comillas “http://ejemplo.com/”. Al escribir el comando para la ejecución de la herramienta, estamos enviando la URL, que en el proceso de Crawler se convierte en la URL base. Seguidamente se realiza el método de Crawler, en dicho método se utiliza librerías para facilitar el proceso. El Crawler son normalmente robots, que realizan la función de inspeccionar aplicaciones web de manera metódica y automatizada. La herramienta DDV tiene el proceso de navegar por la aplicación web especificada recolectando los URLs principales y luego recolecta los hipervínculos y navega por ellos, afirmando que recorre toda la aplicación web. Para el proceso se utilizó la librería “Multiprocessing”, dicha librería ofrece simultaneidad remota y local, y así evitar el bloqueo global de intérprete mediante subprocesos en lugar de hilos. Luego se extrajo la información de la aplicación web, para ello se utilizó la librería “bs4” con la función “BeautifulSoup”, dicha función cumple con lo requerido para explorar las URL a fondo y las URL de los hipervínculos, obteniendo los links. Cabe mencionar que se utilizó el método “html.parser” dentro de la función “BeautifulSoup” para analizar los archivos de la web. En la siguiente figura se muestra el proceso Crawler.

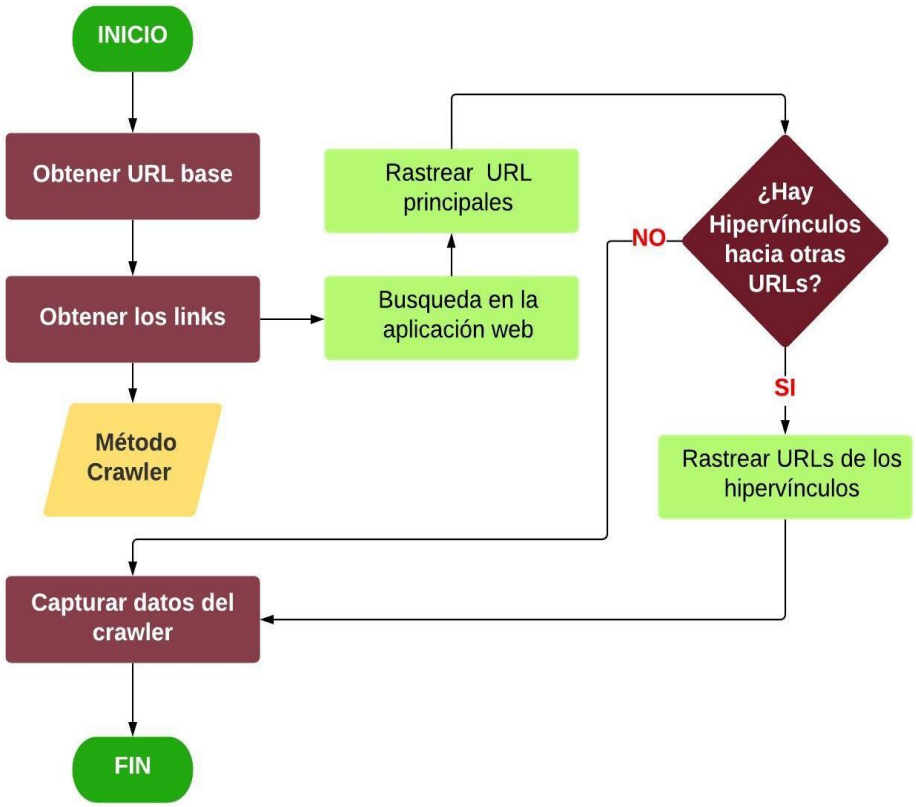

Figura 29: Proceso Crawler en la implementación de la herramienta DDV.

Siguiendo con el proceso, se creó un Script malicioso para realizar la simulación de ataque, se optó por almacenar los script en un vector, obteniendo uno de los tipos de entrada “alert(document.cookie)”, la cadena de ataque es la parte de una variable que almacena dicha cadena y la ubicación a la función, en este caso se usó los <scripts>. Una vez que se construyó el vector de ataque verifico si el método de envío es de tipo POST. Esto se obtiene cuando el Crawler analiza toda la aplicación web y retorna información, dicho retorno se recupera con la función “find_all” y se almacena en una variable para deducir si las URLs son de método POST. Son de método POST cuando él envió del código HTML es “<action>”.

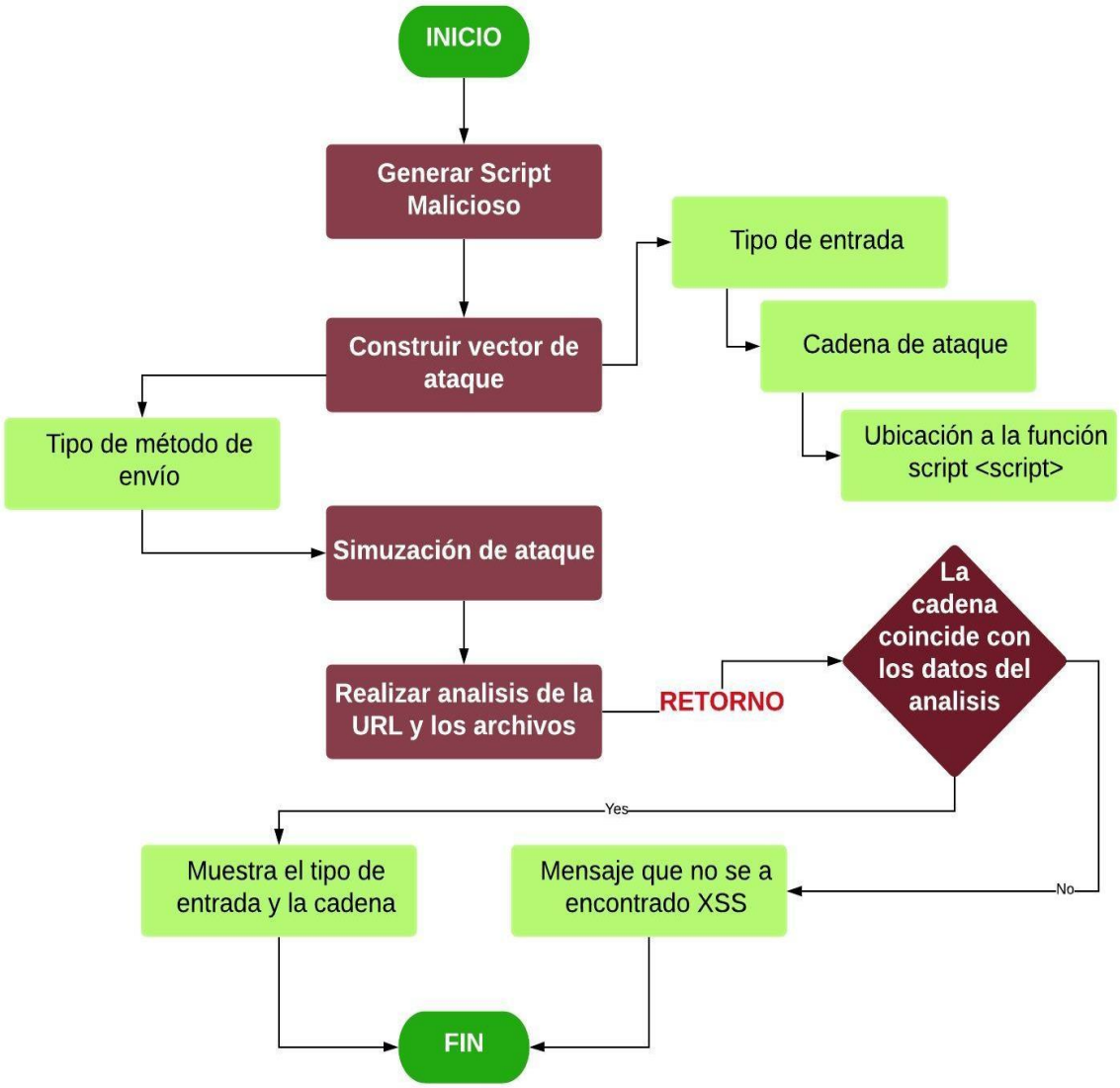

Figura 30: Diagrama de Flujo del proceso de creación de vector de ataques y simulación de ataques de la herramienta DDV.

Para la simulación de ataques se utilizó la prueba de caja negra considerando necesario el envío del vector de ataques hacía el método Crawler funcionando como simulador de ataques, se utilizó “BeautifulSoup” para el rastreo y dentro de dicha función se envió el vector de ataques y especificando que se buscará en el cuerpo del HTML, en el retorno se capturo los datos en una variable. En ambos casos se mostró un reporte del tipo de envió y el lugar a donde se envió, devolverá el resultado de tipo POST mostrando las vulnerabilidades de tipo de form y el envío que puede ser un botón o un <script>.

Por último, se obtuvo la herramienta creada satisfactoriamente con una interfaz en el terminar de Kali Linux, ejecutando desde cualquier posible ubicación, tan solo es necesario conocer la ubicación y ejecutarlo con una serie de comando. Si no la URL de la aplicación a escanear es incorrecta, entonces la herramienta marcara error. A continuación, se muestra la interfaz principal de la herramienta de la técnica DDV.

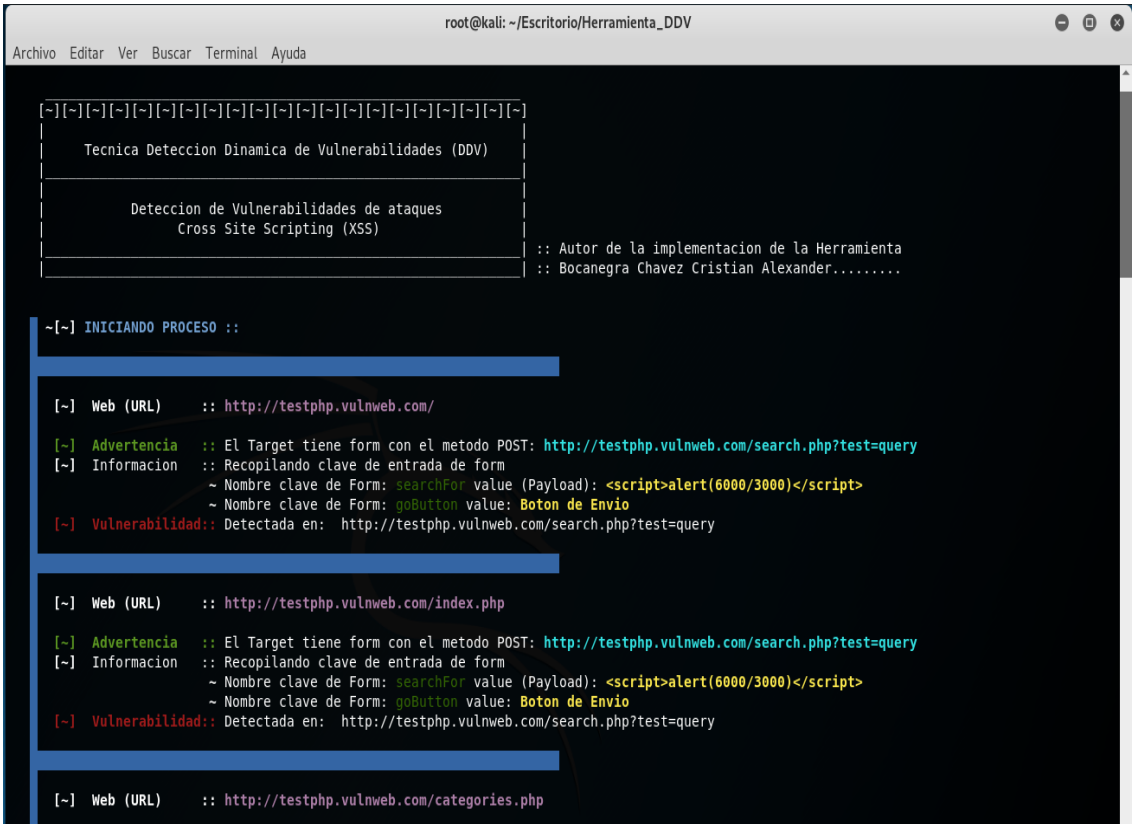

Figura 31: Interfaz principal de la herramienta de detección de vulnerabilidades de ataques de Cross Site Scripting, de la técnica DDV.

b) Técnica Sumidero - Detección de vulnerabilidad (SDV)

La técnica SDV realiza detección de vulnerabilidades de ataques de Cross Site Scripting y proponen reglas de OWASP para prevenir dichos ataques. En la siguiente figura se muestra el proceso completo, pero en la presente investigación se enfocó principalmente en la detección de vulnerabilidades. Así mismo se muestra los procesos que se implementaran para crear la herramienta de detección, siguiendo con un orden, empezando por el análisis de la URL a examinar. Cabe destacar que se seleccionó los procesos principales.

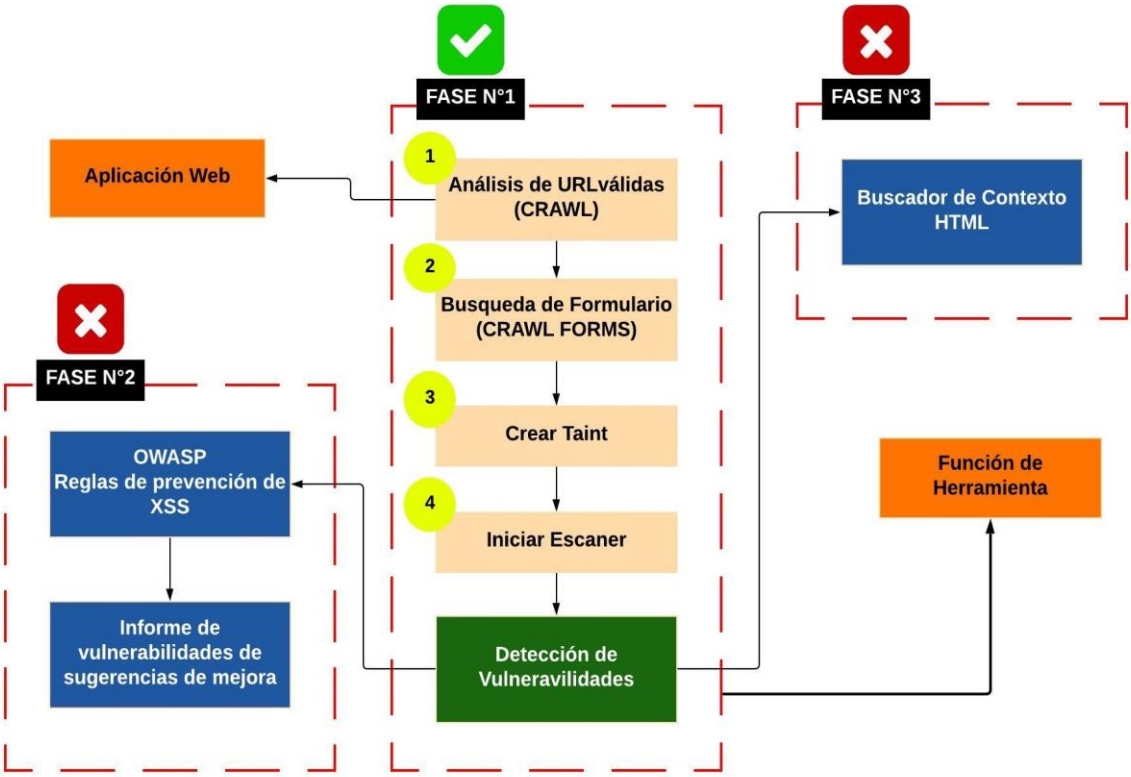

Figura 32: Arquitectura de desarrollo de la técnica Sumidero - Detección de Vulnerabilidades (SDV).

Se grafico un diagrama de flujo del proceso general de toda la etapa de implementación de la herramienta de detección de vulnerabilidades de Cross Site Scripting, mostrando los procesos principales de cada fase. Se realizó un

cambio en el diagrama de flujo con el diagrama de flujo que los autores del articulo habían creado, agregando algunos procesos.

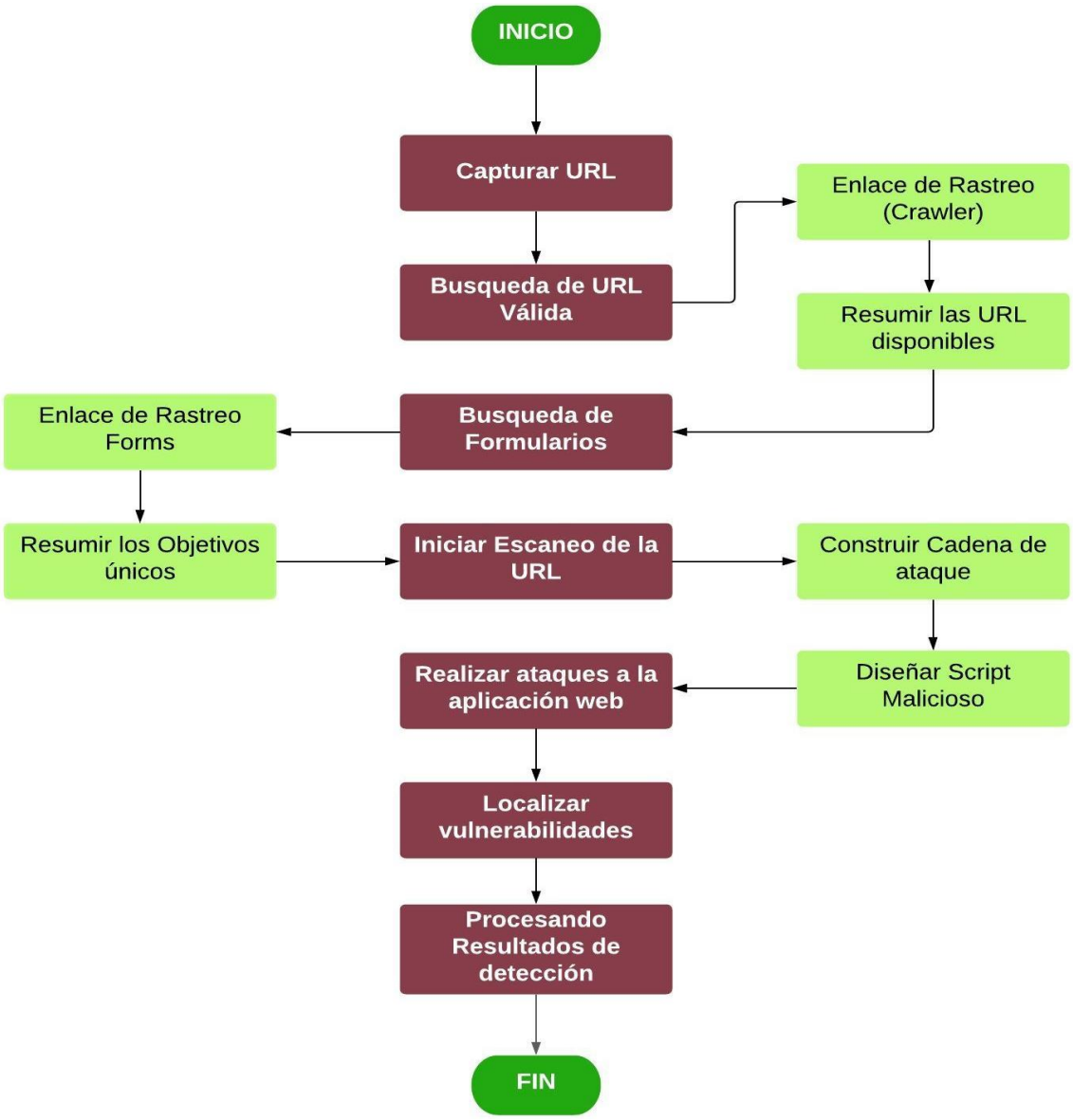

Figura 33: Diagrama de Flujo del proceso general de la fase de implementación de la herramienta SDV. Fuente: Elaboración Propia

En todo el proceso se utilizarán bibliotecas, con el paso de la explicación se irá mencionando cada una de ellas y su respectivo funcionamiento. Primero se tiene que capturar la URL ingresada por el usuario en el terminal de Kali Linux, el comando tendrá parámetros de función de la herramienta, en la siguiente tabla se explica las funciones que se ha implementado.

Tabla 29:

Parámetros para iniciar la herramienta de detección de vulnerabilidades de Cross Site Scripting, SDV

Parámetro	Función
-u	Para indicar que se colocará una URL en el proceso.
“ ”	Se coloca la URL a examinar dentro de las comillas.
--crawl	Método o procedimiento para clasificar e indexar toda la aplicación web. Rastreando el enlace para probar otros enlaces.
-- forms	Rastrear la URL de destino en busca de formularios para probar

Nota: Elaboración Propia.

Colocado primero el tipo de lenguaje de programación de la herramienta, luego el parámetro “-u” haciendo referencia que vamos a examinar en la URL que se escribe dentro de las comillas “http://ejemplo.com/”. En el siguiente diagrama de flujo se muestra el proceso de capturar la URL.

