ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

EVALUACIÓN DE LAS FUNCIONALIDADES DE LOS SISTEMAS DE DETECCIÓN DE INTRUSOS BASADOS EN LA RED DE PLATAFORMAS OPEN SOURCE UTILIZANDO LA TÉCNICA DE DETECCIÓN DE ANOMALÍAS

JOSÉ EDUARDO ARTEAGA PUCHA

Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGISTER EN INTERCONECTIVIDAD DE REDES

Riobamba – Ecuador

Julio – 2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN

EL TRIBUNAL DEL TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, titulado " EVALUACIÓN DE LAS FUNCIONALIDADES DE LOS SISTEMAS DE DETECCIÓN DE INTRUSOS BASADOS EN LA RED DE PLATAFORMAS OPEN SOURCE UTILIZANDO LA TÉCNICA DE DETECCIÓN DE ANOMALÍAS", de responsabilidad del señor José Eduardo Arteaga Pucha, ha sido prolijamente revisado y se autoriza su presentación.

Tribunal:

Ing. Wilson Armando Zúñiga Vinueza M.Sc.
PRESIDENTE **……**

Ing. Boris Fernando Sánchez Moreno Mg.
DIRECTOR **……**

Ing. Paúl Xavier Paguay Soxo Mg.
MIEMBRO **……**

Ing. Juan Carlos Cepeda Pacheco Mg.
MIEMBRO **……**

Riobamba, Julio 2018

DERECHOS INTELECTUALES

Yo, José Eduardo Arteaga Pucha, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo,** y que el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

_____

José Eduardo Arteaga Pucha
N° de Cédula: 020182857-1

Yo, José Eduardo Arteaga Pucha, declaro que el presente proyecto de investigación es de mi autoría y que los resultados obtenidos son auténticos y originales. Los textos constantes en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación de Maestría.

_____

José Eduardo Arteaga Pucha
N° de Cédula: 020182857-1

DEDICATORIA

El presente trabajo se lo dedico a Dios, a mis padres, esposa, hija y a mi familia. A Dios por permitir concluir mi objetivo dándome salud y bendiciones en cada momento. A mis padres por haberme dado la vida y ser mi primer pilar de mi vida. A mi esposa y mi nena por ser los seres más importantes en mi vida y por haber estado siempre a mi lado, a quienes les quite mucho tiempo de convivencia en familia; y a mi familia política que siempre estuvieron conmigo brindándome su apoyo, dándome las fuerzas necesarias para seguir adelante, sus alientos para terminar el objetivo propuesto y muchas cosas más que siempre llevare en mi corazón.

José Eduardo Arteaga Pucha

-

AGRADECIMIENTO

Expreso mi gratitud a un compañero profesor, orientador como es el Ing. Boris Sánchez quien con su ayuda desinteresada permitió que yo culmine este trabajo de investigación con todas las normas establecidas en la Escuela Superior Politécnica de Chimborazo, a los Ingenieros Juan Carlos Cepeda y Paúl Paguay quienes mediante sus consejos y ayuda prestada durante la realización del presente trabajo colaboraron en la culminación de la investigación.

A nuestra Coordinadora de Maestría como es la Ingeniera Blanquita Hidalgo, que desde los inicios de nuestra etapa de estudios de cuarto nivel estuvo ahí para darnos su apoyo incondicional, empuje, perseverancia, y el no rendirse frente a nosotros con la finalidad de que todos nuestros compañeros del programa de Interconectividad de Redes 2015 -2017 obtengamos el título de Magister.

A los todos los Docentes del Programa de Maestría de Interconectividad de Redes que con su perseverancia y desinteresada manera de compartir sus conocimientos lograron que se forme un profesional eficiente y eficaz, gracias por su amistad.

Eduardo Arteaga

CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

INDICE DE ANEXOS

RESUMEN

El presente trabajo investigativo tiene por objeto evaluar las funcionalidades de los sistemas de detección de intrusos (IDS) basados en red de plataformas Open Source utilizando la técnica de detección de anomalías, definiendo varios conceptos acerca de los sistemas de detección de intrusos. Para la evaluación de los IDSes se utilizó una metodología basada en la investigación aplicativa y cuasi-experimental, considerando los conocimientos existentes y la implementación de los casos de: aprendizaje, simulación de ataques y aplicativo, por medio escenarios virtuales, sobre los cuales se instalaron los IDSes Snort, Suricata, Bro y las diferentes herramientas de Benchmark; la correlación de las alertas emitidas tanto por Snort y Suricata utilizando la técnica de detección de anomalías basada en datos estadísticos, permitió determinar los verdaderos negativos (VN) para las alertas efectivas y falsos positivos (FP) para las anomalías. Respecto a la funcionalidad se determinó que Suricata es un 5% mejor que Snort; en cuanto al desempeño se determinó que: Suricata al momento de analizar el conjunto de datos de DARPA 99 que consta de 44'894.776,00 de paquetes tiene un 0% de paquetes perdidos, con respecto a Snort que tiene un 7.6 %; referente a los tiempos de respuesta utilizando el mismo conjunto de datos se evidencia que Suricata emplea un tiempo de 484 segundos y por otra parte a Snort lo toma realizar el análisis en un tiempo de 1086 segundos; y respecto a la seguridad Suricata ofrece una sensibilidad del 33.37% respecto a Snort del 25.63%, luego del análisis estadístico inferencial mediante la prueba Z, se concluye que Suricata ofrece mejores prestaciones para la seguridad de la información que Snort, por lo que se recomienda la implementación de Suricata considerando los pasos descritos en el manual de buenas prácticas de los IDSes de plataformas Open Source.

Palabras claves: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <REDES>, <SNORT>, <SEGURIDAD INFORMÁTICA>, < SISTEMAS DE INFORMACIÓN>, <SURICATA (HERRAMIENTA)>, <BRO (HERRAMIENTA)>, < SISTEMA DE DETECCIÓN DE INTRUSOS (IDS)>, <ANOMALÍAS DE SISTEMAS DE INFORMACIÓN>.

ABSTRACT

The purpose of this research work is to evaluate the functionality of the intrusion detection systems (IDS) based on Open Source platforms using the anomalies detection technique, defining several concepts about intrusion detection systems. For the evaluation of the IDSes, a methodology based on the application and quasi-experimental research was used, considering the existing knowledge and the implementation of the cases of: learning, simulation of attacks and application by means of virtual scenarios, on which they were installed the Snort, Suricata and Bro IDSes and the different Benchmark tools. The correlation of the alerts issued by both Snort and Suricata using the anomaly detection technique based on statistical data, allowed to determine the true negative (TN) for the effective alerts and false positives (FP) for the anomalies. Regarding the functionality, it was determined that Suricata is 5% better than Snort; in terms of performance, it was determined that: Suricata at the time of analyzing the data set of DARPA 99 that consists of 44'894.776,00 of packages has a 0% of lost packages, comparing with Snort that has a 7.6%; with respect to the response times using the same data set, it is evident that Suricata uses a times of 484 seconds and on the other hand Snort takes it to perform the analysis in 1086 seconds; and regarding safety, Suricata offers a sensitivity of 33.37% in relation to Snort that has 25.63%. After the inferential statistical analysis through Z test, it is concluded that Suricata offers better features for information security than Snort, so it is recommended the implementation of Suricata considering the steps described in the manual of good practices of IDSes of Open Source platforms.

Keywords: <TECHNOLOGY AND SCIENCE OF ENGINEERING>, <NETWORKS>, <SNORT>, <COMPUTER SECURITY>, <INFORMATION SYSTEMS>, <SURICATA (TOOL)>, <BRO (TOOL)>, <INTRUSION DETECTION SYSTEM (IDS)>, <ANOMALIES OF INFORMATION SYSTEMS>.

CAPÍTULO I

1. INTRODUCCIÓN

1.1 Problema de la Investigación

1.1.1 Situación problemática

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático.

Los sistemas informáticos pueden ser protegidos desde el ámbito lógico (con el desarrollo de software especializado) o físico (vinculado al mantenimiento eléctrico, por ejemplo).

Las amenazas pueden proceder desde programas dañinos que se instalan en la computadora del usuario (virus) o llegar por vía remota. (Hackers)

La cantidad de intentos de accesos no autorizados a la información que existe en internet ha crecido durante estos últimos años, en la mayoría de las instituciones u organizaciones, estos intentos pueden ser o no detectados, debido a que al momento no han implementado mecanismos de seguridad en su infraestructura tecnológica.

Muchas empresas u organizaciones, normalmente por motivos de coste, han migrado información clave a Internet, exponiéndola hacia el exterior. Además, para la comodidad de los trabajadores que solicitan teletrabajo, las compañías han tenido que "abrir sus puertas" para permitir la conexión a la intranet de la oficina desde su hogar. Desafortunadamente, cuando los atacantes comprometen los sistemas de entrada, ellos también tienen acceso a datos de la empresa u organización. La incorporación de cortafuegos y redes privadas virtuales (VPNs) para permitir de forma segura que los usuarios externos se puedan comunicar con la intranet de la organización ha aliviado un poco el problema; un cortafuego con una política correcta puede minimizar el que muchas redes queden expuestas. Sin embargo, los atacantes están evolucionando constantemente y aparecen nuevas técnicas como los troyanos, gusanos y escaneos silenciosos que atraviesan los cortafuegos mediante protocolos permitidos como HTTP, ICMP o DNS. Los atacantes buscan vulnerabilidades en los pocos servicios que el cortafuego permite y enmascaran sus ataques dentro de estos protocolos, quedando expuesta la red nuevamente.

La cantidad de mensajes publicados en listas de vulnerabilidades como BUGTRAQ ha aumentado de forma exagerada durante los últimos años. Las vulnerabilidades no solo afectan a sistemas tradicionalmente seguros, sino que afectan incluso a sistemas de seguridad: cortafuegos y sistemas de detección de intrusos o IDS (Intrusion Detection Systems). Esto se debe en parte a un crecimiento del número de auditorías que las empresas de software aplican a sus productos y por el aumento de interés en el campo de la seguridad por parte de los profesionales de la informática.

Los atacantes hoy en día también intentan sobrepasar los sistemas de detección de intrusos, ya sea saturándolos de tráfico o bien mediante herramientas que les proporcionan información falsa de lo que pasa por la red. El hecho de que los atacantes están incorporando técnicas anti-IDS a su arsenal, lo que les coloca en situación más ventajosa frente a organizaciones que ni siquiera disponen de un IDS.

Usualmente las instituciones, para el desarrollo de sus actividades cuenta con equipamiento tecnológico critico como son: servidores físicos y servidores virtuales, donde cada uno de ellos tiene diferente funcionalidad, como servidor puede ser: el alojamiento de la página web, correo electrónico institucional, respaldos de información, aplicaciones cliente servidor, impresiones, información técnica acorde al campo estratégico, entre otras. Se puede evidenciar la vulnerabilidad de la seguridad de información en diferentes instituciones del Ecuador.

En el año 2014 en el Ecuador, "las cifras de los ciberataques son alarmantes, según Daniel Molina, experto de la empresa Kaspersky, quien asegura que cerca del 16% de usuarios de la región son víctimas de fraudes informáticos, lo cual suma 60'090.173."

En enero de 2016, un grupo de hackers accedió a los sistemas informáticos de varias universidades privadas del Ecuador para registrar como alumnos a personas que nunca cursaron estudios superiores. Este delito incluyó a una lista de 366 personas que habrían inscrito ilegalmente sus títulos falsos en la base de datos de la Senescyt. El 17 de febrero del 2016, GMS, empresa ecuatoriana con más de 35 años de experiencia en seguridad de la información, realiza un análisis sobre los riesgos cibernéticos a los que se exponen las instituciones educativas, a propósito del reciente suceso ocurrido en enero de 2016 donde una red de hackers vulneró información de universidades privadas para ingresar ilegalmente 366 títulos falsos en la base de datos de la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (Senescyt). Este hecho ha permitido constatar las vulnerabilidades que presenta la red de datos y la importancia que implica para las universidades el utilizar mecanismos de seguridad en sus redes de datos ante la inminente amenaza de ataques avanzados que se están propiciando. (GMS, 2016)

Carlos López, Consultor de Tecnologías de la Información de GMS, señala que existen páginas donde un cibernauta puede encontrar datos no solo de universidades sino de empresas en general. Explica que luego de hallar esta información conocida como "huella digital" los ciber delincuentes penetran las defensas perimetrales, siendo los usuarios atacados al acceder a correos maliciosos o con ayuda de algún infiltrado. Finalmente, cuando el delincuente ya está en la red, la información es manipulada sin dejar huellas, e incluso burlando los sistemas de seguridad de las universidades que no son lo necesariamente robustos o completos. (GMS, 2016)

Por unas causas u otras, las instituciones tanto privadas como públicas carecen parcialmente o tiene deficientes mecanismos de seguridad de la información que hagan frente al aumento del número de ataques que se producen en Internet, como una medida para mitigar los ataques informáticos es utilizar un sistema de detección de intrusos que alerte a los administradores de los servidores y de red, cuando un intruso o eventos anormales se presentan en la red, actualmente la mayoría de instituciones del Ecuador no tienen instalado un sistema de detección de intrusos debido a que no existen evaluaciones de funcionalidad o cuadro comparativos de ventajas y desventajas de los sistemas de detección de intrusos de plataformas Open Source que les permita elegir de manera eficiente y segura un sistema acorde a la infraestructura tecnológica con la que cuentan, en otros casos el desconocimiento de la funcionalidad de los sistemas de detección de intrusos ocasiona que los administradores de servidores o de red lo instalen a modo de prueba para realizar los testeos correspondientes a fin de probar sus funcionamiento, ocasionándoles pierda tiempo y recursos, debido a estos inconvenientes no lo utilizan y en otros casos prefieren utilizar otras alternativas de seguridad o software propietario.

En varios proyectos y estudios se evidencia la utilización de sistemas de detección de intrusos de plataformas Open Source:

(Ramírez A, 2009), elaboró el proyecto "*Estudio de una plataforma de detección de intrusos Open Source*", para el desarrollo de su estudio utilizó Snort, como sistema de detección de intrusos, concluyendo que es tiene una instalación cómoda y un funcionamiento robusto y adecuado para los propósitos de detección de actividad maliciosa y de interfaz amigable.

En el proyecto "*Sistema para la correlación de alertas NIDS basados en anomalías*", con el objeto de llevar a cabo un diseño capaz de correlacionar las alertas emitidas por NIDS basados en anomalías que analizan la carga útil del tráfico de la red, en busca malware. Los principales objetivos han sido la identificación de los falsos positivos, la valoración cualitativa de la probabilidad de que una anomalía sea efectivamente una amenaza y una clasificación cuantitativa que muestre el tipo concreto de amenaza detectada para este estudio se utilizó Snort. (Maestre, 2013)

En el artículo científico acerca de la "*comparación de algoritmos para detección de intrusos en entornos estacionarios y de flujo de dato*", concluyen que la detección de intrusos bajo el enfoque de aprendizaje automático tiene varias deficiencias dada la naturaleza de la propia aplicación. A pesar de ello los investigadores continúan trabajando en lograr soluciones que permitan cubrir las mismas, el despliegue de estas soluciones es lograr que cada red, como sistema autónomo, haga la construcción de su propio conjunto de datos, lo cual debe actualizarse periódicamente debido a la diversidad de aplicaciones y al emergente crecimiento de estas. (Rivero, Ribiero, & Ortiz, 2016)

El presente trabajo de investigación tiene como finalidad realizar una evaluación de funcionalidades de los sistemas de detección de intrusos de plataformas Open Source, debido a que, en los estudios citados anteriormente solo se enfocan en uno solo, como es el caso de Snort, por lo que se ha visto la necesidad de evaluar el resto de sistemas como son: Bro, Suricata y Kismet utilizando la técnica de detección de anomalías, con el objeto de dar a conocer el mejor sistema de detección de intrusos en plataformas Open Source, debido a que en la actualidad el país está aplicando políticas de austeridad, limitando los recursos económicos y dificultado la adquisición de software propietario debido a su costo. Al finalizar el trabajo de investigación se presentará un manual de buenas prácticas de los sistemas de detección de intrusos de plataformas Open Source lo que permitirá a los administradores de servidores o de red, elegir un IDS, conociendo sus ventajas y desventajas y que puede ser adaptado al entorno de trabajo con el fin de garantizar la seguridad de la información de las diferentes unidades a su cargo, ya que en su mayor parte poseen información confidencial y de uso gubernamental para la toma de decisiones, en los diferentes campos de desarrollo del país.

1.1.2 Formulación del Problema

¿La evaluación de funcionalidades de un sistema de detección de intrusos (IDS) de plataformas Open Source utilizando la técnica de anomalías permitirá determinar al mejor IDS, al momento de realizar la detección de amenazas?

1.1.3 Sistematización del Problema

¿Cuáles son los sistemas de detección de intrusos de plataformas Open Source que se utilizan para la seguridad de la información a nivel de red?
¿Cuál es el segmento de red que garantiza una mayor cantidad de detecciones de amenazas por parte de un IDS basado en la red?
¿Cómo medir el rendimiento de los IDSes basados en la red?
¿Cuál es la técnica que mejor evalúa el funcionamiento adecuado de un IDS?

1.2 Justificación de la Investigación

El objetivo principal del presente proyecto de investigación, gira entorno a mostrar los aspectos más importantes que motivan a la investigación de la seguridad de la información respecto a los sistemas de detección de intrusos de plataformas Open Source.

La Secretaría Nacional de la Administración Pública, remitió a las Carteras de Estado el Acuerdo Ministerial Nro. 166, emitido por Secretaría Nacional de la Administración Pública (SNAP) y publicado en el Suplemento del Registro Oficial Nro. 88 del 25 de septiembre de 2013, en el cual se dispone a las entidades de la Administración Pública Central, Institucional y que dependen de la Función Ejecutiva el uso obligatorio de las Normas Técnicas Ecuatorianas NTE INEN-ISO/IEC 27000 para la Gestión de Seguridad de la Información y la implementación del Esquema Gubernamental de Seguridad de la Información. (Castillo C. , 2013)

A fin de dar cumplimiento a las políticas de seguridad definidas en el Esquema Gubernamental de Seguridad de la Información -EGSI-, particularmente la señalada en el punto 6.14. "Incorporar tecnología para la seguridad de los servicios de red como la autenticación, encriptación y controles de conexión de red". (Castillo P. C., 2013)

A pesar de que las instituciones cuentan con mecanismos de seguridad perimetral como es la configuración de un Firewall, Sophos y otras medidas, se ha podido evidenciar que, en los últimos años, los hackers logran vulnerar la seguridad de los sistemas informáticos y de esta manera realizar acciones indebidas en los equipos servidores y finales respecto a los servicios que mantiene las instituciones del estado, para el desarrollo y cumplimiento de sus competencias.

Con la finalidad de fortalecer la seguridad de los sistemas informáticos y de la red, y darles un mayor control a los servicios entrantes es necesario la instalación de sistemas pasivos, que alerten a los administradores en el momento en el que se produzca un ataque, el administrador de servidores o de red será conocedor del ataque dependiendo de la gravedad de éste, podrá tomar decisiones para contrarrestar a tiempo y evitar complicaciones futuras y tener una estadística del origen del ataque informático y a la vez el sistema detección de intrusos podrá almacenar en sus bases de datos para futuras detecciones de ataques informáticos.

Cabe indicar, que en los últimos años el Ecuador viene atravesando problemas económicos muy graves, por lo que se han aplicado políticas de austeridad, limitando de esta manera los recursos económicos a las instituciones públicas, impidiendo a los Directores Tecnológicos realizar adquisiciones de software debido a su alto costo.

De acuerdo al análisis de los estudios de investigación, sobre los sistemas de detección de intrusos o de sus algoritmos de detección, se puede evidenciar, que como caso de estudio siempre se considera a Snort, ocasionando que el resto de sistemas de detección de intrusos de plataformas Open Source, sean desconocidos por la mayoría del personal tecnológico o de áreas encargadas de la seguridad de la información, este desconocimiento provoca que al momento de poder implementar un sistema de detección de intrusos no se tenga opciones conocidas y validadas acorde a los escenarios de trabajo de las instituciones u organizaciones, incurriendo que en muchos casos no sean utilizados como una alternativa para mejorar su seguridad a nivel de red, si no por el contrario pasen desapercibidos, debido a la carencia de estudios en los que se demuestre una evaluación de funcionalidades y su desempeño al momento de detectar amenazas.

Al finalizar el trabajo de investigación, se presentará un manual de buenas prácticas que incluirá la evaluación de las funcionalidades de los diferentes sistemas de detección de intrusos basados en red de plataformas Open Source como son: Snort, Bro y Suricata, que servirá de guía para que los administradores tanto de servidores, como de red, puedan elegir un sistema conociendo sus ventajas y desventajas y sobre todo de libre uso, y que pueda ser adaptable al entorno de trabajo en el menor tiempo posible, esta guía optimizará los tiempos de prueba de los sistemas de detección intrusos al momento de ser implementados dentro su arquitectura de red, debido a que no será necesario validar cada uno de los sistemas, si no que en base a los cuadros comparativos se podrá elegir el mejor sistema de detección de intrusos acorde a las necesidades y capacidades tecnológicas presentes en la institución u organización, esto permitirá garantizar la seguridad de la información a nivel de red, de las diferentes Unidades Departamentales, en su menor tiempo posible, cabe indicar que en su mayor parte manejan información confidencial y de uso gubernamental para la toma de decisiones, en los diferentes campos de desarrollo del país.

1.3 Objetivos de la Investigación

1.3.1 Objetivo General

Evaluar las funcionalidades de los sistemas de detección de intrusos basados en red de plataformas Open Source utilizando la técnica de detección de anomalías.

1.3.2 Objetivos Específicos

- Estudiar los diferentes IDSes basados en red de plataformas Open Source para determinar su funcionalidad.

- Diseñar un esquema de red para la implementación de los sistemas de detección de intrusos (IDS) de plataformas Open Source.

- Definir el Benchmark para determinar el rendimiento de los IDSes frente a los ataques.

- Elaborar un manual de buenas prácticas de los sistemas de detección de intrusos de plataformas Open Source para aportar en la implementación de un IDS de forma óptima.

- Implementar un IDS basado en red de plataforma Open Source utilizando la técnica de detección de anomalías, en un escenario de prueba en la red de datos de la Secretaría de Hidrocarburos para mejorar la seguridad de la información.

1.4 Hipótesis

La implementación de un sistema de detección de intrusos basados en la red de plataforma Open Source utilizando la técnica de detección de anomalías mejorará la seguridad de la información que viaja por la red.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Antecedentes del Problema

(Anónimo, 2002), Los sistemas de detección de instrucciones son un fenómeno relativamente nuevo que emergió a comienzos de los 80. Un buen ejemplo es un estudio que se llevó a cabo en el Instituto de Investigación de Stanford desde julio de 1983 a noviembre de 1986. Conocido como "Proyecto 6169, desarrollo de Técnicas Estadísticas para sistemas de auditoría", el estudio utilizó: Un algoritmo de alta velocidad que podía discriminar de forma precisa entre usuarios basándose en sus perfiles de comportamiento. El proyecto demostró que los usuarios podían distinguirse unos de otros por sus perfiles de comportamiento. Estos procedimientos estadísticos son potencialmente capaces de reducir el tiempo de auditoria por un factor de 100 a la vez que demostraba un alto grado de precisión en la detección de intentos de intrusión.

Urbina en el año 2004 elaboró el proyecto "*Análisis y Evaluación de Sistemas para Detección de Intrusos en Redes de Computadoras*", se presenta la problemática actual que enfrentan las empresas, compañías o instituciones públicas o privadas, al no contar con un sistema de detección de intrusos. Los sistemas de detección de intrusos pertenecen al área aplicada de la seguridad informática encargada de advertir a través de alertas, al administrador de la red, cualquier intento de intrusión, entendiendo como intrusión: la realización de un acto no autorizado como lo es el acceso a un sistema, la ejecución de un programa o ataques a una red de computadoras de área local. La red de área local que se monitoreó fue la del Diario del Istmo de la ciudad de Coatzacoalcos, ésta se analizó y por consiguiente se investigó el software apropiado para las características de esta; el tipo de sistema de detección de intrusos que se implementó fue El basado en Red con los programas de E-trust Intrusion Detection 2.0 de Computer Associates, Snort 2.0.1 y Ettercap 0.6.b. en versiones de prueba (demo). Los resultados obtenidos revelan la vulnerabilidad con que cuenta la red de la empresa a intrusos provenientes de ataques maliciosos (spoofing) y otros con fines publicitarios (banners); a través del protocolo de comunicación HTTP. Se concluye que este es un campo de constante cambio, las intrusiones se encuentran a la orden del día, es por eso que se requiere por parte de los administradores de la red implementar sistemas de detección de intrusos orientados a las vulnerabilidades de la red propia; y por parte de los desarrolladores del software ya sea comercial o de libre distribución; optimizar los

programas existentes para que soporten y procesen tal cantidad de datos e información con el fin de evitar desbordamientos de memoria temporal (buffer) por falta de memoria RAM, cuando estos son capturados del traficó en tiempo real. (Urbina, 2004)

(Ramírez A, 2009), elaboró el proyecto "*Estudio de una plataforma de detección de intrusos Open Source*", Se puede, en primer lugar, concluir que Snort y BASE presentan una instalación cómoda y un funcionamiento robusto y adecuado para los propósitos de detección de actividad maliciosa y de interfaz amigable, respectivamente. Su diseño, al ser modular, permite entender y configurar (en el fichero de configuración y demás ficheros adicionales) comportamientos determinados frente a actividades indeseadas.

Respecto a las herramientas utilizadas, hay que destacar el valor del software libre, que permite modificar y personalizar estas utilidades a través de su ingente comunidad de usuarios. Hay que destacar la formación sencilla y útil de paquetes en Scapy con el propósito de escanear un dispositivo, lo cual se ha logrado hacer parcialmente sin ser detectado por Snort.

Se ha de mencionar, también, que el envío de paquetes fragmentados que atraviesan un NIDS, es un método habitual en la composición de ataques que busquen eludirlo. En nuestro caso, además de comprobar una considerable vulnerabilidad en las versiones previas del Snort actual, se ha conseguido generar un fallo que detiene la ejecución de Snort.

Así mismo, se ha comprobado la sencillez del lenguaje de escritura de reglas del que dispone Snort, al estudiar y escribir unas reglas que alerten y registren un comportamiento especifico, sin ser este un tipo de ataque propiamente dicho. En este caso Snort tiene un desempeño similar a un proxy, a pesar de que, este comportamiento no es uno de sus objetivos primordiales, podrá servir como parte de sistemas de resguardo o backup para un proxy real.

Comentar, a modo de recomendación, que los servidores de Snort deben ser protegidos a través de los otros métodos de seguridad también ya referidos; y que la descarga periódica de las reglas actualizadas desde Internet se hace esencial para la actualización de estos sistemas.

En el proyecto de "*Sistema para la correlación de alertas NIDS basados en anomalías*", con el objeto de llevar a cabo un diseño capaz de correlacionar las alertas emitidas por NIDS basados en anomalías que analizan la carga útil del tráfico de la red, en busca malware. Los principales objetivos han sido la identificación de los falsos positivos, la valoración cualitativa de la probabilidad de que una anomalía sea efectivamente una amenaza y una clasificación cuantitativa

que muestre el tipo concreto de amenaza detectada. Esta última clasificación es llevada a cabo considerando ataques a nivel de paquete o ataques a nivel de trazas de tráfico. (Maestre, 2013)

En el artículo científico "*comparación de algoritmos para detección de intrusos en entornos estacionarios y de flujo de dato*", la detección de intrusos en redes de computadoras a partir del enfoque de aprendizaje automático presenta algunas deficiencias dadas por la propia naturaleza de la aplicación. La principal viene dada por el modesto despliegue de sistemas de detección basados en algoritmos de aprendizaje bajo las restricciones impuestas por los entornos reales. En este artículo se describen y proponen tres variantes de pre procesamiento sobre el conjunto de datos KDD99, incluye selección de atributos. Luego la experimentación se realiza primeramente a partir de evaluar algoritmos representativos en entornos estacionarios sobre las variantes obtenidas a partir de pre procesar KDD99. Por último, dado que el tráfico de red es un flujo constante de datos, en el cual pueden existir variaciones de conceptos relacionadas con las tasas de falsos positivos, unido al hecho de que no se encuentran muchas investigaciones que aborden la detección de intrusos en entornos de flujos de datos, conducen a realizar la comparación de varios algoritmos representativos de flujos de datos. Como resultado se obtiene cuáles son los algoritmos que mejores resultados ofrecen en la detección de intrusos sobre las variantes de pre procesamiento propuestas, tanto para entornos estacionarios como de flujos de datos. (Rivero, Ribiero, & Ortiz, 2016)

Y concluye que la detección de intrusos bajo el enfoque de aprendizaje automático tiene varias deficiencias dada la naturaleza de la propia aplicación. A pesar de ello los investigadores continúan trabajando en lograr soluciones que permitan cubrir las mismas. Algo que resulta fundamental para el despliegue de estas soluciones es lograr que cada red, como sistema autónomo, haga la construcción de su propio conjunto de datos, lo cual debe actualizarse periódicamente debido a la diversidad de aplicaciones y al emergente crecimiento de estas, lo que puede provocar que el tráfico normal se clasifique como algún tipo de ataque. Así mismo surgen nuevos ataques y/o variantes de los ya conocidos.

Existen varios conjuntos de datos disponibles para la evaluación de los algoritmos, pero ninguno logra caracterizar de manera general el tráfico de las redes. KDD99 y sus variantes son los más empleados en la investigación científica, a partir del framework MOA puede construirse una solución personalizada para la red en cuestión, con la posibilidad de evaluar más de un algoritmo, incluso pudiendo desarrollar nuevas variantes de estos.

2.2 Bases teóricas

La seguridad informática comprende un conjunto de medidas de prevención, detección y corrección, orientadas a proteger la confidencialidad, la integridad y la disponibilidad de los recursos informáticos. (Benchimol, 2011). La seguridad de la información se enfoca hacia la protección de la información y del acceso a los sistemas de información, utilización y divulgación o destrucción de la información.

En la actualidad las diferentes instituciones públicas y privadas tienen enormes cantidades de información, sobre sus empleados, productos, servicios y otros netamente confidencial, por lo que requieren contar con políticas de seguridad, para asegurar los derechos de acceso a los datos y a los recursos, mediante diferentes herramientas de control, permitiendo adoptar mecanismos de protección e identificación y alertas respecto al tráfico malicioso que viaja por la red.