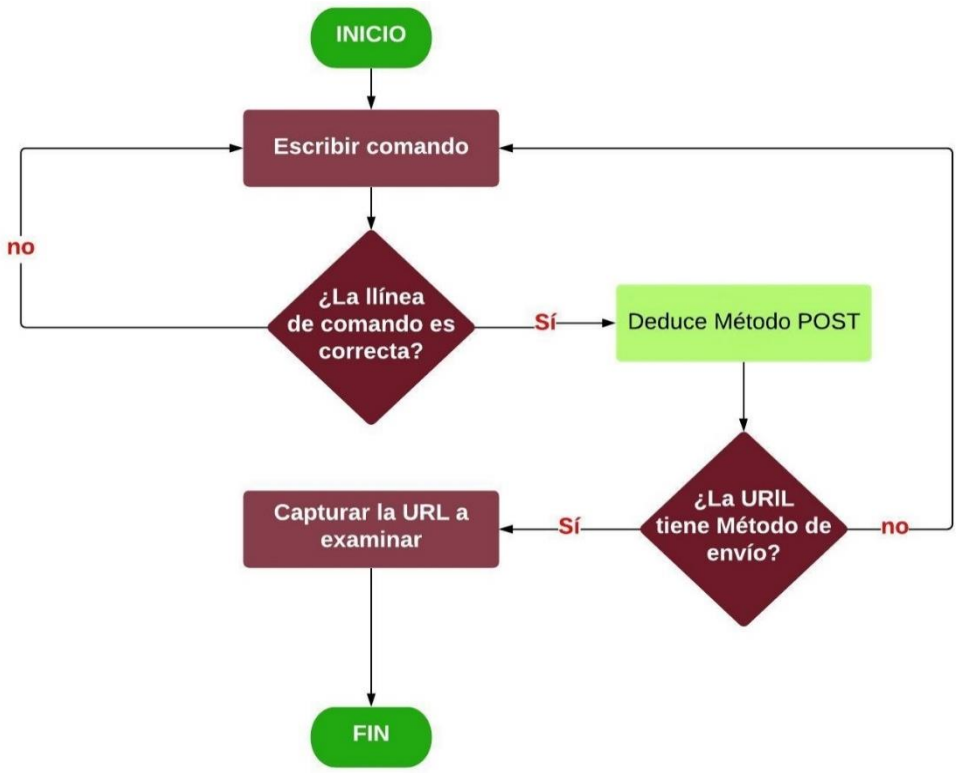

Figura 34: Proceso de Capturar la URL de la herramienta SDV. Fuente: Elaboración Propia.

Seguidamente, los procesos que se utilizará para la herramienta SDV son el “crawler” y el “forms”, es por ello que su funcionalidad se ve reflejada en el comando, ejemplo: python + -u “http://ejemplo.com/” + --crawl + --forms. Una vez escrito el comando correcto, se capturará la URL escrita y empezará con el proceso de la herramienta.

En el segundo proceso la búsqueda de URL validas (Crawl), se empieza a utilizar las funciones de la librería mechanize, tales como “request”, “urlerror”, “proxyHandler”, entre otros, dicha biblioteca permite el fácil manejo de URLs, navegación, cookies y más, brindando un objeto similar a un navegador para interactuar con las aplicaciones web. Continuando, se tiene que analizar todo el contenido y el código que compone la aplicación web. Se empieza convirtiendo la URL de destino en forma de cadena utilizando la librería “urlparse” con la función “Parse()”, luego se utiliza la función “Defaultdict()” de la librería “collections” para crear un elemento predeterminado mientras se realice el crawl, al que se va a acceder llamando al objeto de función que se le pasa al constructor.

Seguidamente se construye el método setProxies(establecer Proxies), para acceder al proxy mediante el http, seguido se tuvo que comparar la URL base con la URL absoluta (la que se convirtió en cadena), con el fin de verificar si está bien convertida la cadena, con la método “startswith()” que devuelve True si la cadena comienza con el valor especificado; de lo contrario, False, si se cumple entonces se empezará a rastrear los enlaces(crawl) para detectar y leer el contenido y el código que lo compone, en el recorrido se rastreará otros enlaces. Se retornará la URL base en una matriz con URL y datos de esta misma, esta función de retornar la URL después del recorrido del Crawl, se le denomino Target. Seguidamente, se construye la URL absoluta con una normalización.

Por último, se resume la cantidad de objetos únicos (cantidad de links encontrados). En el siguiente diagrama de flujo se muestra el proceso comentado

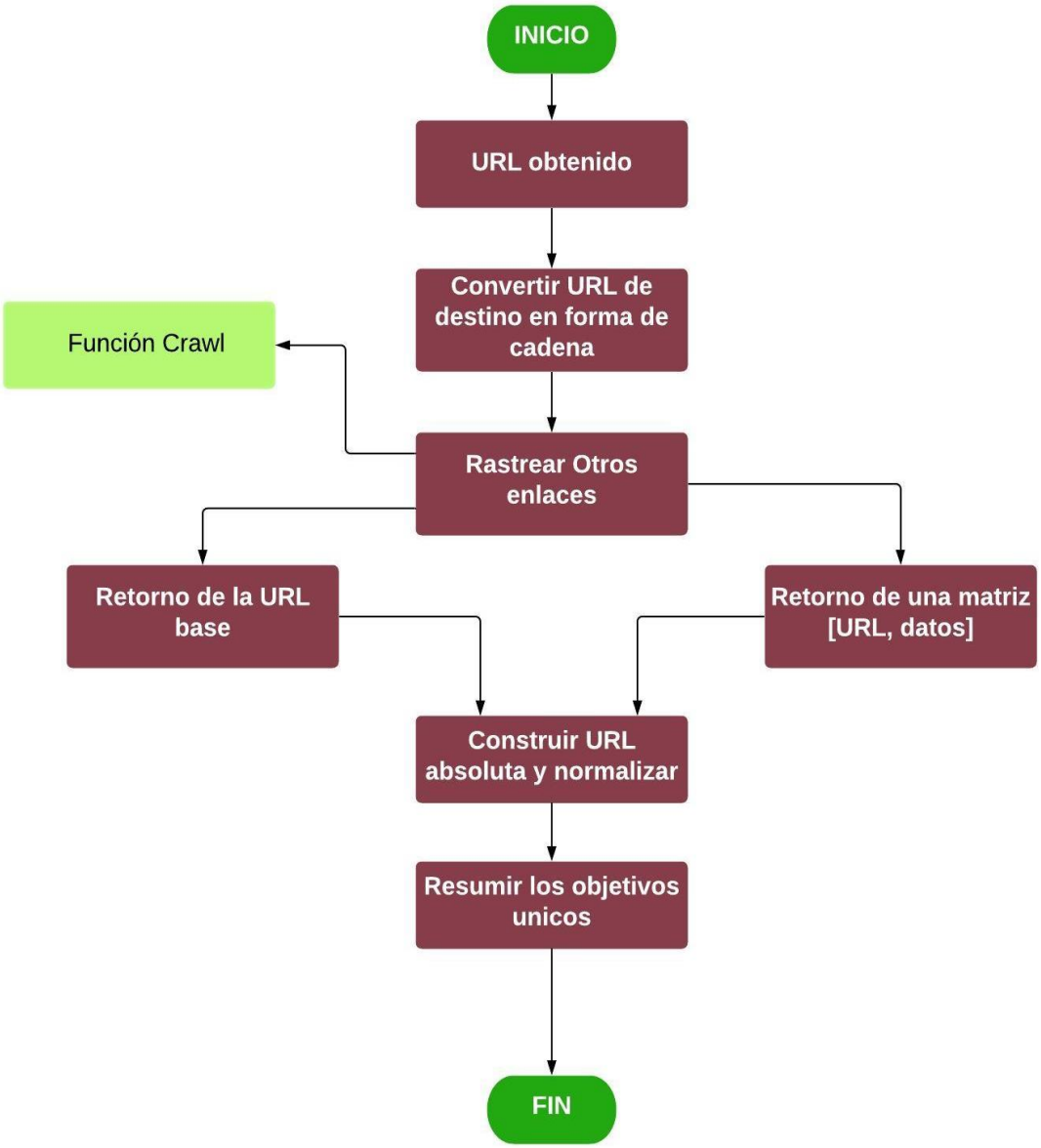

Figura 35: Proceso de búsqueda de URL validas de la herramienta SDV

Continuando con el tercer proceso, búsqueda de formularios. Se accederá a los atributos del navegador, realizando un análisis de cadenas del HTTP, utilizando la librería “user_agents” y “mechanize”. Si el método Agents funciona correctamente entonces se capturará los enlaces externos “http” y de esta manera se deduce que ingreso al servidor, y si no funciona entonces se capturará los enlaces locales o propios de la aplicación web deduciendo que no ingreso al servidor, por último, se resume los links de entrada. A toda función se le conoce como el Crawl Forms.

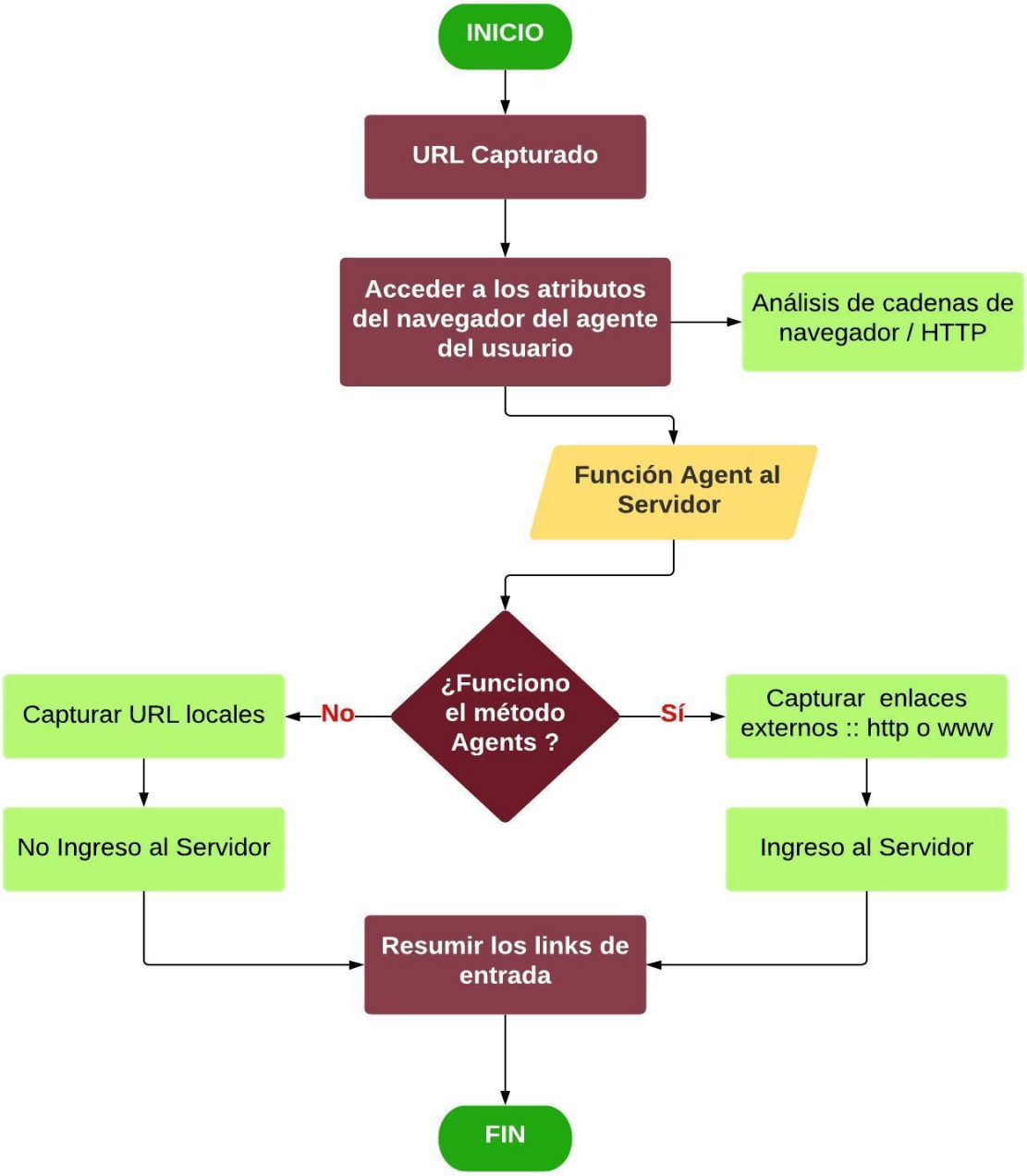

Figura 36: Proceso de búsqueda de URL validas de la herramienta SDV

Procediendo con el cuarto proceso, el escaneo de la URL para detectar vulnerabilidades de Cross Site Scripting, dicho proceso abarca también los ataques a la aplicación web y la detección de las vulnerabilidades encontradas. En el siguiente diagrama de flujo se interpreta la implementación del proceso del escaneo en la aplicación web.

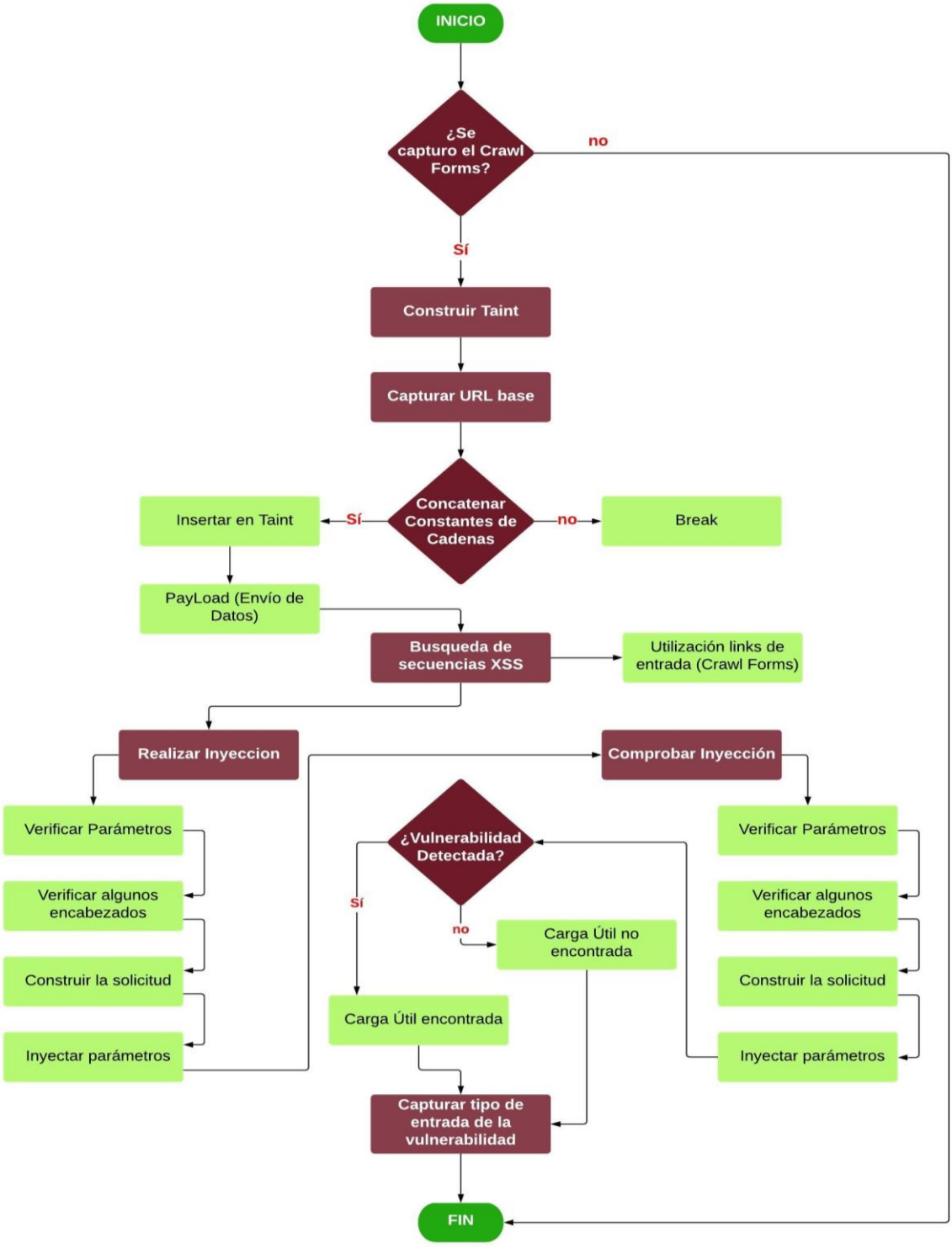

Figura 37: Proceso de escaneo, ataque y localización de vulnerabilidades de ataques Cross Site Scripting la aplicación web de la herramienta DSV.

La condición para la ejecución del proceso de escanear la aplicación web para detectar las vulnerabilidades, son los procesos anteriores Crawl y Crawl Forms,

que sin el Crawl no se puede obtener el proceso Crawl Forms. Si no se cumple con tal condición entonces el programa finalizará, pero si cumple entonces se captura los links del proceso Crawl Forms y se almacena para utilizarlo más adelante. Se utilizará la librería “mechanize”, dicha librería permite la interacción con aplicaciones web para automatizar tal interacción o cuando se desee realizar pruebas de estrés a una aplicación, permitiendo el fácil manejo de URLs, cookies, navegación y otros. Siguiendo con el proceso, se construirá un Taint que es muy usual en la seguridad informática basándose en una técnica útil para una auditoría de seguridad de la superficie de ataque de un programa. Verificando el flujo de entrada del usuario en el código de la aplicación para determinar si una entrada no anticipada puede afectar la ejecución del programa de manera maliciosa.

El Taint necesita una entrada para empezar a funcionar y es por eso que la URL base se tendrá que concatenar con constantes de cadena, si cumple tal condición entonces se insertará en el Taint y empezará a ejecutar, pero si no cumple entonces finalizará el proceso de escaneo. Luego se trabajará con el PayLoad que básicamente es el conjunto de datos que se transmite, excluyendo la cabecera para facilitar la entrega, con la condición de buscar secuencias de Cross Site Scripting, utilizando los links de entrada obtenidos del Crawl Forms. Una vez realizado todos los procesos pre escaneo, se realizará las inyecciones para detectar las vulnerabilidades de tipo Cross Site Scripting, en este proceso se verificará los parámetros de entrada, algunos encabezados y luego se construirá la solicitud de ataque a la aplicación web y finalmente se inyectará de forma correcta. En esta parte ya se tiene los resultados de las inyecciones de ataque a la aplicación web, pero se realiza el mismo proceso de inyecciones para corroborar si se ha detectado de manera correcta las vulnerabilidades. Para dicho proceso se creó el método “resultadosCompactos”, que recibe en una matriz los resultados obtenidos, comparando con el vector de ataques, si son iguales entonces found(encontro) será true, es decir, encontro vulnerabilidades, y si no encontró será false. Por último, se mostrará el tiempo de demora de los procesos de la herramienta SDV y tendremos una condición que lleva al mismo proceso. Si el resultado encontró vulnerabilidades o no encontró vulnerabilidades

entonces se capturará el tipo de entrada de la vulnerabilidad. Llegando a dar por culminado el proceso de escaneo y verificando el buen funcionamiento de la herramienta SDV.

Finalmente, se mostrará los resultados que se obtuvieron de los procesos anteriores, el capturar URL, búsqueda de URL válida (Crawl), búsqueda de formularios (Crawl Forms), Escaneo de la aplicación web realizando ataques y la localización de las vulnerabilidades. Se realizará un reporte de todo lo necesario a mostrar y se limpiará todo el proceso con el fin de dejar la herramienta SDV limpia.

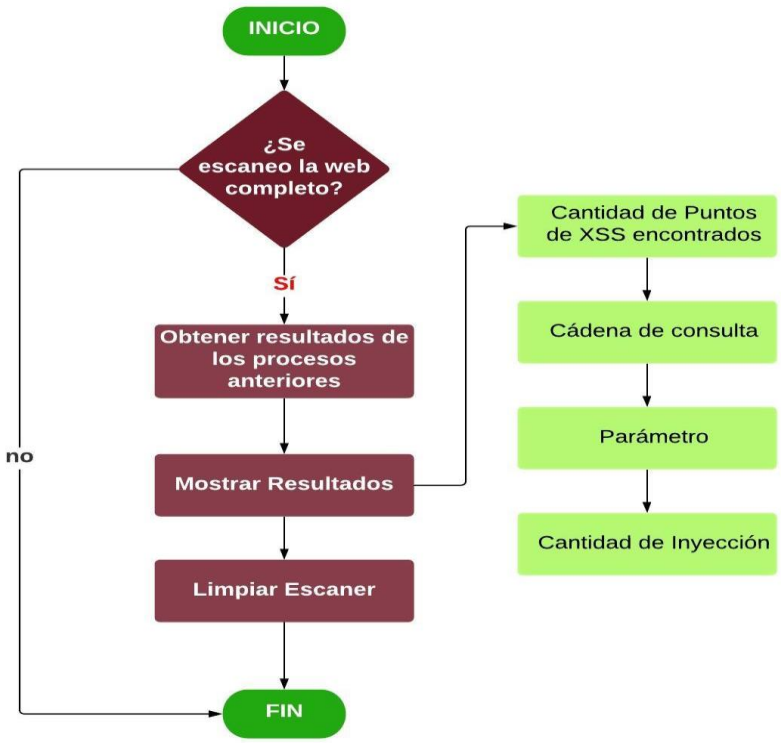

Figura 38: Proceso de Resultados de vulnerabilidades de ataques Cross Site Scripting la aplicación web de la herramienta DSV.