2.2.1 Fases de un ataque informático

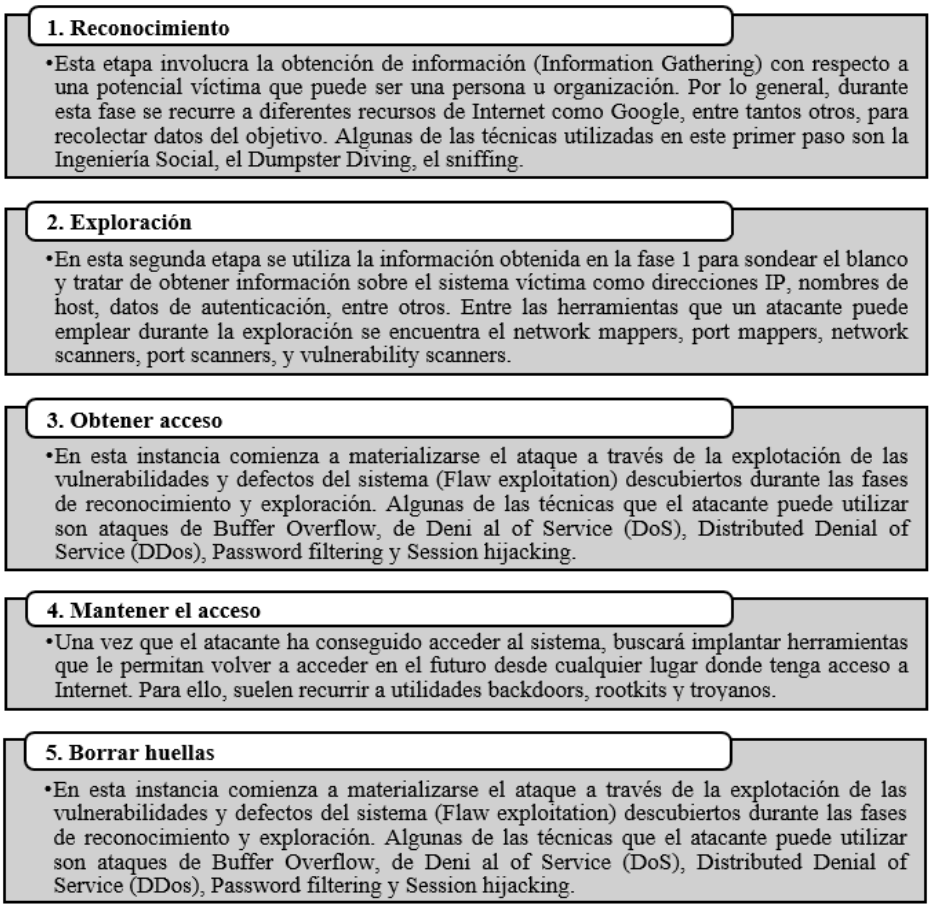

Figura 1-2: Etapas de un ataque informático

Fuente: (Mieres, 2009)

Realizado por: Eduardo Arteaga

En la **Figura 1-2**, se describen las etapas de un ataque informático al momento de su ejecución sobre un sistema:

Conocer las diferentes etapas que conforman un ataque informático, brinda la ventaja de aprender a pensar como los atacantes y a jamás subestimar su mentalidad. Desde la perspectiva del profesional de seguridad, se debe aprovechar esas habilidades para comprender y analizar la forma en que los atacantes llevan a cabo los ataques informáticos. (Mieres, 2009)

2.2.2 Sistema de Detección de Intrusos (IDS)

La detección de intrusiones es la práctica de utilizar herramientas inteligentes y automáticas para detectar intentos de intrusión en tiempo real. Dichas herramientas se llaman Sistemas de Detección de Instrucciones (Intrusion Detection Systems, IDS).

Un Sistema de Detección de Intrusos o IDS (Intrusion Detection System) es una herramienta de seguridad encargada de monitorizar los eventos que ocurren en un sistema informático en busca de intentos de intrusión. (Mira, 2000)

Un intento de intrusión se lo puede catalogar como cualquier intento de comprometer la confidencialidad, integridad, disponibilidad o evadir los mecanismos de seguridad de un sistema informático o de la red de datos.

Las intrusiones se pueden producir de varias formas: a través de atacantes que acceden a los sistemas informáticos desde la web, usuarios con claves de acceso a los sistemas que tratan de obtener mayores privilegios para acceder a otras funcionalidades para los cuales no están autorizados y usuarios autorizados que hacen un mal uso de sus credenciales de acceso que se les han otorgado, medios de almacenamiento infectados y uso malicioso por parte de terceros con el objetivo de producir la caída de servicios, logrando un caos a nivel de organización o institución.

2.2.3 Clasificación de IDS

Los sistemas de detección de intrusos en función de que sistemas vigilan se clasifican en:

2.2.3.1 Sistemas de detección de intrusos de red (NIDS)

Garantizan la seguridad dentro de la red, por lo que necesitan un hardware exclusivo, en donde se implementa el sistema para que pueda verificar todos los paquetes de información que viajan

por la red de datos, con la finalidad de descubrir si se ha generado alguna actividad maliciosa o anormal que pudiese alterar su contenido.

Al momento de configurar los sistemas de detección de intrusos de red (NIDS), es necesario configurar los adaptadores de red del equipo informático en modo promiscuo. Éste es una especie de modo "invisible" en el que no tienen dirección IP. Tampoco tienen una serie de protocolos asignados. Los IDSes pueden ser implementados en diferentes partes de la red, por lo general, se colocan en zonas de fuera de la red con la finalidad de analizar y estudiar los posibles ataques, así como también se colocan zonas internas con la finalidad de analizar las solicitudes que hayan pasado a través las seguridades como puedo que se han realizado desde dentro de la red. (Vialfa, 2017)

2.2.3.2 Sistema de detección de intrusiones en el host (HIDS)

HIDS permiten garantizan la seguridad en el host, pueden ser implementados en diferentes plataformas de sistemas operativos como puede ser: Windows, Solaris, Linux, HP-UX, Aix, etc.

El HIDS actúa como un daemon o servicio estándar en el sistema de un host. El HIDS permiten analizar la información que se encuentra almacenada en registros como pueden ser: los registros del sistema, lastlogs, mensajes y wtmp. También los HIDS permiten realizar la captura de paquetes que viajan por la red que se introducen o salen del host para poder analizar las señales de intrusión producidos por ataques de denegación de servicio, puertas traseras, virus, troyanos, intentos de acceso no autorizado, ataques de desbordamiento de búfer y ejecución de códigos maliciosos. (Vialfa, 2017)

La segunda gran clasificación de los sistemas de detección de intrusos se realiza en función de cómo actúan estos sistemas.

2.2.3.3 Detección de anomalías

Desde que en 1980 James P. Anderson propusiera la detección de anomalías como un método válido para detectar intrusiones en sistemas informáticos, la línea de investigación más activa (esto es, la más estudiada, pero no por ello la más extendida en entornos reales) es la denominada Anomaly Detection IDS, IDSes basados en la detección de anomalías. La idea es a priori muy interesante: estos modelos de detección conocen lo que es “normal” en nuestra red o nuestras máquinas a lo largo del tiempo, desarrollando y actualizando conjuntos de patrones contra los que comparar los eventos que se producen en los sistemas. Si uno de esos eventos (por ejemplo, una

trama procedente de una máquina desconocida) se sale del conjunto de normalidad, automáticamente se cataloga como sospechoso. (Rediris, 2008)

Los IDSes basados en detección de anomalías se basan en la premisa de que cualquier ataque o intento de ataque implica un uso anormal de los sistemas, Pero, ¿cómo puede un sistema conocer lo que es y lo que no es "normal" en nuestro entorno de trabajo? Para conseguirlo, existen dos grandes aproximaciones: o es el sistema el que es capaz de aprenderlo por sí mismo (basándose por ejemplo en el comportamiento de los usuarios, de sus procesos, del tráfico de nuestra red...) o bien se le especifica al sistema dicho comportamiento mediante un conjunto de reglas. La primera de estas aproximaciones utiliza básicamente métodos estadísticos (medias, varianzas...), aunque también existen modelos en los que se aplican algoritmos de aprendizaje automático; la segunda aproximación consiste en especificar mediante un conjunto de reglas los perfiles de comportamiento habitual basándose en determinados parámetros de los sistemas (con la dificultad añadida de decidir cuáles de esos parámetros que con mayor precisión delimitan los comportamientos intrusivos). (Rediris, 2008)

En el primero de los casos (el basado en métodos estadísticos), el detector observa las actividades de los elementos del sistema, activos - sujetos , pasivos - objetos - o ambos, y genera para cada uno de ellos un perfil que define su comportamiento; dicho perfil es almacenado en el sistema, y se actualiza con determinada frecuencia envejeciendo la información más antigua y priorizando la más fresca.

El comportamiento del usuario en un determinado momento se guarda temporalmente en otro perfil, denominado 'perfil actual' (current profile), y a intervalos regulares se compara con el almacenado previamente en busca de desviaciones que puedan indicar una anomalía. (Rediris, 2008)

2.2.3.4 Detección de usos indebidos

El funcionamiento de los IDSes basados en la detección de usos indebidos presupone que se deben establecer patrones para los diferentes ataques conocidos y algunas de sus variaciones; mientras que la detección de anomalías conoce lo normal (en ocasiones se dice que tienen un 'conocimiento positivo', positive knowledge) y detecta lo que no lo es, este esquema se limita a conocer lo anormal para poderlo detectar (conocimiento negativo, negative knowledge). (Rediris, 2008)

2.2.4 Esquema de un IDS

En la presente investigación se basa tipo NIDS utilizando la tecnica de detección de anomalías, que permite probar los diferentes ataques a través de la red. Acontinuación se puede ver el NIDS, colocados en diferentes segmentos de la red, cada uno de ellos con un propósito de asegurar la información, servicios y los recursos, ver **Figura 2-2.**

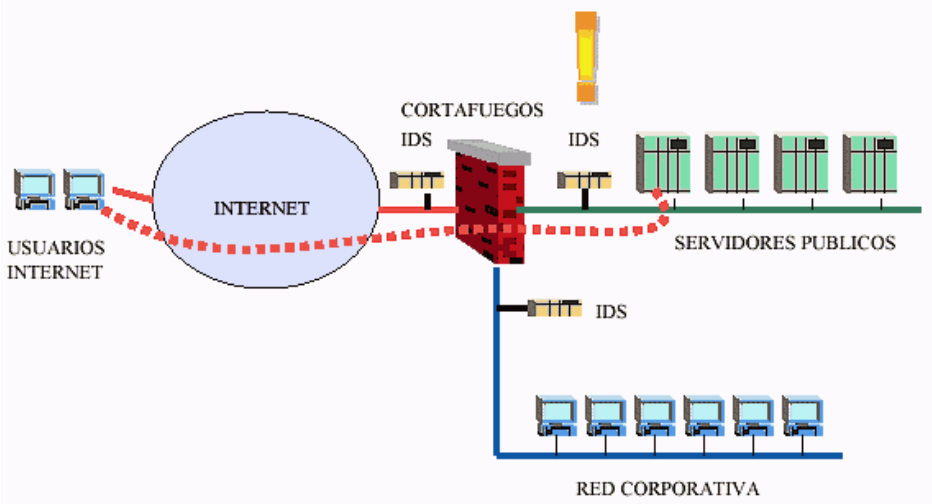

Figura 2-1: Esquema del IDS

Fuente: https://lionsec.net/blog/tecnicas-de-prevencion-de-intrusiones

2.2.5 Localizaciones en la que se puede implementar un IDS

Para el presente estudio de investigación, se han establecido varios puntos de colocación de un IDS, los cuales se encuentran descritos en la **Tabla 1-2**.

Tabla 1-2: Descripción del esquema de localizaciones donde implementar un IDS

Puntos	Ventajas	Desventajas
A: Delante del contrafuegos externo	• Monitorizar el número y el tipo de ataques dirigidos contra la infraestructura de la organización • Detectar ataques cuyo objetivo es el cortajuegos principal.	• No permite detectar ataques que utilicen en sus comunicaciones algún método para ocultar información, como algoritmos de encriptación o estenografía. • En caso de gran cantidad de tráfico de red, puede causar saturación descartando parte de información.

		• El NIDS puede convertirse en blanco fácil si algún atacante logra identificarlo.
B: Detrás del cortafuegos externo	• Se monitorizan intrusiones que logran atravesar el firewall principal. • Detectar ataques a servidores que ofrecen servicios públicos. • En caso de no detectar ataques con éxito, puede reconocer algunas consecuencias de estos, como intentos de conexiones salientes, realizadas desde servidores comprometidos. • Identificación de los ataques y escaneos más comunes permite mejorar la configuración de los cortafuegos.	• No permite identificar ataques que utilicen métodos de encriptación de información. • Normalmente en este segmento de red el N-IDS no puede analizar todo el tráfico, descartando datos. • La seguridad del NIDS mejora con la inclusión de los cortafuegos que lo separa de la red del exterior, sin embargo, esto no excluye tomar medidas adicionales para evitar que pueda ser comprometido por atacantes.
C: Redes principales	• Al haber mayor cantidad de tráfico, hay también mayores posibilidades de encontrar posibles ataques, este hecho se cumple siempre y cuando la cantidad de tráfico no supera la capacidad del NIDS. • Detectar ataques producidos desde dentro de la propia red, como los realizados por personal interno.	• No permite identificar ataques que utilicen métodos de encriptación de información. • No se puede evitar problemas asociados al uso de conmutadores en la red. Las características de estos dispositivos podrían impedir la monitorización de los miembros de la red. • Vulnerabilidad de los sistemas frente ataques internos de la red.
D: Subredes de valor crítico	• Detectar ataques realizados contra elementos críticos de la red. • Dedicar especial atención a los recursos más valiosos de la infraestructura.	• No permite identificar ataques que utilicen métodos de encriptación de información. • No evitan problemas de monitorización relacionados con el uso de conmutadores. • No están estratégicamente bien situados ante ataques de origen interno.
E: Máquinas	• Evitar ataques que utilicen métodos de encriptación de información. • Solventar problemas del uso de conmutadores	• Limitación del sistema de detección debido a la situación de la máquina y red. • Reducción del rendimiento de la máquina que monitoriza. • Que la máquina anfitriona sea comprometida puede traer consecuencias respecto que el detector pierda eficacia, controlado por el atacante, obtener información de la infraestructura, enviar falsas alarmas, etc.

Realizado por: Eduardo Arteaga

2.2.6 Sistemas de detección de intrusos plataformas Open Source

Dentro del proyecto de investigación se hizo uso de los siguientes sistemas de detección de intrusos basados en la red de plataforma Open Source, ver **Figura 3-2**.

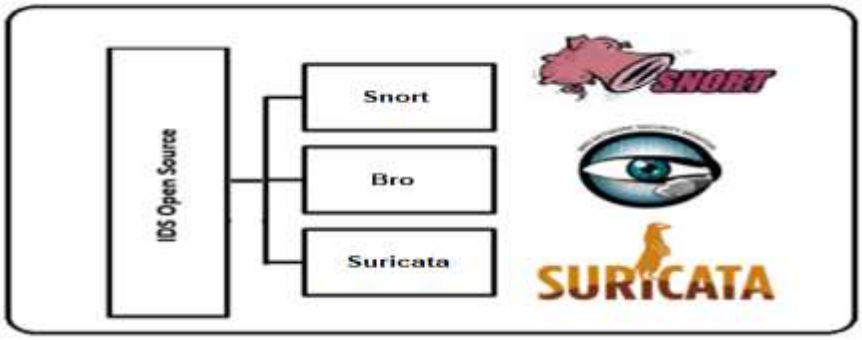

Figura 3-2: IDSes de plataformas Open Source

Realizado por: Eduardo Arteaga

Para esta investigación considerando que Kismet es un sistema de detección de intrusiones para redes inalámbricas 802.11, no se lo tomará en cuenta, ya que su campo de aplicación es distinto al resto de IDSes, y no se podrían establecer cuadros comparativos al respecto.

2.2.7 Snort

Snort es un Sistema de Detección de Intrusos basado en red (NIDS) de plataforma Open Source. Se basa en un lenguaje de creación de reglas en el que se pueden definir los patrones que se utilizarán a la hora de monitorizar el sistema o la red de datos. Snort posee un gran conjunto de reglas y filtros ya predefinidos que se pueden configurar durante su instalación y configuración con la finalidad de que se adapte a las necesidades tecnológicas de la institución u organización, su última versión estable es de diciembre de 2016 (v.2.9.9.0). (Daniel, 2017)

(Cisco & Affiliates, 2014), Snort se puede configurar para que se ejecute en tres modos:

- **Modo Sniffer:** Simplemente lee los paquetes de la red y los muestra para usted en un flujo continuo en la consola (pantalla).

- **Packet Logger:** Registra los paquetes en el disco.

- **El modo NIDS (Network Intrusion Detection System):** Realiza la detección y el análisis del tráfico de red. Este es el modo más complejo y configurable.

2.2.7.1 Arquitectura

Snort cuenta con distintos conectores de software para personalizar su implementación y se encuentra conformado por cuatro componentes básicos: Decodificador, Preprocesador, Motor de detección y Subsistema de alerta y log, ver **Figura 4-2**.

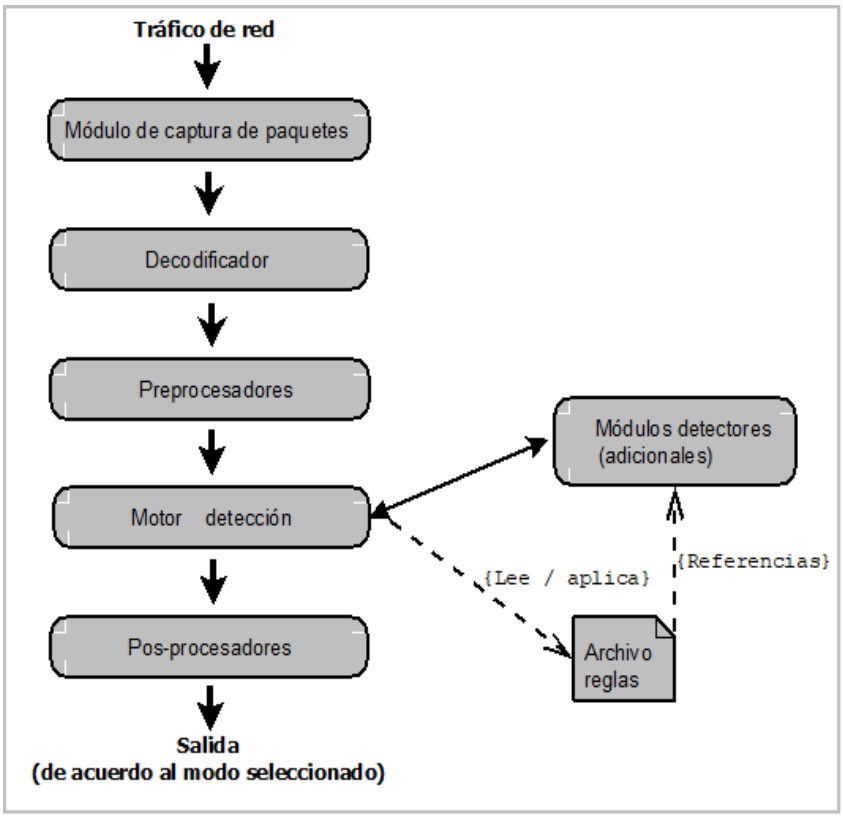

Figura 4-2: Diagrama de operación Snort

Realizado por:: Eduardo Arteaga

2.2.7.2 Ventajas

- Snort puede funcionar como un Sniffer (es decir que podemos ver en consola y en tiempo real el tráfico de la red), registro de paquetes (guardar archivos logs) o como un NIDS normal.

- Snort utiliza un lenguaje de descripción de reglas sencillo y ligero que es flexible y bastante potente.

- Hay un número de pautas simples a recordar al desarrollar las reglas de Snort que ayudarán a salvaguardar su cordura.

2.2.7.3 Elementos del sistema

- ***Reglas***

Una de las principales características que ofrece Snort, es la posibilidad de crear propias reglas ante comportamientos anómalos. También permite definir nuevas reglas de Snort con la finalidad de guardar determinados paquetes en logs, así como descartar aquellos paquetes que no se desea que ingresen en la red. Snort almacena sus reglas predeterminadas dentro del directorio de configuración, bajo el subdirectorio rules y es el mismo que se utiliza para almacenar las nuevas reglas.

Una regla de Snort se divide en dos secciones lógicas como son: el encabezado y las opciones de la regla. Dentro del encabezado de la regla se definen las direcciones IP, las máscaras de red, acción, protocolo, origen y destino de la regla, así como la información de los puertos de origen y de destino. La sección de opciones de regla contiene mensajes de alerta e información sobre que partes del paquete deben ser inspeccionadas para determinar si se debe tomar la acción de regla. (Cisco & Affiliates, 2014)

Figura 2-2: Regla de Snort de ejemplo

Realizado por: Eduardo Arteaga

En **Figura 5-2**, se puede observar la estructura de una regla de Snort, que está definida hasta el primer paréntesis es el encabezado de la regla y la sección entre paréntesis contiene las opciones de la regla. Las palabras antes de los dos puntos en la sección de opciones de regla se llaman palabras clave de opción.

Cabe indicar que todos los elementos que componen una regla deben ser verdaderos, cuando se tenga un conjunto de elementos se pueden hacer uso para formar una declaración o instrucciones lógicas haciendo uso de los operadores AND y OR, respectivamente.

- ***Encabezados de reglas***

Snort para el encabezado de las diferentes reglas, utiliza la siguiente información, de acuerdo con la siguiente **Tabla 2-2**.

Tabla 2-1: Estructura de una regla de Snort

No.	Campo	Descripción
1	***Acciones de reglas***	El encabezado de una regla contiene la información que define quién, dónde y qué de un paquete, así como qué hacer en el caso de que se presente un paquete con todos los atributos indicados en la regla. El primer elemento de una regla es la acción de regla. La acción de regla le indica a Snort qué hacer cuando encuentra un paquete que coincida con los criterios de regla. Existen 8 acciones predeterminadas disponibles en Snort: **1. Alert**: Genera una alerta utilizando el método de alerta seleccionado y registra el paquete. **2. Log**: Registra el paquete. **3. Pass:** Ignorar el paquete. **4. Activate:** Alertar y luego activar otra regla dinámica. **5. Dynamic:** Permanece inactivo hasta que se activa mediante una regla de activación, y luego actúa como una regla de registro. **6. Drop:** Bloqueo y registro del paquete. **7. Reject:** Bloquea el paquete, registrarlo y, a continuación, enviar un restablecimiento TCP si el protocolo es TCP o un puerto ICMP inaccesible mensaje si el protocolo es UDP. 8. **Sdrop:** Bloquea el paquete, pero no lo registra.
2	***Protocolos***	TCP, UDP, ICMP y IP
3	***Direcciones IP***	La siguiente parte de la cabecera de regla trata de la dirección IP y la información de puerto. La palabra clave any se puede utilizar para definir cualquier dirección. Snort no tiene un mecanismo para proporcionar la búsqueda de nombre de host para los campos de dirección IP en el archivo de configuración. Las direcciones están formadas por una dirección IP numérica recta y un bloque CIDR. Las designaciones de CIDR permiten designar grandes espacios de direcciones con sólo unos pocos caracteres.
4	***Números de puerto***	Al momento de crear las reglas, los números de puerto se pueden especificar de varias maneras, incluyendo puertos, definiciones de puertos estáticos, intervalos y por negación. Los puertos estáticos están indicados por un único número de puerto, como 111 para portmapper, 23 para telnet, 80 para http, etc. Los rangos de puertos se indican con el operador de rango (:). El operador de rango puede aplicarse de varias maneras para tomar diferentes significados.
5	***El operador de la dirección***	El operador de dirección (->), indica la orientación o dirección del tráfico al que se aplica la regla. La dirección IP y los números de puerto en el lado izquierdo del operador de dirección se consideran como el tráfico procedente del host de origen y la dirección y la información del puerto en el lado derecho del operador es el host de destino.

No.	Campo	Descripción
		También hay un operador bidireccional, que está (< >), esto le indica a Snort que considere los pares de direcciones / puertos en la orientación de origen o de destino, esto es útil para grabar, analizar ambos lados de una conversación, por ejemplo, sesiones de Telnet o POP3.
6	**Activar / Reglas Dinámicas**	Las reglas de activación son como alertas, pero también le dicen a Snort que agregue una regla cuando se produzca un evento de red específico. Las reglas dinámicas son iguales que las reglas de registro, excepto que se habilitan dinámicamente cuando se apaga el ID de la regla de activación. Utilizan campos de opción como son: activated_by y count.

Realizado por: Eduardo Arteaga

- ***Acciones de reglas***

El encabezado de una regla contiene la información que define quién, dónde y qué de un paquete, así como qué hacer en el caso de que se presente un paquete con todos los atributos indicados en la regla. El primer elemento de una regla es la acción de regla. La acción de regla le indica a Snort qué hacer cuando encuentra un paquete que coincida con los criterios de regla. Existen 8 acciones predeterminadas disponibles en Snort:

2.2.7.4 Plugins

Nombre	Descripción
ACID	Consola web para visualizar registros de Snort.
BASE	Evolución del ACID
IDS Policy Manager	Facilita el manejo de los preprocesadores y de las salidas de Snort
Inline Snort	Sistema de prevención de intrusos
SAM	Monitor de Alertas de Snort
Snort Log Parser	Analiza los mensajes de archivo de alertas de Snort
SnortSnarf	Consola web de Snort que permite almacenar los incidentes y realizar informes
SnortSMS	Consola web que permite administrar varios sensores de Anomalías
Spade	Módulo detector de Anomalías

Figura 6-2: Módulos y complementos para Snort

Fuente: (Gómez, 2009)

Realizado por: Eduardo Arteaga

En **Figura 6-2**, se muestra un resumen de algunos módulos y complementos existentes para Snort.

2.2.8 Bro

Bro es un sistema de detección de intrusiones para UNIX/Linux Open Source, algo distinto a Snort y Suricata. En cierto modo, Bro es tanto un IDS basado en anomalías como en firmas. El tráfico capturado generará una serie de eventos. Por ejemplo, un evento podría ser un inicio de sesión de usuario a un FTP, conexión a servicio web o casi cualquier cosa. Bro es un Intérprete de Políticas Script. Con su propio lenguaje de administración (Bro-Script) ofrece posibilidades muy interesantes. Por poner un ejemplo de su versatilidad, con Bro permite descargar ficheros encontrados en nuestro entorno, remitirlos para un análisis de malware, notificar si se encuentra un problema y después introducir en la lista negra la fuente del mismo, incluso permite gestionar el apagado del equipo remoto del usuario que lo descargó. Además, es capaz de detectar más patrones de actividad que la mayoría de IDS.

El análisis de red reporta varios tipos de registros divididos según el protocolo y características como puede ser HTTP, DNS, SSL, FTP, sesiones IRC, SMTP, etc.

Bro dispone de una serie de scripts o Políticas escritos en un lenguaje nativo de Bro. Estas políticas describen qué tipo de actividades se consideran sospechosas. En base a esto, se dispone de una gran cantidad de políticas para la detección de las actividades sospechosas más comunes. Estas políticas suponen una especie de capa más en la estructura de Bro. Estas políticas generarán, dado el caso, cientos de eventos dependiendo del tipo de protocolo / script.

2.2.8.1 Ventajas

- Gran capacidad de análisis a nivel de protocolo y las políticas especializadas y configurables y la capacidad de ser una herramienta de análisis forense.

- Es de elevado rendimiento y capacidad para gestionar grandes volúmenes de tráfico.

- Sus alertas pueden ser configuradas para generar eventos de log, alertas en tiempo real y hasta ejecución de comandos de sistema.

- Las políticas o policies, además de generar logs por actividad sospechosa, puede generar logs de actividad normal dependiente de las configuraciones dadas.

- A través del lenguaje de scripts de políticas o policies, se pueden crear políticas específicas para un entorno de red o una actividad concreta de acuerdo con las necesidades tecnológicas de una institución.

- Se puede ejecutar Bro para detección en tiempo real usando una determinada interface de red o leyendo un fichero ***.pcap***.

2.2.8.2 Arquitectura

La arquitectura de Bro IDS según (Sanz, 2015), se basa en dos componentes:

- **Motor de eventos (event engine):** Reduce el flujo de paquetes, organizándolos para ser llevados a un nivel superior.

- **Interprete de Scripts (policy script interpreter):** acciones a tomar cuando se detecta una actividad determinada.

2.2.8.3 Elementos del sistema

- ***Scripts***

Bro incluye un lenguaje de scripting basado en eventos que proporciona los medios principales para que una organización amplíe y personalice la funcionalidad de Bro. Prácticamente toda la producción generada por Bro es, de hecho, generada por los guiones Bro. Es casi más fácil considerar a Bro como una entidad detrás de las cámaras que procesa conexiones y genera eventos mientras que el lenguaje de scripting de Bro es el medio a través del cual los simples mortales pueden lograr la comunicación.

Bro scripts efectivamente notifica que, si hay un evento de un tipo definido, entonces vamos a tener la información sobre la conexión para que existente para realizar alguna función en él. Por ejemplo, el archivo ssl.log es generado por un script de Bro que recorre toda la cadena de certificados y emite notificaciones si alguno de los pasos a lo largo de la cadena de certificados no es válido. Todo este proceso está configurado diciéndole a Bro que si vea un servidor o cliente emita un mensaje SSL HELLO, con la información acerca de esa conexión. (Writing Bro Scripts, 2017)

- ***La cola de eventos y los controladores de eventos***

El lenguaje de scripting de Bro es impulsado por eventos, que es un cambio de velocidad con respecto a otros de lenguajes de script con los que la mayoría de los usuarios tienen experiencia en su manejo. Scripting en Bro depende de manejar los eventos generados por Bro, ya que procesa el tráfico de red, altera el estado de las estructuras de datos a través de esos eventos y toma decisiones sobre la información proporcionada. En este enfoque de scripting a menudo puede causar confusión a los usuarios que vienen a Bro de un lenguaje procedural o funcional, pero una vez que el usuario inicia con interacción con la parte funcional este se hace más claro con cada exposición.

El núcleo de Bro actúa para colocar los eventos en una "cola de eventos" ordenada, permitiendo a los manejadores de eventos procesarlos de la siguiente manera, el primero en llegar primero en ser atendido. En efecto, esta es la funcionalidad principal de Bro, ya que, sin las secuencias de comandos escritas para realizar acciones discretas en eventos, habría poco o ningún resultado utilizable. Como tal, una comprensión básica de la cola de eventos, los eventos que se generan y la forma en que los manejadores de eventos procesan esos eventos es una base no sólo para aprender a escribir guiones para Bro sino para entender la funcionalidad de Bro.

La mayoría de los eventos generados por Bro se definen en los archivos de que función, que actúan como base para la documentación de eventos en línea. Estos comentarios en línea se compilan en un sistema de documentación en línea con Broxygen. Ya sea iniciando un script desde cero o leyendo y manteniendo el script de otra persona, tener las definiciones de eventos incorporadas disponibles es un recurso excelente para tener a mano. Para la versión 2.0, los desarrolladores de Bro han puesto un esfuerzo significativo en la organización y documentación de cada evento. Este esfuerzo resultó en archivos de función incorporados organizados de modo que cada entrada contiene un nombre de evento descriptivo, los argumentos pasados al evento y una explicación concisa de las funciones que se usan.

- ***El tipo de datos de registro de conexión***

De los eventos definidos por Bro, un número abrumadoramente grande de ellos se pasa el tipo de datos de registro de conexión, en efecto, lo que la convierte en la columna vertebral de muchas soluciones de secuencias de comandos. El registro de conexión en sí es una masa de tipos de datos anidados utilizados para rastrear el estado de una conexión durante su vida útil. Mientras que Bro es capaz de procesar a nivel de paquete, sus fortalezas se encuentran en el contexto de una conexión entre un originador y un respondedor. Como tal, hay eventos definidos para las partes

primarias del ciclo de vida de la conexión, como veremos a continuación de la pequeña selección de eventos relacionados con la conexión.

Bro utiliza ampliamente las estructuras de datos anidadas para almacenar el estado y la información obtenida del análisis de una conexión como una unidad completa. Para desglosar esta colección de información, tendrá que hacer uso del delimitador de campo de Bro ***$***. Por ejemplo, el host de origen se hace referencia por ***cidorig_h***, que se refiere a orig_h que es un miembro de id que es un miembro de la estructura de datos referida como c que se pasó al manejador de eventos. Dado que el puerto de respuesta cidresp_p es 80/tcp, es probable que los scripts HTTP de base de Bro puedan rellenar el registro de conexión. Se pueden cargar los scripts de base/protocols/http y comprobar la salida del script.

Bro usa el signo de dólar como delimitador de campo y existe una correlación directa entre la salida del registro de conexión y el formato apropiado de una variable no referenciada en los scripts. La salida del script cuando contienen los grupos de información se recoge entre paréntesis, lo que correspondería al $ delimitador en un script Bro.

- ***Tipos de datos y estructuras de datos***

✓ ***Alcance***

Es importante tener una buena comprensión de los diferentes niveles disponibles de alcance en Bro y los momentos apropiados para usarlos dentro de un script. Las declaraciones de variables en Bro vienen en dos formas. Las variables se pueden declarar con o sin una definición en el formulario ***SCOPE*** name: ***TYPE*** o ***SCOPE*** name = ***EXPRESSION*** respectivamente; Cada uno de los cuales produce el mismo resultado si ***EXPRESSION*** evalúa al mismo tipo que ***TYPE***. La decisión sobre el tipo de declaración a utilizar es probable que sea dictada por la preferencia personal y la legibilidad.

✓ ***Variables globales***

Una variable global se utiliza cuando el estado de la variable necesita ser rastreado. Si bien hay algunas advertencias, cuando un script declara una variable utilizando el ámbito global, ese script está otorgando acceso a esa variable desde otros scripts. Sin embargo, cuando un script utiliza la palabra clave module para dar al script un espacio de nombres, se debe prestar más atención a la declaración de globales para garantizar el resultado deseado. Cuando se declara un global en un script con un espacio de nombres hay dos posibles resultados. En primer lugar, la variable sólo

está disponible en el contexto del espacio de nombres. En este escenario, otros scripts dentro del mismo espacio de nombres tendrán acceso a la variable declarada mientras que los scripts que utilizan un espacio de nombres diferente o ningún espacio de nombres en conjunto no tendrán acceso a la variable. De forma alternativa, si se declara una variable global dentro de un bloque ***export { ... }*** , esa variable está disponible para cualquier otro script a través de la convención de nomenclatura de ***MODULE::variable_name.*** (Writing Bro Scripts, 2017)

✓ ***Constantes***

Bro también hace uso de constantes, que se denotan por ***const***. A diferencia de los globales, las constantes sólo se pueden establecer o alterar en el momento de análisis si se ha utilizado el atributo &redef . Después (en tiempo de ejecución) las constantes son inalterables. En la mayoría de los casos, las constantes re-definibles se utilizan en los scripts de Bro como contenedores para las opciones de configuración. Por ejemplo, la opción de configuración para registrar contraseñas descifradas de flujos HTTP se almacena en ***HTTP::default_capture_password.***

✓ ***Variables locales***

Mientras que los globales y las constantes están ampliamente disponibles en el ***scriptland*** por diversos medios, cuando una variable se declara con un alcance local, su disponibilidad se restringe al cuerpo del evento o función en la cual fue declarado. Las variables locales tienden a usarse para valores que sólo se necesitan dentro de un ámbito específico y una vez que el procesamiento de un script pasa más allá de ese ámbito y ya no se utiliza, la variable se elimina. Bro mantiene los nombres de los locales por separado de los globalmente visibles.

✓ ***Estructuras de datos***

Algunas de las características más interesantes de los tipos de datos se revelan cuando se utiliza dentro de una estructura de datos, Bro para sus estructuras de datos utiliza varios tipos de datos conocidos y para una plataforma de supervisión de la seguridad de la red tiene un conjunto bastante robusto de tipos de datos centrados en la red, como son: Int, contar, doble, Bool, Addr, Puerto, subred, intervalo y patrón.

✓ ***Conjuntos***

Los conjuntos se utilizan para almacenar los elementos singulares del mismo tipo de datos. En esencia, se puede pensar en ellos como “un conjunto único de números enteros” o “un conjunto

único de direcciones IP". Mientras que la declaración de un conjunto puede diferir según el tipo de datos que están siendo recogidos, el conjunto siempre contendrá elementos únicos y los elementos en el conjunto será siempre del mismo tipo de datos. Tales requisitos hacen el conjunto perfecto para el tipo de datos de información que ya se encuentra de forma única natural, como puertos o direcciones IP.

- ✓ ***Tablas***

Una tabla en Bro es un mapeo de una clave para un valor o rendimiento, cada clave en la tabla debe ser única para preservar un mapeo uno a uno de las claves de los valores que pueden contener cualquier valor.

2.2.8.4 Plugins

La distribución principal de Bro consiste en una serie de componentes individuales que también se pueden descargar y utilizar por separado. En la **Tabla 3-2**, se muestra la descripción al respecto.

Tabla 3-2: Módulos y complementos para Bro

Módulo	**Descripción**
BinPAC	BinPAC es un lenguaje de alto nivel para describir analizadores de protocolo y genera código C ++. Actualmente se mantiene y se distribuye con la distribución Bro Network Security Monitor
Bro-aux	Pequeñas herramientas auxiliares para Bro.
Broccoli	La Biblioteca de Comunicación Bro Client.
BroControl	Un shell interactivo para administrar las instalaciones de Bro
Broccoli-python	Este módulo de Python proporciona enlaces para Broccoli, la biblioteca de comunicación del cliente de Bro. En general, los enlaces proporcionan la misma funcionalidad que la API C de Broccoli.
Broccoli-ruby	Esta es la extensión de brócoli-rubí para Ruby que proporciona acceso a la API de Broccoli. El brócoli es una biblioteca para comunicarse con el sistema de detección de intrusiones Bro.
BTest	El btest es un marco simple para escribir pruebas unitarias. Tomando libremente algunas ideas de otros paquetes, su principal objetivo es proporcionar un controlador fácil de usar y sencillo para un conjunto de pruebas basadas en shell. Cada prueba consiste en un conjunto de líneas de comando que se ejecutará, y el éxito se determina en función de sus códigos de salida. Btest viene con algunas herramientas adicionales que se pueden utilizar dentro de estas pruebas para comparar la producción con una línea de base previamente establecida.
Capstats	Capstats es una pequeña herramienta para recopilar estadísticas sobre la carga actual de una interfaz de red, utilizando libpcap o la interfaz nativa de Endace. Reporta estadísticas por intervalo de tiempo y / o para el tiempo de ejecución total de la herramienta

PySubnetTree	El paquete PySubnetTree proporciona una estructura de datos Python SubnetTree que asigna las subredes que se dan en la notación CIDR (incluidas las versiones IPv6 correspondientes) a los objetos Python. Las búsquedas se realizan por coincidencia de prefijo más largo.
Trace-summary	Trace-summary es una secuencia de comandos Python que genera desgloses del tráfico de red, incluyendo listas de los principales hosts, protocolos, puertos, etc. Opcionalmente, puede generar salida por separado para el tráfico entrante vs. saliente, por subred y por tiempo- intervalo.

Fuente: https://www.bro.org/download/index.html
Realizado por: Eduardo Arteaga

2.2.8.5 Vectores

Los vectores realizan gran parte de la misma funcionalidad que las matrices asociativas con enteros sin signo como sus índices. Sin embargo, son más eficientes que eso y que permiten el acceso ordenado. Como tal cualquier momento que necesita para almacenar datos de forma secuencial del mismo tipo. Los vectores son una colección de objetos, todos los cuales son del mismo tipo de datos, a los que elementos se pueden agregar de forma dinámica o eliminado. Los vectores utilizan el almacenamiento contiguo para sus elementos, el contenido de un vector se puede acceder a través de un desplazamiento numérico cero-indexada. El formato para la declaración de un vector es: ***SCOPE v: vector of T***, donde **V** es el nombre de su vector, y **T** es el tipo de datos de sus usuarios.

2.2.5.6 Cómo elevar avisos

La necesidad para indicar cuando un comportamiento específico que ha sido detectado. Bro mantiene la filosofía que corresponde a cada operador para indicar las conductas en las que están interesados y como tal Bro viene con un gran número de secuencias de comandos de directiva que detectar comportamientos que pueden ser de interés, pero no pretende adivinar qué conductas son capaces de "acción". En efecto, el Bro trabaja para separar el acto de detección y la responsabilidad de informar. Con el Framework de aviso es simple para elevar un aviso por cualquier comportamiento que sea detectado.

Para subir un anuncio en Bro, se necesita proporcionar un aviso ***Notice::type*** exportando y luego hacer una llamada de aviso para suministrar una notificación ***Notice::Info*** de información. A veces la llamada de notificación incluye sólo el ***Notice::Type***, y un breve mensaje. Existen, sin embargo, considerablemente más opciones disponibles a la hora de elevar los avisos como se ve en la definición de ***Notice::Info***. El único campo en ***Notice::Info*** cuyos atributos lo convierten en un campo obligatorio es el campo ***note.*** Aun así, los buenos modales siempre son importantes,

incluyendo un breve mensaje en ***$msg*** y, cuando sea necesario, el contenido del registro de conexión en ***$conn*** junto con el ***Notice::type*** tienden a comprenden la información mínima requerida para un anuncio para ser considerados útiles. Si la variable ***$conn*** se suministra el marco de aviso se completarán automáticamente los campos de ***$id*** y ***$src***. Otros campos que comúnmente se incluyó ***$identifier y $suppress_for*** están construidos entorno a la característica de supresión automática del Framework. (Writing Bro Scripts, 2017)

Un ejemplo de elevar avisos se trata de un script de política predeterminada que plantea un aviso cuando un inicio de sesión SSH ha sido detectado heurísticamente y hostname que originan el levantamiento de sospechas. Efectivamente, la secuencia de comandos intenta definir una lista de hosts que nunca deben originar tráfico SSH, como servidores DNS, servidores de correo, etc. Para ello, el script se adhiere a la separación de detección y notificación al detectar un comportamiento y elevar un aviso. Si no se actúa en ese anuncio, que es decidido por la Directiva del aviso local, pero la secuencia de comandos intenta proporcionar tanta información como sea posible.

2.2.9 Suricata

Suricata es una herramienta escalable. Este monitor de seguridad hace uso de las funciones multi-hilo de manera que solo con ejecutarse en una instancia el monitor balanceará su carga entre todos los procesadores disponibles, evitando incluso alguno de ellos si así de ser necesario. Gracias a ello, esta herramienta es capaz de procesar un ancho de banda de hasta 10 gigabits por segundo sin que ello repercuta sobre el rendimiento. (Elhacker, 2017)

Esta herramienta también es capaz de identificar los principales protocolos de red, siendo capaz de controlar en todo momento todo el tráfico que se genera en el sistema y controlando posibles amenazas de malware. (Elhacker, 2017)

Suricata dentro de sus funcionalidades también permite controlar los diferentes archivos que viajan por la red de datos, siendo capaz de identificar varios formatos diferentes, realizar comprobaciones de algoritmos MD5 con la finalidad de verificar que no ha sido modificado y también es capaz de extraer temporalmente ciertos archivos para identificar posible malwares y códigos maliciosos que han sido inyectados.

Suricata trabaja actualmente de acuerdo con los avances tecnológicos actuales, se planteó contribuir con dos aspectos fundamentales faltantes: una interfaz de administración, que facilite su gestión como solución empresarial y un módulo de detección de anomalías. (Astudillo Herrera, Jimenez Macias, & Ortiz Flores, 2012)

2.2.9.1 Características

Suricata es el nombre de un proyecto de software libre para un motor Sistema de Detección y Prevención de Intrusos o de manera abreviada IDS/IPS; fue desarrollado por la comunidad de OISF (Open Information Security Foundation). (Elhacker, 2017)

En la **Figura 7-2,** se muestran las características más relevantes de Suricata.

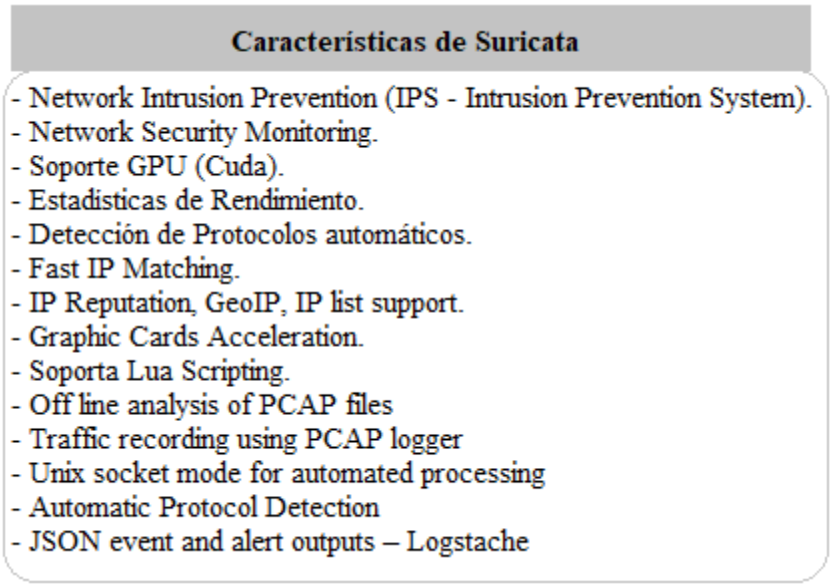

Figura 7-2: Características de Suricata

Fuente: http://blog.elhacker.net/2017/04/ids-ips-suricata-reglas-rules.html
Realizado por: Eduardo Arteaga

Distribuciones

- SELKS & Amsterdam
- SecurityOnion
- pfSense & OPNsense

Herramientas de Administración

- Evebox
- Scirius
- Kibana

Eventos de Procesamiento

- Mobster
- Barnyard2
- Logstash

2.2.9.2 Reglas

Una regla o firma, consiste en: acción, cabecera y opciones de regla, en la **Figura 8-2** se muestra un ejemplo de una regla de Suricata.

Figura 8-2: Estructura de una regla del IDS Suricata

Fuente: http://blog.elhacker.net/2017/04/ids-ips-suricata-reglas-rules.html
Realizado por: Eduardo Arteaga

En la **Tabla 4-2**, se describe los campos que debe contener un cabecera de una regla de Suricata, al momento de realizar su configuración.

Tabla 4-2: Rule Header

Action	Alert, pass, drop, reject
Protocols	TCP, UDP, ICMP, IP, HTTP, FTP, TLS, SMB, DNS
Source IP	$HOME_NET
Source Port	Any
Direction	->
Destination IP	$EXTERNAL_NET
Destination Port	80

Realizado por: Eduardo Arteaga

2.2.9.3 Plugins

Suricata, para un óptimo funcionamiento necesita tener instalado los siguientes complementos adicionales y opcionales, en la **Tabla 5-2**, se muestra el listado de los complementos de Suricata.

Tabla 5-2: Complementos de Suricata

Complementos	Tipo
Libcpre	Adicional
Libnet 1.1.x	Adicional
Libyaml	Adicional
Libpcap	Adicional
Libz	Adicional
Libpthread	Adicional
Libnetfilter_queue	Opcional
Libfnetlink	Opcional
Libcap-ng	Opcional
Libpfring	Opcional

Realizado por: Eduardo Arteaga

2.2.10 Benchmark

Los Benchmarks permiten averiguar el funcionamiento un Sistema de Detección de Intrusos frente a los ataques que se producen en la red, con la finalidad de cerciorarse de que el IDS, funciona correctamente y que las configuraciones realizadas se ajustan a los requerimientos tecnológicos de la institución u organización.

A la hora de configurar un sistema de detección de intrusos de forma correcta no es una tarea fácil de realizar, ya que actualmente existen varios softwares en el mercado para escoger y muchos de ellos no son adecuados debido a que no se pueden adaptar a la arquitectura tecnológica utilizada para garantizar la seguridad de la información que viaja por la red.

2.2.10.1Evaluación de IDS

Para detectar los ataques en los diferentes escenarios se utilizará la técnica de detección de anomalías.

En los IDSes basados en anomalías consiste en analizar el tráfico que viaja por la red para ver si el comportamiento de los usuarios se clasifica como ataque. Para ello, el sistema de detección de intrusos genera un autómata en el que asocia las comunicaciones a un determinado estado, y dependiendo de la actividad va cambiando la comunicación de estado hasta que se termine la comunicación o que llegue a un estado que se considera como un ataque. (Giménez, 2008)

La técnica de detección de anomalías sería perfecta siempre que este actualizada su base de datos de anomalías, que será fundamental para considerar como un ataque las posibles intrusiones a red; y todas sus combinaciones y variaciones posibles. Pero esta labor es imposible ya que no se pueden guardar todas las variaciones de cualquier ataque. Además, porque un IDS tiene que procesar casi en tiempo real los paquetes ya que de nada sirve que un IDS informe de lo que detectó en días anteriores. (Giménez, 2008)

En el momento que un sistema de detección de intrusos detecta una amenaza esta toma una decisión, éste puede tomarla de forma correcta o incorrecta. De acuerdo con los cuatro posibles estados existentes, ver **Figura 9-2.**

Figura 9-2: Estados de un IDS para una amenaza

Fuente: (Giménez, 2008)
Realizado por: Eduardo Arteaga

En la **Figura 10-2**, se describe la clasificación de un evento según la naturaleza para un IDS en el que se determina una detección normal o con anomalías.

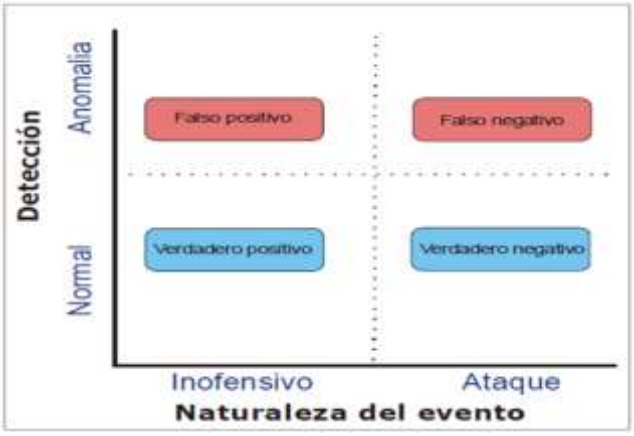

Figura 10-2: Naturaleza de un evento

Fuente: (Giménez, 2008)
Realizado por: Eduardo Arteaga

Las altas tasas de falsos positivos y de falsos negativos pueden minar los motivos para usar un IDS. Los falsos positivos ocupan tiempo o y recursos cuando el IDS genera falsas alarmas. Peor aún son los falsos negativos, que son todos los ataques que el IDS falla en detectar. Estas tasas complican la justificación del empleo de un IDS, pudiendo ser consecuencia de su arquitectura y configuración. Un IDS no debe ocupar recursos innecesarios. La mayor parte de alarmas probablemente son falsas. Para reducir el tiempo de trabajo de un IDS, hay que reducir el mayor número posible de las tasas de falsos negativos y de falsos positivos detectados. Hay que encontrar una solución que sea viable. Si ya se tiene fijada la arquitectura del IDS, se podrá modificar sus configuraciones con la finalidad de lograr un mayor rendimiento posible, atendiendo a las características de la red de datos y a las necesidades tecnológicas de esta, logrando de esta forma mitigar la tasa de errores que se pueden generar. (Giménez, 2008)

Para ver las mejoras de funcionamiento de un IDS, las pruebas de referencia deberían de realizarse sobre varios escenarios que contenga diferentes configuraciones de modo que los resultados puedan ser comparados. Las diferencias entre los resultados de las pruebas permitirán identificar el IDS que se adaptada a las características de la arquitectura de la red y de mejor desempeño al momento de detectar amenazas.

Los cambios tienen que estar basados en la reacción del sistema de detección de intrusos considerando las pruebas a las que se le someten durante el caso de pruebas. El número de alarmas tiene que ser comparado en base a la configuración del IDS utilizada. De las alarmas se tienen que encontrar información de cómo fueron producidas que herramientas polivalentes fueron empleadas para realizar la simulación de ataques como, por ejemplo: por un ataque genuino, un falso positivo, o si es posible determinar algunos falsos negativos que pueden producir un mal desempeño del IDS.

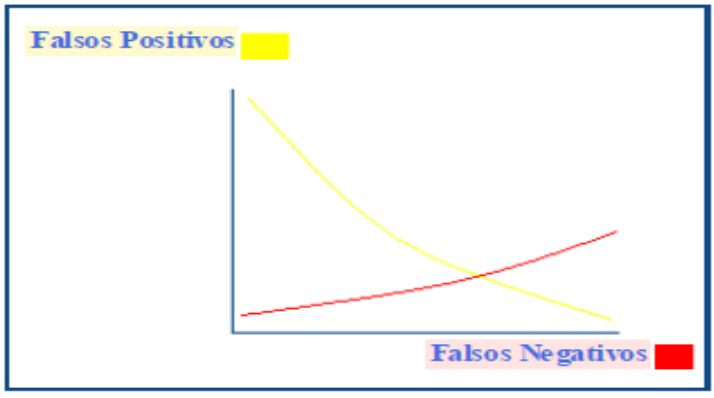

Figura 11-2: El modelo de la tasa de error en un IDS

Fuente: (Giménez, 2008)
Realizado por: Eduardo Arteaga

Los resultados obtenidos de las alertas detectadas por el IDS deberían estar en la zona cercana al lugar donde la tasa de falsos positivos y la de falsos negativos se cruzan, como se puede observar en el **Figura 11-2**.

(Giménez, 2008) indica que, en la evaluación de un IDS probando su configuración, se basa en encontrar algunos elementos que puedan al mayor rendimiento posible, por lo que es necesario averiguar:

- Disponibilidad de los métodos de prueba de referencia.
- Importantes criterios de prueba de configuración basados en la metodología.
- Estudiar las pruebas de penetración usadas sobre IDS, en lo referente a aspectos como el punto hasta el cual la prueba puede ser usada para mejorar las configuraciones, analizar las ventajas y los puntos débiles de las metodologías y del software empleado, etc.

El objetivo perseguido es determinar un IDS de software libre, que funcione de manera eficiente al momento de generar las alertas, probando los aspectos importantes de su funcionalidad, con el objetivo de mejorar la configuración y así reducir las tasas de error que pueden conducir a una mejor respuesta de seguridad y a una liberación de recursos. Para ello, se utilizarán un conjunto de herramientas, que reciben el nombre de Benchmarks.

2.2.10.2Herramientas para el aprendizaje y evaluación.

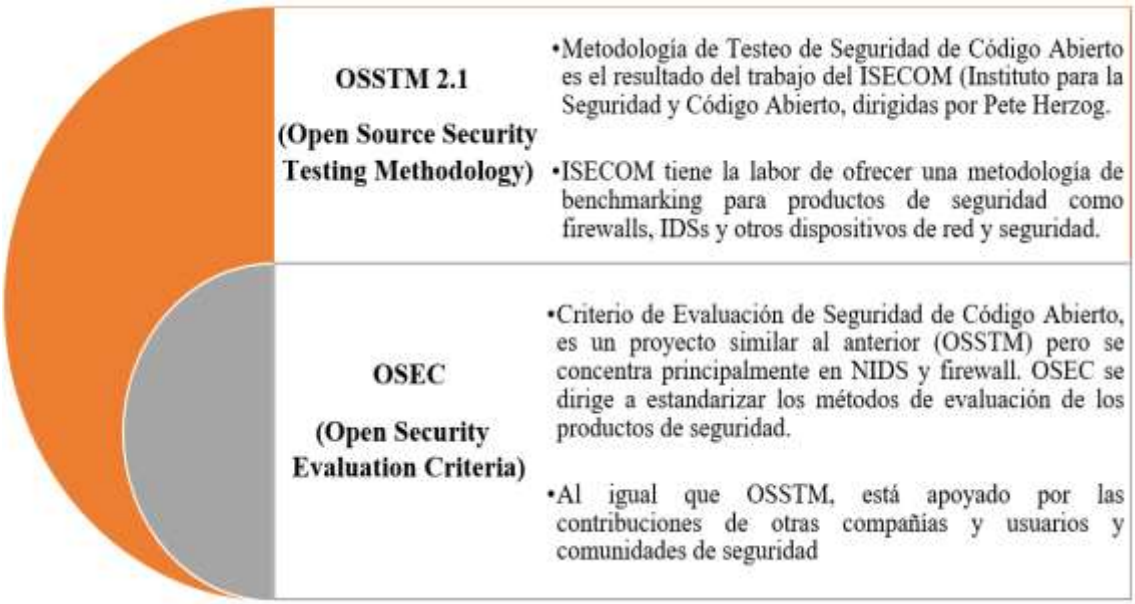

Figura 12-2: Metodologías de aprendizaje y evaluación de los IDS

Realizado por: Eduardo Arteaga

Para realizar el testeo y la evaluación de los sistemas de detección de intrusos, se cuenta con una variedad de metodologías y herramientas.

En la **Figura 12-2**, se describen las metodologías de aprendizaje y evaluación para los sistemas de detección de intrusos.

Entre las metodologías más importantes que existen para el aprendizaje y evaluación de los IDSes es la construcción de modelos de datos. Una vez desarrollado el modelo, se debe evaluar su precisión a través de los datos obtenidos como son: la tasa de aciertos, número de falsos positivos y número de falsos negativos. Para lo cual se usarán los conjuntos de datos tomados del internet que representen la actividad de una red tanto normal como anormal.

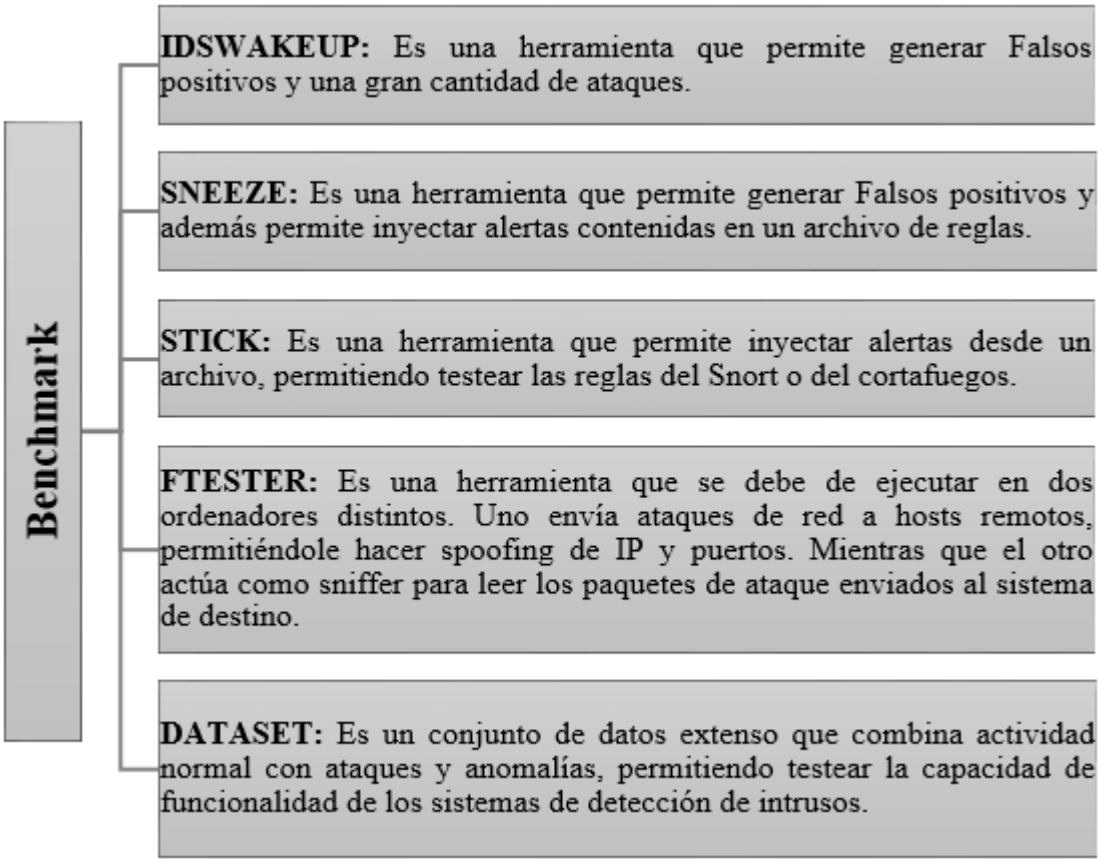

Figura 13-2: Benchmark

Realizado por: Eduardo Arteaga

Fuente: (Giménez, 2008)

A parte de estas metodologías de evaluación y aprendizaje, se encuentran otros Benchmarks de plataforma de libre distribución, que permiten a los usuarios generar grandes cantidades de ataques distintos y también se pueden utilizar para los ataques las propias reglas del IDS, con la

finalidad de poder evaluar su capacidad de detección y su desempeño, algunos de estos Benchmarks, se describen en la **Figura 13-2**.

Otro de los métodos más utilizados en el desarrollo de los sistemas de detección de intrusos consiste, en utilizar el tráfico de red generado en los mismos departamentos de la institución, añadiendo cierta actividad maliciosa a través de varias herramientas polivalentes. Cabe indicar que esta metodología tiene ciertas limitaciones con respecto a la topología de la red que este diseñada.

Para la evaluación de los IDSes de plataforma Open Source se utilizarán los DATASET 99, que contienen 3 semana de tráfico de entrada y salida.