El reporte del escáner muestra resultados obtenidos, empezando por la cantidad de puntos de vulnerabilidad de Cross Site Scripting, luego la URL base, seguido del método de envío, la cadena de consulta, el parámetro que se ha inyectado y por último en donde se ha encontrado la vulnerabilidad.

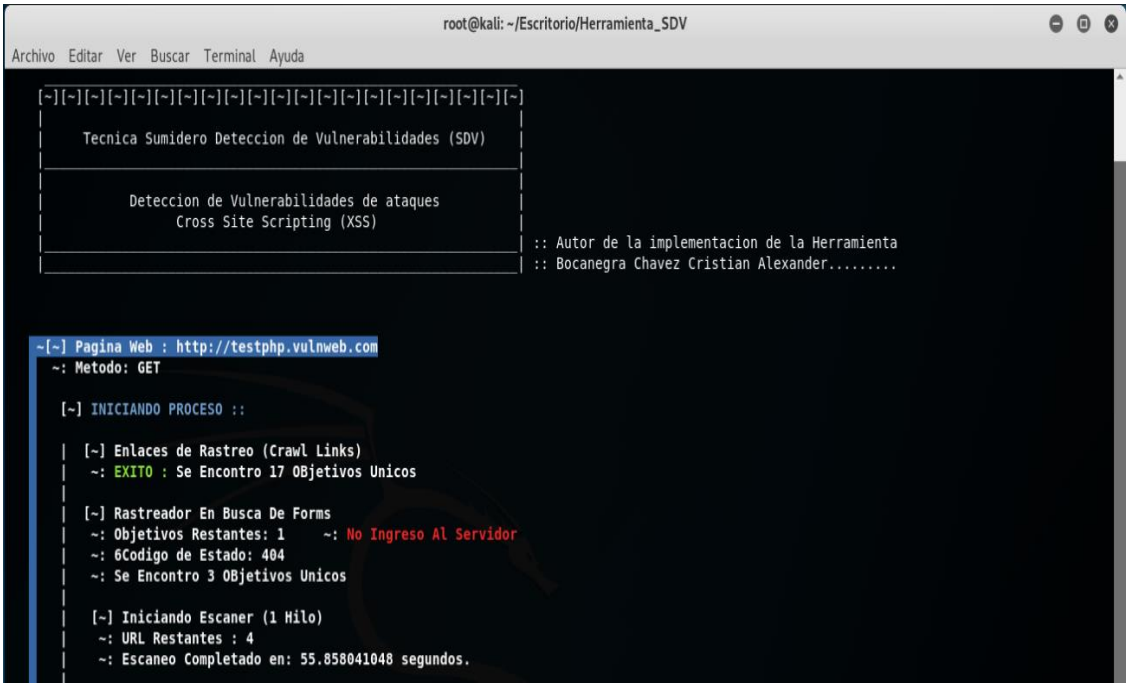

Figura 39: Interfaz principal de la herramienta de detección de vulnerabilidades de ataques de Cross Site Scripting, de la técnica SDV. Fuente: Elaboración Propia

3. Aplicar las Herramientas de las técnicas en el caso de estudio

Se aplicó las herramientas de ambas técnicas en el caso de estudio, pero solo se muestra en el presente trabajo la captura de pantalla de un escaneo a un solo caso de estudio por las dos herramientas, concluyendo con 6 pruebas en total. A continuación, se muestra el proceso general de las dos herramientas implementadas

Figura 40: Proceso general de las dos herramientas implementadas de deteción de vulnerabilidades de ataques de Cross Site Scripting. Fuente: Elaboración Propia.

Técnica DDV

Se utilizó la herramienta de detección en Kali Linux ejecutado desde la consola. Primero se realiza la ubicación de la herramienta y luego se ejecuta el siguiente comando: nombre del lenguaje ‘python3’, nombre de la carpeta de la herramienta ‘xss-ddv.py’, target URL ‘u’ y URL de la aplicación web ‘http://testphp.vulnweb.com/’. Por último, se ejecuta y escanea la aplicación, brindando un resultado.

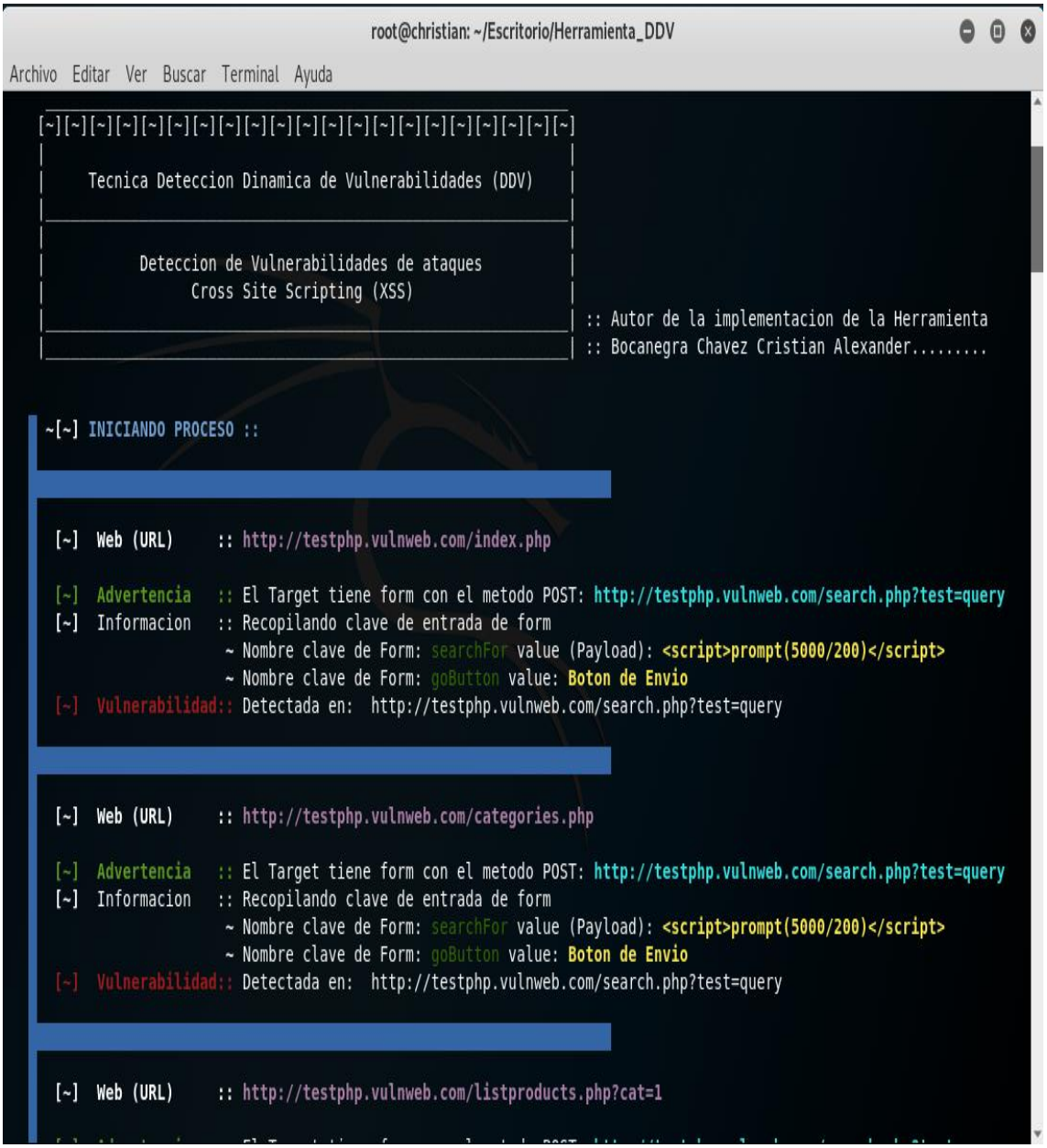

Figura 41: Técnica DDV aplicada en el caso de estudio. Fuente: Elaboración Propia.

Técnica SDV

De tal manera se utilizó la herramienta en Kali Linux ejecutado desde la consola. Funciona igual que la técnica DDV, pero con comando diferente: nombre del lenguaje ‘python’, nombre de la carpeta de la herramienta ‘xss-sdv.py’, target URL ‘u’, URL de la aplicación web ‘http://testphp.vulnweb.com/’, rastrear la URL de destino para probar ‘--crawl’ y rastrear URL de destino en busca de formularios para probar ‘--forms’. Por último, se ejecuta y escanea la aplicación, brindando un resultado.

Figura 42: Técnica SDV aplicada en el caso de estudio. Fuente: Elaboración Propia.

iii. Evaluar las técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting.

1. Crear ficha técnica y seleccionar Scripts.

1.1. Crear ficha técnica

En el presente trabajo se necesitó crear una ficha técnica para la anotación de resultados de las pruebas realizadas al caso de estudio con las herramientas de las técnicas DDV y SDV. Por los que se tomó como referencia un guía para crear dicho formato de la ficha técnica. (Bracho & Naranjo, 2019), argumentan elementos conceptuales y metodológicos para el diseño de los instrumentos de evaluación de los proyectos de evaluación, de difusión y uso, proponiendo tres fases para concluir con la ficha técnica, pero en el presente trabajo se consideró necesario solo utilizar dos fases.

Conceptualizando el instrumento de evaluación, se afirmó que se necesita anotar los datos obtenidos por cada prueba realizada, anotando el número de prueba, tiempo de ejecución del proceso escáner y tiempo de respuesta total de la herramienta, de manera que cada prueba tendrá un espacio para su anotación, y se promediará el total de todos los datos apuntados. Finalmente, en el desarrollo del instrumento de evaluación, se pensó necesario que se debe hacer uso de tablas para la anotación necesaria de la información obtenida. Dicha ficha se puede visualizar en el **ANEXO 8**.

1.2. Seleccionar Scripts

Se consideró necesario seleccionar scripts de ataques para poder utilizarlos en el caso de estudio, para comparar con las pruebas realizadas por las dos técnicas DDV y SDV. Se seleccionará de fuentes web confiables y creados por el autor del presente trabajo, esto se pude visualizar en el **ANEXO *9*.**

2. Seleccionar método de evaluación.

Se necesito métricas o herramientas que ayuden a evaluar los resultados obtenidos, para ellos se consideró necesario utilizar la matriz de confusión, siendo un modelo de clasificación basado en aprendizaje automático, y es capaz de predecir a qué clase va a pertenecer una nueva instancia. El modelo a evaluar utiliza 4 opciones que lo conforman (positivos o negativo de una matriz binaria), por lo tanto, sería verdadero positivo, verdadero negativo, falso negativo y falso positivo. Los errores se ubican en los falsos positivos y falsos negativo, y los datos verdaderos están en los verdaderos positivos y verdaderos negativos.

		Datos de herramienta	
		Positivo	**Negativo**
Datos Reales	**Positivo**	Verdaderos Positivos (VP)	Falsos negativos (FN)
	Negativo	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 43: Estructura Matriz de Confusión

3. Ejecutar Pruebas

En el presente trabajo se evaluó dos dimensiones: tiempo y eficiencia, de las técnicas Detección Dinámica de Vulnerabilidades y Sumidero Detección de Vulnerabilidades. En el tiempo se estableció conocer los valores de dos indicadores: tiempo de ejecución del proceso escáner de la herramienta y tiempo de ejecución total de la herramienta. Se entendió que el proceso de escáner demora más que los procesos iniciales y en la presente investigación se centró en tal proceso para la detección de vulnerabilidades de Cross Site Scripting. De tal manera que se consideró necesario tener los dos indicadores. Evaluando el promedio del total de pruebas realizadas en cada caso de estudio, en base al tiempo que demora en ejecutar el proceso de escáner y el proceso total de cada herramienta para la detección de vulnerabilidades de ataques de Cross Site

Scripting. Se realizo escaneos al caso de estudio, cada herramienta realizo tres escaneos por cada caso de estudio, teniendo tres escaneos por técnica y en total fueron seis, cronometrando ambos indicadores de evaluación de tiempo. Finalmente se promediaron las tres pruebas en base a los dos indicadores. A continuación, se muestra en las dos tablas los resultados adquiridos de la dimensión tiempo, de las dos técnicas implementadas en la presente investigación.

Tabla 30:

Resultados de las pruebas realizadas al caso de estudio con la técnica Detección Dinámica de Vulnerabilidades (DDV) obteniendo los indicadores tiempo de ejecución y respuesta

Técnica Detección Dinámica de Vulnerabilidades (DDV)		
Caso de estudio	**Tiempo de ejecución del proceso escáner**	**Tiempo disponible total de la herramienta**
Prueba 1	53 segundos	55 segundos
Prueba 2	52 segundos	53 segundos
Prueba 3	50 segundos	51 segundos
Promedio	51 segundos	53 segundos

Nota: Elaboración Propia

Tabla 31:

Resultados de las pruebas realizadas al caso de estudio con la técnica Sumidero – Detección de Vulnerabilidades (SDV) obteniendo los indicadores tiempo de ejecución y respuesta.

Técnica Sumidero - Detección de Vulnerabilidades (SDV)		
N° de prueba	**Tiempo de ejecución del proceso escáner**	**Tiempo disponible total de la herramienta**
Prueba 1	30 segundos	32 segundos
Prueba 2	28 segundos	30 segundos
Prueba 3	27 segundos	29 segundos
Promedio	28.3 segundos	30.33 segundos

Nota: Elaboración Propia

Luego se dio a conocer la eficiencia de las dos técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting, utilizando la herramienta matriz de confusión. Básicamente permite a la presente investigación visualizar el desempeño de cada herramienta de las dos técnicas. Para la utilización de la matriz de confusión se necesita los datos obtenidos de las pruebas de cada técnica, para ello se necesita promediar los resultados de cada técnica aplicada al caso de estudio. Una vez obtenido los datos, se pasa a completar los campos de la matriz de confusión que se compone de dos clases: datos reales y datos de la herramienta de cada técnica.

Necesitando conocer el verdadero positivo que viene hacer la cuantía de valores efectivos que se clasificaron de forma correcta como valores positivos del arquetipo, luego se conoce el verdadero negativo que viene hacer la cuantía de valores negativos que se clasificaron de manera correcta como valores negativos del arquetipo, seguidamente conocer el falso negativo que es la cantidad de valores positivos que se clasificaron de manera incorrecta como valores negativos, por último, el falso positivo que viene hacer la cuantía de valores negativos que se clasificaron de forma incorrecta como valores positivos. Se completa la matriz y se evalúa cuatro clasificadores: precisión que predice los positivos, sensibilidad que el tipo se positivo, especificidad que el tipo es negativo y exactitud de mostrar de manera correcta. Cabe destacar que se todos los resultados de los clasificadores se mostrarán en porcentaje (%). Finalmente se calcula los valores de los clasificadores con sus operaciones para determinar el porcentaje de cada uno de ellos.

Precisión	$PR = \frac{VP}{VP + FP}$
Sensibilidad	$SE = \frac{VP}{VP + FN}$
Especificidad	$ES = \frac{VN}{VN + FP}$
Exactitud	$EX = \frac{VP + VN}{VP + FP + FN + VN}$

Figura 44: Operaciones de los clasificadores de la matriz de confusión

Seguidamente, se contabilizo la cantidad de vulnerabilidades que tiene cada caso estudio, ejecutando los scripts en cada una de ellas. En la siguiente figura se muestra un ejemplo de cómo respondió la aplicación web al inyectar el script.

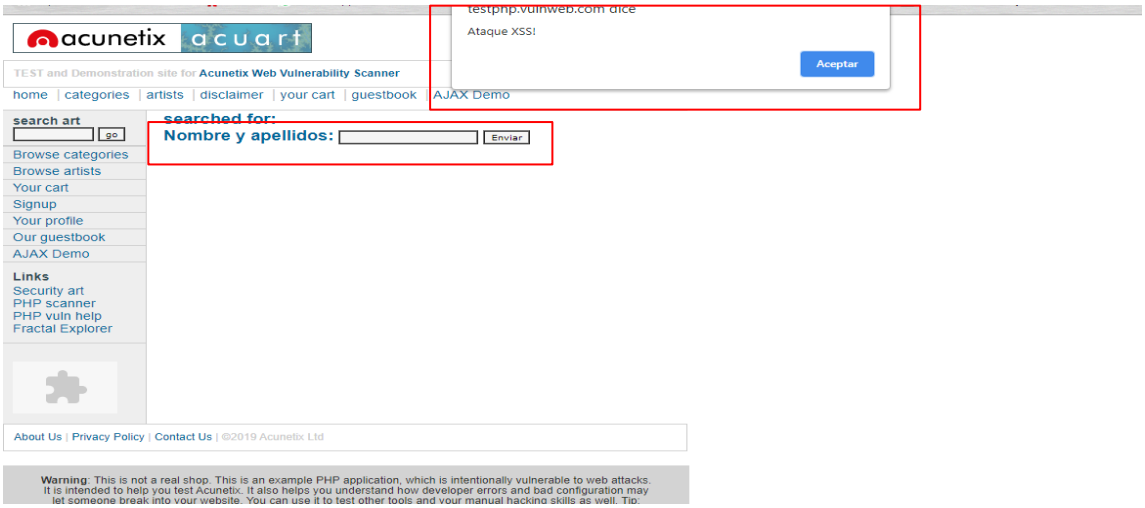

Figura 45: Resultado de haber ejecutado un Script de ataque XSS. Fuente: Elaboración Propia

Se realizaron un total de 3 pruebas, los resultados se muestran a continuación en las dos siguientes tablas, cuyos resultados fueron promediados.

Tabla 32:

Promedio de datos de vulnerabilidades y no vulnerabilidades de las pruebas realizadas al caso de estudio con la herramienta de la técnica Detección Dinámica de Vulnerabilidades (DDV) y de los datos reales.

Técnica Detección Dinámica de Vulnerabilidades (DDV)				
N° de prueba	**Datos de Herramienta**		**Datos Reales**	
	Cantidad de vulnerabilidad detectadas	**Cantidad de vulnerabilidad no detectadas**	**Cantidad de vulnerabilidad detectadas**	**Cantidad de vulnerabilidad no detectadas**
Prueba 1	15	0	15	1
Prueba 2	14	1	15	1
Prueba 3	14	1	15	1
Promedio	14	1	15	1

Nota: Elaboración Propia

Tabla 33:

Promedio de datos de vulnerabilidades y no vulnerabilidades de las pruebas realizadas al caso de estudio con la herramienta de la técnica Sumidero – Detección de Vulnerabilidades (SDV) y de los datos reales.

Técnica Sumidero - Detección de Vulnerabilidades (SDV)				
N° de prueba	**Datos de Herramienta**		**Datos Reales**	
	Cantidad de vulnerabilidad detectadas	**Cantidad de vulnerabilidad no detectadas**	**Cantidad de vulnerabilidad detectadas**	**Cantidad de vulnerabilidad no detectadas**
Prueba 1	3	12	15	1
Prueba 2	2	13	15	1
Prueba 3	3	12	15	1
Promedio	3	12	15	1

Nota: Elaboración Propia

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones.

A) Como punto de partida de la presente investigación se procedió a la selección de técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas, optando como referencia de la IEEE, la cual se seleccionó cinco técnicas. Procediendo a la evaluación de indicadores más resaltantes por los autores de cada técnica, coincidiendo con dos indicadores “exactitud y flexibilidad”, de tal manera sean evaluadas. Se seleccionaron dos de ellas, las que obtuvieron mayor puntaje en la evaluación con un resultado de 8.

B) Se procedió a seleccionar el caso de estudio utilizando la metodología “El estudio de caso y su implementación en la investigación”, permitiendo culminar cinco fases de la metodología y luego se comparó las características del caso de estudio con características de una aplicación web de microempresa, teniendo como resultado una alineación de características positivas. Seguidamente, la técnica Dinámica de Detección de Vulnerabilidades (DDV) al momento de realizar su implementación, demostró tener más procesos utilizando el Crawler como proceso principal para captar las URLs y ubicación, creación de vector de ataques y la simulación del vector de ataques en pruebas de caja negra. Recolectando datos necesarios para coincidir en plenitud con la detección de vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas.

C) Las pruebas realizadas en el caso de estudio fueron muy cercanas en el tiempo de ejecución del proceso escáner y tiempo de respuesta total de la herramienta en ambas técnicas. Ejecutándose las pruebas en Kali Linux porque dicho sistema operativo basado en Debian está diseñado principalmente para la auditoría y seguridad informática en general, proporcionado gran ayuda al momento de ejecutar las herramientas en el caso de estudio.