2.2.10.3Herramientas polivalentes

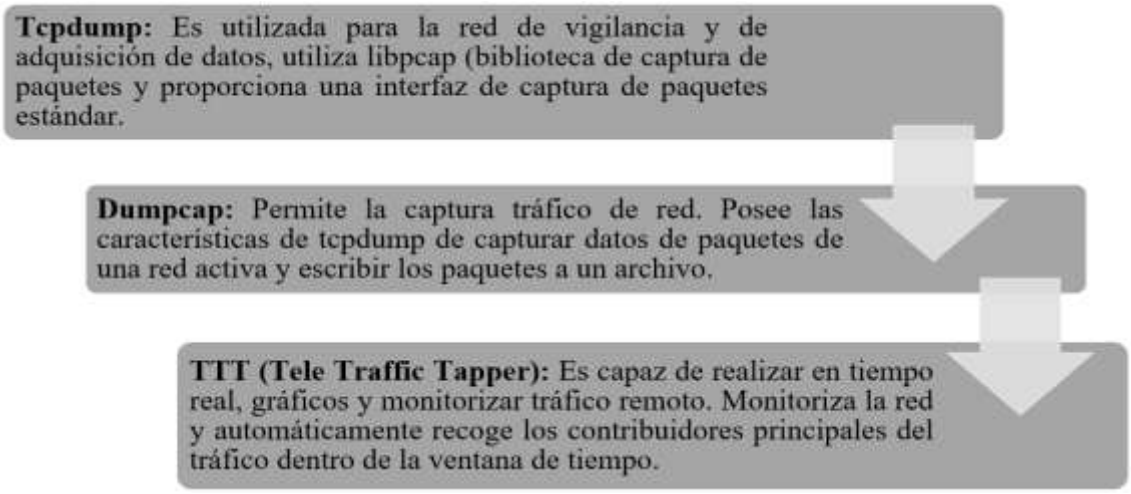

Figura 14-2: Captura de Datos

Fuente: (Giménez, 2008)
Realizado por: Eduardo Arteaga

Existe una variedad de herramientas de plataforma Open Source, que se pueden utilizar en conjunto con los Benchmarks para realizar tareas como: inyectar paquetes en la red, analizar el desempeño de la red, facilitar las tareas de administración de logs, la visualización de las tramas de paquetes que viajan por la red. En las **Figuras 14-2, 15-2 y 16-2**, se encuentra la descripción de las herramientas polivalentes más importantes. (Giménez, 2008)

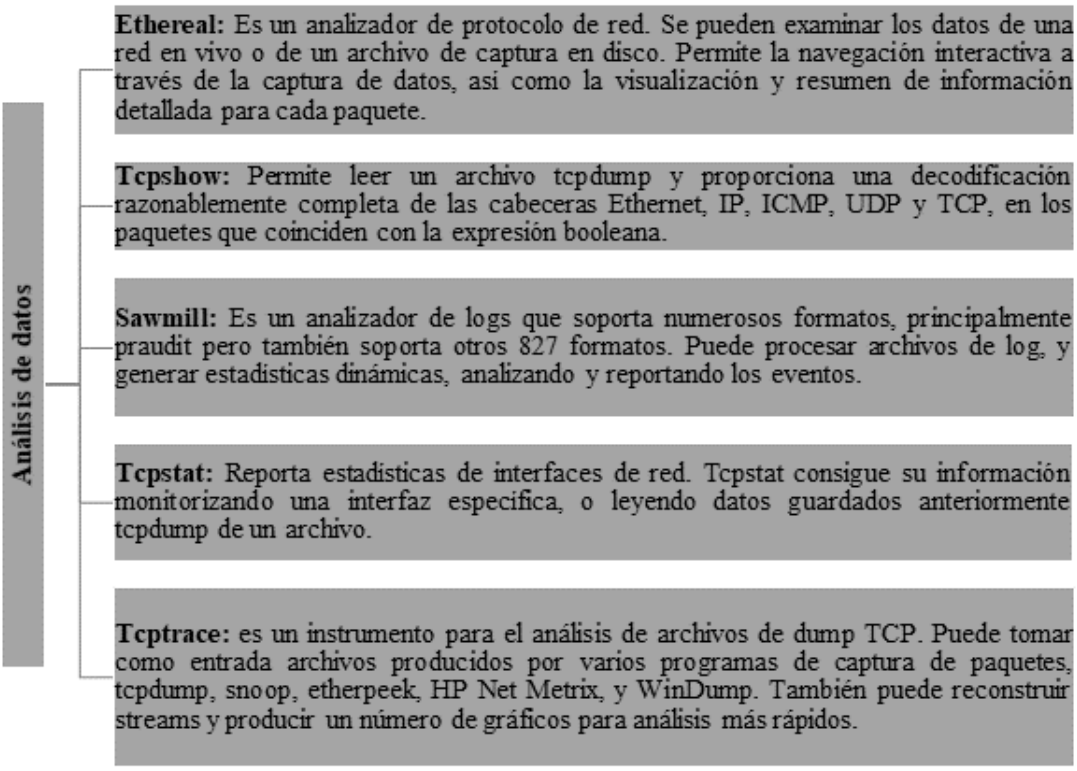

Figura 15-2: Análisis de datos

Fuente: (Giménez, 2008)

Realizado por: Eduardo Arteaga

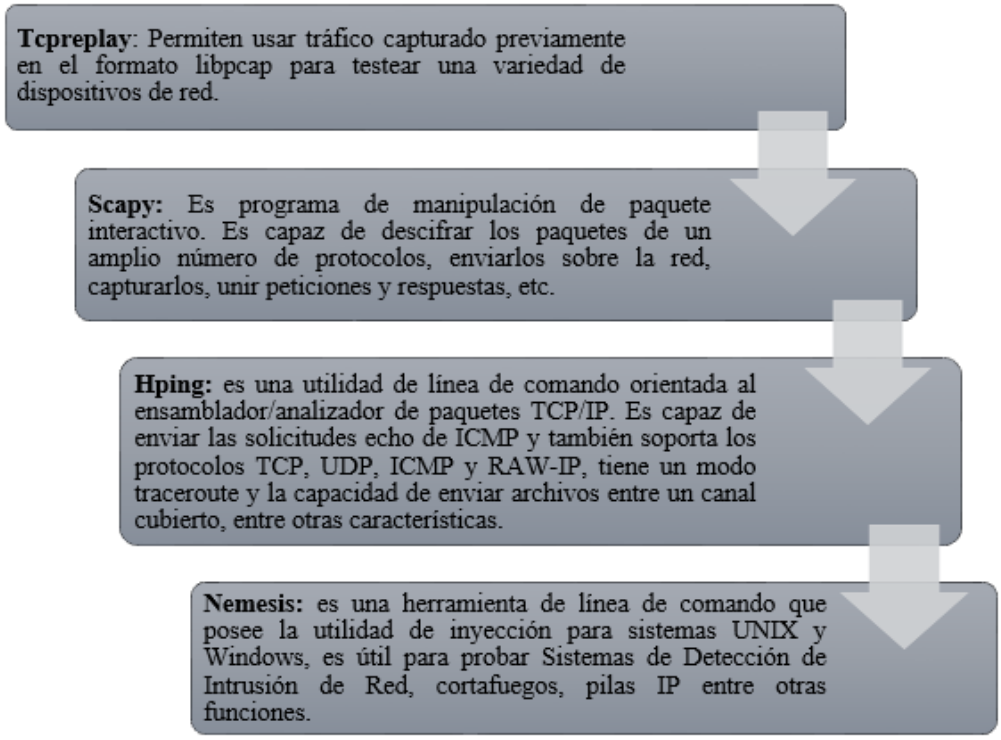

Figura 16-2: Manipulación y de inyección de paquetes

Fuente: (Giménez, 2008)

Realizado por: Eduardo Arteaga

CAPÍTULO III

3. METODOLOGÍA DE LA INVESTIGACIÓN

3.1 Tipo de investigación

Para la evaluación de las funcionalidades de los sistemas de detección de intrusos basados en la red de plataformas Open Source utilizando la técnica de detección de anomalías, se basó en dos tipos de investigación: aplicativa y cuasi-experimental.

- **Aplicativa:** En base a los conocimientos existentes, de investigaciones de los sistemas de detección de intrusos basados en la red de plataformas Open Source, se estableció la metodología y las herramientas de evaluación utilizando la técnica de detección de anomalías para Snort, Suricata y Bro.

- **Cuasi-experimental:** Para el análisis de los diferentes sistemas de detección de intrusos basados en la red de plataformas Open Source, se implementaron tres escenarios virtuales, para la instalación de: Snort, Suricata y Bro, para los cuales se crearon escenarios de ataques, mediante la técnica de detección de anomalías, para determinar la mejor opción de acuerdo a los últimos requerimientos de seguridad de la información.

3.2 Diseño de la Investigación

Para alcanzar los objetivos propuestos y responder a las interrogantes del presente trabajo de investigación, se planteó un diseño experimental en donde se propone realizar la evaluación de sistemas de detección de intrusos de plataformas Open Source basados en red utilizando la técnica de detección de anomalías, para lo cual se crearán varios escenarios virtuales utilizando Virtual-Box y otros, que permitirá la instalación de los diferentes sistemas de detección de intrusos de plataformas Open Source y través de clientes simulados se realizaron los escenarios de ataques con la finalidad de verificar su: precisión, rendimiento, tiempos de respuesta, tolerancia a fallos.

Para la comprobación de la hipótesis se implementó el escenario con los IDSes del caso de estudio para analizar el tráfico de la red de datos de la Secretaría de Hidrocarburos.

3.3 Métodos de Investigación

Los métodos de investigación utilizados en el desarrollo del presente trabajo de investigación son:

- **Método deductivo,** permitió la comprensión de conceptos, principios, definiciones, políticas en el ámbito de la seguridad específicamente en este trabajo de investigación lo relacionado a la detección de intrusos en la red.

- **Método inductivo**, permitió mediante el estudio de casos, hechos y situaciones determinar el mejor sistema de detección de intrusos basados en la red de plataforma Open Source que se acople a la situación de estudio.

- **Método Analítico**, mediante el proceso mental, se descompuso en partes un hecho o una idea, para mostrarlas, describirlas, numerarlas y para explicar las causas de los hechos o fenómenos a estudiar relativos a los sistemas de detección de intrusos y su aplicabilidad.

- **Método Sintético**, se reconstruyeron y unificaron ciertos elementos circundantes al problema de investigación para facilitar la comprensión cabal del mismo.

3.4 Enfoque de la Investigación

En la presente investigación se utilizó una combinación de los enfoques tanto cualitativo como cuantitativo para:

- Realizar la observación y evaluación de los sistemas de detección de intrusos de plataforma Open Source.
- Establecer las suposiciones o ideas de la evaluación de los sistemas de detección de intrusos de plataforma Open Source como producto de la observación y evaluaciones realizadas.
- Comprobar y demostrar el grado en que las suposiciones o ideas tienen fundamento.
- Analizar las suposiciones o ideas sobre la base de las pruebas o del análisis de los sistemas de detección de intrusos de plataforma Open Source.
- Proponer nuevas observaciones y evaluaciones para esclarecer, modificar, cimentar y/o fundamentar las suposiciones o ideas; o incluso para generar otras acerca de los IDSes de plataforma Open Source.

3.5 Alcance de la investigación

3.5.1 Explicativo

La presente investigación es explicativa debido al análisis de las causas y efectos de la relación entre las variables identificadas que llevará a una conclusión.

3.5.2 Correlacional

Esta investigación tiene carácter correlacional debido al propósito de mostrar los resultados de la variable independiente, en este caso, la evaluación de las funcionalidades de los IDSes basados en red de plataformas Open Source utilizando la técnica de anomalías permitirá identificar al mejor IDS.

3.6 Población de estudio

Para el presente trabajo de investigación, se consideró como población de estudio al número de eventos generados en la simulación de ataques en los diferentes escenarios de pruebas.

3.7 Unidad de análisis

El presente proyecto tiene por objeto evaluar las funcionalidades de los sistemas de detección de intrusos basado en red de plataforma Open Source, utilizando la técnica de detección de anomalías, para lo cual se evaluaron los resultados generados en cada uno de los escenarios de simulación de eventos de ataques informáticos.

3.8 Tamaño de la muestra

Para determinar el tamaño de la muestra se utilizó la siguiente fórmula, cabe mencionar que se desconoce el valor de la población total:

$$N = \frac{Z^2 \; x \; p \; x \; q}{d^2} \; ; \; N = \frac{(1.96)^2 \; x \; 0.05 \; x \; 0.95}{(0.03)^2}; \quad \text{N} = 203$$

En donde:

***N** = Tamaño de la muestra.*

***Z** = Nivel de confianza (1.96 = 95%).*

***p** = Probabilidad de éxito, o proporción esperada (0.05 = 50%).*

***q** = Probabilidad de fracaso (0.95).*

d^2 *= Precisión (0.03 =3%).*

3.9 Técnica de recopilación de datos primarios y secundarios

Las técnicas que se utilizaron en la investigación del proyecto son:

Búsqueda de información: Se investigó información necesaria acerca del objeto de estudio de la investigación para su desarrollo, utilizando las fuentes secundarias disponibles.

Pruebas: Permitió realizar experimentos en máquinas virtuales con la finalidad que evaluar los eventos generados por los sistemas detección de intrusos de plataforma Open Source, al momento de realizar la simulación de los ataques.

Observación directa: Permitió evaluar los resultados de las pruebas realizadas en los escenarios de virtualizados, con los diferentes eventos generados.

Análisis: Permitió cuantificar y calificar los resultados de la investigación.

Las fuentes que se tomaron como base para esta investigación son, se muestran en la **Tabla 1-3.**

Tabla 1-3: Fuentes de información para la investigación

FUENTES PRIMARIAS	FUENTES SECUNDARIAS
• Los eventos generados por los IDSes al momento de sufrir un ataque.	• Papers (Revistas científicas). • Investigaciones realizadas. • Publicaciones. • Libros. • Máquinas virtuales • Observaciones. • Textos. • Videos.

Realizado por: Eduardo Arteaga

3.10 Identificación de variables

3.10.1 Variable Independiente

Implementación de un sistema de detección de intrusos basado en la red de plataforma Open Source utilizando la técnica de detección de anomalías.

3.10.2 Variable dependiente

Seguridad de la información

3.11 Operacionalización de variables

En la **Tabla 2-3**, se muestra el detalle de la matriz de consistencia del proyecto de investigación.

Tabla 2-3: Matriz de consistencia

Problema	Objetivo general	Hipótesis	Variables	Indicadores	Índice	Técnicas	Instrumentos
¿La evaluación de funcionalidades de un sistema de detección de intrusos (IDS) de plataformas Open Source utilizando la técnica de anomalías permitirá determinar al mejor IDS, al momento de realizar la detección de amenazas?	Evaluar las funcionalidades de los sistemas de detección de intrusos basados en red de plataformas Open Source utilizando la técnica de detección de anomalías.	La implementación de un sistema de detección de intrusos basados en la red de plataforma Open Source utilizando la técnica de detección de anomalías mejorará la seguridad de la información que viaja por la red.	**Independiente** Implementación de un sistema de detección de intrusos basado en la red de plataforma Open Source utilizando la técnica de detección de anomalías.	• Casos de implementación	• Verdaderos negativos • Falsos positivos • Falsos negativos	• Testeo. • Toma de datos. • Escaneos. • Análisis de tráfico. • Informes.	• Putty. • VirtualBox. • IDSes • Base • Snorby • Barnyard2 • Mysql
			Dependiente Seguridad de la información.	• Funciones.	• Características	• Análisis de tráfico. • Observación. • Ataques a los IDS	• VirtualBox. • Wireshark. • Nmap. • Hydra • Nikto. • Putty. • Kaly Linux • Metasploitable.
				• Desempeño.	• Tiempos de respuesta • Paquetes perdidos		
				• Seguridad.	• Sensibilidad		

Realizado por: Eduardo Arteaga

3.12 Instrumentos de recolección de datos primarios y secundarios

Para la realización de ataques, se emplearon las herramientas disponibles en la distribución Kali Linux, Hping3, Nmap, Metasploitable[1].

Los instrumentos utilizados para recabar información de proyecto de investigación son:

- VirtualBox
- Wireshark
- Nmap
- Nikto
- Putty
- Snort
- Bro
- Suricata
- Kali Linux
- Metasploitable
- Filezilla

3.13 Instrumentos para procesar datos recopilados

Para el procesamiento de los datos recabados en la investigación se utilizaron los siguientes instrumentos:

- VirtualBox
- Nmap
- Nessus
- Nikto
- Programa estadístico (SBS)
- Metasploitable
- Base
- Barnyard

[1] **Metasploitable:** Son máquinas las cuales poseen determinados puertos abiertos corriendo determinados servicios. Puertos los cuales por estar corriendo servicios *no actualizados* o diversos motivos son vulnerables (no todos) y nos permiten realizar una intrusión a la máquina.

- Snortreport
- Snorby
- Excel
- Geogebra
- Edraw

3.14 Diseño de escenarios

Para la creación del laboratorio virtual, se utilizó un equipo con las siguientes características:

- Equipo: Laptop HP ENVY
- Procesador: Intel Core i7-4700MQ de 2.40 GHz
- RAM: 16 GB
- Almacenamiento 1TB
- Tipo de sistema: Windows 10, de 64 bits
- Tarjeta de red: Intel Centrino Wireless-N 2230

Utilizando el software Oracle WM Virtual Box en su versión 5.1.26, se crearon las 7 máquinas virtuales con los recursos tecnológicos necesarios para la instalación del sistema operativo base de Centos 7 sobre el cual se procedió con la instalación y configuración de los IDSes del estudio de investigación, herramientas de testeo, las mismas que se encuentran conectadas entre sí a través de la red interna, para interactuar unas a modo de servidor y otras a modo cliente, para generar los ataques necesarios, ver **Figura 1-3.**

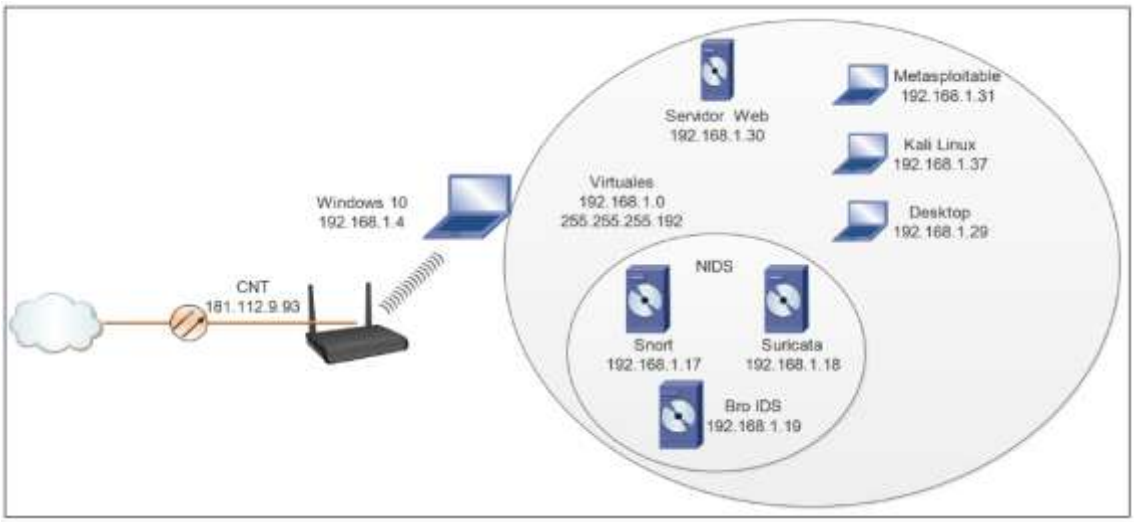

Figura 1-3: Esquema lógico del escenario de simulación

Realizado por: Eduardo Arteaga

El esquema lógico cuenta con varios segmentos de red que representan: la red interna (LAN), y red externa (WAN). cabe indicar que dentro de cada segmento se implementaron las máquinas virtuales con los IDSes y los clientes para simular los ataques.

La descripción de los segmentos de red y dispositivos generales se encuentran estructurados de la siguiente manera:

Escenario 1: Snort

Para el primer escenario, se utilizó la arquitectura descrita en la siguiente tabla, en donde se incluye la instalación de las herramientas necesarias para el análisis de la red, ver **Tabla 3-3.**

Tabla 3-1: Arquitectura del escenario 1

Descripción	Red	Características	Sistema Operativo	Herramientas
IDS Snort	Red interna	Memoria RAM 4 GB 2 procesadores Disco Duro de 60 GB	Centos 7	Snort (ver Anexo A) Barnyard2 (ver Anexo B) Adodb (ver Anexo C) Base (ver Anexo D) PHP, Httpd y Mysql Snortreport
Kali Linux	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 30 GB	Kali Linux	Hydra Hping3 Nmap Nikto2
Metasploitable	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 8 GB	Ubuntu 8	
Servidor WEB	Red Interna	Apache Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	Httpd
Cliente Centos	Red Interna	Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	
Pc Anfitrión	Red Interna y Externa	Memoria RAM 16 GB 8 procesadores Disco Duro de 1 TB	Windows 10	Virtual Box WireShark Putty FileZilla

Realizado por: Eduardo Arteaga

En el escenario 1, en la máquina virtual IDS Snort previamente creada e instalado el software base Centos 7, se procede a instalar Snort versión 2.9.9.0, de acuerdo con el Anexo A, una vez finalizado su instalación, se procede con la instalación y configuración de Barnyard2 para Snort de acuerdo con el Anexo B., con la finalidad de almacenar y procesar las salidas binarias unified2 de Snort en una base de datos MySQL, después se procede a instalar el conjunto de bibliotecas que permite la abstracción de las bases de datos para PHP como es ADOdb, ver Anexo C, finalmente se instala la interfaz web en PHP como es BASE de acuerdo con el Anexo D.

Escenario 2: Suricata

Para el segundo escenario, se utilizó la arquitectura descrita en la siguiente tabla, en donde se incluye la instalación de las herramientas necesarias para el análisis de la red, ver **Tabla 4-3.**

Tabla 4-3: Arquitectura del escenario 2

Descripción	Red	Características	Sistema Operativo	Herramientas
IDS Suricata	Red interna	Memoria RAM 4 GB 2 procesadores Disco Duro de 60 GB	Centos 7	Suricata (ver Anexo E) Snorby (ver Anexo F) Barnyard2 (ver Anexo G) Httpd Mysql
Kali Linux	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 30 GB	Kali Linux	Hydra Hping3 Nmap Nikto2
Metasploitable	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 8 GB	Ubuntu 8	
Servidor WEB	Red Interna	Apache Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	Httpd
Cliente Centos	Red Interna	Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	
Pc Anfitrión	Red Interna y Externa	Memoria RAM 16 GB 8 procesadores Disco Duro de 1 TB	Windows 10	Virtual Box WireShark Putty FileZilla

Realizado por: Eduardo Arteaga

En el escenario 2, en la máquina virtual IDS Suricata previamente creada e instalado el software base Centos 7, se procede a instalar Suricata versión 2.9.10, de acuerdo con el Anexo E, una vez finalizado su instalación, se procede con la instalación y configuración de la aplicación web Snorby de acuerdo con el Anexo F; con la finalidad de almacenar y procesar las salidas de Suricata en una base de datos MySQL se procede a instalar Barnyard2 de acuerdo con el Anexo G.

Escenario 3: Bro IDs

Para el tercer escenario, se utilizó la arquitectura descrita en la siguiente tabla, en donde se incluye la instalación de las herramientas necesarias para el análisis de la red, ver **Tabla 5-3.**

Tabla 5-3: Arquitectura del escenario 3

Descripción	Red	Características	Sistema Operativo	Herramientas
IDS BRO	Red interna	Memoria RAM 4 GB 2 procesadores Disco Duro de 60 GB	Centos 7	Bro (ver Anexo H) BroControl (ver Anexo I)
Kali Linux	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 30 GB	Kali Linux	Hydra Hping3 Nmap Nikto2
Metasploitable	Red interna	Memoria RAM 4 GB 1 procesador Disco Duro de 8 GB	Ubuntu 8	
Servidor WEB	Red Interna	Apache Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	Httpd
Cliente Centos	Red Interna	Memoria RAM 4 GB 1 procesador Disco Duro de 40 GB	Centos 7	
Pc Anfitrión	Red Interna y Externa	Memoria RAM 16 GB 8 procesadores Disco Duro de 1 TB	Windows 10	Virtual Box WireShark Putty FileZilla

Realizado por: Eduardo Arteaga

En el escenario 3, en la máquina virtual IDS BRO previamente creada e instalado el software base Centos 7, se procede a instalar BRO IDS versión 2.5.1, de acuerdo con el Anexo H, una vez finalizado su instalación, se procede con la instalación y configuración BroControl ver Anexo I.

Localización de los NIDS

Considerando las bases teóricas expuestas en la **Tabla 1-2,** acerca de la descripción del esquema de localizaciones donde implementar un IDS, para el presente estudio de investigación se consideró ubicar el IDS detrás del cortajuegos, con la finalidad de verificar las amenazas existentes en la red interna, las mismas que pueden ser generadas por los usuarios al momento de ingresar a sitios maliciosos o al hacer uso de dispositivos infectados con algún tipo de amenaza informática. Al momento de verificar la seguridad de una red, la mayoría de las instituciones cuentan con dispositivos para garantizar la seguridad de la red como son: Firewall, Sophos y Herramientas de monitoreo a nivel de red externa.

Esquema de red

Los escenarios planteados permiten efectuar ataques DOS al servidor DNS y HTTP internos y externos, con respecto al direccionamiento general:

- LAN 192.168.1.0/26 gw (192.168.1.1)
- WAN 181.112.9.93 (CNT Dinámica)

Tabla 2-3: Esquema de red de los sistemas virtualizados y anfitrión

RED	Tipo	S.O.	IP
WAN	Salida internet	N/A	181.112.9.93
LAN	Estación atacante	Kali Linux 3.18	192.168.1.37
	Estación normal/anfitrión	Windows 10	192.168.1.4
	Metasploitable	Centos 7	192.168.1.31
	Servidor WEB	Centos 7/Httpd	192.168.1.30
	Desktop	Centos 7	192.168.1.29
IDS	Snort	Centos 7	192.168.1.17
	Suricata	Centos 7	192.168.1.18
	Bros	Centos 7	192.168.1.19

Realizado por: Eduardo Arteaga

En la **Tabla 6-3**¸ se muestra el esquema de red de los sistemas virtualizados y anfitrión, definidos para los casos planteados en el proyecto de investigación.

De acuerdo con el conjunto de reglas

Para obtener resultados comparables sobre los indicadores a investigar, se hace uso de las reglas actualizadas propias de cada NIDS.

- Snort EmergingThreats Snort-2.9.0, actualizada a la fecha 01/09/2017.
- Suricata EmergingThreats, actualizada a la fecha 01/09/2017.

La cantidad de reglas para las dos herramientas NIDS son 1.170 para Snort, 1.193 para Suricata, actualizadas al año 2017. Sin embargo, se hace énfasis y uso para las intrusiones DOS, DNS, SCAN y otros. Cabe indicar que estas reglas pueden descargarse de las páginas oficiales con la finalidad de mantener actualizado las reglas.

Por otra parte, BRO IDS, se basa en un esquema políticas, directicas y plugins que son cargadas al momento de su instalación, también permite la posibilidad de descargar nuevos plugins desarrollados por la comunidad.

3.15 Descripción de los criterios de evaluación y sus parámetros

En este estudio se consideran las características más importantes de cada una de las técnicas de análisis más habituales utilizadas por los sistemas para la detección de intrusiones basados en red. En base a las siguientes premisas concernientes al descubrimiento de intrusiones en función de: minimización de detección errónea, maximización de la correcta detección de intrusiones, facilidad de despliegue, rapidez de despliegue y otros aspectos descritos a continuación:

- Facilidad de despliegue.
- Minimización de detección errónea: representa a la ausencia o mínima cantidad de falsos positivos detectados.
- Maximización de detección correcta: grado de confianza con relación a que las intrusiones sean detectadas correctamente, a esta definición corresponde falsos negativos detectados.
- Rapidez de despliegue: representa si existe un periodo de entrenamiento requerido antes de la implementación y uso de la técnica.

- Precisión de detección de intrusiones habituales: si la técnica detecta eficazmente intrusiones o ataques habituales y bien conocidos.
- Optimización con respeto al tiempo: el tiempo que representa procesar los eventos y emitir alertas correspondientes a los ataques sufridos.

3.16 Ponderación de evaluación

Con el fin de efectuar la evaluación de Snort y Suricata, se utiliza el método de evaluación sumaria escala de Likert y para ello se crea la siguiente **Tabla 7-3**. con las ponderaciones de evaluación cualitativamente, cuantitativamente y en una escala porcentual. La escala gradual es de 0 a 5 puntos de acuerdo con el cumplimiento de los indicadores expuestos.

Tabla 7-3: Ponderación de evaluación de los IDSes

Calificación cualitativa	Valor asignado	Porcentaje
No existe	0	0%
Malo	1	20%
Regular	2	40%
Bueno	3	60%
Muy bueno	4	80%
Excelente	5	100%

Realizado por: Eduardo Arteaga

3.17 Benchmark

Una vez instalados los NIDS y previa evaluación de su funcionalidad, se determina que: Snort y Suricata están basados en firmas y su funcionalidad se basa en el conocimiento sobre malwares y utilizan el mismo conjunto de reglas; por otro lado, Bro IDS se basa en el comportamiento y mantiene una estructura basada en la interpretación de políticas, plugins y scripts que son cargados al momento de su instalación, razón por la cual dentro del proceso de investigación realizado, se descarta la comparación de Bro IDS, debido a que su arquitectura a evaluar difiere con las características de Snort y Suricata y no se podrían establecer parámetros de igual similitud.

Los Benchmarks permitieron averiguar el funcionamiento de los Sistemas de Detección de Intrusos del proyecto de investigación frente a los ataques, con la finalidad de verificar de que el IDS funciona correctamente. Para evaluar el rendimiento de Snort y Suricata se ha definido Benchmarks de software libre.

Para la detección de los ataques por parte de los IDS, se utilizó la técnica de detección de anomalías basado en datos estadísticos, para lo cual se utilizaron casos de entrenamiento y simulación de ataques, mediante los cuales se analizó el tráfico de la red, con la finalidad de evaluar el comportamiento de los eventos generados y su clasificación, ya sea como un verdadero negativo, falso positivo y verdadero positivo; con la finalidad de comprobar la hipótesis de la investigación se realizó un caso aplicativo de evaluación de tráfico real de la red de datos de la Secretaría de Hidrocarburos, utilizando los escenarios virtuales.

3.18 Análisis del rendimiento de los IDSes

3.18.1 *Recolección de información*

Como fuente primaria, se generó tráfico normal y anormal en la red de cada uno de los escenarios, con la utilización de herramientas polivalentes como son: Hping3, Hydra, Nmap, Nikto. Con la finalidad de realizar la comprobación de la hipótesis se captura de tráfico de la red de datos de los servidores de la Secretaría de Hidrocarburos, el cual fue facilitado con la ayuda del administrador de servidores de la institución, utilizando la herramienta Wireshark, la captura se realizó en diferentes días en horarios distintos, obteniéndose dos archivos de captura en formato. pcap.

3.18.2 Interpretación de la información

Para realizar un análisis de las alertas emitidas por lo IDSes en los diferentes escenarios, se utilizaron las herramientas Snortreport, BASE, SNORBY, Wireshark y los logs, con finalidad de realizar la correlación de alertas.

Para los archivos .pcap capturados, se encuentran desglosados por protocolo de red, lo que permite tener una idea aproximada de las reglas que debería activarse si Snort y Suricata, encuentran patrones característicos de actividad maliciosa.

3.19 Procesamiento y Análisis

Para el procesamiento de la información por parte de los IDSes, se han definido 3 casos:

- Caso 1: Entrenamiento
- Caso 2: Simulación
- Caso 3: Aplicativo

3.19.1 Caso 1: Entrenamiento

Con la finalidad de evaluar los sistemas de detección de intrusos como son: Snort y Suricata, se estableció la utilización de los conjuntos de datos de DARPA 99, los mismos que están diseñados para realizar la evaluación de las tasas de falsas alarmas y las tasas de detección de los diferentes IDSes del proyecto de investigación. Los conjuntos de datos de DARPA 99, constan de tres semanas de tráfico, en el que se incluye tráfico normal y malicioso, los mismos que fueron utilizados para el caso de aprendizaje con los escenarios de Snort y Suricata, En la **Tabla 8-3** se muestran los resultados obtenidos por Suricata en el caso de entrenamiento con tráfico normal.

Tabla 8-3: Alertas detectadas por Suricata en el caso de entrenamiento con tráfico normal

N°	Alertas	Clasificación	Número
1	ET CHAT IRC JOIN command	Misc activity	21534
2	ET CHAT IRC NICK command	Misc activity	22666
3	ET CHAT IRC PING command	Misc activity	19499
4	ET CHAT IRC PONG response	Misc activity	1084
5	ET CHAT IRC PRIVMSG command	Misc activity	28004
6	ET CHAT IRC USER command	Misc activity	15061
7	ET POLICY Outbound Multiple Non-SMTP Server Emails	Misc activity	2991
8	SURICATA DNS flow memcap reached		7356
9	SURICATA DNS malformed request data		149
10	SURICATA DNS malformed response data		147
11	SURICATA DNS Not a request		803
12	SURICATA DNS Not a response		138
13	SURICATA DNS Unsollicited response		7501
14	SURICATA HTTP invalid request field folding	Generic Protocol Command Decode	123
15	SURICATA HTTP request field missing colon	Generic Protocol Command Decode	94
16	SURICATA HTTP URI terminated by non-compliant character	Generic Protocol Command Decode	512
17	SURICATA HTTP request field too long	Generic Protocol Command Decode	616
18	SURICATA HTTP response header invalid	Generic Protocol Command Decode	12
19	SURICATA HTTP responsefield missing colon	Generic Protocol Command Decode	135
20	SURICATA STREAM 3way handshake with ack in wrong dir	Generic Protocol Command Decode	467

N°	Alertas	Clasificación	Número
21	SURICATA STREAM 3way handshake wrong seq wrong ack	Generic Protocol Command Decode	28
22	SURICATA STREAM CLOSEWAIT ACK out of window	Generic Protocol Command Decode	255
23	SURICATA STREAM CLOSEWAIT FIN out of window	Generic Protocol Command Decode	58
24	SURICATA STREAM CLOSEWAIT invalid ACK	Generic Protocol Command Decode	95
25	SURICATA STREAM ESTABLISHED invalid ack	Generic Protocol Command Decode	19285
26	SURICATA STREAM ESTABLISHED packet out of window	Generic Protocol Command Decode	42218
27	SURICATA STREAM ESTABLISHED SYN resend with different seq	Generic Protocol Command Decode	35
28	SURICATA STREAM ESTABLISHED SYNACK resend with different ACK	Generic Protocol Command Decode	35
29	SURICATA STREAM excessive retransmissions	Generic Protocol Command Decode	4
30	SURICATA STREAM FIN invalid ack	Generic Protocol Command Decode	175
31	SURICATA STREAM FIN out of window	Generic Protocol Command Decode	92
32	SURICATA STREAM Packet with invalid ack	Generic Protocol Command Decode	7777
		Totales	**198949**

Realizado por: Eduardo Arteaga

Una vez procesado el conjunto de datos de Darpa 99 correspondiente a las 2 semanas de tráfico normal por Suricata, se obtiene un total de 198949 alertas, las mismas que de acuerdo con el IDS las clasifica en una categoría determinada como se observa en la **Tabla 8-3**, este conjunto de alertas servirá de línea base para determinar los falsos positivos (FP) en las evaluaciones siguientes con respecto a Suricata.

Una vez que se determinada la línea de falsos positivos para Suricata, se procede con el análisis de conjunto de datos de Darpa 99 de la semana 3 utilizando Suricata, conjunto de datos que contiene tráfico malicioso. Una vez terminado el análisis de los paquetes se obtienen 235970 alertas, las mismas que fueron correlacionadas considerando la línea base de falsos positivos (FP) de la **Tabla 8-3** y se obtienen 166 verdaderos negativos (VN) y 235804 falsos positivos (FP), en la **Tabla 9-3** se muestra el detalle de los resultados obtenidos.

Tabla 3-3: Correlación de alertas detectadas por Suricata en el caso de entrenamiento

N°	Alertas	Clasificación	VN	FP	Número
1	SURICATA SMTP invalid reply	Generic Protocol Command Decode	75		75
2	ET CHAT IRC JOIN command	Misc activity		14529	14529
3	ET CHAT IRC NICK command	Misc activity		14511	14541
4	ET CHAT IRC PING command	Misc activity		12651	12651
5	ET CHAT IRC PONG response	Misc activity		698	698
6	ET CHAT IRC PRIVMSG command	Misc activity		144529	144529
7	ET CHAT IRC USER command	Misc activity		15437	15437
8	ET POLICY Outbound Multiple Non-SMTP Server Emails	Misc activity		3346	3346
9	SURICATA DNS flow memcap reached			1047	1047
10	SURICATA DNS malformed request data			164	164
11	SURICATA DNS malformed response data			206	206
12	SURICATA DNS Not a request			129	129
13	SURICATA DNS Not a response			154	154
14	SURICATA DNS Unsollicited response			1351	1351
15	SURICATA HTTP invalid request field folding	Protocol-command-decode		58	58
16	SURICATA HTTP request field missing colon	Protocol-command-decode		60	60
17	SURICATA HTTP response field missing colon	Protocol-command-decode	30		30
18	SURICATA HTTP response header invalid	Generic Protocol Command Decode		16	16
19	SURICATA HTTP URI terminated by non-compliant character	Protocol-command-decode		4	4
20	SURICATA SMTP no server welcome message	Protocol-command-decode	60		60
21	SURICATA STREAM 3way handshake right seq wrong ack evasion	Protocol-command-decode	1		1
22	SURICATA STREAM 3way handshake with ack in wrong dir	Generic Protocol Command Decode		10	10
23	SURICATA STREAM 3way handshake wrong seq wrong ack	Generic Protocol Command Decode		21	21
24	SURICATA STREAM ESTABLISHED invalid ack	Generic Protocol Command Decode		5820	5820
25	SURICATA STREAM ESTABLISHED packet out of window	Generic Protocol Command Decode		15098	15098
26	SURICATA STREAM FIN invalid ack	Generic Protocol Command Decode		620	620
27	SURICATA STREAM FIN out of window	Generic Protocol Command Decode		35	35

N°	Alertas	Clasificación	VN	FP	Número
28	SURICATA STREAM Packet with invalid ack	Generic Protocol Command Decode		5310	5310
			Totales	**166**	**235804**	**235970**

Realizado por: Eduardo Arteaga

Una vez procesado el conjunto de datos de Darpa 99 correspondiente a 2 semanas de tráfico normal por Snort, se obtiene un total de 361423 alertas, las mismas que de acuerdo con el IDS las clasifica en una categoría determinada como se observa en la **Tabla 9-3**, este conjunto de alertas servirá de línea base para determinar los falsos positivos (FP) en las evaluaciones siguientes con respecto a Snort.

Tabla 10-3: Alertas detectadas por Snort en el caso de entrenamiento

N°	Alertas	Clasificación	Número
1	FIN number is greater than prior FIN	Potentially Bad Traffic	1
2	(ftp_telnet) Telnet Subnegotiation Begin Command without Subnegotiation End	Generic Protocol Command Decode	2
3	(http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE	Unknown Traffic	47373
4	(http_inspect) JAVASCRIPT WHITESPACES EXCEEDS MAX ALLOWED	Unknown Traffic	29
5	(http_inspect) LONG HEADER	Potentially Bad Traffic	468
6	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	Unknown Traffic	66466
7	(http_inspect) SIMPLE REQUEST	Unknown Traffic	232
8	(http_inspect) UNESCAPED SPACE IN HTTP URI	Unknown Traffic	1202
9	(http_inspect) UNKNOWN METHOD	Unknown Traffic	50
10	(spp_sdf) SDF Combination Alert	Senstive Data	13040
11	(spp_ssh) Protocol mismatch	Detection of a non-standard protocol or event	229509
12	Bad segment, adjusted size <= 0	Potentially Bad Traffic	44
13	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	2974
14	FIN number is greater than prior FIN	Potentially Bad Traffic	16
15	Limit on number of overlapping TCP packets reached	Potentially Bad Traffic	2
16	Reset outside window	Potentially Bad Traffic	15
		Totales	**361423**

Realizado por: Eduardo Arteaga

En la **Tabla 10-3** se muestra los resultados obtenidos por Snort en el caso de entrenamiento con tráfico normal.

Una vez que se determinada la línea de falsos positivos para Snort, se procede con el análisis de conjunto de datos de Darpa 99 de la semana 3 utilizando Snort, conjunto de datos que contiene tráfico malicioso. Una vez terminado el análisis de los paquetes se obtienen 232363 alertas, las mismas que fueron correlacionadas considerando la línea base de falsos positivos (FP) de la **Tabla 10-3** anterior y se obtienen 20021 verdaderos negativos (VN) y 212342 falsos positivos (FP), en la **Tabla 11-3** siguiente se muestra el detalle al respecto.

Tabla 11-3: Correlación de alertas detectadas por Snort en etapa de entrenamiento

N°	Alertas	Clasificación	VN	FP	Número
1	(ftp_telnet) FTP bounce attempt	Potentially Bad Traffic	0	16	16
2	(http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE	Unknown Traffic	20019	0	20019
3	(http_inspect) JAVASCRIPT WHITESPACES EXCEEDS MAX ALLOWED	Unknown Traffic	0	4	4
4	(http_inspect) LONG HEADER	Potentially Bad Traffic	0	151	151
5	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	Unknown Traffic	0	28842	28842
6	(http_inspect) SIMPLE REQUEST	Unknown Traffic	0	263	263
7	(http_inspect) UNESCAPED SPACE IN HTTP URI	Unknown Traffic	0	752	752
8	(http_inspect) UNKNOWN METHOD	Unknown Traffic	0	28	28
9	(IMAP) Unknown IMAP4 command	Generic Protocol Command Decode	0	3	3
10	(spp_sdf) SDF Combination Alert	Senstive Data	0	6384	6384
11	(spp_ssh) Protocol mismatch	Detection of a non-standard protocol or event	0	175119	175119
12	Bad segment, adjusted size	Potentially Bad Traffic	0	1	1
13	Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	0	721	721
14	FIN number is greater than prior FIN	Potentially Bad Traffic	0	20	20
15	Reset outside window	Potentially Bad Traffic	0	38	38
16	SENSITIVE-DATA Email Addresses	Senstive Data	2	0	2
		Totales	**20021**	**212342**	**232363**

Realizado por: Eduardo Arteaga

3.19.2 Caso 2: Simulación

Para la evaluación de los IDSes del proyecto de investigación en el caso de simulación, se desarrolló un ambiente de pruebas para el cual se generaron ataques: DoS (Denial Of Service - Ataques de Denegación de Servicios), R2L (Remote to Local - Ataques de acceso Remoto a Local), U2R (User to Root – Ataques de Usuario a Súper usuario) y Probing (Sondeo de redes). Para realizar la exploración de la red y la generación de ataques se utilizó Kali Linux, con las siguientes aplicaciones:

Nmap: Mediante esta utilidad de código abierto y gratuito se realizó la exploración de las redes internas de los diferentes escenarios con la finalidad de realizar un sondeo de seguridad de los puertos de los equipos tanto de servidores como de host, en la imagen siguiente se muestra un escaneo con Nmap hacia un host, ver **Figura 2-3.**

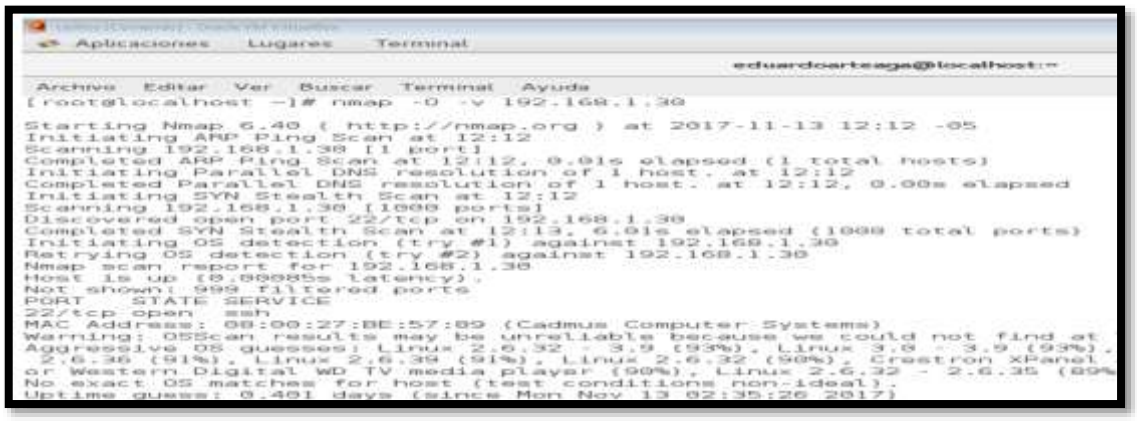

Figura 1-3: Escaneo con detección de servicios utilizando Nmap

Realizado por: Eduardo Arteaga

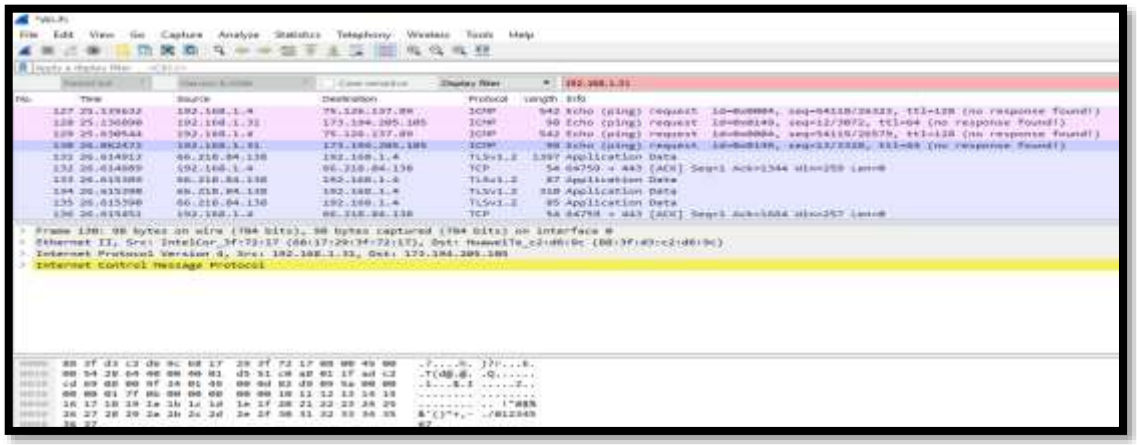

Figura 3-2: Análisis del comportamiento de la red capturado con Wireshark

Realizado por: Eduardo Arteaga

Wireshark: Permitió realizar el análisis de paquetes de red, con la finalidad de determinar los protocolos, el comportamiento y captura de tráfico de la red, en la **Figura 3-3** se puede apreciar el monitoreo de la red.

NIKTO2: Nikto2 se utilizó para realizar pruebas exhaustivas a los servidores web caso de estudio, para verificar los elementos de configuración del servidor, como la presencia de varios archivos de índice, las opciones del servidor HTTP, identificar los servidores web y el software instalados, en la **Figura 4-3** se muestra la ejecución de Nikto hacia un servidor web.

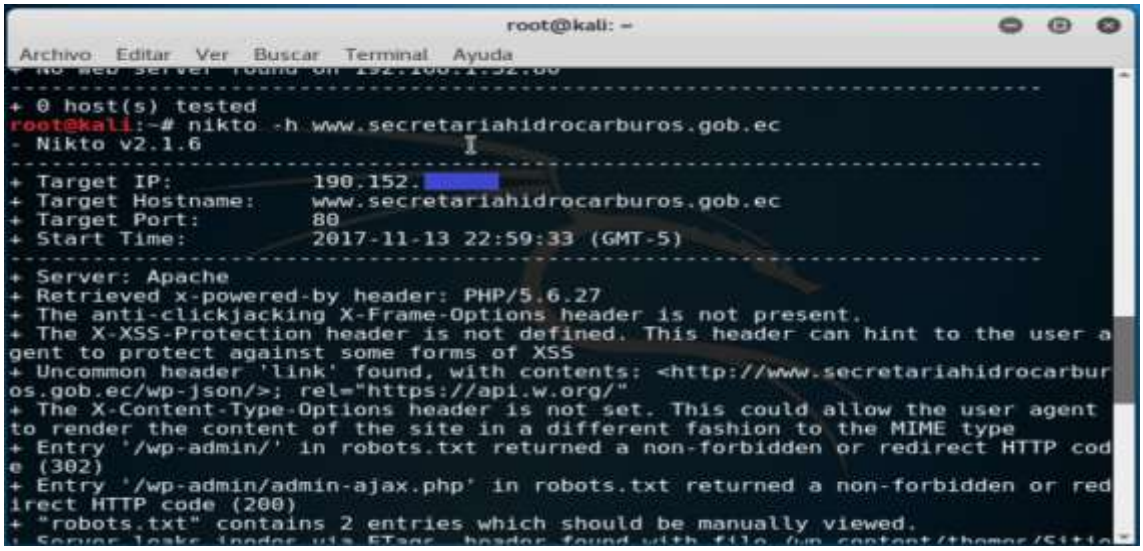

Figura 4-3: Ejecución de Nikto2

Realizado por: Eduardo Arteaga

HPING3: Mediante el uso de la herramienta Hping se enviaron peticiones de eco TCP, ICMP, UDP y RAW-IP hacia los equipos de la red de los diferentes escenarios, en la **Figura 5-3,** se muestra su ejecución.

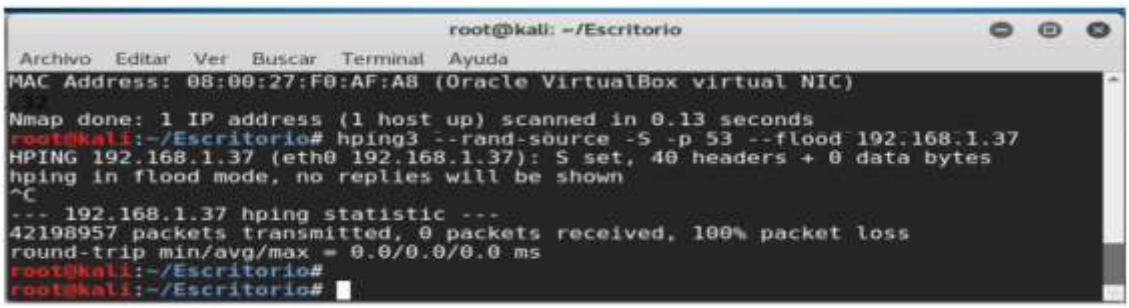

Figura 5-3: Ejecución de Hping3

Realizado por: Eduardo Arteaga

Hydra: Mediante Hydra se lanzaron ataques de fuerza bruta hacía servidores y equipos, con la finalidad de adivinar las contraseñas de estos al hacer uso de un diccionario de datos. En la **Figura 6-3**, se puede ver su ejecución.

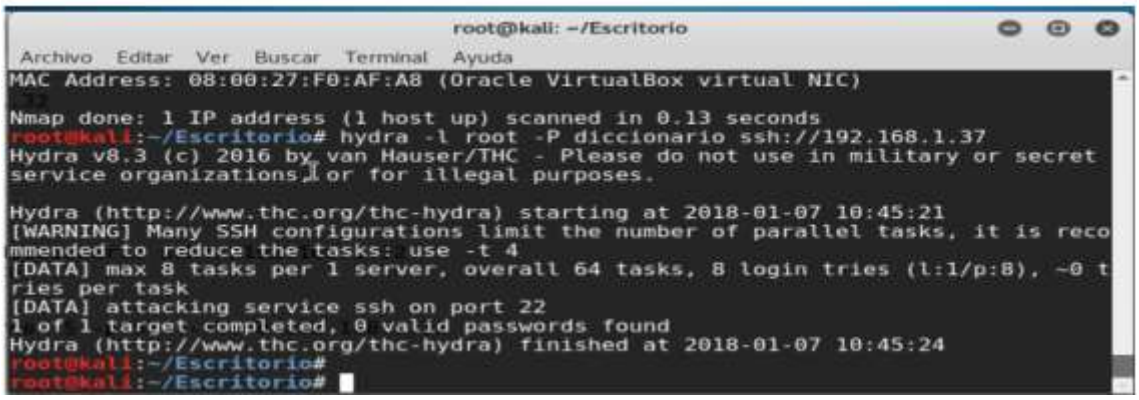

Figura 6-3: Ejecución de Hydra

Realizado por: Eduardo Arteaga

Una vez finalizado la ejecución de las herramientas descritas anteriormente, las cuales permitieron realizar la simulación de ataques hacía los servidores y host del escenario del Suricata, se obtienen los siguientes resultados, ver **Tabla 12-3**.

Tabla 4-3: Descripción de intrusiones detectadas por Suricata caso de simulación

Item	Alerta	Clasificación	VN	NP
1	signature:"ET WEB_SERVER Possible CVE-2014-6271 Attempt in Headers"	category:"Attempted Administrator Privilege Gain"	122	0
2	signature:"SURICATA HTTP Host header invalid"	category:"Generic Protocol Command Decode"	3	0
3	signature:"SURICATA HTTP invalid content length field in request"	category:"Generic Protocol Command Decode"	3	0
4	signature:"SURICATA STREAM 3way handshake with ack in wrong dir"	category:"Generic Protocol Command Decode"	0	2
5	signature:"SURICATA STREAM 3way handshake wrong seq wrong ack"	category:"Generic Protocol Command Decode"	0	2
6	signature:"SURICATA STREAM bad window update"	category:"Generic Protocol Command Decode"	1	0
7	signature:"SURICATA STREAM excessive retransmissions"	category:"Generic Protocol Command Decode"	1	0
8	signature:"SURICATA STREAM Packet with invalid ack"	category:"Generic Protocol Command Decode"	1	0

Item	Alerta	Clasificación	VN	NP
9	signature:"SURICATA STREAM SHUTDOWN RST invalid ack"	category:"Generic Protocol Command Decode"	1	0
10	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 22"	category:"Misc Attack"	1	0
11	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 26"	category:"Misc Attack"	1	0
12	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 28"	category:"Misc Attack"	1	0
13	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 49"	category:"Misc Attack"	1	0
14	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 52"	category:"Misc Attack"	1	0
15	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 53"	category:"Misc Attack"	1	0
16	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 56"	category:"Misc Attack"	1	0
17	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 59"	category:"Misc Attack"	1	0
18	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 60"	category:"Misc Attack"	1	0
19	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 71"	category:"Misc Attack"	1	0
20	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 81"	category:"Misc Attack"	2	0
21	signature:"ET CINS Active Threat Intelligence Poor Reputation IP group 90"	category:"Misc Attack"	1	0
22	signature:"ET COMPROMISED Known Compromised or Hostile Host Traffic group 12"	category:"Misc Attack"	1	0
23	signature:"ET COMPROMISED Known Compromised or Hostile Host Traffic group 2"	category:"Misc Attack"	1	0
24	signature:"ET COMPROMISED Known Compromised or Hostile Host Traffic group 31"	category:"Misc Attack"	1	0
25	signature:"ET DROP Dshield Block Listed Source group 1"	category:"Misc Attack"	17	0
26	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 1"	category:"Misc Attack"	1993	0
27	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 10"	category:"Misc Attack"	2910	0
28	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 11"	category:"Misc Attack"	1964	0
29	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 12"	category:"Misc Attack"	2876	0

Item	Alerta	Clasificación	VN	NP
30	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 13"	category:"Misc Attack"	9151	0
31	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 14"	category:"Misc Attack"	6051	0
32	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 15"	category:"Misc Attack"	3747	0
33	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 16"	category:"Misc Attack"	1837	0
34	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 17"	category:"Misc Attack"	355	0
35	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 18"	category:"Misc Attack"	96	0
36	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 19"	category:"Misc Attack"	97	0
37	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 2"	category:"Misc Attack"	29664	0
38	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 20"	category:"Misc Attack"	244	0
39	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 21"	category:"Misc Attack"	81	0
40	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 22"	category:"Misc Attack"	145	0
41	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 23"	category:"Misc Attack"	845	0
42	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 24"	category:"Misc Attack"	57	0
43	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 25"	category:"Misc Attack"	86	0
44	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 27"	category:"Misc Attack"	549	0
45	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 28"	category:"Misc Attack"	10	0
46	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 29"	category:"Misc Attack"	71	0
47	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 3"	category:"Misc Attack"	1086	0
48	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 30"	category:"Misc Attack"	90	0
49	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 31"	category:"Misc Attack"	404	0
50	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 32"	category:"Misc Attack"	570	0

Item	Alerta	Clasificación	VN	NP
51	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 33"	category:"Misc Attack"	608	0
52	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 34"	category:"Misc Attack"	1831	0
53	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 4"	category:"Misc Attack"	119	0
54	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 5"	category:"Misc Attack"	18	0
55	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 6"	category:"Misc Attack"	1700	0
56	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 7"	category:"Misc Attack"	204	0
57	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 8"	category:"Misc Attack"	2504	0
58	signature:"ET DROP Spamhaus DROP Listed Traffic Inbound group 9"	category:"Misc Attack"	2887	0
59	signature:"ET TOR Known Tor Exit Node Traffic group 1"	category:"Misc Attack"	1	0
60	signature:"ET TOR Known Tor Exit Node Traffic group 14"	category:"Misc Attack"	1	0
61	signature:"ET TOR Known Tor Exit Node Traffic group 30"	category:"Misc Attack"	1	0
62	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 1"	category:"Misc Attack"	1	0
63	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 109"	category:"Misc Attack"	1	0
64	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 129"	category:"Misc Attack"	1	0
65	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 14"	category:"Misc Attack"	1	0
66	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 265"	category:"Misc Attack"	1	0
67	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 30"	category:"Misc Attack"	1	0
68	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 309"	category:"Misc Attack"	1	0
69	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 316"	category:"Misc Attack"	1	0
70	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 375"	category:"Misc Attack"	1	0
71	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 455"	category:"Misc Attack"	1	0

Item	Alerta	Clasificación	VN	NP
72	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 458"	category:"Misc Attack"	1	0
73	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 462"	category:"Misc Attack"	1	0
74	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 495"	category:"Misc Attack"	1	0
75	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 515"	category:"Misc Attack"	2	0
76	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 528"	category:"Misc Attack"	1	0
77	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 569"	category:"Misc Attack"	1	0
78	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 627"	category:"Misc Attack"	1	0
79	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 669"	category:"Misc Attack"	1	0
80	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 688"	category:"Misc Attack"	1	0
81	signature:"ET TOR Known Tor Relay\/Router (Not Exit) Node Traffic group 689"	category:"Misc Attack"	1	0
82	signature:"ET POLICY Possible Kali Linux hostname in DHCP Request Packet"	category:"Potential Corporate Privacy Violation"	1	0
83	signature:"ET WEB_SERVER ColdFusion administrator access"	category:"Web Application Attack"	5	0
84	signature:"ET WEB_SERVER ColdFusion componentutils access"	category:"Web Application Attack"	1	0
		Σ	**75046**	**4**

Número total de alertas detectadas: 75050

Observaciones:

Falsos negativos (FN)

- Escaneo de Nmap = 2
- Escaneo de Hydra = 2

Realizado por: Eduardo Arteaga

Según la **Tabla 12-3,** de resultados se puede apreciar que Suricata determinó 75046 verdaderos negativos (VN), 4 falsos positivos (FP) y 4 falsos negativos, cabe indicar que los verdaderos positivos están determinados con la no presencia de alertas al momento de la ejecución de las herramientas de simulación de ataques, como en este caso con Nmap y Hydra sobre ciertos

equipos de la red, para lo cual por cada herramienta que no generó alertas se le otorgó el valor de la unidad, desconociendo la cantidad de alertas que pudiese generar sobre el equipo atacado.

Las alertas más relevantes detectadas por Suricata son: ET DROP Spamhaus DROP Listed Traffic Inbound group xx, de la categoría Misc Attack.

En la **Figura 7-3**, se puede observar como Suricata por medio de Snorby presenta las alertas detectadas en el caso de simulación de ataques, las mismas que se encuentran clasificadas de acuerdo con su prioridad como son alta, media y baja.