D) Finalmente, se comparó los resultados obtenidos de ambas técnicas, concluyendo que ambas técnicas detectan vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas, teniendo una tasa de Verdaderos Negativos sobresalientes, pero en los Verdaderos Positivos la técnica DDV sobresale con promedio de 14 vulnerabilidades detectadas de 15 vulnerabilidades reales y la técnica SDV su promedio de vulnerabilidades es de 3 de 15 vulnerabilidades reales. Así mismo, la técnica DDV mostro un alto porcentaje en los indicadores de eficiencia, en la precisión un 93.33%, sensibilidad un 93.33%, especificidad un 93.75% y en la exactitud un 93.54%, mientras que la técnica SDV mostro un bajo porcentaje en los indicadores de eficiencia, en la precisión un 20.00%, sensibilidad 75.00%, especificidad un 55.55% y en la exactitud un 58.06%. De tal manera la técnica DDV detecta mejor las vulnerabilidades de ataques de Cross Site Scripting en aplicaciones web de microempresas, logrando una mejora en cuanto a la ayuda hacía los programadores que desarrollan aplicaciones web para microempresas, y para los encargados de la área de sistemas o informática de microempresas, mejorando la seguridad de su aplicaciones web, detectando las vulnerabilidades de ataques de Cros Site Scripting y poder mejorar el código fuente para no tener ataques informático futuros.

4.2. Recomendaciones.

Muchos desarrolladores cuando inician a crear aplicaciones web para microempresas escriben código sin pensar en un ataque informático, es por ello que se recomienda aplicar dichas herramientas en las aplicaciones web para detectar vulnerabilidades de ataques de Cross Site Scripting y así saber su ubicación y poder prevenir dichos ataques.

La técnica Sumidero – Detección de Vulnerabilidades procesa un rastreo de vulnerabilidades básica por tal motivo que no detecta profundamente las vulnerabilidades, es por ello que se le recomienda mejorar en ese punto.

Se recomienda trabajar con aplicaciones web vulnerables aquellas personas que se dediquen a desarrollar técnicas de detección de vulnerabilidades, con el fin de visualizar si la técnica desarrollada esta por un buen camino en detectar las vulnerabilidades de ataques Cross Site Scripting. También que se ejecute en Kali Linux que es un sistema operativo que apoya a la seguridad informática, para detectar ataques y vulnerabilidades, ayudando a la presente investigación.

Se recomienda seguir con la investigación de detección de vulnerabilidades de ataques de Cross Site Scripting porque es un tema muy delicado en cuanto a privacidad de microempresas y a los usuarios que utilizan la aplicación web. De manera que, si no lo hacen, la microempresa puede sufrir pérdidas de información y datos, también para el usuario porque puede sufrir robos de cuentas bancarias, contraseñas, suplantación de identidades falsas, etc.

Finalmente, se les recomienda a los trabajadores de las microempresas que se encargan del área de informática o sistemas, que exijan a los desarrolladores que han creado la aplicación web, aplicar dichas herramientas de las dos técnicas de detección de vulnerabilidades de ataques de Cross Site Scripting para prevenir ataques informáticos, mejorando el código fuente y algunas configuraciones.

REFERENCIAS.

Cebrián, A., María, J., Guzmán Sacristán, A., Laguna Durán, P., & Bailón, M. (2014). *Ataques a Aplicaciones Web.* Universitat Oberta de Catalunya 74.

Alzahrani, Abdulrahman, Ali Alqazzaz, Ye Zhu, Huirong Fu, & Nabil Almashfi. (2017). *Web Application Security Tools Analysis.* Proceedings - 3rd IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2017, 3rd IEEE International Conference on High Performance and Smart Computing, HPSC 2017 and 2nd IEEE International Conference on Intelligent Data and Securit 237-42. doi: 10.1109/BigDataSecurity.2017.47.

Amutio Gómez,M. (2012). *Metodología de Análisis y Gestión de Riesgos de Los Sistemas de Información*. 5-75.

Asensio Hildago, L. (2014). *Seguridad En Aplicaciones Web : Una Visión Práctica.* Leganés, España.

Babiker, Mohammed, Enis Karaarslan, & Yasar Hoscan. (2018). *Web Application Attack Detection and Forensics: A Survey.* 6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding 2018-Janua:1–6. doi: 10.1109/ISDFS.2018.8355378.

Baojiang, Cui, Long Baolian, & Hou Tingting. (2014). *Reverse Analysis Method of Static XSS Defect Detection Technique Based on Database Query Language.* Proceedings - 2014 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2014 487–91. doi: 10.1109/3PGCIC.2014.99.

Blanco Bueno, C. (2015). *Ingenieria Del Software II.* CreaOve Commons BY-NC-SA, 85.

Bracho González,T., & Naranjo Piñera, B. (2019). *Guía Para La Elaboración de Instrumentos de Evaluación.* Instituto Nacional Para La Evaluación de La Educación (INEE) 145.

Campetella, M. (2015). *Aspectos Legales de La Seguridad Informática*. OWASP The Open Web Application Security. 1–33.

Cano, J. (2008). *Entendiendo La Inseguridad de La Información*. Editorial creativity and research. California, Estados Unidos, 1-8.

Chapple, Mike, James, M., Stewart, & Darril Gibson. (2018). *Malicious Code and Application Attacks*. CISSP, Eighth Edition 915–48. doi: 10.1002/9781119549567.ch21.

Choi, Hyunsang, Seongjin Hong, Sanghyun Cho, & Young Gab Kim. (2018). *HXD: Hybrid XSS Detection by Using a Headless Browser*. Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology, CAIPT 2017 2018-Janua:1–4. doi: 10.1109/CAIPT.2017.8320672.

Costas, J. (2003). *Seguridad Informática*. Edited by R.-M. Editorial. Madrid.

Deng, G., & Robert, H. (1995). *Distributed Systems.* Second Edition. Vol. 18.

Dunham, Ken, & Gold Honors. (2007). *Mitigating Malicious Code.* Information Systems Security 16(4):233–38. doi: 10.1080/10658980701585314.

ESET. (2018). *Eset Security Report Latinoamérica (2018)*. We Live Security 3–15.

ESET, & Welivesecurity. (2019). *Eset Security Report Latinoamerica 2019.* Eset Security Report Latinoamérica 2019 (December):4–10

ESET, & Welivesecurity. (2020). *Eset Security Report Latinoamérica 2020.* Eset Security Report Latinoamérica 2020 (October):3–29.

Fogie, Seth, Jeremiah Grossman, Robert Hansen, & Anton Rager. (2007). *Cross Site Scripting Attacks Xss Exploits and Defense.*

Garcia Alfaro, J., & Navarro Arribas, G (2007). *Prevención de Ataques de Cross-Site Scripting En Aplicaciones Web*. Actas de la Recsi, 1–9.

Gómez, V., Álvaro, & Suárez, R. (2006). *Sistemas de Información. Herramientas Prácticas Para La Gestión Empresarial.* Ra-Ma.

Grabiel, Gallardo Avilés. (2014). *Seguridad En Base de Datos y Aplicaciones Web*.

Mexico: 2° Edición. edited by I. C. Academy.

Guamán Quinche, R. (2011). *Seguridad En Entornos Web Para Sistemas de Gestión Académica*. 1–47.

Guo, Xiaobing, Shuyuan Jin, & Yaxing Zhang. (2015). *XSS Vulnerability Detection Using Optimized Attack Vector Repertory.* Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2015 29–36. doi: 10.1109/CyberC.2015.50.

Gupta, Kunal, Rajni Ranjan, S. , & Manish Dixit. (2018). *Cross Site Scripting (XSS) Attack Detection Using Intrustion Detection System*. Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems, ICICCS 2017 2018-Janua:199–203. doi: 10.1109/ICCONS.2017.8250709.

Gupta, Mukesh Kumar, Mahesh Chand, G., & Girdhari Singh. (2014). *A Context-Sensitive Approach for Precise Detection of Cross-Site Scripting Vulnerabilities.* 2014 10th International Conference on Innovations in Information Technology, IIT 2014 7–12. doi: 10.1109/INNOVATIONS.2014.6987553.

Hou, Xin-yu, & Yu-peng Chen. (2018). *A Dynamic Detection Technique for XSS Vulnerabilities.* 2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC) 34–43. doi: 10.1109/ICNISC.2018.00016.

Householder, Allen, Kevin Houle, & Chad Dougherty. (2002). *Computer Attack Trends Challenge Internet Security.* IEEE Computer Magazine (4):5–7.

Huang, Shih Kun, Han Lin Lu, Wai Meng Leong, & Huan Liu. (2013). *CRAXweb: Automatic Web Application Testing and Attack Generation.* Proceedings - 7th International Conference on Software Security and Reliability, SERE 2013 208–17. doi: 10.1109/SERE.2013.26.

Martínez Leyva, J. (2009). *Infeccion Malware En El Mundo Actual*. Sabia-Tic, 3-74

ISACA. (2013). *Guía de Auto-Evaluación: Usando COBIT 5*. Barcelona: ISACA®,

24.

Jiménez Chaves, V. (2012). *El Estudio de Caso y Su Implementación En La Investigación Viviana*. Revista Internacional de Investigación En Ciencias Sociales 8(1):141–50.

Kulkarni, Yogini A., & Rajendra G. Kaduskar. (2010). *Security against Malicious Code in Web Based Applications*. Proceedings - 3rd International Conference on Emerging Trends in Engineering and Technology, ICETET 2010 286–91. doi: 10.1109/ICETET.2010.53.

Larrieu, E. (2015). *Ciberataque.* Buenos Aires: Ciencias - Económicas (UBA), 1–58.

Guitiérrez del Moral, L. (2014). *Curso Ciberseguridad y Hacking Ético*. España, Sevilla: 1° Edicion. edited by P. R. Libros.

Li, Yonghao. (2009). *Cross-Site-Scripting (XSS) - Attacking and Defending.* Degree Programme in Information Technology 2:1–79.

Liu, Yuan, Wenbing Zhao, Dan Wang, & Lihua Fu. (2016). *A XSS Vulnerability Detection Approach Based on Simulating Browser Behavior.* 2015 IEEE 2nd International Conference on InformationScience and Security, ICISS 2015. doi: 10.1109/ICISSEC.2015.7370974.

López, C., & Quezada. J. (2016). *Ataques Informáticos Mo.* México: Entretenimiento de libros, 20–60.

López, R. (2015). *Fundamentos de Seguridad*. EJE 1 Conceptualicemos, 1–30.

Marashdih, Abdalla Wasef, & Zarul Fitri, Z. (2017). *Detection and Removing Cross Site Scripting Vulnerability in PHP Web Application.* Proceedings - 2017 International Conference on Promising Electronic Technologies, ICPET 2017 26–31. doi: 10.1109/ICPET.2017.11.

Martínez, E. (2017). *7 de Los Ciberataques Más Famosos de La Historia.* Tendencias y Tecnología. Retrieved (http://www.icorp.com.mx/blog/ciberataques-mas-famosos/).

Mewara, Bhawna, Sheetal Bairwa, Jyoti Gajrani, & Vinesh Jain. (2015). *Enhanced Browser Defense for Non-Persistent Cross-Site Scripting.* Proceedings - 2014 3rd International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2014. doi: 10.1109/ICRITO.2014.7014761.

Universidad Nacional Autonoma, (2019). *Documentos De Seguridad Informática.* Seguridad de Información. Retrieved (https://www.seguridad.unam.mx/historico/documento/index.html).

Mohammadi, Mahmoud, Bill Chu, Heather Richter Lipford, & Emerson Murphy-Hill. (2016). *Automatic Web Security Unit Testing: XSS Vulnerability Detection.* Proceedings of the 11th International Workshop on Automation of Software Test - AST ’16 78–84. doi: 10.1145/2896921.2896929.

Mohammadi, Mahmoud, Bill Chu, Heather Richter Lipford, & Murphy-Hill, E. (2016). *Automatic Web Security Unit Testing*. 78–84. doi: 10.1145/2896921.2896929.

Mokbal, Fawaz Mahiuob, M., Wang Dan, Azhar Imran, Lin Jiuchuan, Faheem Akhtar, & Wang Xiaoxi. (2019). *MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique.* IEEE Access 7:100567–80. doi: 10.1109/access.2019.2927417.

Montero, I. (2002). *Classification and Description of Research Methodologies*. Madrid: International Journal of Clinical adn Health Psychology, 1-7.

Mourad, Azzam, Hadi Otrok, & Sara Ayoubi. (2011). *Toward Systematic Integration of Security Policies into Web Services.* Proceedings - 2011 European Intelligence and Security Informatics Conference, EISIC 2011 220–23. doi: 10.1109/EISIC.2011.48.

Nguyen, Trong Kha, & Seong Oun, H. (2017). *Large-Scale Detection of DOM-Based XSS Based on Publisher and Subscriber Model. Proceedings - 2016 International Conference on Computational Science and Computational Intelligence, CSCI 2016 975–80. doi: 10.1109/CSCI.2016.0187.*

Nirmal, K., Janet, L., & Kumar, M. (2018). *Web Application Vulnerabilities - The Hacker's Treasure.* Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018 (Icirca):58–62. doi: 10.1109/ICIRCA.2018.8597221.

Pacheco Ortega, R. (2009). *Códigos Maliciosos.* Sevilla: TAC Seguridad, 1-21.

Pranathi, K., Kranthi, S., Srisaila, A., & Madhavilatha, P. (2018). *Attacks on Web Application Caused by Cross Site Scripting.* Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018 (Iceca):1754–59. doi: 10.1109/ICECA.2018.8474765.

Prassi, T. (2018). *Ataques a Aplicaciones Web: Fuente Más Frecuente de Fuga de Datos.* 2018 Ataques Web.

Godoy Lemus, R. (2014). *Seguridad de La Información.* Guatemala: Memory deseart, 1-10.

Ruse, Michelle, & Samik Basu. (2013). *Detecting Cross-Site Scripting Vulnerability Using Concolic Testing.* Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013 633–38. doi: 10.1109/ITNG.2013.97.

Santos, R., Ordinez, L., & Eggly, G. (2019). *El Enfoque de Cajas Negra y Blanca Para La Enseñanza de Sistemas Embebidos.*

Shrivastava, Ankit, Santosh Choudhary, & Ashish Kumar. (2017). *XSS Vulnerability Assessment and Prevention in Web Application.* Proceedings on 2016 2nd International Conference on Next Generation Computing Technologies, NGCT 2016 (October):850–53. doi: 10.1109/NGCT.2016.7877529.

Singh Ahuja, Mini, Dr Jatinder Singh Bal, & Varnica. (2014). *Web Crawler: Extracting the Web Data*. International Journal of Computer Trends and Technology 13(3):132–37. doi: 10.14445/22312803/ijctt-v13p128.

Soleimani, Hamed, Mohmmad Ali Hadavi, & Arash Bagherdaei. (2018). *WAVE:*

Black Box Detection of XSS, CSRF and Information Leakage Vulnerabilities. 2017 14th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology, ISCISC 2017 99–104. doi: 10.1109/ISCISC.2017.8488361.

Sommerville, I. (2011). *Pruebas de Software.* Colombia: Libros de Ingeniería 206–208.

Subramaniyaswamy, V., Kalyani Gopireddy, V., & Likhitha Naladala. (2018). *Securing Web Applications from Malware Attacks Using Hybrid Feature Extraction in XSS.* International Journal of Pure and Applied Mathematics 119(12):13367–85.

Tarazona, C. (2006). *Amenazas informáticas y seguridad de la información*. CSI/FBI Computer Crime and Security Survey 137–46.

Voutssas, M. (2010). *Preservación Documental Digital y Seguridad Informática.* Investigación Bibliotecológica 24(50):127–55.

Wang, Rui, Xiaoqi Jia, Qinlei Li, & Daojuan Zhang. (2015). *Improved N-Gram Approach for Cross-Site Scripting Detection in Online Social Network.* Proceedings of the 2015 Science and Information Conference, SAI 2015 1206–12. doi: 10.1109/SAI.2015.7237298.

ANEXOS.

ANEXO 1: Encuesta hacia los programadores sobre la seguridad de aplicaciones web al momento de escribir el código fuente.

UNIVERSIDAD SEÑOR DE SIPÁN	Encuesta		
	Tema	Dirigido para:	Programadores web
	SEGURIDAD DE APLICACIONES WEB AL MOMENTO DE ESCRIBIR EL CÓDIGO FUENTE	Versión:	00
		Hoja:	100-120

ENCUESTA HACIA LOS PROGRAMADORES SOBRE LA SEGURIDAD DE APLICACIONES WEB AL MOMENTO DE ESCRIBIR EL CÓDIGO FUENTE

1. ¿Usted desarrolla aplicaciones web?

   SI ☐ NO ☐

2. ¿Qué tiempo lleva desarrollando aplicaciones web?

   _____

3. ¿Utiliza algún Framework para desarrollar aplicaciones web?

   SI ☐ NO ☐

4. ¿Qué es más importante en el desarrollo de la aplicación web?

   Que cumpla con todos los procesos requeridos por el usuario ☐

   Que sea adaptable a todo tipo de dispositivo “responsive” ☐

   Que la seguridad de información y usuario sea prioridad ☐

   Que el interfaz sea agradable para el usuario ☐

5. ¿Las aplicaciones web desarrolladas por usted son para microempresas?

   SI ☐ NO ☐

6. ¿Qué tiempo aproximado demora en desarrollar una aplicación web?

   _____

7. ¿Se fija en la seguridad de la aplicación web desarrollada?

   SI ☐ NO ☐

8. ¿Escribe el código fuente de la aplicación web tomando como prioridad los procesos que debe cumplir la aplicación web?

   SI ☐ NO ☐

9. ¿Realiza pruebas de seguridad en la aplicación web desarrollada?

SI ☐ NO ☐

10. ¿Usted aplica reglas de prevención de vulnerabilidad de código fuente de la aplicación web desarrollada?

SI ☐ NO ☐

11. ¿Alguna vez a intentado atacar a su misma aplicación web que ha desarrollado?

SI ☐ NO ☐

ANEXO 2: Resolución de aprobación del proyecto de investigación

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N° 0930-2019/FIAU-USS

Chiclayo, 22 de julio de 2019

VISTO:

El Acta de Reunión N° de fecha 22 de julio de 2019,, para la ejecución de la Tesis titulada: *"COMPARACIÓN DE TÉCNICAS DE DETECCIÓN DE VULNERABILIDADES DE ATAQUES CROSS SITE SCRIPTING (XSS) EN APLICACIONES WEB DE MICROEMPRESAS"*, presentada por el(los) estudiante(s) **BOCANEGRA CHÁVEZ CRISTIAN ALEXANDER** de la Escuela Académico Profesional de **INGENIERÍA DE SISTEMAS** y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: *"La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";*

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO 1°: APROBAR, el Proyecto de Tesis denominado ***"COMPARACIÓN DE TÉCNICAS DE DETECCIÓN DE VULNERABILIDADES DE ATAQUES CROSS SITE SCRIPTING (XSS) EN APLICACIONES WEB DE MICROEMPRESAS"***, perteneciente a la Línea de Investigación **TECNOLOGÍAS DE LA INFORMACIÓN - SEGURIDAD INFORMÁTICA,** a cargo del(los) estudiante(s) **BOCANEGRA CHÁVEZ CRISTIAN ALEXANDER,** de la Escuela Académico Profesional de **INGENIERÍA DE SISTEMAS.**

ARTÍCULO 2°: ESTABLECER, que la inscripción de la Tesis se realice a partir de emitida la presente resolución, y tendrá una vigencia máxima de 02 años.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

UNIVERSIDAD SEÑOR DE SIPÁN S.A.
Dr. Andres Alberto Ruiz Gómez
DECANO DE LA FACULTAD DE INGENIERIA ARQUITECTURA Y URBANISMO

UNIVERSIDAD SEÑOR DE SIPÁN S.A.C.
Mg. Luis Roberto Larrea Colchado
SEC. ACADÉMICO FACULTAD DE INGENIERÍA ARQUITECTURA Y URBANISMO

Cc: Dirección de Investigación, CPGYT, Interesados, Archivo

ADMISIÓN E INFORMES
074 481610 - 074 481632
CAMPUS USS
Km. 5, carretera a Pimentel
Chiclayo, **Perú**

www.uss.edu.pe

ANEXO 3: Revisión y Selección de Técnicas de tipo Cross Site Scripting

Sitio web de la Plataforma SCImago Journal & Country Rank (SJR), que se utilizara para clasificar las bases de datos en la cual se realizan la búsqueda de los artículos relacionados.