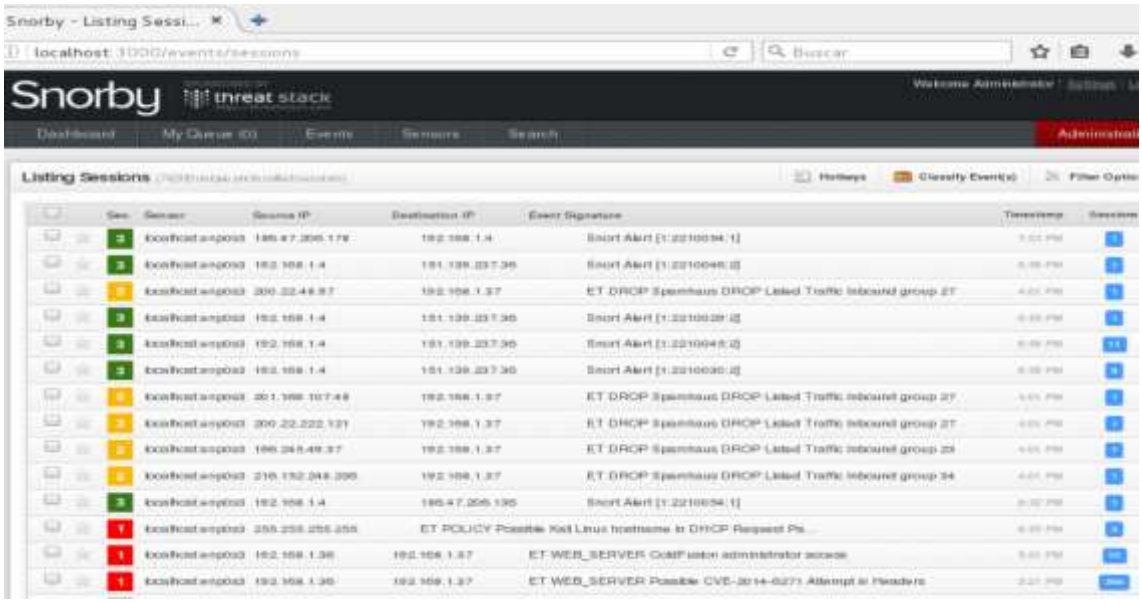

Figura 7-3: SNORBY front end de Suricata

Realizado por: Eduardo Arteaga

Una vez finalizado la ejecución de las herramientas descritas anteriormente, las cuales permitieron realizar la simulación de ataques hacía los servidores y host del escenario del Snort, se obtienen los siguientes resultados, ver **Tabla 13-3**:

Tabla 5-3: Descripción de intrusiones detectadas por Snort en el caso de simulación

Item	Alerta	Clasificación	VN	FP
1	[snort] http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	Unknown	0	66
2	[snort] http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE	Unknown	2	0
3	[snort] http_inspect: UNESCAPED SPACE IN HTTP URI	Unknown	0	8

Item	Alerta	Clasificación	VN	FP
4	[snort] http_inspect: POST W/O CONTENT-LENGTH OR CHUNKS	Unknown	1	0
5	[snort] http_inspect: UNKNOWN METHOD	Unknown	0	2
6	[snort] http_inspect: HTTP RESPONSE HAS UTF CHARSET WHICH FAILED TO NORMALIZE	Unknown	1	0
7	[snort] http_inspect: LONG HEADER	bad-unknown	0	1
8	[snort] http_inspect: NON-RFC DEFINED CHAR	bad-unknown	5	0
9	[snort] stream5: Reset outside window	bad-unknown	0	12
10	[snort] stream5: Bad segment, overlap adjusted size less than/equal 0	bad-unknown	3	0
11	[snort] stream5: TCP Small Segment Threshold Exceeded	bad-unknown	0	3
12	[snort] SERVER-WEBAPP JBoss JMX console access attempt	attempted-recon	2	0
13	[snort] SERVER-WEBAPP JBoss web console access attempt	attempted-recon	2	0
14	[snort] OS-OTHER Bash CGI environment variable injection attempt	attempted-recon	122	0
15	[snort] MALWARE-BACKDOOR phpMyAdmin server_sync.php backdoor access attempt	Trojan-activity	4	0
16	[snort] SQL generic sql with comments injection attempt - GET parameter	Web-application-attack	3	0
17	[snort] POLICY-OTHER Adobe ColdFusion component browser access attempt	Policy-violation	1	0
18	[snort] POLICY-OTHER Adobe ColdFusion admin interface access attempt	Policy-violation	5	0
19	[snort] sensitive_data: sensitive data global threshold exceeded	Sdf	33	0
20	Escaneo de Nikto		0	1
21	Escaneo de Nmap		0	2
22	Escaneo de Hydra		0	2
23	Escaneo de Hping3		0	1
		Σ	**184**	**92**

Número total de alertas detectadas: 276

Observaciones:

Falsos negativos (FN)

- Escaneo de Nikto -1
- Escaneo de Nmap -2
- Escaneo de Hydra -2
- Escaneo de Hping3 -1

Realizado por: Eduardo Arteaga

Según la **Tabla 13-3** de resultados se puede apreciar que Snort determinó 184 verdaderos negativos (VN), 92 falsos positivos (FP) y 6 falsos negativos (FN), cabe indicar que los verdaderos positivos esta determinados con la no presencia de alertas al momento de la ejecución de las herramientas de simulación de ataques, como en este caso con Nmap, Nikto, Hping3 y

Hydra sobre ciertos equipos de la red, para lo cual por cada herramienta que no generó alertas se le otorgó el valor de la unidad, desconociendo la cantidad de alertas que pudiese generar sobre el equipo atacado.

Las alertas más relevantes detectadas por Snort son: OS-OTHER Bash CGI environment variable injection attempt, de la categoría attempted-recon.

En la **Figura 8-3**, se puede observar como Snort por medio de BASE presenta las alertas detectadas en el caso de simulación de ataques, las mismas que se encuentran clasificadas de en una categoría determinada.

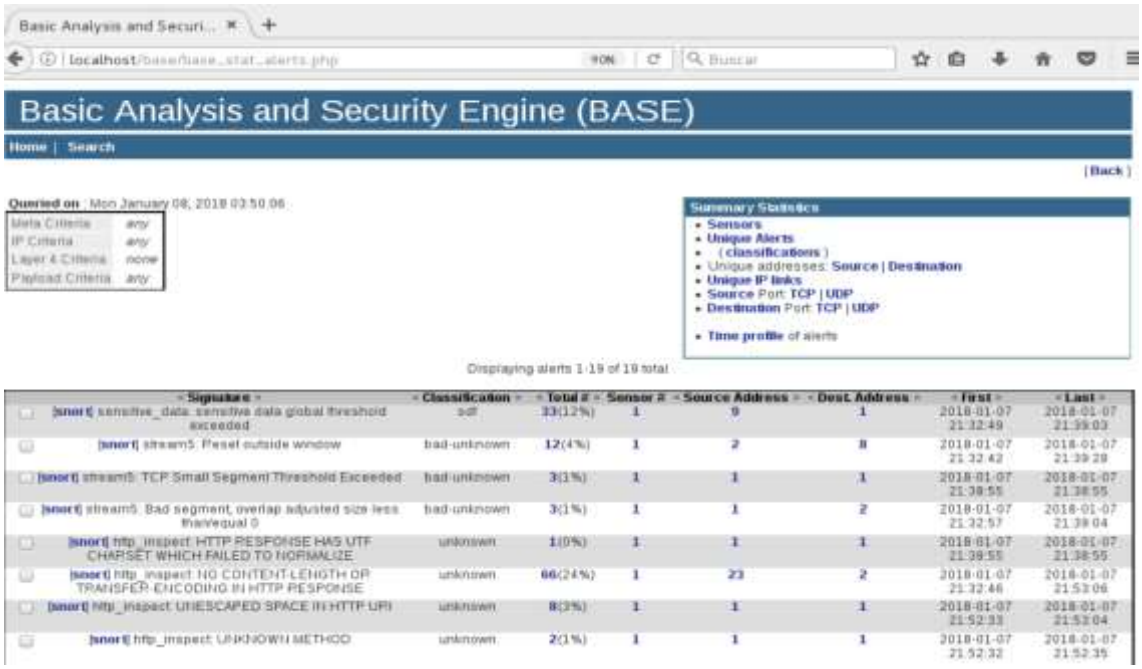

Figura 8-3: BASE Front End de Snort

Realizado por: Eduardo Arteaga

3.19.3 Caso 3: Aplicativo

Con la finalidad de cumplir con los objetivos propuesto en el presente proyecto de investigación, y analizar y procesar los datos de los indicadores de la variable dependiente, se procedió con la implementación de un escenario de pruebas, para analizar, escanear y estudiar el tráfico generado en la red de servidores de la Secretaría de Hidrocarburos (ver Anexo J), de acuerdo con lo definido en el benchmark, los IDSes que se usaron en este escenario fueron: Snort y Suricata.

Una vez procesado los paquetes de información por Snort, se obtienen un total de 2652 alertas las mismas que al ser correlacionadas, se obtienen 22 verdaderos negativos y 2630 falsos positivos, el detalle al respecto se puede apreciar en la **Tabla 14-3**.

Tabla 14-3: Descripción de alertas detectadas mediante Snort caso aplicativo

Alerta	Clasificación	VN	FP	Total
Reset outside window (puertos 88,389)	Potentially Bad Traffic	0	1678	2652
(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	Unknown Traffic	0	455	
Consecutive TCP small segments exceeding threshold	Potentially Bad Traffic	0	454	
SDF Combination Alert	Senstive Data	0	8	
(http_inspect) LONG HEADER	Potentially Bad Traffic	0	6	
TCP Timestamp is missing	Potentially Bad Traffic	22	0	
Bad segment, adjusted size <= 0		0	29	
Σ		**22**	**2630**	

Realizado por: Eduardo Arteaga

De acuerdo con los resultados obtenidos existe un valor muy alto de falsos positivos con respecto a las alertas Reset outside window, previo análisis de la alerta se determina que esta corresponde a la ejecución de las ventanas del navegador web. Para evitar alertas irrelevantes por parte de Snort se puede configurar una regla para descartar dichas alertas y evitar la saturación del IDS.

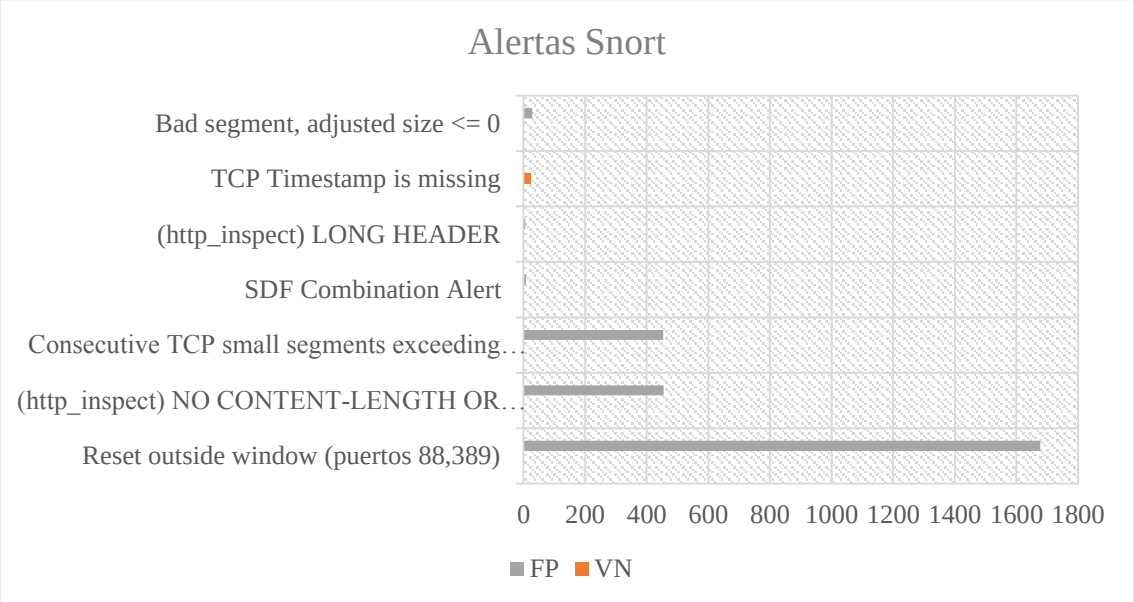

Figura 9-3: Alertas detectadas mediante de Snort

Realizado por: Eduardo Arteaga

En la **Figura 9-3**, se puede apreciar de manera representativa la cantidad de verdaderos negativos y falsos positivos detectados por Snort, evidenciándose que existe número muy bajo de verdaderos negativos lo que involucra que la red de la Secretaría de Hidrocarburos al momento del análisis de su tráfico de red se encuentra generando tráfico malicioso, lo que podría estar ocasionando ciertos intentos de robo de información.

Una vez procesado los paquetes de información del caso aplicativo por Suricata, se obtienen un total de 38525 alertas las mismas que al ser correlacionadas, se obtienen 27 verdaderos negativos y 38498 los falsos positivos, el detalle al respecto se puede apreciar en la **Tabla 15-3**.

De acuerdo con los resultados existe un valor muy alto de falso positivos con respecto a las alertas TCPv4 invalid checksum, previo análisis de la alerta se determina que esta corresponde a mal formaciones del paquete TCP debido que dichos paquetes son generados en un ambiente virtual y se produce un error de comprobación de no válida del TCP. Para evitar alertas irrelevantes por parte de Suricata se puede configurar una regla para descartar dichas alertas y evitar la saturación del IDS.

Tabla 6-3: Descripción de alertas detectadas mediante Suricata

Alerta	Clasificación	VN	FP	Total
ET POLICY Cleartext Wordpress Login	Potential Corporate Privacy Violation	8	0	38525
TLS invalid record version	Generic Protocol Command Decode	9	0	
TLS invalid record /traffic	Generic Protocol Command Decode	9	0	
TCPv4 invalid checksum		0	37448	
STREAM CLOSEWAIT invalid ACK	Generic Protocol Command Decode	0	14	
STREAM SHUTDOWN RST invalid ack	Generic Protocol Command Decode	1	0	
UDPv4 invalid checksum		0	33	
STREAM ESTABLISHED packet out of window	Generic Protocol Command Decode	0	86	
STREAM ESTABLISHED invalid ack	Generic Protocol Command Decode	0	448	
STREAM Packet with invalid ack	Generic Protocol Command Decode	0	464	
STREAM 3way handshake with ack in wrong dir	Generic Protocol Command Decode	0	3	
STREAM 3way handshake wrong seq wrong ack	Generic Protocol Command Decode	0	2	
	Σ	**27**	**38498**	

Realizado por: Eduardo Arteaga

En la **Figura 10-3**, se puede apreciar de manera representativa la cantidad de verdaderos negativos y falsos positivos detectados por Suricata, evidenciándose que existe número muy bajo de verdaderos negativos lo que involucra que la red de la Secretaría de Hidrocarburos al momento del análisis de su tráfico de red se encuentra generando tráfico malicioso, lo que podría estar ocasionando ciertos intentos de robo de información.

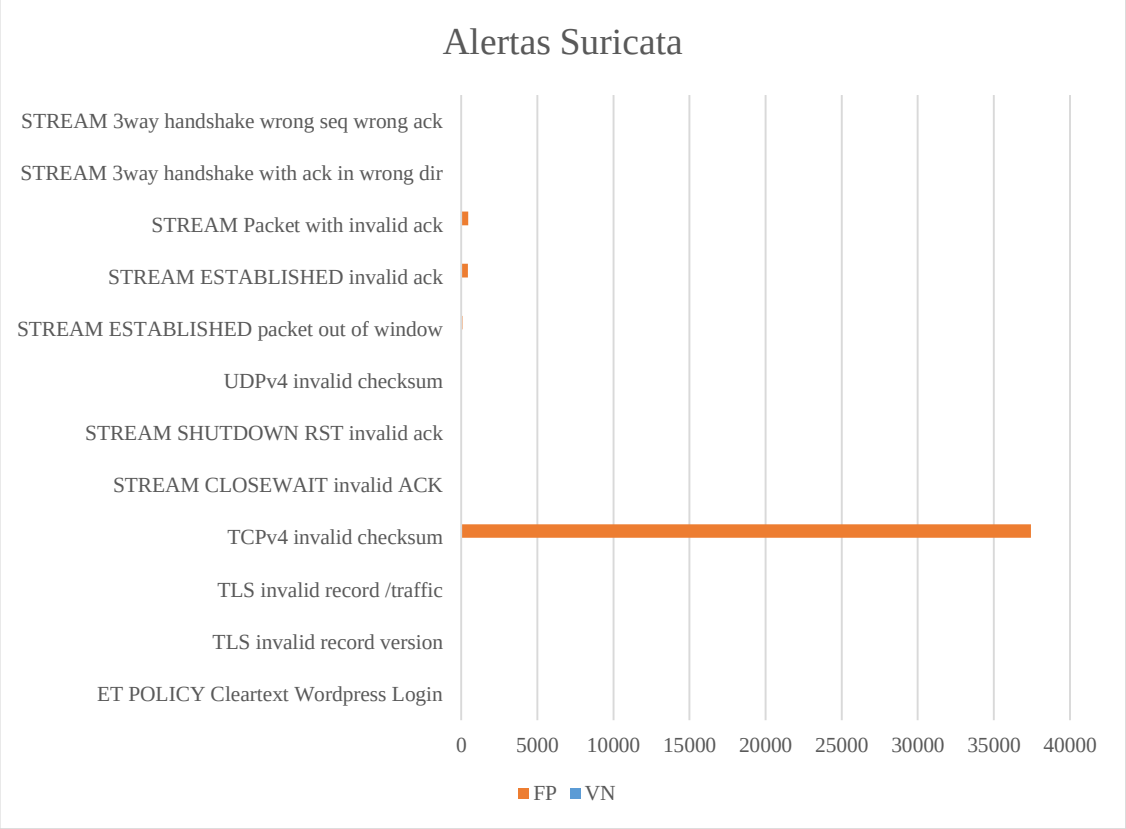

Figura 10-3: Alertas detectadas mediante Suricata

Realizado por: Eduardo Arteaga

3.20 Procesamiento y análisis de los indicadores de la variable independiente

La variable independiente está representada por la implementación un sistema de detección de intrusos basado en la red de plataforma Open Source utilizando la técnica de detección de anomalías. En este apartado se realiza un análisis de las herramientas NIDS Open Source en función de cada indicador, el cual a su vez contiene diferentes parámetros de estudio para su evaluación.

Para el presente estudio se utilizaron las versiones 2.9.9 de Snort, 2.9.10 de Suricata, y la **Tabla 7-3** de ponderación de evaluación para estimar valores cuali-cuantitativos y porcentuales.

3.20.1 Indicador 1: Funciones

Con el fin de proceder con la evaluación del indicador de funciones, se consideró los valores definidos en la siguiente **Tabla 16-3**, fundamentada en el método de evaluaciones sumarias (escala de Likert), en la que se crea una escala con valores graduales de 0 a 5 puntos, de acuerdo con el cumplimiento de los criterios expuestos. Además, cada valor está representado por su respectiva valoración cualitativa y porcentual.

Tabla 16-3: Ponderación de evaluación del indicador 1

Calificación cualitativa	Valor asignado	Porcentaje	Descripción
No existe (N/A)	0	0%	No aplicable para la asignación de un valor cuantitativo o no posee ese indicador o característica evaluada.
Deficiente	1	20%	Esta cualidad cuyo equivalente cuantitativo es igual a 1, será otorgada a las herramientas que no cumplan o cumplan de manera deficiente con el objetivo del indicador.
Regular	2	40%	Esta cualidad cuyo equivalente cuantitativo es 2, será otorgado a la herramienta que cumpla de manera insuficiente el indicador.
Bueno	3	60%	Esta cualidad cuyo equivalente cuantitativo es 3, será otorgado a la herramienta que cumplan parcialmente el indicador.
Muy bueno	4	80%	Esta cualidad cuyo equivalente cuantitativo es 4, será otorgado a la herramienta que cumpla casi en su totalidad el indicador.
Satisfactorio	5	100%	Esta cualidad cuyo equivalente cuantitativo es 5, será otorgado a la herramienta que cumpla en su totalidad el indicador.

Realizado por: Eduardo Arteaga

De acuerdo con el análisis expuesto en el Anexo K, acerca de la comparación de funciones que inciden directamente con la detección de intrusiones de los IDSes, se obtiene los siguientes resultados de estudio, en la **Tabla 17-3** se muestra de manera resumida los indicadores y sus respectivos parámetros evaluados.

Tabla 17-3: Resumen de la evaluación del indicador 1

Caracteríticas	Evaluación	
	Snort	Suricata
Altamente escalable	4	5
Protocolos de red	4	5
Gestiona distintos módulos salida	*3*	4

Soporte IPV6	5	5
Aplicaciones extendidas	5	5
Subherramientas para respuestas activas	5	4
Reputación IP	5	5
GeoIP	5	5
Promedio	4,5	4.75

Realizado por: Eduardo Arteaga

3.20.2 Indicador 2: Desempeño

Para la evaluación del indicador 2, se considera el desempeño de los sistemas de detección de intrusos del caso de estudio en condiciones estresantes o situaciones en el que se comprometa de manera lícita o ilícita los recursos de red posiblemente por ataques DOS no detectados. Cabe indicar que los sistemas deben ser capaces de procesar el tráfico de manera normal, rápida y eficiente.

Básicamente el correcto desempeño de los sistemas NIDS Open Source, esta relaciona directamente con las características del sistema hardware/software en el que se encuentre implementado el entorno a proteger.

Para analizar el desempeño de los sistemas de detección de intrusos, se empleó los archivos (.tcpdump) de DARPA99, los cuales contienen 2 semanas de tráfico normal y una semana de tráfico anormal. Estos archivos fueron analizados tanto por Snort como por Suricata considerando las mismas características para las máquinas virtuales.

En la **Tabla 18-3**, se muestra el número de paquetes perdidos y los tiempos de respuesta obtenidos al momento de realizar el análisis de 30 archivos .tcpdump del conjunto de datos DARPA 99, que se usaron para el caso de entrenamiento.

Tabla 18-3: Evaluación del desempeño de los sistemas de detección de intrusos

N°	Paquetes	Suricata		Snort	
		Paquetes perdidos	Tiempo de respuesta (s)	Paquetes perdidos	Tiempo de respuesta (s)
1	1492331	0%	40	0%	23
2	1237119	0%	24	0%	36
3	1726319	0%	10	63%	38

N°	Paquetes	Suricata		Snort	
		Paquetes perdidos	Tiempo de respuesta (s)	Paquetes perdidos	Tiempo de respuesta (s)
4	1947815	0%	32	0%	64
5	1483419	0%	16	74%	29
6	1362869	0%	10	53%	33
7	1157328	0%	12	0%	36
8	1616710	0%	12	0%	38
9	1807060	0%	8	38%	62
10	1349635	0%	10	0%	28
11	2106744	0%	32	0%	41
12	1831648	0%	29	0%	40
13	1849753	0%	15	0%	41
14	1359136	0%	20	0%	15
15	1635425	0%	15	0%	41
16	1542614	0%	14	0%	39
17	1374431	0%	14	0%	37
18	1760879	0%	17	0%	65
19	1096660	0%	8	0%	12
20	1536736	0%	17	0%	63
21	1753377	0%	23	0%	35
22	1585120	0%	18	0%	45
23	1011149	0%	6	0%	12
24	1563069	0%	14	0%	36
25	1362422	0%	9	0%	30
26	1337777	0%	12	0%	33
27	1454035	0%	14	0%	41
28	888139	0%	6	0%	10
29	1412645	0%	13	0%	35
30	1252412	0%	14	0%	28
Σ	**44894776**	**0%**	**484**	**7.6%**	**1086**

Realizado por: Eduardo Arteaga

Una vez obtenidos los resultados se evidencia que: Respecto al porcentaje de paquetes perdidos Suricata tiene un 0%, con respecto a Snort que tiene un 7.6 %; referente a los tiempos de respuesta se evidencia que Suricata realiza el análisis de 44894776 paquetes en un tiempo de 484 segundos y por otra parte la misma cantidad de paquetes a Snort analizar lo toma un tiempo de 1086 segundos, por lo que se concluye que en la evaluación del indicador 2; Desempeño, Suricata es mayormente eficiente respecto a Snort, debido que no tiene paquetes perdidos y sus tiempo de respuesta son inferiores.

3.20.3 Indicador 3: Seguridad

Este indicador se refiere a la posibilidad de detección correcta (sensibilidad) y por el contrario la posibilidad de no detectar intrusiones (FN). De acuerdo con el problema investigado se considera importante realizar la evaluación de este indicador en los casos de entrenamiento, simulación en el ambiente de pruebas (ataques DOS y monitorización) y aplicativo.

Sensibilidad

El indicador de sensibilidad permitir determinar la capacidad de NIDS en clasificar los eventos de entrada correctamente como normal o intrusiva.

El valor de sensibilidad está determinado con la posibilidad de predecir casos positivos es decir intrusiones catalogadas como verdaderos negativos (VN) y se determina por la división entre el número total de VN y la sumatoria de VN + FP (falsos positivos). El valor será un valor menor a 1, que multiplicado por 100 es el valor porcentual. Mientras el valor sea más cercano a 1, la herramienta se aproxima a un valor sensible de 100%:

$$S = \frac{\Sigma\, VN}{\Sigma(\, VN+FP)} * 100\ \%$$

Nomenclatura:

S *= Sensibilidad*
VP *= Verdaderos negativos*
FP *= Falsos Positivos*

Tabla 19-3: Cálculo de la sensibilidad de los IDSes

Indicador	Snort	Suricata
Caso entrenamiento	*9.42%*	*0.07%*
Caso Simulación	*66.67%*	*99.99%*
Caso aplicativo	*0.82%*	*0.07%*
Valor promedio	**25.63%**	**33.37%**

Realizado por: Eduardo Arteaga

Para obtener el valor de sensibilidad de los IDSes, se utilizaron los datos de los diferentes casos de estudios como son: entrenamiento, simulación y aplicativo, los resultados obtenidos, se muestran en la **Tabla 19-3.**

De acuerdo con los resultados obtenidos se puede concluir que: Suricata tiene una sensibilidad de predecir casos positivos de un valor promedio de 33.37 % y por otro lado Snort tiene un valor promedio de 25.63%, estos valores son obtenidos de acuerdos a los casos de estudio del proyecto de investigación.

3.21 Aporte del IDS para mejorar la seguridad de la información

Al momento de realizar el análisis del tráfico de la red de datos de la Secretaría de Hidrocarburos con el IDS Suricata, se evidenciaron alertas con la siguiente descripción "***ET POLICY Cleartext Wordpress Login"*** de prioridad alta. Una vez detectada la alerta se procedió con su análisis respectivo encontrándose que se trataba de una falla se seguridad de en el sistema de gestión de contenidos que consistía en que al momento de realizar un login de acceso el nombre de usuario y la contraseña se envían en texto claro a través de Internet lo que estaba poniendo en riesgo que esos datos sean robados por personas que pudiesen dañar la integridad de la información o detener el servicio de la página web.

Como medida de prevención y afín de mitigar este inconveniente de seguridad se determinaron algunos procedimientos a implementarse en el sistema de gestión de contenidos como es la implementación del protocolo de Https.

CAPÍTULO IV

4. RESULTADOS Y DISCUSIONES

4.1 Análisis de los resultados obtenidos

Con la finalidad de evaluar las variables independiente y dependiente, se procede con la evaluación de los resultados de los indicadores obtenidos en los diferentes casos como son: entrenamiento, simulación y aplicativo.

INDICADOR 1: FUNCIONES

La evaluación de los sistemas de detección de intrusos de acuerdo con sus funcionalidades se representa en la **Figura 1-4.**

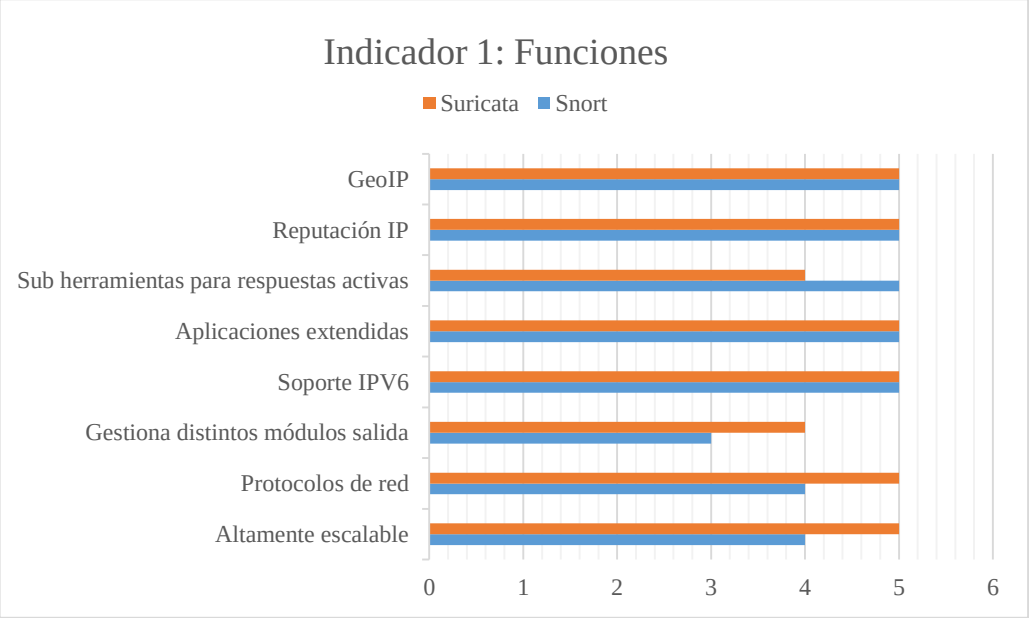

Figura 1-4: Valoración de las funcionalidades de los sistemas de detección de intrusos

Realizado por: Eduardo Arteaga

El promedio del resultado de la evaluación del indicador 1 es de: 4, 50 puntos para Snort y 4,75 para Suricata. Cabe indicar que esta evaluación está sustentada principalmente por búsqueda de información en sitios web oficiales de las herramientas NIDS Open Source.

Finalmente, de acuerdo con los resultados porcentuales obtenidos en los 8 criterios del indicador 1 funciones de los IDSes, se estipula que: Suricata es la solución más prometedora con un porcentaje total de 95 % frente a 90 % de Snort. La diferencia de los resultados obtenidos entre Suricata y Snort es sutil.

INDICADOR 2: DESEMPEÑO

Para la evaluación del desempeño de los sistemas de detección de intrusos, se utilizó los paquetes de entrada y salida de DARPA 99, que corresponden a 2 semanas de entrenamiento con paquetes libre de amenazas y 1 una semana a paquetes que contiene amenazas. Con la finalidad de evaluar este indicador se hace uso de los datos de la etapa de entrenamiento, para lo cual se utilizó 44894776 (100%) paquetes divididos en 30 archivos de extensión .pcap.

Una vez analizados los archivos correspondientes por los sistemas de detección intrusos Snort y Suricata respectivamente, se puede evidenciar, que: Suricata para analizar los 30 archivos de acuerdo con el detalle mostrado en la **Tabla 1-4**, toma un tiempo promedio de 16,13 segundos, en cambio Snort lo realiza en un tiempo promedio de 36,20 segundos, cabe indicar que Suricata posee cierta ventaja respecto a Snort, debido a que es multihilo y puede equilibrar sus procesos.