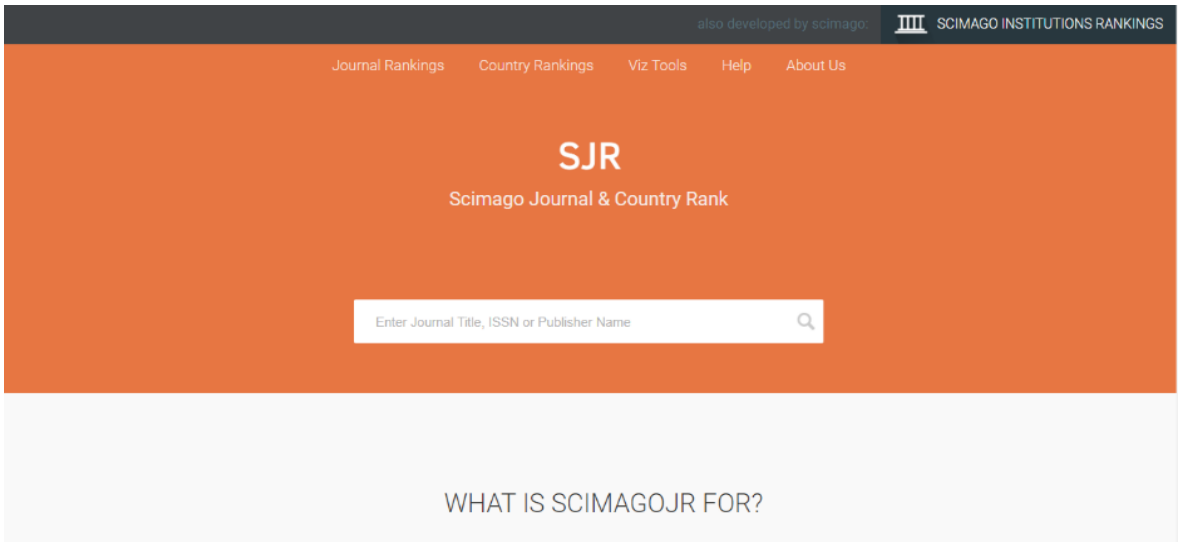

Figura 46: Figura 35: Sitio Web Scimago Journal & Country Rank (SJR). Fuente: scimagojr.com

Resultados que se obtiene al buscar detección de vulnerabilidades de ataques de Cross Site Scripting.

	Title	Type	SJR	H index	Total Docs. (2018)	Total Docs. (3years)	Total Refs. (2018)	Total Cites (3years)	Citable Docs. (3years)	Cites / Doc. (2years)	Ref. / Doc. (2018)
1	IEEE Transactions on Cybernetics	journal	3.548 Q1	81	619	898	12590	10010	878	11.47	20.34
2	IEEE Transactions on Automatic Control	journal	3.233 Q1	260	677	1495	14169	9981	1484	6.20	20.93
3	Researchgate	journal	2.875 Q1	236	677	1495	14169	9981	1484	6.20	20.93
4	Foundations and Trends in Systems and Control	journal	2.875 Q1	7	5	8	534	53	8	4.80	106.80
5	IEEE Transactions on Fuzzy Systems	journal	2.794 Q1	170	407	457	13604	4245	451	8.80	33.43
6	33rd International Conference on Machine Learning, ICML 2016	conference and proceedings	2.718	38	0	342	0	4172	337	5.14	0.00
7	IEEE Transactions on Industrial Electronics	journal	2.400 Q1	236	1190	2627	27531	25178	2605	8.70	23.14

Figura 47: Resultado del filtro de la plataforma SCImago Journal & Country Rank. Fuente: scimagojr.com

Resultado de la búsqueda de investigaciones que implementan técnicas de detección de vulnerabilidades de ataques Cross Site Scripting.

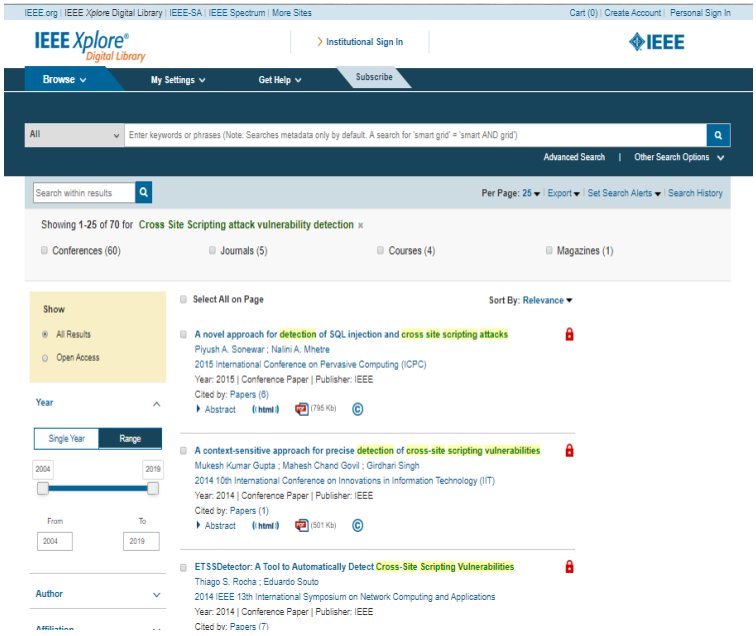

Figura 48: Resultado crudo de la búsqueda IEEE. Fuente: ieeexplore.ieee.org/

ANEXO 4: Descarga e instalación de Software de Virtualización Virtual Box

Se descargará el software de virtualización VirtualBox versión 6.0.14. Se acudió al sitio web del software, donde se ve claramente que se puede utilizar para varias plataformas. Para la presente investigación se necesita para el Sistema Operativo Windows.

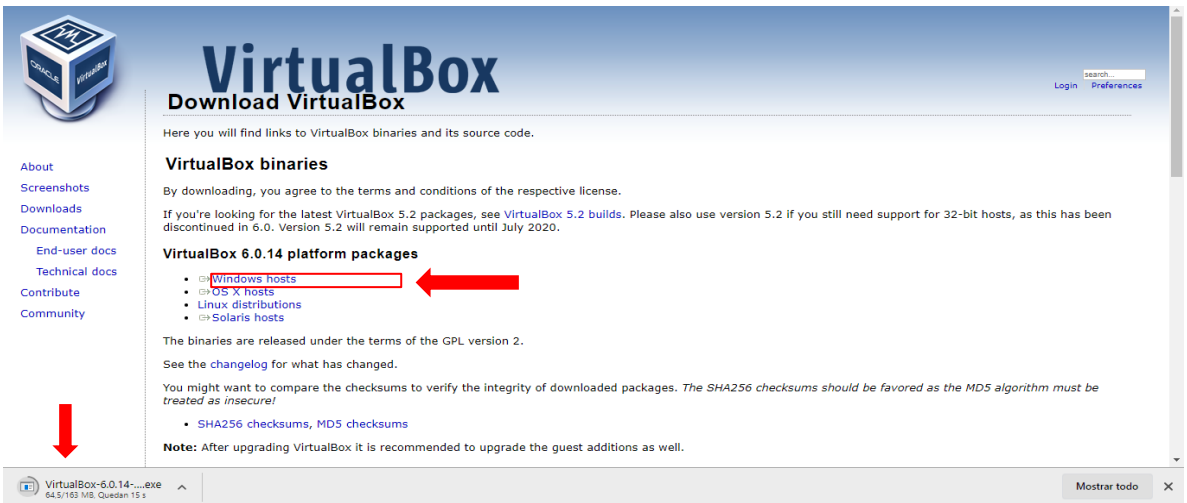

Figura 49: Sitio Web de descarga del software de virtualización VirtualBox versión 6.0.14. Fuente: virtualbox.org/wiki/Downloads

Para la instalación de VirtualBox, primero se tiene que ejecutar como administrador en el ordenador, seguido de ello aparecerá una pantalla de bienvenida, en el cual se dará clic en el botón ‘Next >’ para continuar con la instalación

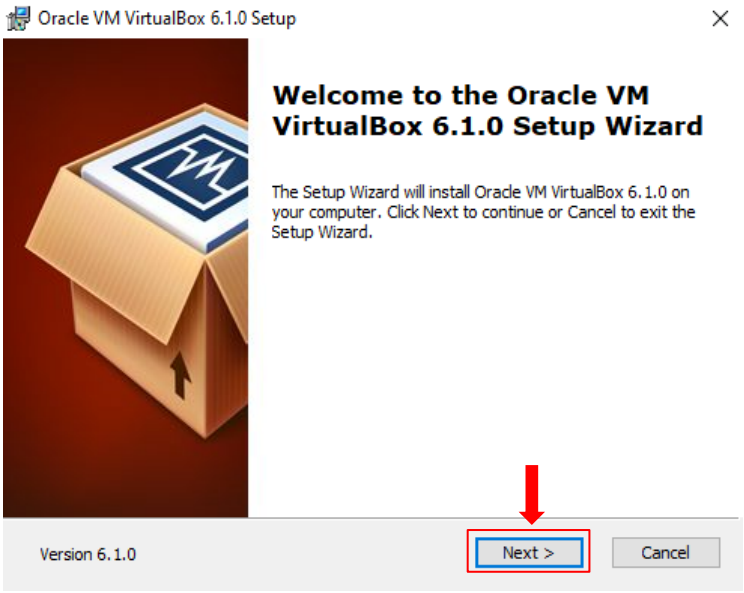

Figura 50: Ventana de bienvenida. Fuente: Instalación Virtual Box

Luego aparecerá la pantalla de seleccionar la forma en que se desea instalar las funciones y la ubicación de la instalación del VirtualBox. Para dicho trabajo se dejará tal como está y se dará clic en ‘Next >’ para avanzar con la instalación.

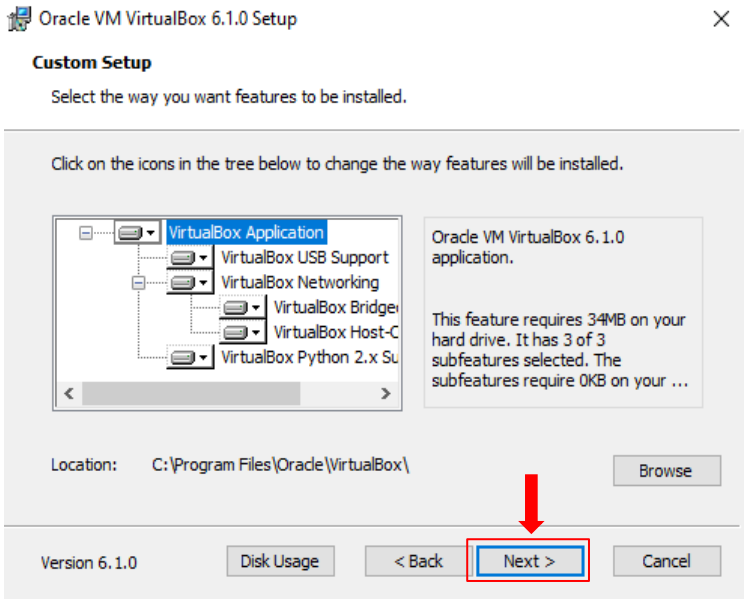

Figura 51: Ventana de localización. Fuente: Instalación Virtual Box

Después en la siguiente pantalla de selección de casilleros de las funciones que se realizará la instalación. Propiamente se dejará tal como está y se dará clic en 'Next'

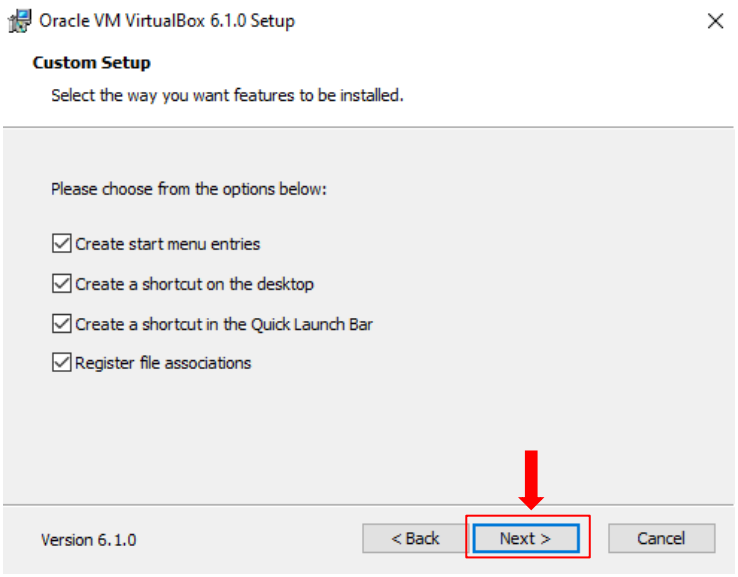

Figura 52: Ventana de selección de las funciones. Fuente: Instalación Virtual Box

Seguidamente terminará la configuración de la instalación de VirtualBox, por lo tanto, se dará clic en 'Yes' para proceder con la instalación en el ordenador.

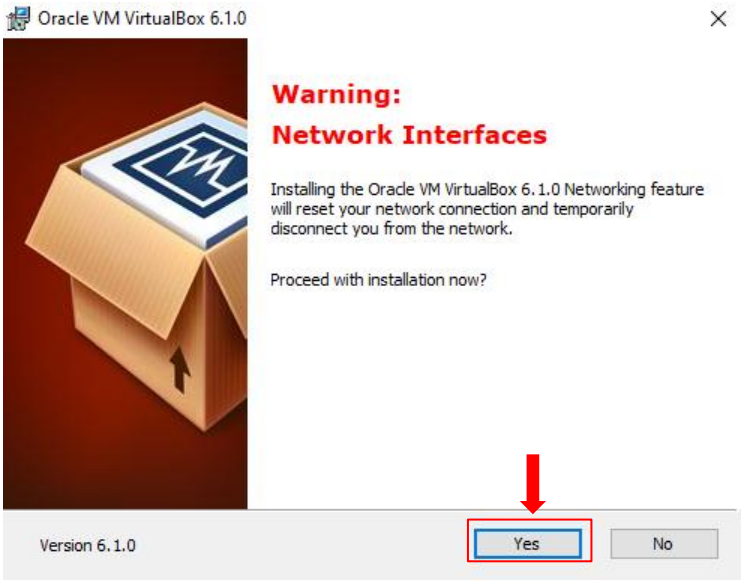

Figura 53: Ventana de culminación de la configuración. Fuente: Instalación Virtual Box

Luego iniciará con la instalación en el ordenador

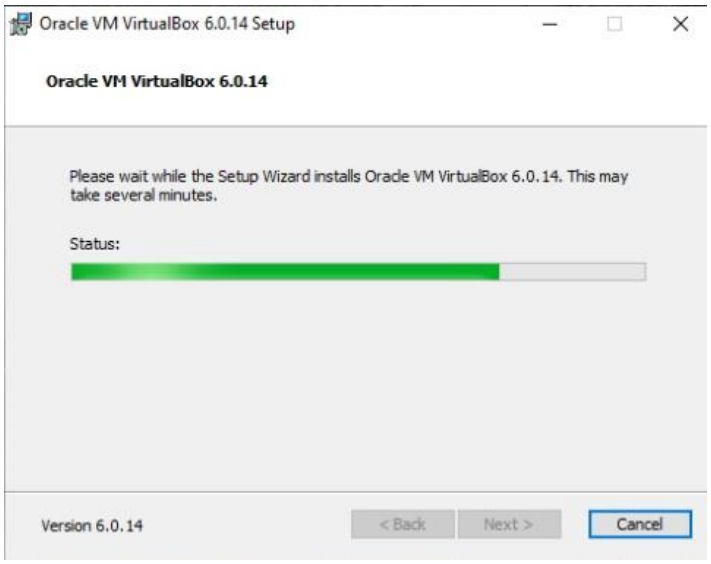

Figura 54: Ventana del proceso de instalación. Fuente: Instalación Virtual Box

Finalmente se terminará la instalación y se procederá a dar clic en 'Finish' para concluir, dejando marcado el casillero 'Start Oracle VM VirtualBox 6.0.14 after installation' para iniciar con el software

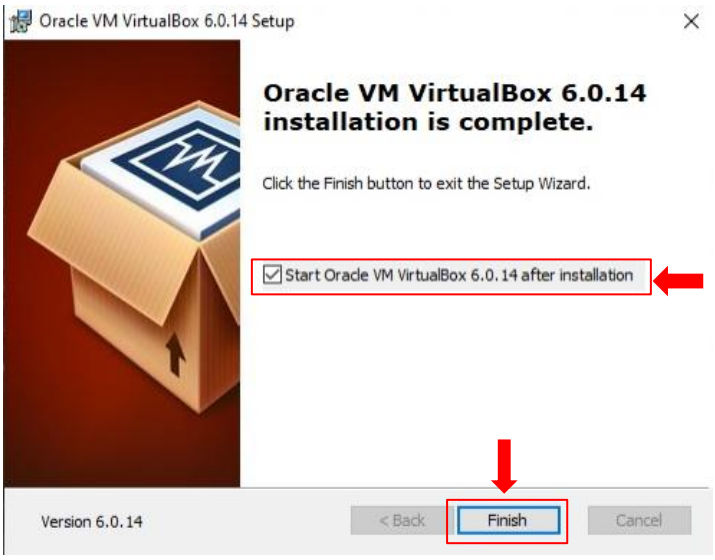

Figura 55: Ventana de finalización de la instalación. Fuente: Instalación Virtual Box

ANEXO 5: Descarga de Kali Linux

Descargar la imagen ISO de Kali Linux desde su sitio web. Se trabajará con la versión 2019.3 en 64 bit, para ello se descargará haciendo clic en la opción 'Kali Linux 64-bit VirtualBox' direccionando a otra ventana de la web.

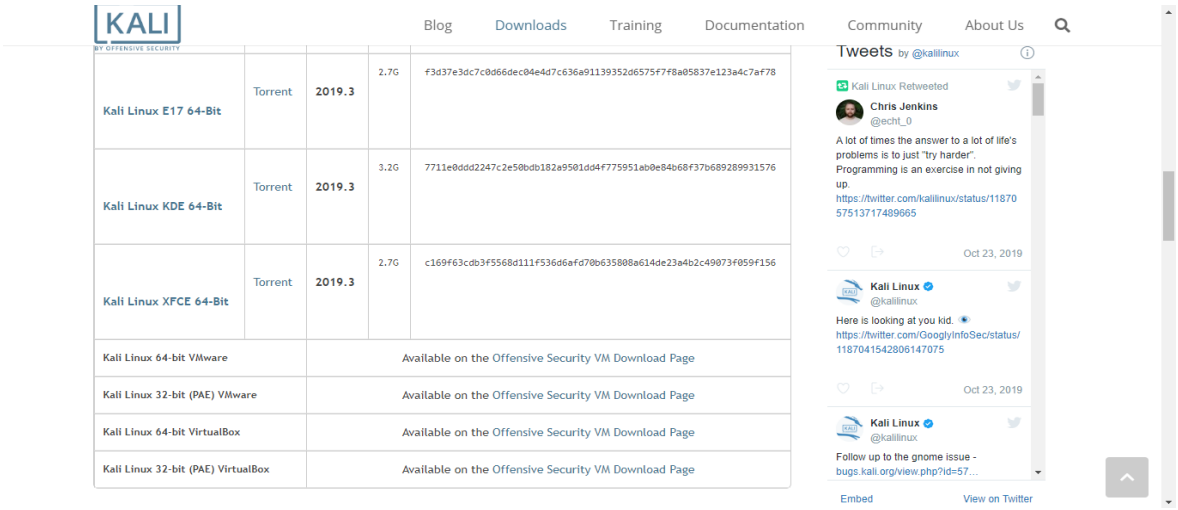

Figura 56: Sitio Web de descarga de Kali Linux. Fuente: kali.org/downloads/

Seguidamente direcciona a otro sitio web llamado 'offensive-security' para continuar con la descarga. Se dará clic en la opción 'Kali Linux VirtualBox 64-Bit' dando origen a la descarga automática de Kali Linux.

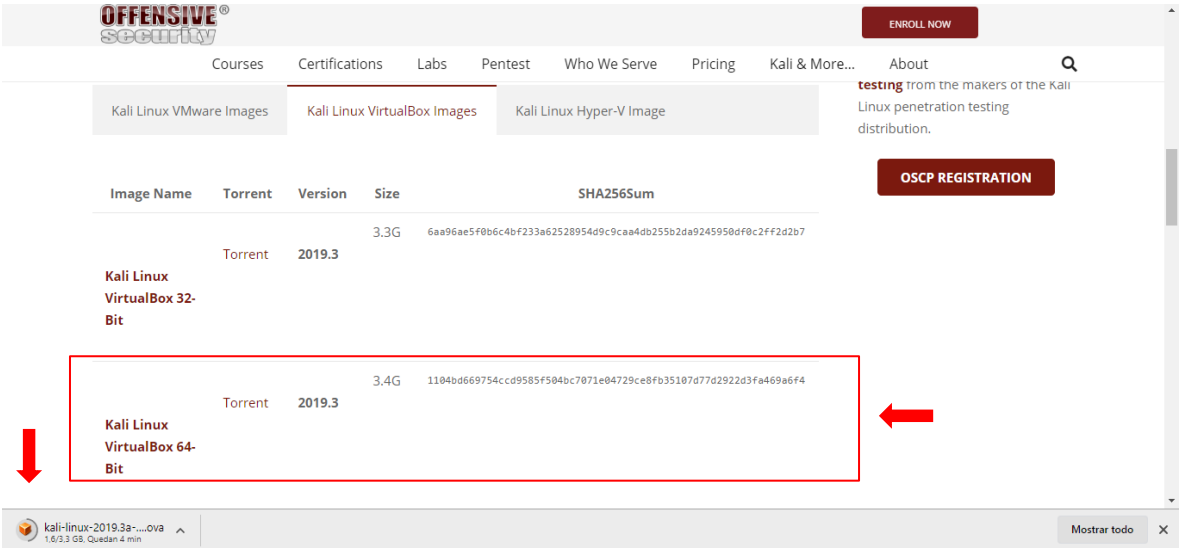

Figura 57: Sitio Web de descarga de la Imagen ISO Kali Linux. Fuente: kali.org/downloads/

ANEXO 6: Montar Imagen ISO Kali Linux en Virtual Box

Virtualizar la ISO Kali Linux en VirtualBox. Para ello se debe dar clic en la opción 'Nueva'.

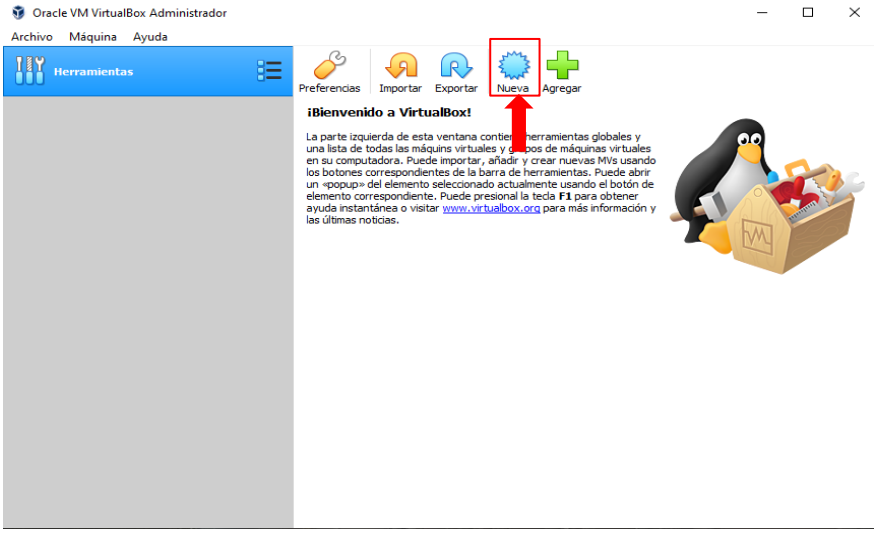

Figura 58: Ventana principal VirtualBox donde se montará la Imagen ISO de Kali Linux. Fuente: Programa de Virtual Box

En la siguiente ventana se configura la creación de la máquina virtual, escribiendo un nombre y eligiendo el sistema operativo. Se escribe 'Kali Linux', se selecciona el tipo el sistema operativo 'Linux' y en la versión se selecciona 'Other Linux (-bit)' porque no se encuentra Kali. Finalmente se da clic en 'Next' para seguir con la configuración de la máquina virtual.

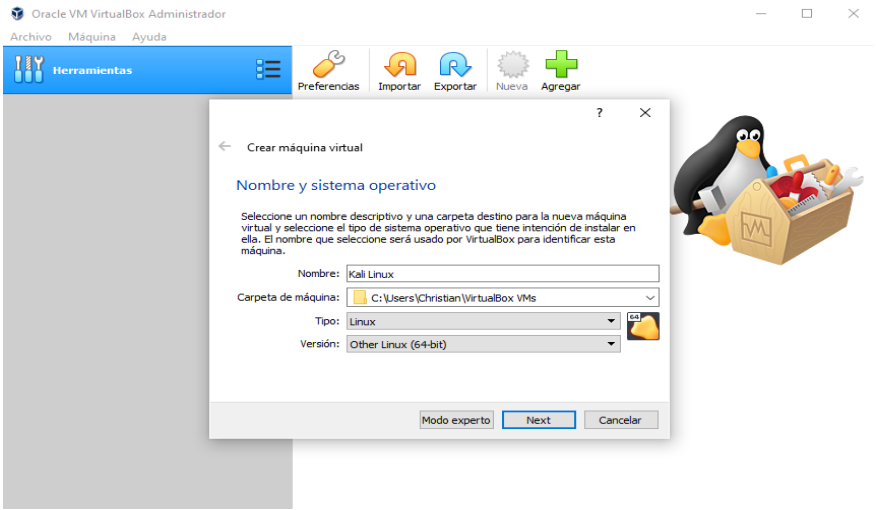

Figura 59: Ventana para crear la máquina virtual con el sistema operativo Kali Linux. Fuente: Programa de Virtual Box

Luego se selecciona el tamaño de memoria de la máquina virtual. Se selecciona 4 MB para Kali Linux.

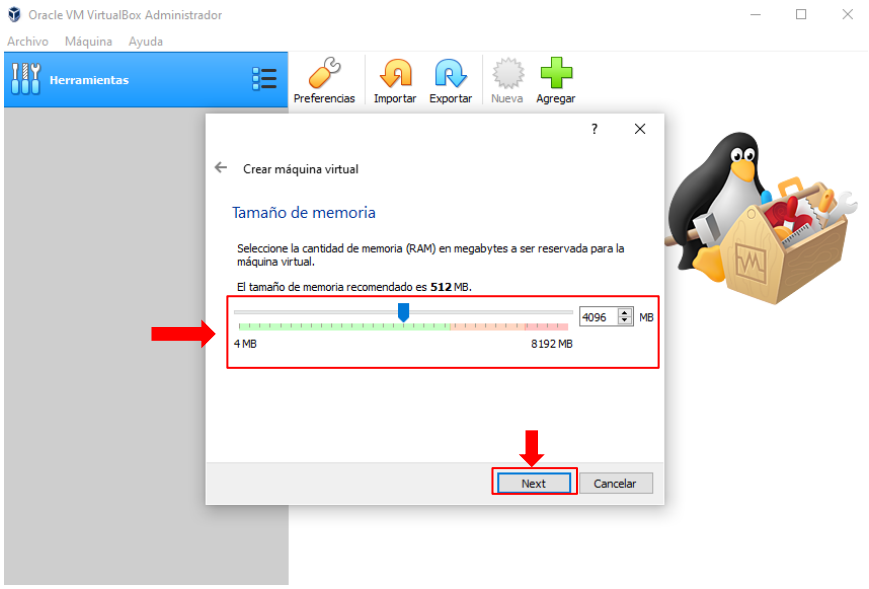

Figura 60: Tamaño de memoria de la máquina virtual Kali Linux. Fuente: Programa Virtual Box

Después se elige el disco duro, para Kali Linux se selecciona en la opción 'Crear un disco duro ahora' y seguido se da clic en 'Crear'.

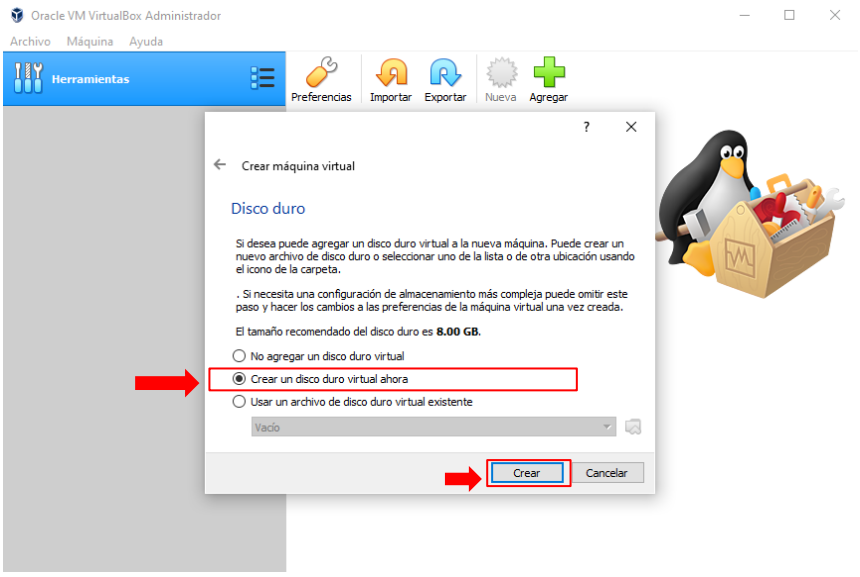

Figura 61: Seleccionar el disco duro de Kali Linux. Fuente: Programa de Virtual Box

Seguidamente aparece la ventana de la creación de disco duro virtual en el cual se seleccionará la opción ‘VDI (VirtualBox Disk Image)’ y se da clic en ‘Next’.

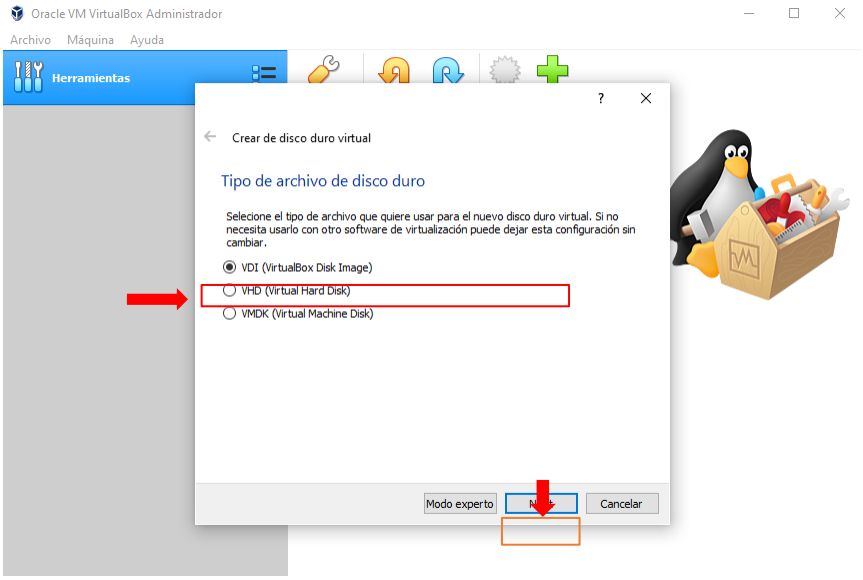

Figura 62: Selección de tipo de archivo del disco duro de Kali Linux. Fuente: Programa Virtual Box

En la siguiente ventana se selecciona el archivo de unidad de disco duro virtual, para Kali Linux se utilizará la opción ‘Reservado dinámico’ y luego clic en ‘Next’

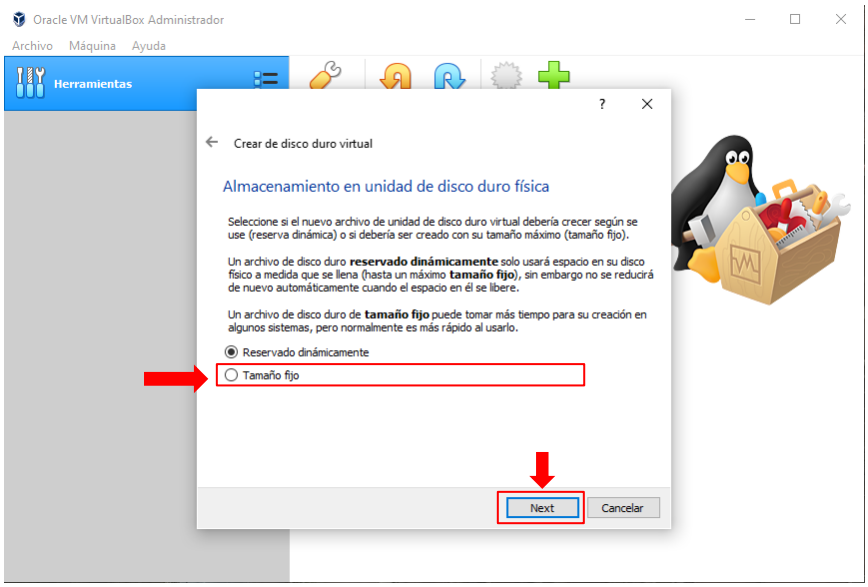

Figura 63: Selecciona el archivo de unidad de disco duro virtual de Kali Linux. Fuente: Programa Virtual Box

Seguidamente se selecciona la ubicación y tamaño que tendrá la máquina virtual Kali Linux. Para dicha creación se utilizará en la misma ubicación que genera el Software, se selecciona 8.00 Gb de tamaño y luego ‘Crear’.

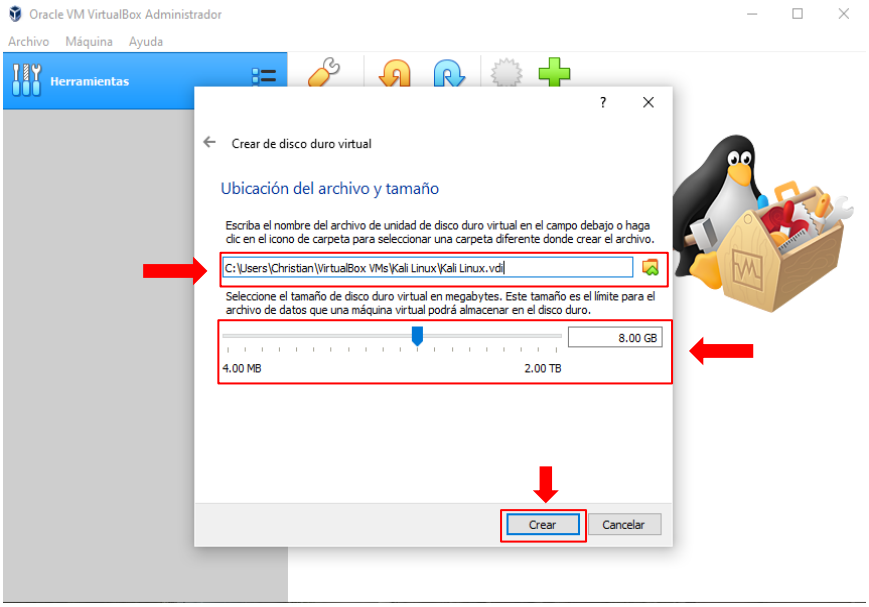

Figura 64: Ubicación y tamaño de la máquina virtual Kali Linux. Fuente: Programa Virtual Box

Finalmente se crea la máquina Virtual

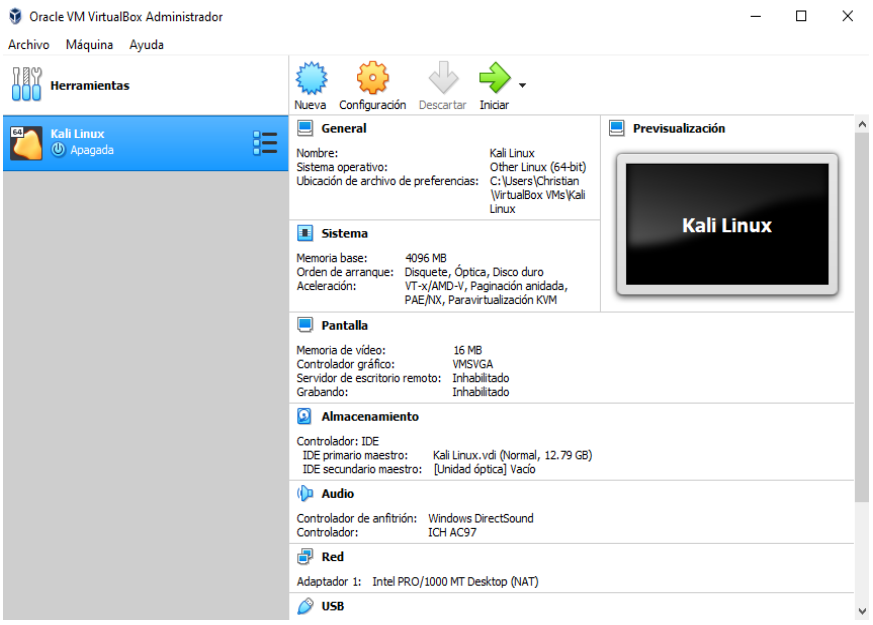

Figura 65: Máquina virtual Kali Linux creada con éxito. Fuente: Programa Virtual Box

Instalación de Kali Linux en la máquina virtual. Se da doble clic en la máquina virtual Kali Linux y se abre una ventana donde se selecciona el disco de inicio, donde se busca dónde está la imagen ISO de Kali Linux descargado anteriormente y se selecciona para empezar con la instalación en la máquina virtual.

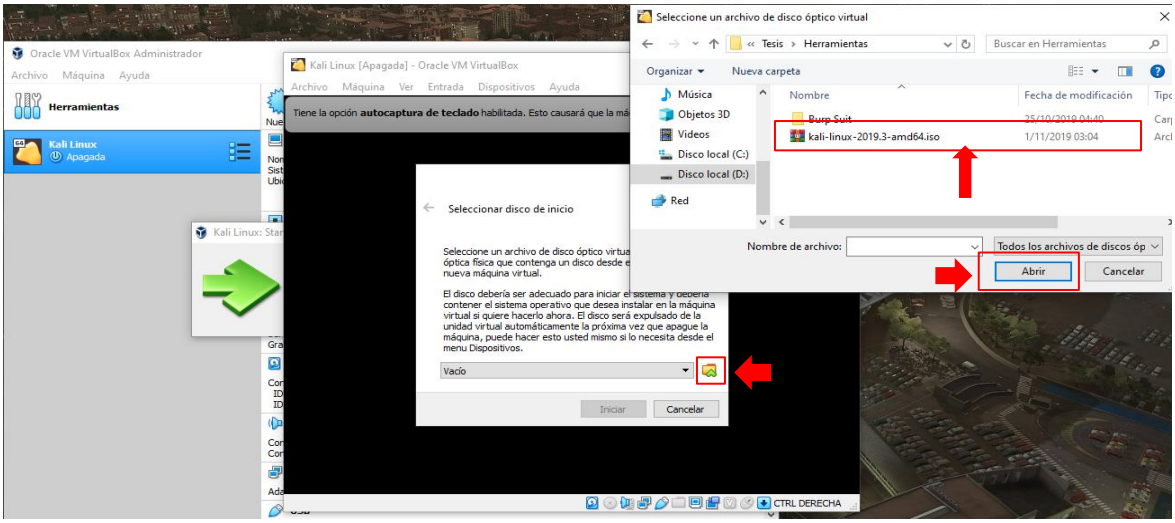

Figura 66: Selección de disco de inicio de la imagen ISO Kali Linux. Fuente: Programa Virtual Box

Luego se selecciona ‘Graphical install’ para instalar directamente porque las otras opciones se instalan por consola.

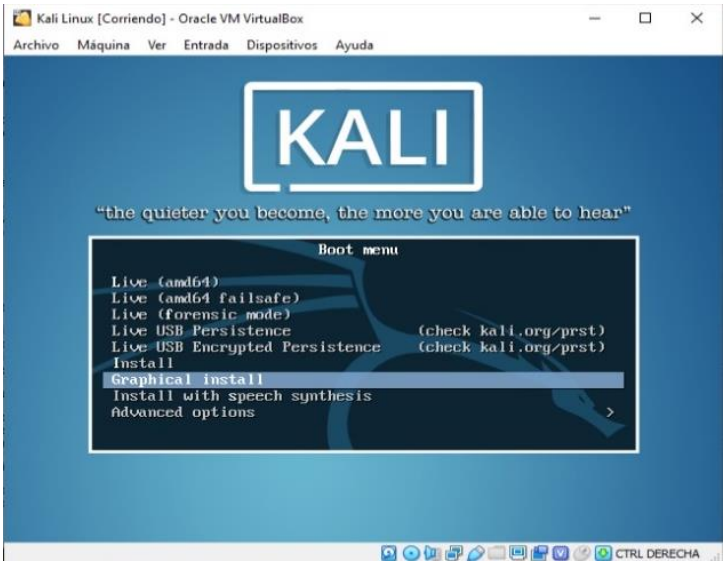

Figura 67: Ventana principal de la Instalación de Kali Linux. Fuente: Programa Virtual Box

Siguiendo con el procedimiento, se selecciona el lenguaje, para ello se busca ‘Spanish’ y se selecciona, luego en ‘Continue’.