Respecto a la cantidad de alertas emitidas al momento del entrenamiento tanto con tráfico normal como anormal, se obtienen los siguientes datos al respecto, Suricata muestra un total de 593786, y por otra parte Snort emite 225924, de esta manera se evidencia que Suricata emite mayor cantidad de alertas al respecto.

Tabla 1-4: Caso de entrenamiento de Suricata y Snort

N°	Paquetes	Suricata			Snort		
		Paquetes perdidos	Tiempo de respuesta (s)	Total de alertas	Paquetes perdidos	Tiempo de respuesta (s)	Total de alertas
1	1492331	0%	40		0%	23	
2	1237119	0%	24		0%	36	
3	1726319	0%	10		63%	38	
4	1947815	0%	32		0%	64	
5	1483419	0%	16	593786	74%	29	225924
6	1362869	0%	10		53%	33	
7	1157328	0%	12		0%	36	
8	1616710	0%	12		0%	38	
9	1807060	0%	8		38%	62	

N°	Paquetes	Suricata			Snort		
		Paquetes perdidos	Tiempo de respuesta (s)	Total de alertas	Paquetes perdidos	Tiempo de respuesta (s)	Total de alertas
10	1349635	0%	10		0%	28	
11	2106744	0%	32		0%	41	
12	1831648	0%	29		0%	40	
13	1849753	0%	15		0%	41	
14	1359136	0%	20		0%	15	
15	1635425	0%	15		0%	41	
16	1542614	0%	14		0%	39	
17	1374431	0%	14		0%	37	
18	1760879	0%	17		0%	65	
19	1096660	0%	8		0%	12	
20	1536736	0%	17		0%	63	
21	1753377	0%	23		0%	35	
22	1585120	0%	18		0%	45	
23	1011149	0%	6		0%	12	
24	1563069	0%	14		0%	36	
25	1362422	0%	9		0%	30	
26	1337777	0%	12		0%	33	
27	1454035	0%	14		0%	41	
28	888139	0%	6		0%	10	
29	1412645	0%	13		0%	35	
30	1252412	0%	14		0%	28	
Σ	**44894776**	**0%**	**484**	**593786**	**228%**	**1086**	**225924**

Realizado por: Eduardo Arteaga

También se puede constatar, que de acuerdo con la **Tabla 1-4**, Suricata no tiene pérdidas de paquetes y por otro lado Snort tiene una pérdida promedio de paquetes equivalente al 1,32%. Por lo que se puede concluir que Suricata tiene un mejor desempeño respecto a Snort ya que sus tiempos son cortos a relación de Snort y no genera pérdidas de paquetes.

Tabla 2-4: Resumen de la evaluación del indicador desempeño

Criterio	Suricata		Snort	
	Cuantitativo	Cualitativo	Cuantitativo	Cualitativo
Tiempo de respuesta	5	Satisfactorio	3	Bueno
Paquetes perdidos	5	Satisfactorio	3	Bueno

Realizado por: Eduardo Arteaga

En la **Tabla 2-4,** se muestra el resumen de la valoración del indicador de desempeño de acuerdo con los resultados obtenidos en el caso de entrenamiento de Suricata y Snort.

INDICADOR 3: SEGURIDAD

Con la finalidad de evaluar este indicador, se procede a cuantificar las alertas detectados por los sistemas de detección de intrusos de los casos de estudio, en los escenarios propuestos para la evaluación de estos, en la **Tabla 3-4** se muestra los resultados obtenidos:

Tabla 3-4: Cantidad de alertas emitidas por los IDSes

Caso	Snort	Suricata
Entrenamiento	232363	235970
Simulación de ataques	276	75050
Aplicativo	2652	38525
Totales	235291	349545

Realizado por: Eduardo Arteaga

En la **Figura 2-4**, se muestra de forma gráfica la representación de las alertas emitidas por Snort y Suricata, en los diferentes casos de evaluación propuestos en el proyecto de investigación.

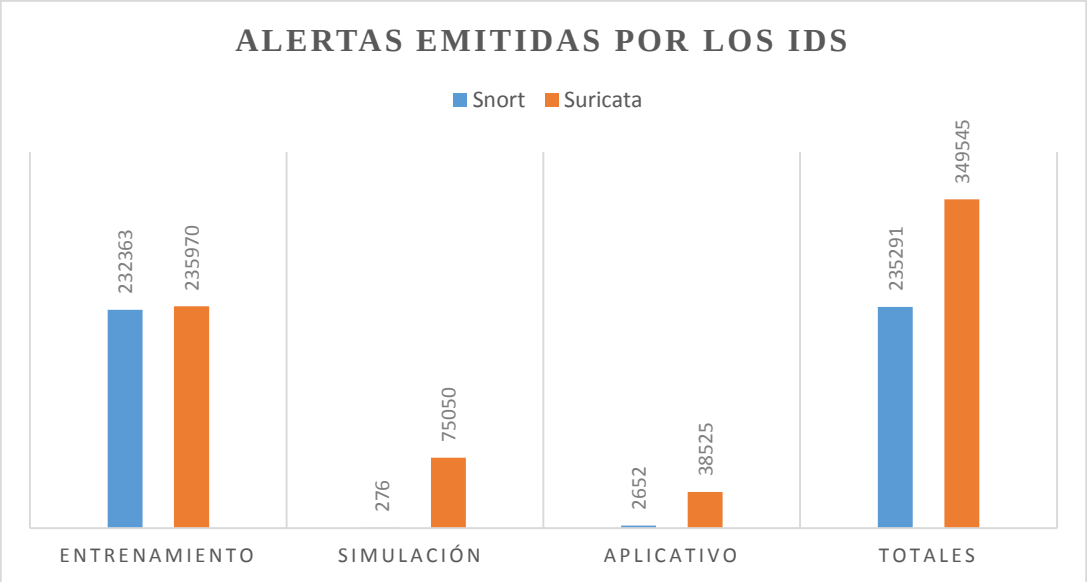

Figura 2-4: Distribución del total de las alertas emitidas por los IDSes

Realizado por: Eduardo Arteaga

Para la correlación de las alertas emitidas por Snort y Suricata, se utilizaron los datos estadísticos de las etapas de entrenamiento, simulación y aplicativo; las alertas se clasificaron como verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN), los resultados obtenidos se muestran en la **Tabla 4-4.**

Tabla 4-1: Correlación de las alertas emitidas por los sistemas de detección de intrusos

Caso	Snort			Suricata		
	VN	FP	FN	VN	FP	FN
Entrenamiento	20021	212342	0	166	235804	0
Simulación de ataques	184	92	6	75046	4	4
Aplicativo	22	2630	0	27	38498	0
Totales	20227	215064	6	75239	274306	4

Realizado por: Eduardo Arteaga

En la **Figura 3-4**, se muestra de forma gráfica la representación de la correlación de las alertas como VF, FP y VN emitidas en por Snort y Suricata, en los diferentes casos de evaluación propuestos en el proyecto de investigación.

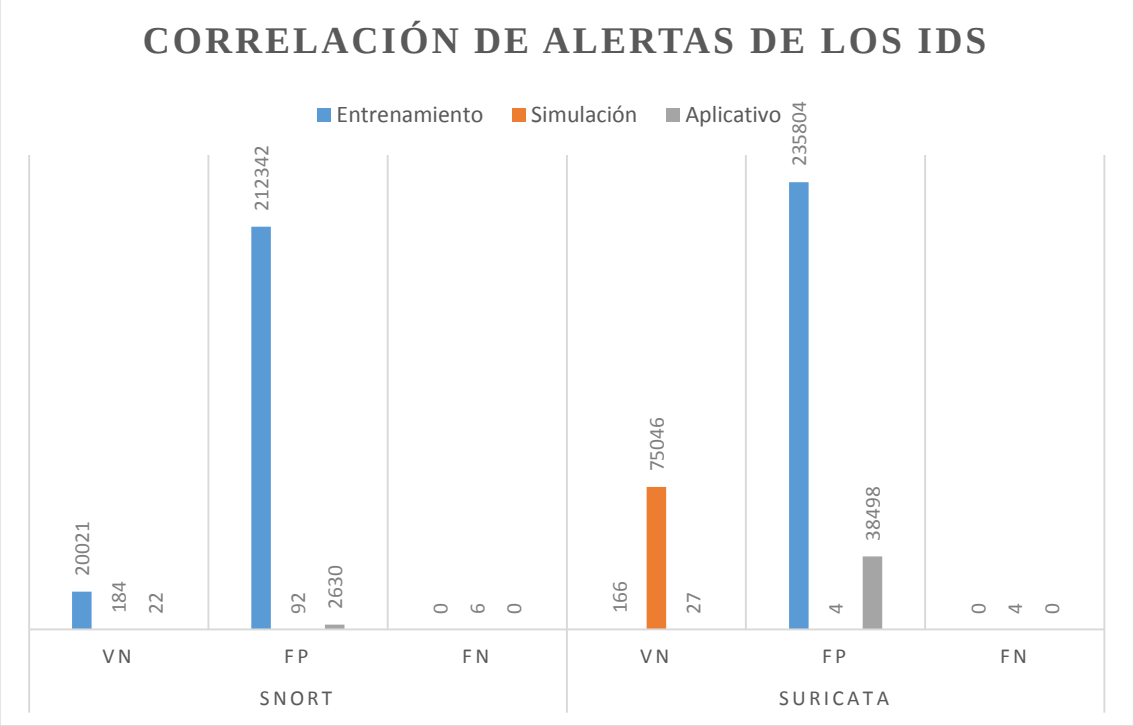

Figura 3-4: Correlación de las alertas emitidas por los IDSes

Realizado por: Eduardo Arteaga

Cabe indicar que, los falsos positivos (FP), se los consideraron en el caso simulación, debido a que los sistemas de detección de intrusos no emitieron ninguna alerta al momento de generar ataques a través de Hydra, Nikto, Hping3 y otros, razón por la cual, por cada intrusión no detecta se otorgó un valor de la unidad, considerando que se desconoce la cantidad de alertas que pudiese generar al respecto, tanto Snort como Suricata.

4.2 Comprobación de la hipótesis

Una vez que se analizaron los indicadores de evaluación de los sistemas de detección de intrusos de plataforma Open Source, utilizando la técnica de detección de anomalías, se observa que Suricata tienes dos indicadores con calificación de satisfactorio y uno regular, por otra parte Snort tienes dos indicadores con calificación de bueno y uno deficiente; Determinándose que Suricata tiene mejores prestaciones para mejorar la seguridad de la red respecto a Snort, de acuerdo a los resultados de los indicadores de funciones, desempeño y seguridad en los diferentes casos de estudio, para más detalle se puede ver en la **Tabla 5-4** el resumen de la evaluación de los IDSes.

Tabla 5-4: Resumen de los indicadores de evaluación de los IDSes

Indicador	Snort		Suricata	
	Cuantitativo	**Cualitativo**	**Cuantitativo**	**Cualitatitvo**
Funciones	4,5	Muy bueno	4,75	Satisfactorio
Desempeño	3	Bueno	5	Satisfactorio
Seguridad	1.28	Deficiente	1,67	Regular

Realizado por: Eduardo Arteaga

La verificación de la hipótesis se realiza con la prueba de Z, para lo cual se hizo uso de la cantidad de alertas y verdaderos negativos detectados por los IDSes Snort y Suricata, obtenidos en el caso aplicativo, para la evaluación inductiva de la verificación de la Hipótesis de la investigación se utiliza la prueba de Z, con un nivel de significancia del 5%.

Para la verificación de hipótesis es necesario el planteamiento de hipótesis nula que desaprueba el análisis de proceso de control y de hipótesis alternativa.

4.3 Planteamiento de la hipótesis

- **Hipótesis Alterna Ha**

La implementación de un sistema de detección de intrusos basados en la red de plataforma Open Source utilizando la técnica de detección de anomalías mejorará la seguridad de la información que viaja por la red.

- **Hipótesis Nula Ho**

La implementación de un sistema de detección de intrusos basados en la red de plataforma Open Source utilizando la técnica de detección de anomalías no mejorará la seguridad de la información que viaja por la red.

4.4 Prueba de Z

La fórmula de prueba de Z utilizada para la comprobación de la hipótesis es la de comparación de dos proporciones:

$$Z = \frac{P1 - P2}{\sqrt{P(1-P)(\frac{1}{n1} + \frac{1}{n2})}}$$

Z *= Valor de Z*
n1 *= Tamaño del primer grupo*
n2 *= Tamaño del segundo grupo*
P1 *= Proporción del primer grupo*
P2 *= Proporción del segundo grupo*
P *= Proporción total*

4.5 Datos utilizados en la comprobación de la hipótesis

Una vez cuantificados los valores de los indicadores de funciones, desempeño y seguridad tanto de Snort y Suricata, se determina que Suricata ofrece mayores prestaciones para garantizar la seguridad de una red de datos, considerando los resultados expuesto anteriormente y con la finalidad de demostrar la hipótesis de la investigación, se harán uso de los datos obtenidos en el caso aplicativo como son:

1. Verdaderos negativos (VN).
2. Cantidad de alertas detectadas por los IDSes.

4.6 Valores determinados para el proceso de la comprobación de la hipótesis

En el caso aplicativo, se realizó el análisis de los paquetes de información de la red de servidores de la Secretaría de Hidrocarburos, para lo cual se obtuvieron una cantidad de alertas, las mismas que fueron correlacionadas como verdaderos negativos y falsos positivos, con la finalidad de proceder con la comprobación de la hipótesis utilizando la prueba Z de proporciones, se procede a utilizar una la muestra de los datos de investigación, para lo cual se utilizó la cantidad de los verdaderos negativos y el total de alertas emitidas por los IDSes, para poder determinar si efectivamente la cantidad de verdaderos negativos es menor o similar a la cantidad de falsos positivos y de esta manera poder determinar al mejor IDS que permita mejorar la seguridad de la información que viaja por la red, en la **Tabla 6-4**, se puede observar el conjunto de datos a utilizar.

Tabla 6-4: Conjunto de valores

Conjunto de datos		
Eventos	**Snort**	**Suricata**
Total de alertas	*47*	*156*
VN	*7*	*9*

Realizado por: Eduardo Arteaga

4.7 Cálculo de Z

Para el cálculo de la prueba de Z se hará uso del conjunto de valores de la **Tabla 6-4**, con una significancia del 5%.

$$Z = \frac{0.149 - 0.058}{\sqrt{0.788(1 - 0.788)(\frac{1}{47} + \frac{1}{156})}}$$

$$Z = \frac{0.091}{0.0020}$$

$$Z = 45.5$$

Calculo del valor crítico de Z

De acuerdo con el planteamiento, se ha determinado que se trata de una prueba de una cola.

$Z(1 - \alpha) = Z(1 - 0.05)$

$= Z(0.95)$

$= 1.65$ // De acuerdo con la tabla de Z

De acuerdo con el valor obtenido de Z, como nivel de confianza $\alpha = 0,05$ se obtiene: $Z = 45,5$, y el valor de crítico de $Z = 1,65$; se concluye que el valor de Z es mayor que el valor crítico de Z, por lo tanto, se rechaza la hipótesis nula, y se acepta la hipótesis de la investigación, considerando que mientras mayor cantidad de VN emitidos por un IDS, este será más eficiente al momento de garantizar la seguridad de la información que viaja por la red, cabe indicar que la cantidad de FP se los puede descartar una vez analizados y determinado su naturaleza sin atentar la seguridad de la información.

En la **Figura 4-4**, se puede observar la representación gráfica de la prueba de z en la campana de Gauss, en el que puede observar la zona de aceptación de la hipótesis.

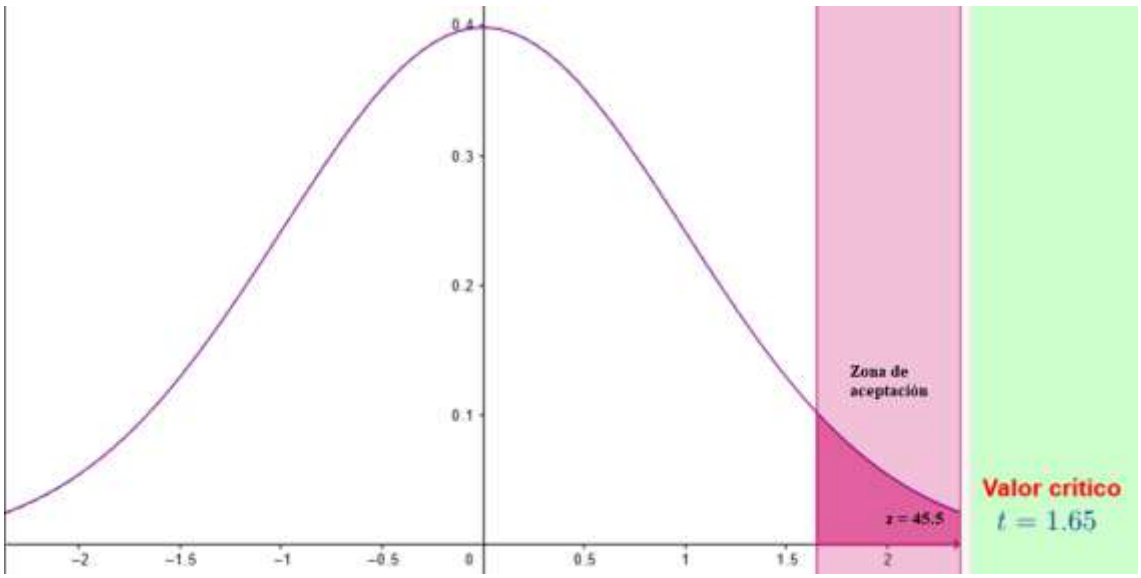

Figura 4-4: Gráfica de Z

Realizado por: Eduardo Arteaga

CAPÍTULO V

5. PROPUESTA

5.1 Título de la propuesta

Manual de buenas prácticas de los sistemas de detección de intrusos de plataformas Open Source

5.2 Introducción

En la actualidad las instituciones tanto privadas como públicas carecen parcialmente o tiene deficientes mecanismos de seguridad de la información que hagan frente al aumento del número de ataques que se producen en Internet, como una medida para mitigar los ataques informáticos es utilizar un sistema de detección de intrusos que alerte a los administradores de los servidores y red, cuando un intruso o eventos anormales se presenten en la red, actualmente la mayoría de instituciones del Ecuador no tienen instalado un sistema de detección de intrusos, debido a que no existen evaluaciones de funcionalidad o cuadro comparativos de ventajas y desventajas de los sistemas de detección de intrusos de plataformas Open Source, que les permita elegir de manera eficiente y de forma segura un sistema acorde a la infraestructura tecnológica con la que cuentan, en otros casos el desconocimiento de la funcionalidad de los sistemas de detección de intrusos ocasiona que los administradores de servidores o de red, lo instalen a modo de prueba para realizar los testeos correspondientes a fin de probar su funcionamiento, lo que ocasiona que se pierda tiempo y recursos, debido a estos inconvenientes no lo utilizan y en otros casos prefieren utilizar otras alternativas de seguridad o software propietario.

Con el manual de buenas prácticas de los sistemas de detección de intrusos de plataforma de software libre, se describe los pasos necesarios para su implementación y los resultados de la evaluación de sus funcionalidades, con la finalidad de reducir los tiempos al momento de elegir un IDS que se adapte a las necesidades de la institución y su proceso de instalación.

5.3 Objetivo

Elaborar un manual de buenas prácticas de los sistemas de detección de intrusos de plataformas Open Source para aportar en la implementación de IDSes de forma óptima.

5.4 Fundamentación de la propuesta

EL estudio de los sistemas de detección de intrusos basados en la red de plataforma Open Source, utilizando la técnica de anomalías, promoverá las bases y requerimientos necesarios para la implementación de un NIDS de forma adecuada y eficiente y además fomentará la investigación de temas relacionados.

5.5 Descripción de la propuesta

Los procedimientos y sistematización de los pasos para una implementación adecuada y eficiente de Suricata, se encuentran plasmados a continuación.

5.5.1 Que IDS elegir

En este de estudio de investigación se ha realizado una evaluación de IDSes de plataforma Open Source, con la finalidad de dar a conocer a los administradores de servidores y de redes una herramienta que permita mejorar la seguridad de la información que viaja por la red.

Al momento de implementar un IDS, de acuerdo con el análisis realizado en el presente estudio de investigación, se concluyó que el mejor sistema de detección de intrusos de plataforma Open Source es Suricata. Para la efectiva detección de este se recomienda realizar la instalación de Suricata, considerando los pasos descritos en presente manual en el que se incluye donde colocar, instalación y configuración, ya que son fundamentales para un funcionamiento correcto y efectivo del sistema de detección de intrusos, al obviar ciertos procedimientos se puede poner en riesgo la seguridad de la red, ya que el IDS puede estar haciendo un mal trabajo al momento de realizar el escaneo de la red.

5.5.2 Donde colocar el IDS

En el presente proyecto de investigación, se había indicado que un IDS puede ser colocado en varios puntos de la red como son:

A. Delante de contrafuegos externo
B. Detrás de cortafuegos externo
C. Redes principales
D. Subredes de valor crítico
E. Máquinas

Se recomienda instalar el IDS en el punto B, ya que ofrece varias ventajas al respecto como son:

- Se monitorizan intrusiones que logran atravesar el firewall principal.
- Detectar ataques a servidores que ofrecen servicios públicos.
- En caso de no detectar ataques con éxito, puede reconocer algunas consecuencias de estos, como intentos de conexiones salientes, realizadas desde servidores comprometidos.
- Identificación de los ataques y escaneos más comunes permite mejorar la configuración de los cortafuegos.
- Monitorizar la red LAN ya que muchas amenazas son provocadas por usuarios internos al momento de ingresar a páginas web maliciosas o al insertar medios extraíbles contaminados en sus equipos informáticos.

En la **Figura 1-5**, se puede observar el esquema de red, y donde debe ubicarse Suricata.

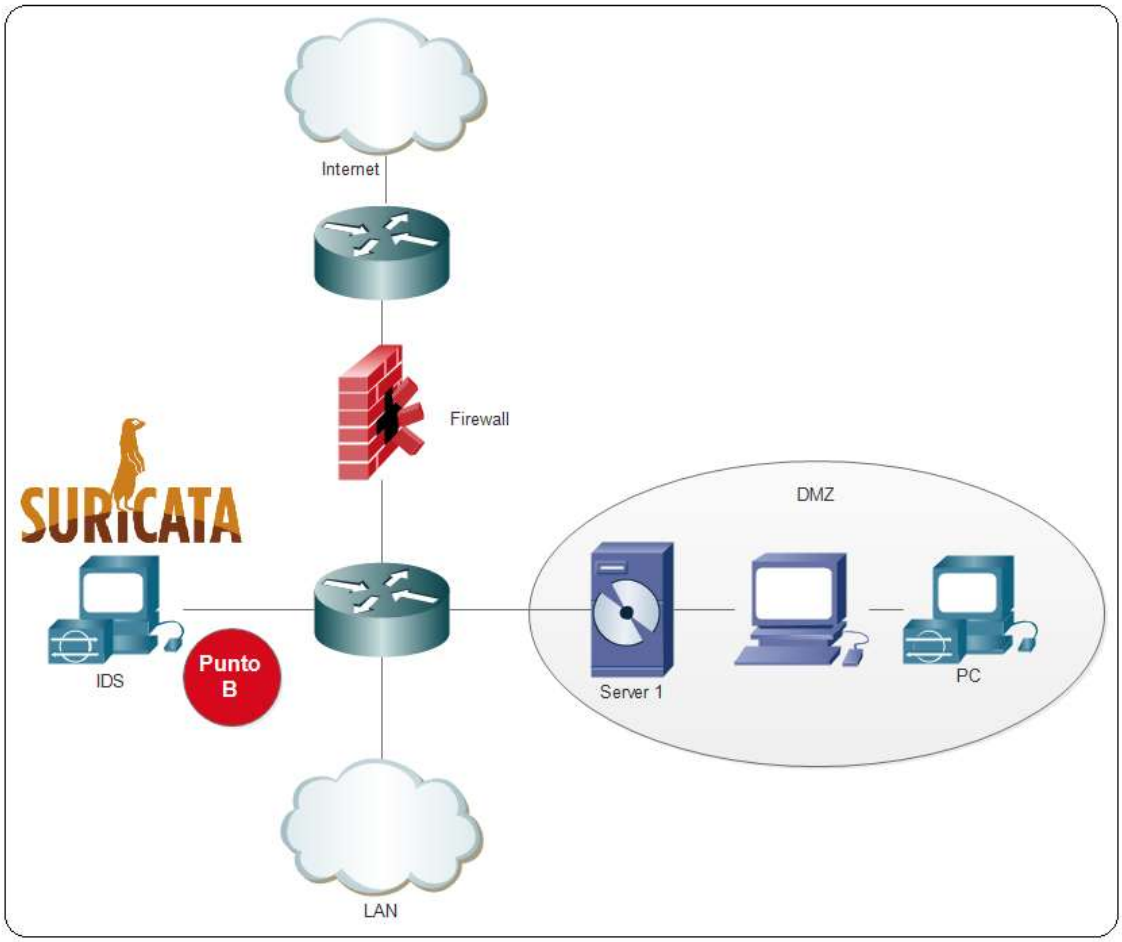

Figura 1-5: Donde ubicar Suricata

Realizado por: Eduardo Arteaga

5.5.3 Instalación de Suricata

Una vez definido el equipo donde se va a instalar Suricata, se debe definir como sistema base Centos 7, ya que los pasos descritos están definidos para esta plataforma Open Source.

Paso 1: Instalar las dependencias

Antes de iniciar con la instalación de Suricata, es necesario realizar la instalación de las dependencias, para lo cual se debe utilizar las siguientes líneas de comandos:

```
yum install wget gcc-c++ libpcap-devel libnet-devel pcre-devel automake autoconf libtool make libyaml-devel zlib-devel file-devel jansson-devel nss-devel // instalación de dependencias.
```

Paso 2: Instalación de Suricata

Para la instalación de Suricata, descargar el último código fuente desde el sitio oficial de Suricata http://suricata-ids.org/download/, y construirlo. Para lo cual se debe seguir la siguiente secuencia de comandos

```
wget http://www.openinfosecfoundation.org/download/suricata-2.0.8.tar.gz // Descargar archivos de Suricata
tar -xvf suricata-2.0.8.tar.gz // descomprimir los archivos
cd suricata-2.0.8 // acceder al directorio
./configure --sysconfdir=/etc --localstatedir=/var // configuración de archivos
make // compilación de archivos de Suricata
make install // instalación de Suricata
```

El código fuente de Suricata viene con archivos de configuración por defecto. Por lo que se tiene que instalar los archivos de configuración por defecto utilizando el siguiente comando.

```
sudo make install-conf // instalación de los archivs de Suricata
```

Para instalar reglas de Suricata, se debe ejecutar el siguiente comando.

```
sudo make install-rules // descarga de reglas de Suricata
```

El comando de instalación de las reglas de Suricata, realiza la descarga de la instantánea actual de los conjuntos de reglas disponibles en la comunidad de EmergingThreats.net , y los almacena en el directorio /etc/suricata/rules, como se puede ver en la **Figura 2-5,** el detalle de las reglas almacenadas.

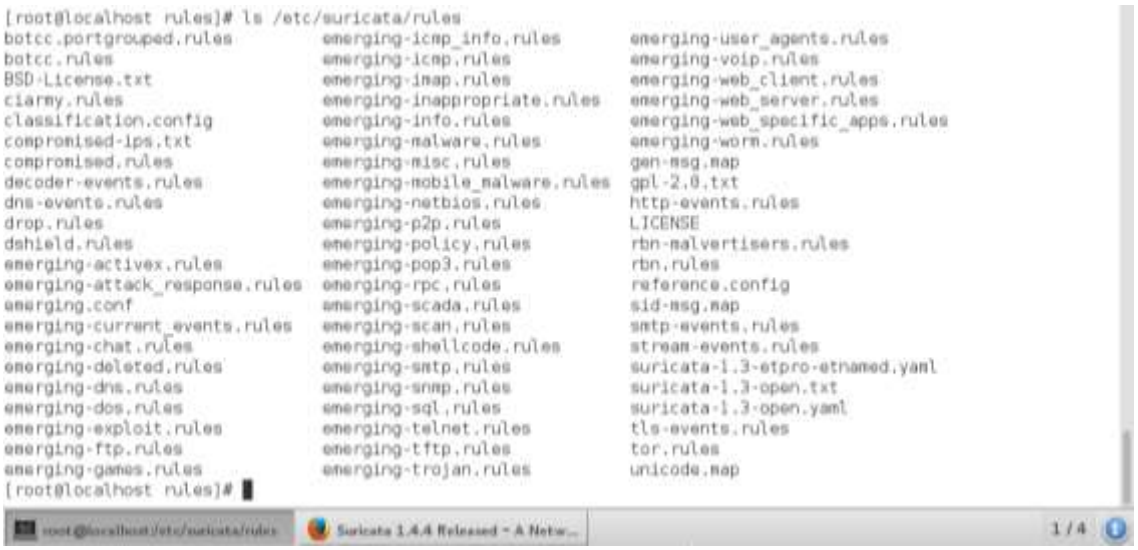

Figura 2-5: Directorio de reglas de Suricata

Realizado por: Eduardo Arteaga

Paso 3: Configuración de Suricata

Para la configuración de Suricata se editar el archivo suricata.yaml, mediante el editor Vi, para lo cual se utilizó el siguiente comando:

vi /etc/suricata/suricata.yaml // editar el archivo de configuración

Las reglas de configuración básica de Suricata son:

La variable "default-log-dir", debe apuntar a la ubicación de los archivos de registro de los logs de Suricata.

default-log-dir: /var/log/suricata/ // ubicación de los logs de Suricata

Variables por configurar

HOME_NET: "[192.168.122.0/24]" // Red local
EXTERNAL_NET: "! $HOME_NET" // Cualquier otra red

HTTP_PORTS: "80" // Número de puertos de servicios
SHELLCODE_PORTS: "!80"
SSH_PORTS: 22 // puerto SSH

Bajo la sección "threading" de Suricata, puede especificar con la afinidad del CPU para los diferentes hilos de ejecución de Suricata, al momento de realizar un análisis de tráfico de la red. Por defecto, la afinidad de la CPU está desactivado (“set-cpu-affinity: no"), lo que significa que las discusiones de Suricata serán programados en cualquiera de los núcleos del CPU disponibles. De forma predeterminada, Suricata creará un hilo "detectar" para cada núcleo de CPU, este comportamiento puede ajustarse especificando la siguiente variable "detect-thread-ratio: X". Esto creará *X* * *Y* detectar hilos, donde Y es el número total de núcleos de CPU del host.

threading:
set-cpu-affinity: no // afinidad del CPU
detect-thread-ratio: 1.5 // número de hilos

Con la configuración definida anteriormente, Suricata creará 1,5 * hilos de detección Y, donde Y es el número total de núcleos de CPU en el host.