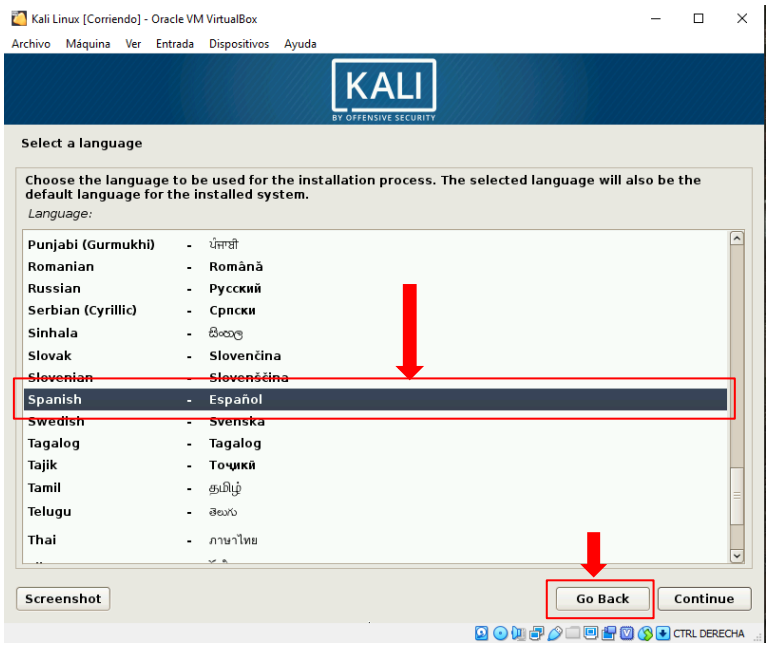

Figura 68: Ventana de selección de lenguaje de la instalación de Kali Linux. Fuente: Programa Virtual Box

Luego se selecciona la ubicación del ordenador, esto para que se fije la zona horaria y fecha. Se busco ‘Perú’ y luego en clic en ‘Continuar’

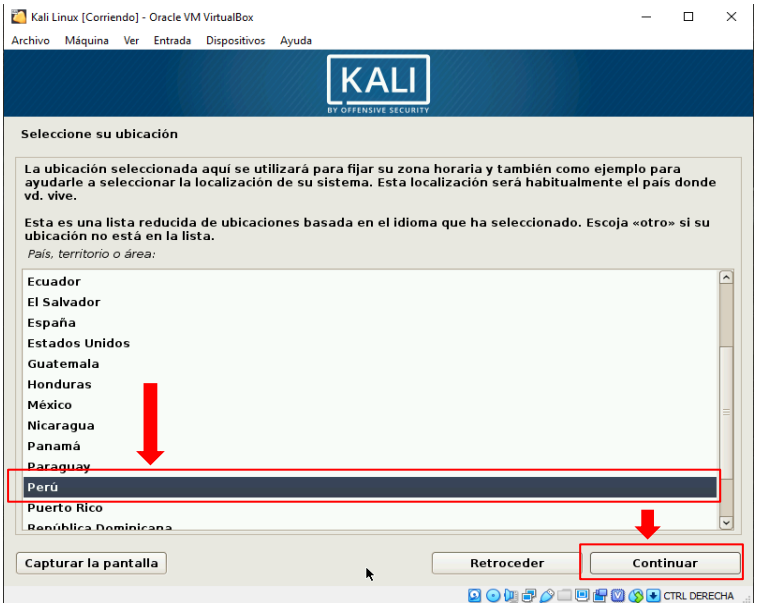

Figura 69: Ventana de selección de la ubicación del ordenador para fijar la zona horaria y fecha de la instalación Kali Linux. Fuente: Programa Virtual Box

En la ventana de selección de teclado, se busca en la entrada de teclado ‘Latinoamérica’ y luego ‘Continuar’.

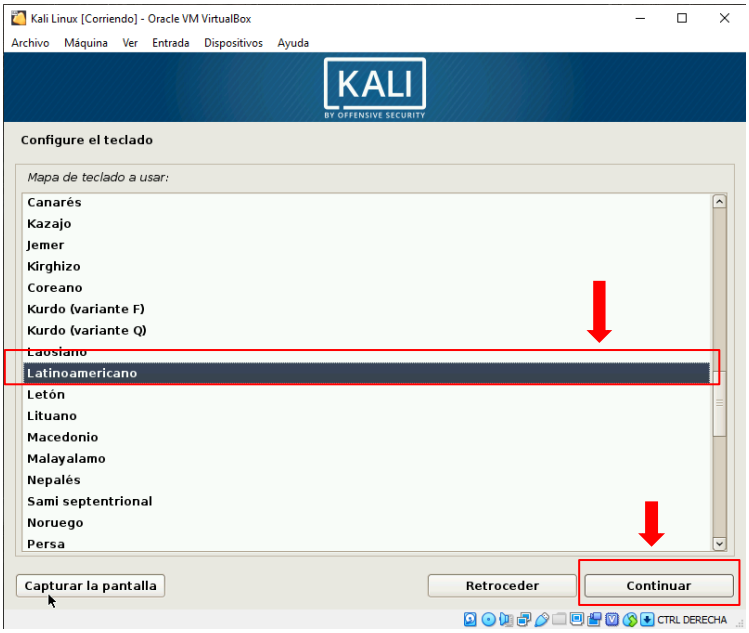

Figura 70: Ventana de selección de teclado de la instalación Kali Linux. Fuente: Programa Virtual Box

Termina la primera fase de la configuración y se empieza a cargar los componentes adicionales que necesita Kali Linux

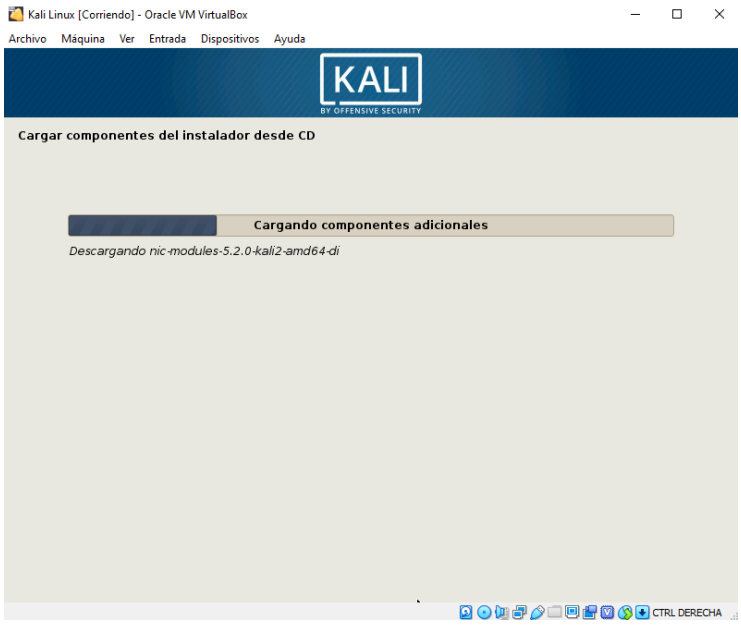

Figura 71: Ventana de carga de componentes adicionales de la instalación Kali Linux. Fuente: Programa Virtual Box

Después de concluir la carga de los componentes adicionales se continua la segunda fase de la instalación, se escribe el nombre de la máquina, el cual se colocará el nombre del investigador y luego en ‘Continuar’

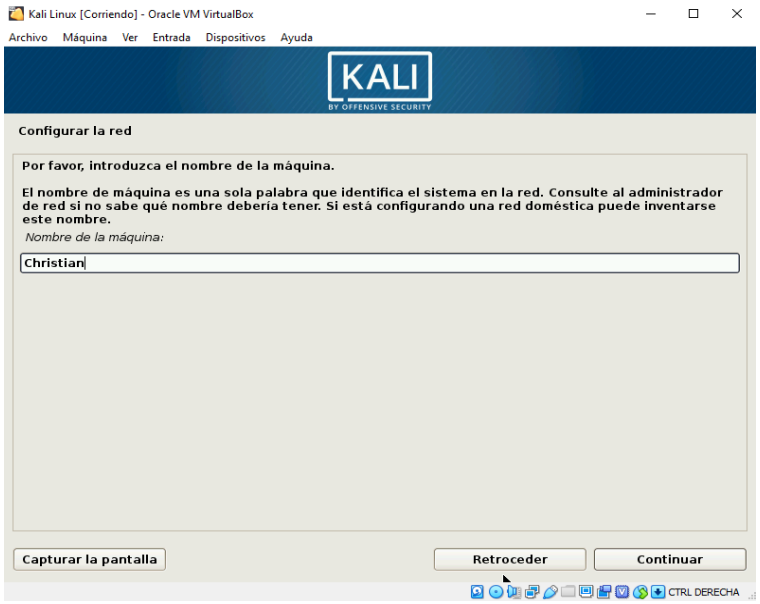

Figura 72: Ventana asignar un nombre a la máquina de la instalación Kali Linux. Fuente: Programa Virtual Box

Después se debe configurar la contraseña, asignándole al Kali Linux una contraseña la cual se debe repetir y deben de coincidir. Seguido se da clic en ‘Continuar’.

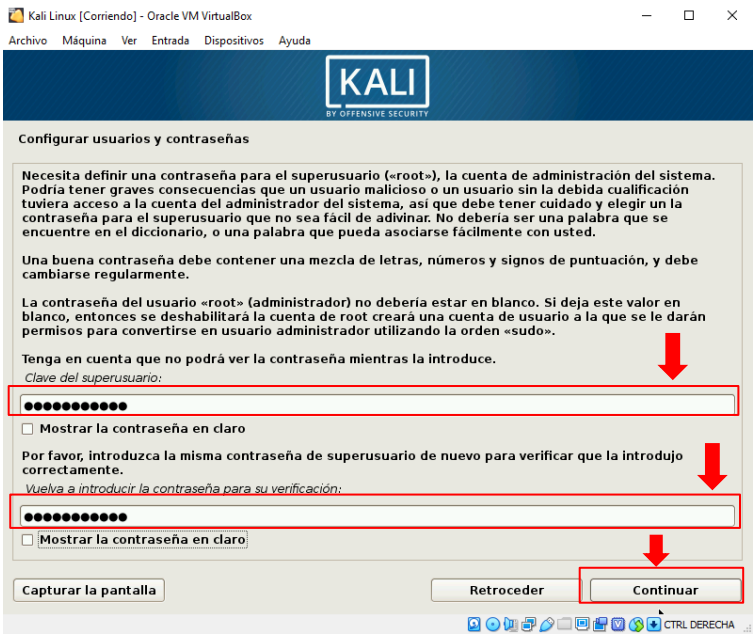

Figura 73: Ventana de asignación de contraseña para la máquina en instalación Kali Linux. Fuente: Programa Virtual Box.

Procede a configurar el reloj, terminando con la segunda fase de la configuración de la instalación, y espera a que avance hacia la siguiente ventana.

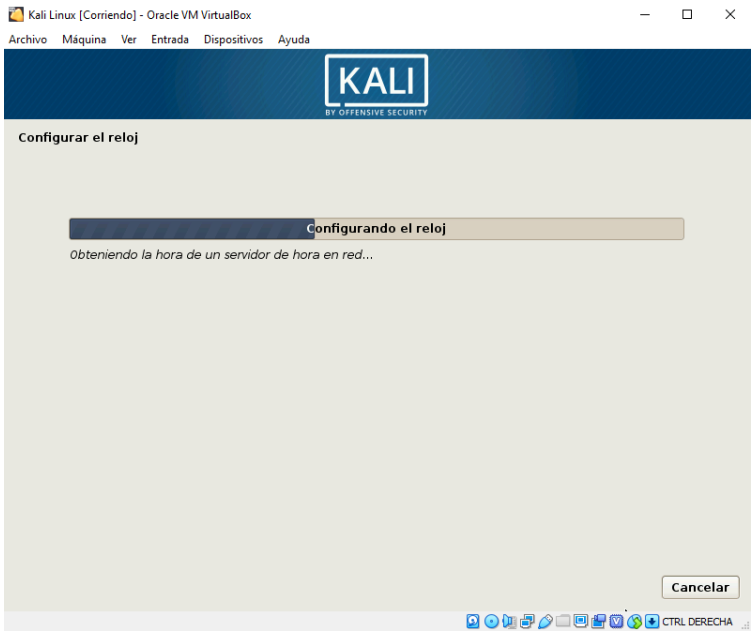

Figura 74: Ventana de configuración automática del reloj de la instalación Kali Linux. Fuente: Programa Virtual Box.

Seguidamente empieza la fase de partición de discos. Se selecciona la partición de discos, en el cual se elige ‘Guiado – utilizar todo el disco’ y luego en ‘Continuar’

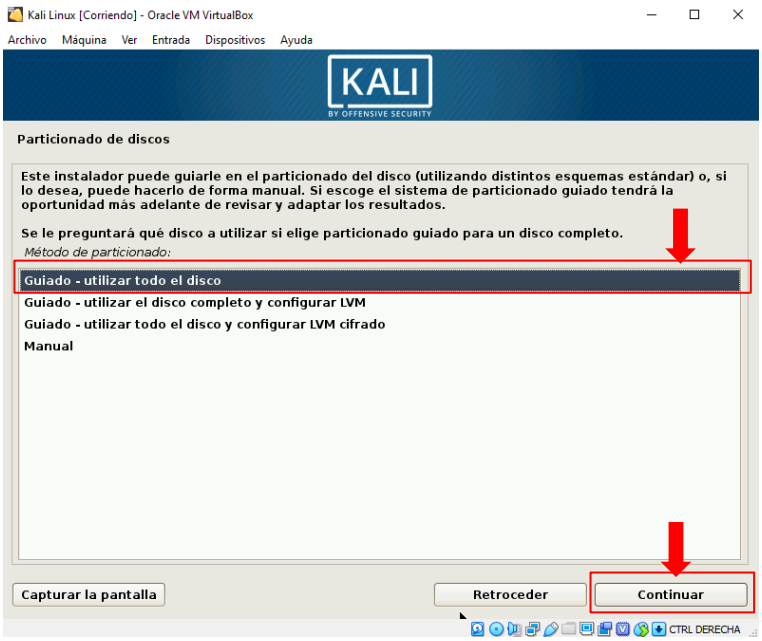

Figura 75: Ventana de selección la partición de discos de la instalación Kali Linux. Fuente: Programa Virtual Box.

Luego se seleccionó el disco donde se borrará los datos almacenados anteriormente (en este caso no hay datos, pero se llega a seguir con la configuración). Se elige el único disco que se tiene y luego en ‘Continuar’.

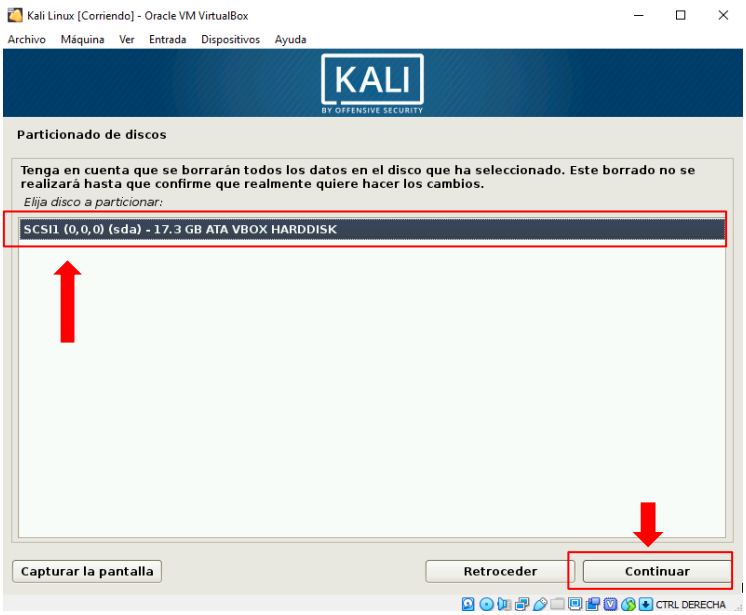

Figura 76: Ventana de selección de disco para borrar los datos almacenados en la instalación Kali Linux. Fuente: Programa Virtual Box.

Termina la partición de discos y los cambios que se realizaron, se selecciona la opción ‘Finalizar el particionado y escribir los cambios en el disco’ y luego en ‘Continuar’.

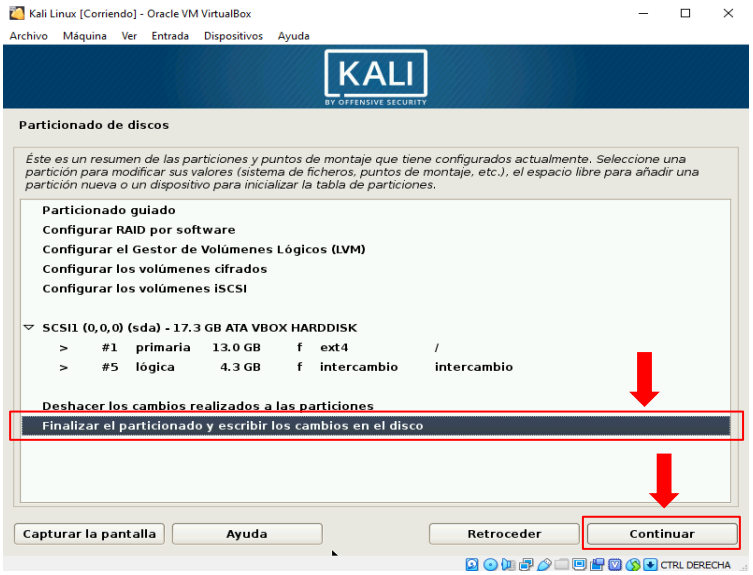

Figura 77: Ventana de Finalizar con la partición de discos y los cambios realizados en la instalación Kali Linux. Fuente: Programa Virtual Box.

Finaliza la fase de partición, solo se selecciona la opción ‘Si’ para terminar con la fase y seguir con la instalación.

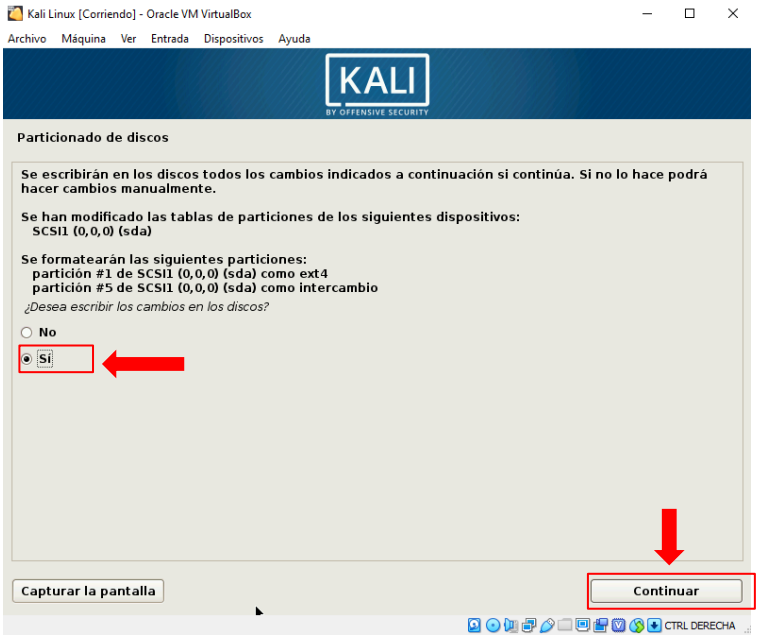

Figura 78: Ventana de finalización de la partición de discos en la instalación Kali Linux. Fuente: Programa Virtual Box.

Después de las fases de configuración de Kali Linux se empezó a instalar.

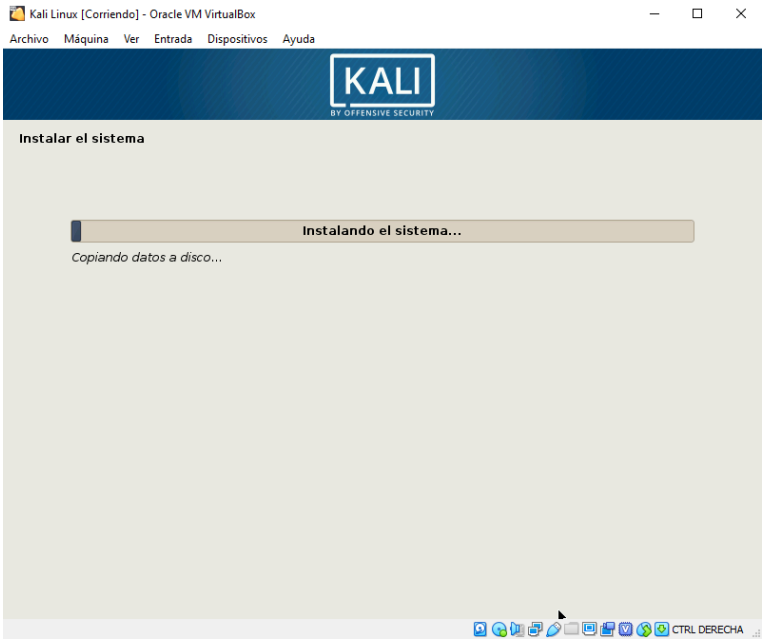

Figura 79: Ventana de proceso de la instalación del sistema Kali Linux. Fuente: Programa Virtual Box.

Después se interrumpe la instalación de Kali Linux para preguntar si se desea utilizar la réplica en red, por lo cual se seleccionó en la opción ‘Si’ y luego en ‘Continuar’.

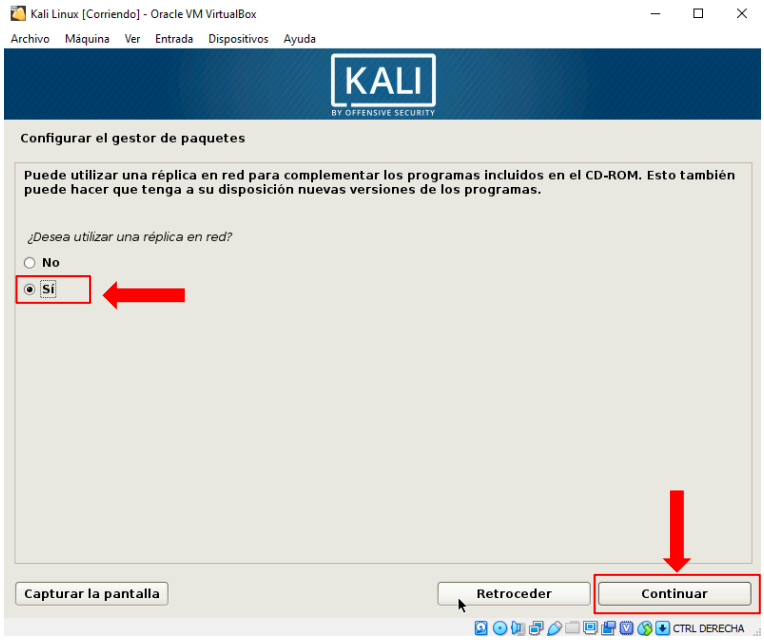

Figura 80: Ventana de configuración del gestor de paquetes en la instalación Kali Linux. Fuente: Programa Virtual Box.

Prosigue con la configuración automática del APT

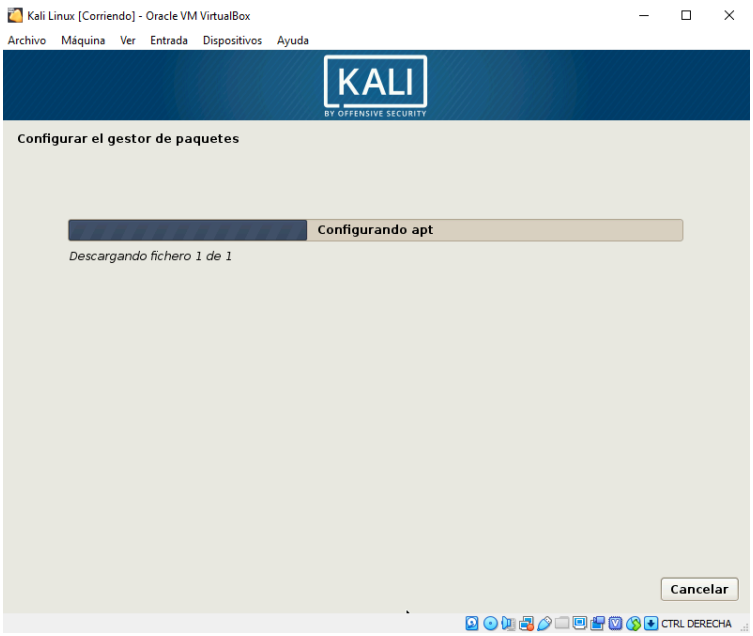

Figura 81: Ventana de configuración automática de gestor de paquetes en la instalación Kali Linux. Fuente: Programa Virtual Box.

Sigue la instalación del cargador de arranque GRUB en el disco duro, por lo cual se secciona en la opción ‘Si’ y luego en ‘Continuar’.

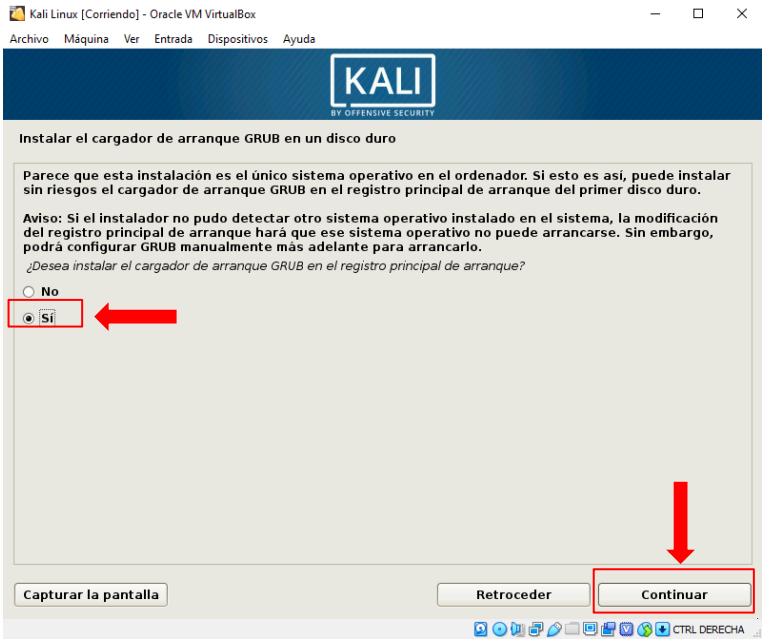

Figura 82: Ventana de instalación del cargador de arranque GRUB en el disco duro en la instalación Kali Linux. Fuente: Programa Virtual Box.

Seguidamente se elige el disco duro, por lo cual se elegio ‘/dev/sda (ata-VBOX_HARDDISK_VB637e3f6f-a845c5b0)’ y luego en ‘Continuar’.

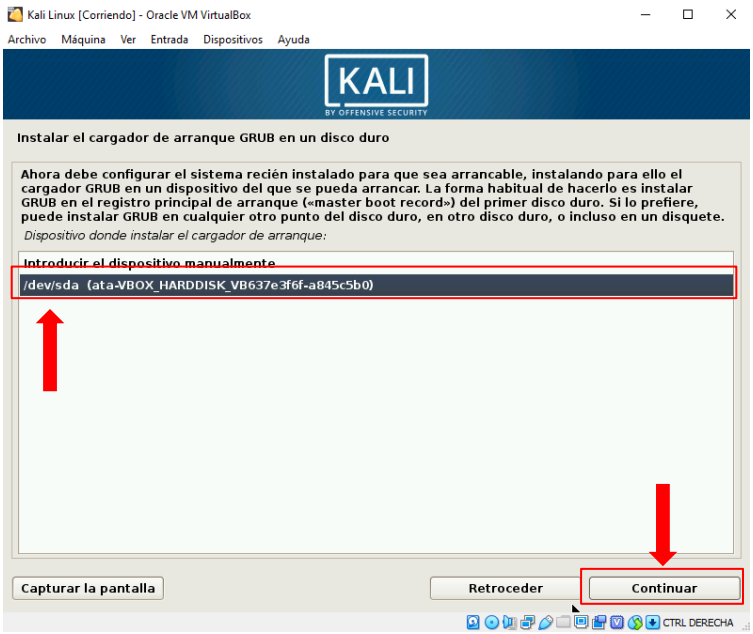

Figura 83: Ventana de selección de disco duro para instalar el arranque GRUB en la instalación Kali Linux. Fuente: Programa Virtual Box.

Se espera que termine el proceso de instalación del cargador de arranque GRUB.

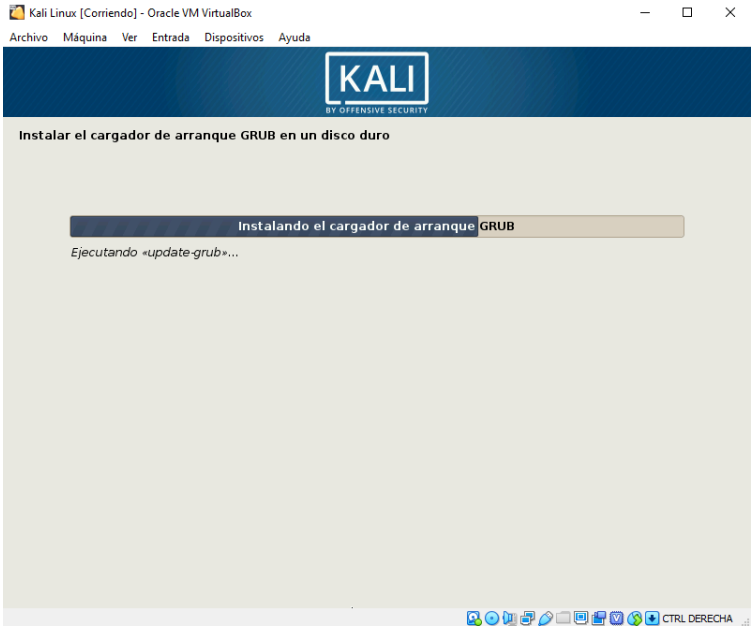

Figura 84: Venta del proceso de instalación del cargador de arranque GRUB en la instalación Kali Linux. Fuente: Programa Virtual Box.

Finalmente se termina la instalación Kali Linux, seguido se da clic en 'Continuar' para iniciar con el sistema operativo.

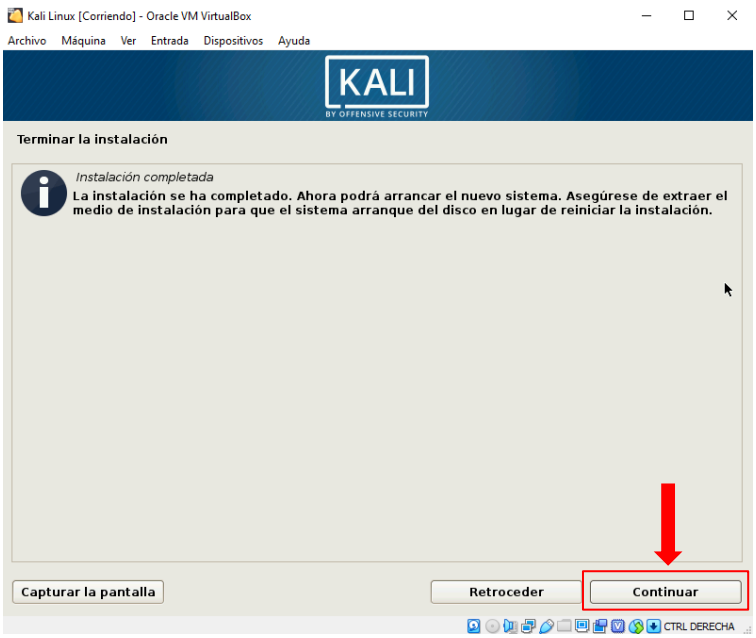

Figura 85: Ventana de finalización de la instalación Kali Linux. Fuente: Programa Virtual Box.

Por último, se inicia el sistema escribiendo el usuario y la contraseña en Kali Linux.

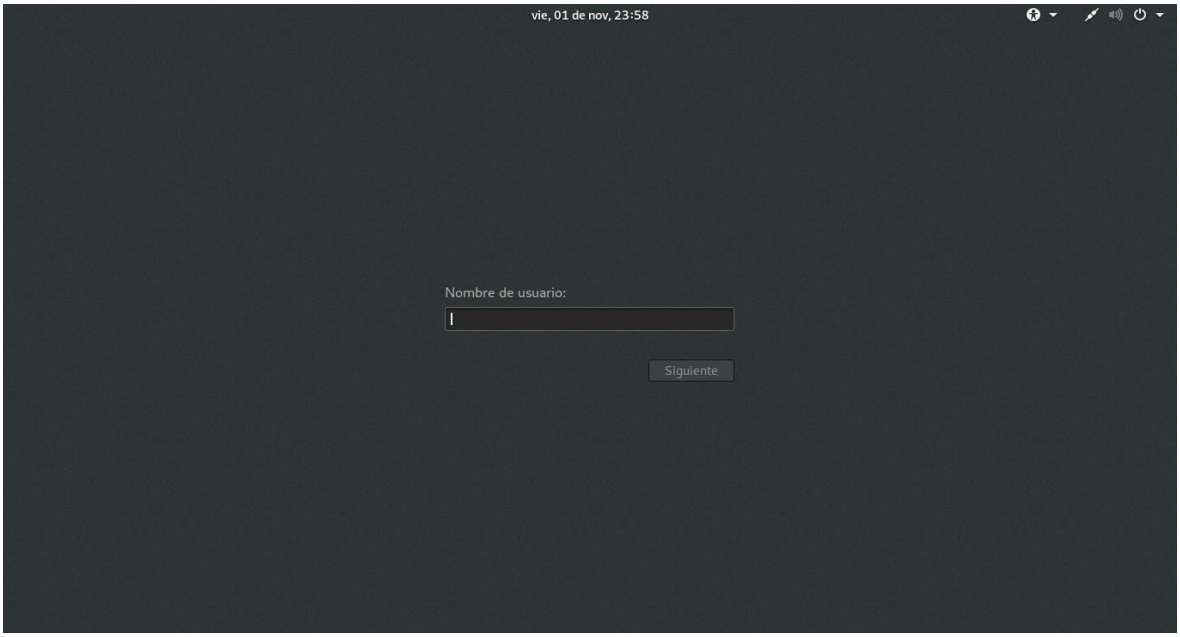

Figura 86: Ventana de Inicio de Sesión de Kali Linux. Fuente: ISO de Kali Linux

El inicio de sesión es correcto y muestra la ventana de inicio de Kali Linux

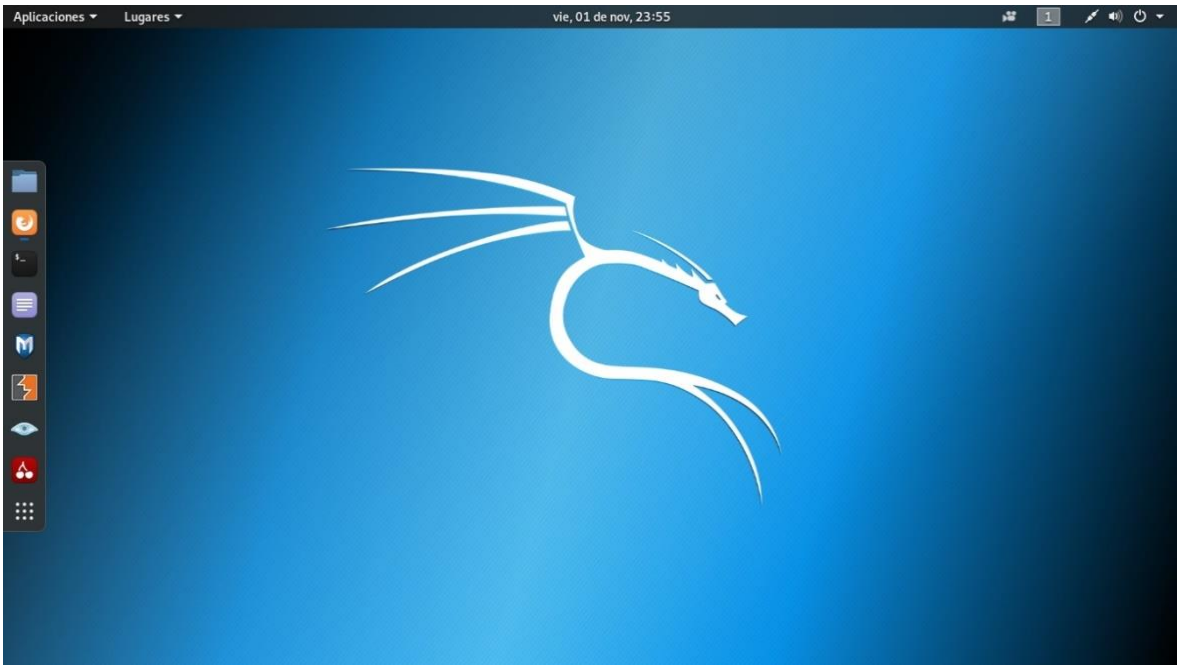

Figura 87: Ventana principal de Kali Linux. Fuente: ISO de Kali Linux

ANEXO 7: Descarga e instalación de Atom, gestor de lenguaje de programación

Se ingresa al sitio web de Atom, luego se da clic en ‘Descargar’

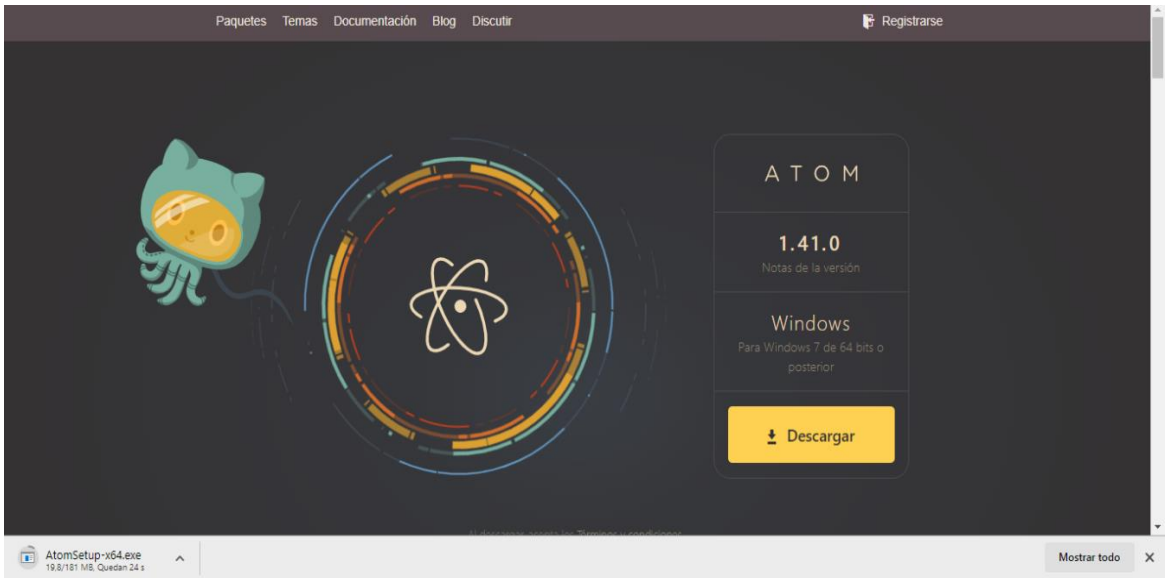

Figura 88: Sitio Web de descarga de Atom. Fuente: atom.io

Se da clic en el software Atom y empieza automáticamente la instalación

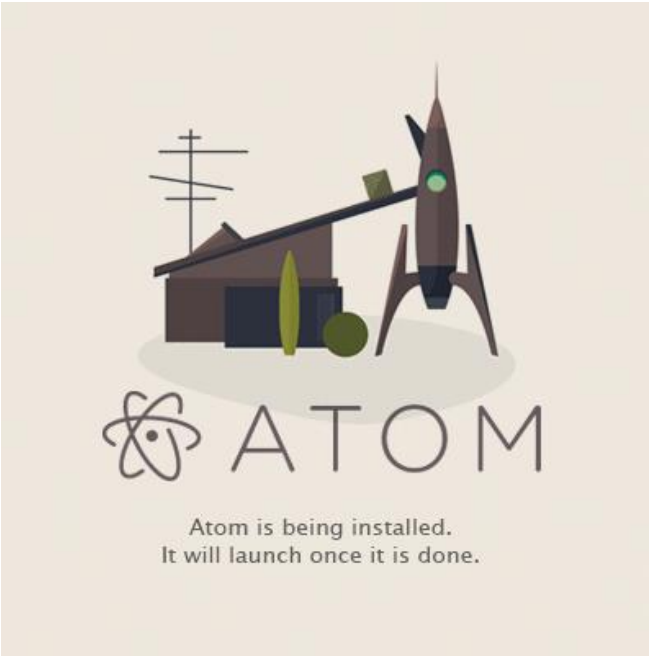

Figura 89: Ventana de instalación de Atom. Fuente: Atom

ANEXO 8: Ficha técnica para la anotación de las pruebas realizadas al caso de estudio por las técnicas DDV y SDV.

Tabla 34:

Ficha técnica creada para la anotación de pruebas realizadas

Técnica Detección Dinámica de Vulnerabilidades (DDV)		
N° de prueba	**Tiempo de ejecución del proceso escáner**	**Tiempo disponible total de la herramienta**
Prueba 1	1:26	1:36
Prueba 2	0:39	0.36
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
Pruebas N	Tiempo N	Tiempo B
Total	Promedio	Promedio

Nota: Elaboración Propia.

ANEXO 9: Selección de Scripts para aplicarlos en el caso de estudios.

Tabla 35:

Selección de Scripts para aplicarlos en el caso de estudio.

Script
<form name="formulario" method="post" action="data.php"> Nombre y apellidos: <input type="text" name="nombre" value=""> <input type="submit" /> </form> <a onmouseover="alert('Ataque XSS!')" <script>alert("Ataque XSS")</script>

```
<script>document.write('<img
src="http://testphp.vulnweb.com/search.php?cookie='%2Bdocument.cookie%2B'
"/>');<script>
<img src="images/logo.gif" onerror="alert('Ataque XSS!')"/>
```

Nota: Elaboración Propia.