Paso 4: Ejecución

Para verificar la ejecución de Suricata, antes de su lanzamiento, para el modo de captura .pcap, es necesario desactivar cualquier característica de paquetes offloead como puede ser: LRO / GRO en la tarjeta de red del equipo informático utilizado, que está escuchando en Suricata, ya que estas funciones pueden interferir con la captura de paquetes en vivo de la red analizada.

Para desactivar las características de LRO/GRO en la interfaz de red, se tiene que utilizar la siguiente línea de comandos.

sudo ethtool -K enp0s3 gro off lro off

Se debe considerar que, dependiendo de la tarjeta de red del equipo informático utilizado, es posible que aparezca algún mensaje de advertencia, que puede pasar por alto. Simplemente significa que la tarjeta de red no es compatible con las características de LRO.

Cannot change large-receive-offload // No se puede cambiar a recibir-offload

Suricata es compatible con diferentes modos de funcionamiento. Un RunMode determina cómo los diferentes hilos se utilizan para el IDS al momento de analizar el tráfico de la red. En la **Figura 3-5,** se muestra la ejecución del comando para listar los runmodes disponibles de Suricata.

sudo /usr/local/bin/suricata --list-runmodes

```
[root@localhost rules]# sudo /usr/local/bin/suricata --list-runmodes
----------------------------------------- Runmodes ------------------------------------------
| RunMode Type      | Custom Mode       | Description
|-------------------------------------------------------------------------------------------
| PCAP_DEV          | single            | Single threaded pcap live mode
|                   ---------------------------------------------------------------------
|                   | auto              | Multi threaded pcap live mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pcap live mode.  Packets from each flow are assigned to
 a single detect thread, unlike "pcap_live_auto" where packets from the same flow can be processed by any detect
 thread
|                   ---------------------------------------------------------------------
|                   | workers           | Workers pcap live mode, each thread does all tasks from acquisition to
 logging
|-------------------------------------------------------------------------------------------
| PCAP_FILE         | single            | Single threaded pcap file mode
|                   ---------------------------------------------------------------------
|                   | auto              | Multi threaded pcap file mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pcap file mode.  Packets from each flow are assigned to
 a single detect thread, unlike "pcap-file-auto" where packets from the same flow can be processed by any detect
 thread
|-------------------------------------------------------------------------------------------
| PFRING(DISABLED)  | auto              | Multi threaded pfring mode
|                   ---------------------------------------------------------------------
|                   | autofp            | Multi threaded pfring mode.  Packets from each flow are assigned to a
single detect thread, unlike "pfring_auto" where packets from the same flow can be processed by any detect threa
d
|                   ---------------------------------------------------------------------
|                   | single            | Single threaded pfring mode
|                   ---------------------------------------------------------------------
|                   | workers           | Workers pfring mode, each thread does all tasks from acquisition to lo
gging
|-------------------------------------------------------------------------------------------
```

Figura 3-5: Runmodes Suricata

Realizado por: Eduardo Arteaga

El RunMode predeterminado utilizado por Suricata es autofp (que significa "auto equilibrio para flujos altos de carga" o" auto flow pinned load balancing"). En este modo, los paquetes de cada flujo distinto se asignan a un solo hilo "detectar". Los flujos se asignan a las discusiones con el menor número de paquetes sin procesar. (Orovengua, 2016)

$ sudo /usr/local/bin/suricata -c /etc/suricata/suricata.yaml -i enp0s3 --init-errors-fatal

```
|-------------------------------------------------------------------------------------------
[root@localhost rules]# sudo /usr/local/bin/suricata -c /etc/suricata/suricata.yaml -i enp0s3 --init-errors-fata
l
9/10/2017 -- 20:41:24 - <Notice> - This is Suricata version 2.0.8 RELEASE
9/10/2017 -- 20:41:30 - <Warning> - [ERRCODE: SC_ERR_PCAP_CREATE(21)] - Using Pcap capture with GRO or LRO activ
ated can lead to capture problems.
9/10/2017 -- 20:41:30 - <Notice> - all 2 packet processing threads, 3 management threads initialized, engine sta
rted.
root@localhost:/etc/suricata/rules   Suricata 1.4.4 Released - A Netw...   1 / 4
```

Figura 4-5: Ejecución Suricata

Realizado por: Eduardo Arteaga

En la **Figura 4-5**, se muestra la ejecución de Suricata en modo pcap, para lo cual se utiliza la siguiente línea de comandos:

5.5.4 Instalación de Snorby

Para empezar con su instalación es necesario la instalación de Develoment Tools y las siguientes dependencias, para lo cual se ejecuta los siguientes comandos:

```
yum groupinstall "Development Tools" –y // instalación de herramientas de desarrollo
```

```
yum install openssl-devel readline-devel libxml2-devel libxslt-devel urw-fonts libX11-devel libXext-devel git fontconfig-devel libXrender-devel unzip wget xorg-x11-server-Xvfb libyaml libyaml-devel gdbm-devel db4-devel libffi-devel ethtool httpd httpd-devel ImageMagick ImageMagick-devel curl libcurl libcurl-devel libmnl-devel gcc zlib-devel jansson-devel libnet-devel libnetfilter_queue-devel mariadb mariadb-devel mariadb-server java-1.8.0-openjdk –y // instalación de herramientas de dependencias de Snorby
```

Compilar Ruby, para lo cual se procede a descargar en el directorio correspondiente y posterior su instalación utilizando la siguiente secuencia de comandos.

```
cd /usr/src // acceder al directorio src
wget -c https://cache.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-p551.tar.gz //descargar Ruby
tar zxvf ruby-1.9.3-p551.tar.gz  // Descomprimir los archivos
cd ruby-1.9.3-p551/ // acceder al directorio
./configure --prefix=/usr // configuración de archivos
 make && make install // compilación e instalación de Ruby
```

Finalmente se puede verificar la versión de Ruby instalada

```
ruby –v // ver la versión de Ruby
ruby 1.9.3p551 (2014-11-13 revisión 48407) [x86_64-linux]
```

Habilitar los servicios de Apache y Mariadb

```
systemctl enable httpd //habilitar Apache
systemctl enable mariadb // habilitar Mariadb
```

Instalación de la dependencia wkhtmltox-0.12, utilizando la siguiente secuencia de comandos

```
cd /usr/src/ // acceder al directorio src
wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.4/wkhtmltox-
0.12.4_linux-generic-amd64.tar.xz // descargar los archivos de wkhtmltox-0.12
tar Jxvf wkhtmltox-0.12.4_linux-generic-amd64.tar.xz  // descomprimir los archivos
cd wkhtmltox/ // acceder a al directorio
./configure  // configuración de archivos
cd bin/ // acceder al directorio bin
mv wkhtmltopdf /usr/bin/wkhtmltopdf // renombrar los directorios
mv wkhtmltoimage /usr/bin/wkhtmltoimage // renombrar los directorios
```

Instalación de Rails y Rake

```
gem install bundler rails
gem install rake --version=0.9.2
```

Descarga de Snorby en el directorio correspondiente y realizar el copiado de archivos.

```
cd /var/www/html/ // acceder al directorio
git clone git://github.com/Snorby/snorby.git // clonar los archivos de Snorby
cd snorby/config/ // acceder al directorio
cp database.yml.example database.yml // renombrar los archivos
cp snorby_config.yml.example snorby_config.yml // renombrar los archivos
```

Configuración de archivos de Snorby.

```
vim /var/www/html/snorby/config/snorby_config.yml // editar archivo snorby_config.yml
#/var/www/snorby/html/config/snorby_config.yml // ruta del archivo de configuración
production:
  baseuri: ''
  domain: 'snorby.prueba.com.br' // dominio
  wkhtmltopdf: /usr/bin/wkhtmltopdf // ruta de wkhtmltopdf
  ssl: false
  mailer_sender: 'example@gmail.com' // correo de envió
  geoip_uri:
http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz // geoip
  rules: - ""
  authentication_mode: database // modo de autenticación
```

```
timezone_search: true // habilitación de la zona horaria
time_zone: 'America/Guayaquil' // zona horaria
```

Para el acceso remoto de Suricata editamos el siguiente archivo y procedemos a es la dirección correspondiente.

```
vim /etc/my.cnf // editar archive my.cnf
[mysqld]
[...]
bind-address            = 0.0.0.0 // dirección
```

Iniciar servicios de Mariadb

```
systemctl start mariadb // iniciar servicios de Mariadb
```

Creación del password de root para Mariadb

```
mysqladmin -u root password 'password' // pasword del usuario root
```

Creación de la base de datos Snorby y usuarios correspondientes

```
mysql -u root –p // acceso a la consola de Mariadb
CREATE DATABASE snorby; // creación de la base de datos snorby
GRANT ALL PRIVILEGES ON snorby.* TO snorby@'%' IDENTIFIED BY 'claveuser'; //
creación de usuario snorby
GRANT ALL PRIVILEGES ON snorby.* TO snorby@'localhost' IDENTIFIED BY 'claveuser'; //
privilegios al usuario snort
FLUSH PRIVILEGES; // refrescar privilegios
Exit // salir de la consola
```

Configuración del archivo de conexión de a la base de datos de Snorby database.yml

```
vim /var/www/html/snorby/config/database.yml //editar el archive de configuración
# /var/www/html/snorby/config/database.yml
snorby: &snorby // descripción
adapter: mysql //tipo
username: snorby // usuario
```

```
password: "claveuser" //clave de mysql
host: localhost // nombre del host

production:
database: snorby // nombre de la base de datos
<<: *snorby
```

Despliegue de Snorby

```
cd /var/www/html/snorby/config/ // ruta de los archivos de configuración
bundle install // instalación de componentes
bundle exec rake snorby: setup RAILS_ENV=production // iniciar el servicio de Snorby
```

Instalación de Passenger gem ruby

```
gem install passenger
```

Instalación del módulo de Passenger para Apache

```
passenger-install-apache2-module -a
```

Creación del módulo de configuración de Apache.

```
vim /etc/httpd/conf.modules.d/passenger.conf
LoadModule passenger_module /usr/lib/ruby/gems/1.9.1/gems/passenger-
5.0.6/buildout/apache2/mod_passenger.so //configuración del módulo de passenger para
apache
```

Creación del archivo de configuración para el módulo de Passenger

```
vim /etc/httpd/conf.d/passenger.conf
<IfModule mod_passenger.c>
   PassengerRoot /usr/lib/ruby/gems/1.9.1/gems/passenger-5.0.6
   PassengerDefaultRuby /usr/bin/ruby
</IfModule>
```

Creación del host virtual para el uso de Snorby.

```
vim /etc/httpd/conf.d/snorby.conf

<VirtualHost *:80> // host virtual
    ServerName snorby.prueba.com.br // nombre del servidor
    DocumentRoot /var/www/html/snorby/public
    RailsEnv production
    <Directory /var/www/html/snorby/public>
     AllowOverride all
     Options -MultiViews
    </Directory>
     ServerSignature Off
     LogLevel info
     CustomLog /var/log/httpd/snorby.prueba.com.br-access.log combined
     ErrorLog /var/log/httpd/snorby.prueba.com.br-error.log
</VirtualHost>
```

Otorgar Permisos al directorio de Snorby.

```
chown -R apache: apache /var/www/html/snorby // permisos al directorio de Snorby
```

Reiniciar Apache.

```
systemctl restart httpd // reiniciar apache
```

Ahora se pueden chequear los archivos de los logs de Snorby.

```
cd /var/log/httpd/ // ruta d elos archivos de los logs
tail -f /var/log/httpd/snorby.prueba.com.br-* // ver los últimos logs
```

Creación del servicio para iniciar y parar Snorby, en la **Figura 5-5,** se muestra las configuraciones del servicio.

```
vim /usr/lib/systemd/system/snorby.service // creación del servicio de snorby.service
```

```
[Unit]
Description=Snorby ConfiServ // nombre del servicio
After=syslog.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/etc/snorby/snorby-start // iniciar
ExecStop=/etc/snorby/snorby-stop // parar
[Install]
WantedBy=multi-user.target // tipo de servicio
```

Figura 5-2: Configuración del servicio de Snorby

Realizado por: Eduardo Arteaga

Habilitación de Snorby.

```
systemctl enable snorby
```

5.5.5 Instalación de Barnyard2

Para proceder con la instalación de Barnyard2, es necesarios previamente instalar las siguientes dependencias, utilizando las siguientes líneas de comandos:

```
yum install libpcre3 libpcre3-dbg libpcre3-dev build-essential automake autoconf libtool libpcap-dev libnet1-dev mysql-client libmysqlclient16-dev // dependencias de Barnyard2
```

Para la instalación de Barnyard2, se deben seguir las siguientes líneas de comandos:

```
cd /srv //directorio para descargar el archivo
git clone https://github.com/firnsy/barnyard2.git // clonación de los archivos de Barnyard2
cd barnyard2/ //acceder a la carpeta
autoreconf -fvi -I ./m4 //autoconfiguración de los archivos para el make
/autogen.sh // creación del script
./configure --with-mysql -with-mysql-libraries=/usr/lib64/mysql/ //configuración de archivos para mysql de 64
make // compilación de archivos
make install // instalación de la compilación
```

Copiamos el archivo de configuración.

```
cd barnyard2/etc
cp barnyard2.conf /etc/suricata/
```

Editamos las configuraciones del archivo barnyard2.conf de los siguientes parámetros.

```
gedit barnyard2.conf // abrir el archivo de configuración
config reference_file:      /etc/suricata/reference.config //habilitar ruta de archivos
config classification_file: /etc/suricata/classification.config // habilitar ruta de archivos
config gen_file:            /etc/suricata/rules/gen-msg.map // habilitar ruta de archivos
config sid_file:            /etc/suricata/rules/sid-msg.map // habilitar ruta de archivos
.....
.....
output database: log, mysql, user=snorbyuser password=password_mysql dbname=snorby
host=localhost // cadena de conexión a la base de datos de Mariadb
.....
.....
sensor_name=enp0s3 // nombre la tarjeta de red
```

Creación de la carpeta de logs de barnyard2 y archivo.

```
mkdir /var/log/barnyard2
touch /var/log/suricata/suricata.waldo
```

Habilitar el registro unified2 en suricata.yaml:

```
# alert output for use with Barnyard2
- unified2-alert: // nombre de la alerta
enabled: yes //estado habilitado
filename: unified2.alert // tipo de archivo de la alerta
```

5.5.6 Actualización de Reglas

El conjunto de reglas a utilizar en un IDS debe ser descargado de las páginas oficiales de la comunidad de EmergingThreats.net y debe mantener periódicamente actualizado con finalidad de que el IDS pueda detectar nuevas.

5.5.7 Testeo del IDS

Una vez finalizado la implementación de Suricata es necesario realizar un testeo de funcionalidad con la finalidad de verificar que el IDS genere las alertas correspondientes a los ataques que pudiesen generarse, para lo cual es necesario utilizar herramientas que permiten simular de ataques DOS y monitorización como son: Hping3, Hydra, Nmap, Nikto, entre otras, es fundamental que estas herramientas se ejecuten sobre sistemas operativos sin mayores seguridades como puede ser un metasploitable, a fin validar la configuración del IDS para cerciorarnos de que este generando alertas correspondientes a eventos de ataques sobre la red.

5.5.8 Configuración de nuevas reglas

Un IDS al momento de su instalación, cargar todas las reglas por defecto, por lo que generará alertas de todo tipo, esto dependerá mucho de la infraestructura que se esté utilizando, como se había indicado en el proyecto de investigación muchos paquetes de red se crean con erros de comprobación de TCP debido a que estos paquetes se generaron en un ambiente virtual, esto puede variar dependiendo de programa utilizado para virtualizar, por lo que es muy importante una vez determinadas las reglas que generar falsos positivos, realizar la configuración de una nueva regla o realizar la actualización correspondiente para evitar que esas alertas ocasionen el consumo de recursos innecesarios del IDS.

5.5.9 Iniciar aplicaciones

Para iniciar Suricata, Snorby y Barnyard2, es necesario seguir la siguiente secuencia, para lo cual es necesario abrir 3 terminales o consolas:

Terminal 1: Iniciar Suricata

Para iniciar el Suricata se debe utilizar la siguiente línea de comandos:

```
sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3 -D
```

Nota: Cabe indicar que el nombre de la tarjeta de red (enp0s3) puede variar dependiendo de la configuración o versión del sistema base.

Terminal 2: Iniciar Barnyard2

Para iniciar el Barnyard2 se debe utilizar la siguiente línea de comandos:

```
sudo barnyard2 -c /etc/suricata/barnyard2.conf -f unified2.alert -d /var/log/suricata -w /var/log/suricata/suricata.waldo -D
```

Terminal 3: Iniciar Snorby

Para verificar las alertas emitas por Suricata es necesario acceder a Snorby a través del navegador utilizando el siguiente url http://localhost:3000, utilizando email: snorby@example.com password: snorby, ver **Figura 6-5**.

Figura 6-5: Pantalla de inicio de Snorby

Realizado por: Eduardo Arteaga

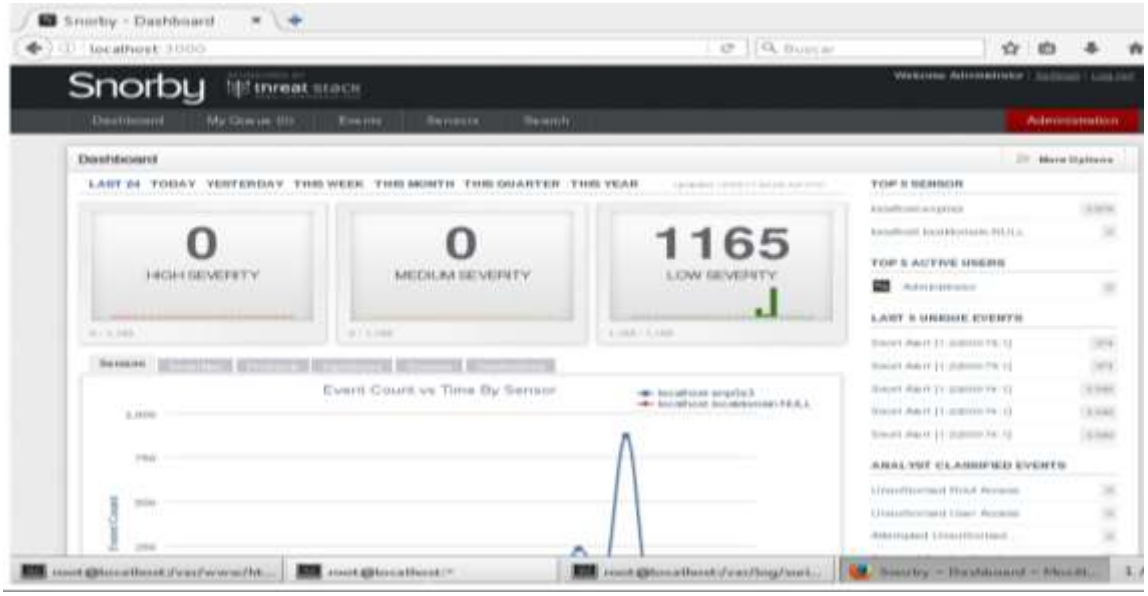

Figura 7-5: Interfaz de monitoreo de Snorby

Realizado por: Eduardo Arteaga

Para iniciar el Snorby se debe utilizar las siguientes líneas de comandos:

```
cd /var/www/html/snorby
bundle exec rails server –e production
```

Una vez que se ha iniciado correctamente se puede apreciar a través del navegador web el monitoreo de la Red. En primera instancia Suricata clasificará las alertas de acuerdo con su prioridad en baja, media y alta, para facilitar la toma de decisiones al respecto, ver **Figura 7-5**.
En la parte izquierda de la **Figura 7-5**, se podrá visualizar el listado de las últimas alertas generadas Para mayor detalle se pue dar un clic sobre una de ellas y se desplegará la información al respecto como puede ser las direcciones IP, descripción de la alerta y otras, ver **Figura 8-5**.

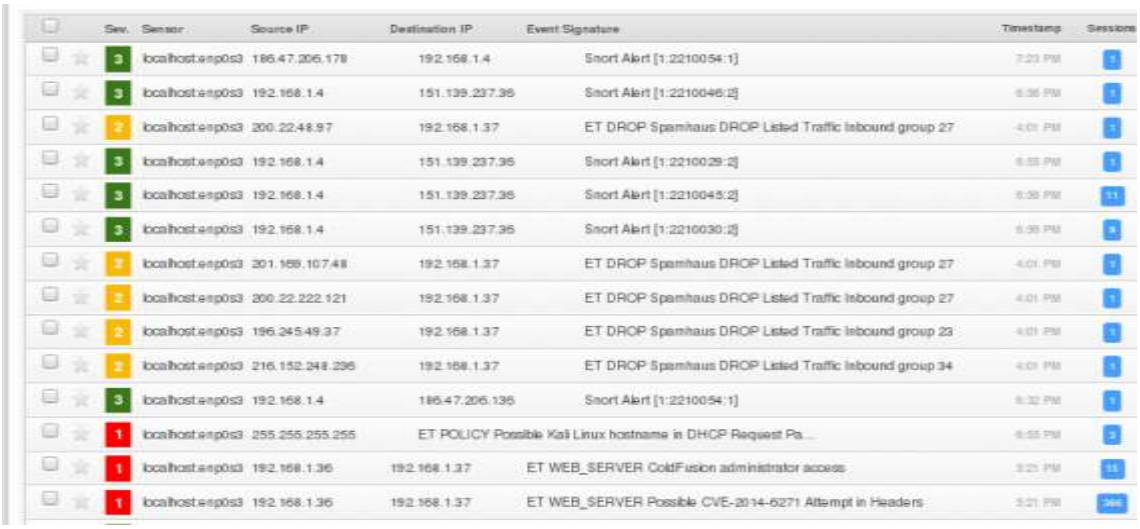

Figura 8-5: Información de alertas de Suricata

Realizado por: Eduardo Arteaga

CONCLUSIONES

- En el proyecto de investigación se ha realizado la implementación de IDSes de plataforma Open Source, que utilizan la técnica de detección de anomalías, como son Bro IDS, Snort y Suricata, en diferentes escenarios, considerando que Snort y Suricata mantienen una estructura basada en firmas de conocimiento y por otra parte BRO IDS se basa en una estructura de políticas de scripts (comportamiento), razón por la cual no se considera al momento de la evaluación de los indicadores propuestos, debido que su esquema de funcionalidad es distinto y por ende sus resultados serían discordantes.

- Luego del análisis se concluye que: en cuanto a la funcionalidad Suricata es un 5% mejor que Snort; en cuanto al desempeño se determinó que: Suricata al momento de analizar el conjunto de datos de DARPA 99 que consta de 44894776 paquetes tiene un 0% de paquetes perdidos, con respecto a Snort que tiene un 7.6 %; referente a los tiempos de respuesta utilizando el mismo conjunto de datos se evidencia que Suricata emplea un tiempo de 484 segundos y por otra parte a Snort lo toma realizar el análisis en un tiempo de 1086 segundos; y respecto a la seguridad Suricata ofrece una sensibilidad del 33.37% respecto a Snort del 25.63%. Luego del análisis estadístico inferencial mediante la prueba Z, con la utilización de los datos del caso aplicativo se establece que Suricata ofrece una mejor seguridad que Snort.

- Con la finalidad de garantizar que un IDS pueda analizar todo el tráfico que viaja en una red datos es necesario implementarlo delante del firewall para que nos permita realizar una monitorización de las intrusiones que logran atravesar el firewall principal, detectar ataques a servidores que ofrecen servicios públicos, intentos de conexiones salientes e identificar los ataques y escaneos más comunes que pudiesen ocurrir.

- Para poder cuantificar las alertas que emite los IDS, se emplearon diferentes herramientas polivalentes como son: Hydra, Nmap, Nikto, Hping, entre otras, cabe indicar que cada una estas herramientas tienen un objetivo específico sobre las máquinas que están siendo atacadas.

- La elaboración del manual de buenas prácticas de los sistemas de detección de intrusos de plataformas Open Source, sistematiza los pasos necesarios para la implementación de un sistema de IDS, los mismos que están verificados al 2017, y de acuerdo con las versiones propuestas, ya que al momento de realizar este manual en conjunto con la

implementación de los laboratorios, se encontraron que ciertos pasos descritos en manuales y documentación de la WEB, estaban obsoletos e incompletos, causando muchas molestias al momento de instalar, por lo que este trabajo servirá de base para realizar la implementación de Suricata de forma óptima, eficiente y segura, y de esta manera ser un complemento más, para aportar a la seguridad de la información que viaja por la red.

- Al analizar el tráfico de la red de datos de la Secretaría de Hidrocarburos utilizando el IDS Suricata, se determinaron varios verdaderos negativos que provenían desde el sistema de gestión de contenidos, los mismos que al ser corregidos permitieron mejorar la seguridad de la información que viaja por la red de datos.

RECOMENDACIONES

- Mantener actualizado las reglas del sistema de detección de intrusos con periodicidad, permitirá que el sistema alerte sobre nuevas amenazas que viajan por la red, ya sea esta externa o interna que pueden vulnerar la seguridad de la información.

- Un sistema de detección de intrusos alerta sobre nuevas amenazas, por lo que es importante complementarlo con un sistema de prevención de intrusos, con la finalidad de saber que acción se debe tomar al respecto, como puede ser denegar el servicio o permitir el acceso, de acuerdo con las reglas establecidas.

- Utilizar el manual de buenas prácticas permitirá implementar un sistema de detección de intrusos de plataforma Open Source por parte de los administradores de servidores y redes de manera eficiente y segura, ya que los pasos descritos en el manual corresponden a laboratorios de casos prácticos dentro de los casos de estudios de los IDSes.

- La ubicación de un IDS debe estar priorizado hacia los servicios del Core de la institución u organización, ya que serán el centro de ataques de terceros con la finalidad de detener servicios tecnológicos o robar información, cabe indicar que el IDS solo alerta de amenazas porque es muy importante fusionar con otras herramientas para denegar los servicios requeridos por los intrusos o atacantes de la red.

- Considerando la importancia de la seguridad de la información dentro de las instituciones se recomienda que se realice un análisis de la usabilidad de las herramientas de análisis del tráfico, con finalidad de que se determine el grado de uso de herramientas de plataforma Open Source y su eficiencia.

BIBLIOGRAFÍA

Alfaia, R. (2017 de 04 de 05). Installing and Configuring Snorby on CentOS 7. Obtenido de Alfaia com Linux : http://alfaiacomlinux.blogspot.com/2017/04/installing-and-configuring-snorby-on.html

Anónimo. (2002). Linux máxima seguridad. México, Madrid, Santafé de Bogota, Buenos Aires, Caracas, Lima, Montevideo, San Juan, San José, Santiago, Sao Paulo, White Plains: Prentice Hall.

Astudillo Herrera, J. A., Jimenez Macias, A. A., & Ortiz Flores, F. M. (19 de 01 de 2012). Adaptación del ids/ips suricata para que se pueda convertir en una solución empresarial. Obtenido de Repositorio Dspace: http://www.dspace.espol.edu.ec/xmlui/handle/123456789/19502

Barrios, j. (20 de 07 de 2016). Introducción a Iptables. Obtenido de Alcance libre: http://www.alcancelibre.org/staticpages/index.php/introduccion-iptables

Benchimol, D. (2011). Hacking. Banfield: Gradi S.A.

Britos, J. (01 de 09 de 2010). Detección de Intrusiones en redes de datos con captura distribuida y procesamiento estadístico.

Castillo, C. (25 de 9 de 2013). Acuerdo Ministerial 166. Eesquema gubernamental de seguridad de la informacion egsi. Quito, Pichincha, Ecuador.

Castillo, P. C. (25 de 09 de 2013). Esquema gubernamental de seguridad de la información egsi. San Francisco de Quito, Pichincha, Ecuador. Obtenido de https://www.gobiernoelectronico.gob.ec/wp-content/uploads/downloads/2016/06/Esquema-Gubernamental-de-Seguridades-de-la-Informaci%C3%83%C2%B3n.pdf

Cisco, & Affiliates. (2014). Snort users manual 2.9.11. Obtenido de http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node3.html

Corletti, A. (2011). Seguridad por niveles. Madrid.

Daniel, O. (21 de 03 de 2017). Qué es Snort. Obtenido de https://openwebinars.net/blog/que-es-snort/

Elhacker. (23 de 04 de 2017). IDS/IPS Suricata . Obtenido de http://blog.elhacker.net/2017/04/ids-ips-suricata-reglas-rules.html

Giménez, G. M. (2008). Utilización de sistemas de detección de intrusos como elemento de seguridad perimetral. Obtenido de www.adminso.es/recursos/Proyectos/PFC/PFC_marisa.pdf

GMS. (18 de 02 de 2016). Obtenido de GMS presentó recomendaciones para prevenir ataques cibernéticos en instituciones educativas: http://canalnewsecuador.com/2016/02/18/gms-presento-recomendaciones-para-prevenir-ataques-ciberneticos-en-instituciones-educativas/

Gómez, J. (2009). Optimización de sistemas de detección de intrusos en red utilizando técnicas de computacionales avanzadas. Obtenido de https://books.google.com.ec/books?id=QTdBAQAAQBAJ&pg=PA69&lpg=PA69&dq=anomalias+de+los+sistemas+de+detecci%C3%B3n+de+intrusos&source=bl&ots=ZZXavJAmnP&sig=JwcfDAsZjsESdcmbwlX-zzrSCpo&hl=es&sa=X&redir_esc=y#v=onepage&q&f=false

https://lionsec.net. (2016). Técnicas de prevención de intrusiones (IDS/IPS). Obtenido de https://lionsec.net/blog/tecnicas-de-prevencion-de-intrusiones/

Jean-Philippe, L. (s.f.). Suricata,Snorby and Barnyard2 set up guide. Obtenido de Suricata: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricata_Snorby_and_Barnyard2_set_up_guide

Maestre, J. (01 de 06 de 2013). Sistema para la correlación de alertas de NIDS basados en anomalías. Obtenido de http://eprints.ucm.es/22651/1/thesis.pdf

Mieres, J. (01 de 01 de 2009). Ataques informáticos. Obtenido de Debilidades de seguridad comúnmente explotadas: https://www.evilfingers.com/publications/white_AR/01_Ataques_informaticos.pdf

Mira, E. (2000). Implantación de un sistema de detección de intrusos en la Universidad de Valencia. Obtenido de https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwi_ntfCiOvZAhWphOAKHcTgDYoQFggnMAA&url=https%3A%2F%2Fwww.rediris.es%2Fcert%2Fdoc%2Fpdf%2Fids-uv.pdf&usg=AOvVaw2F8CZvyJ7sQ7Dd3DdTdlkW

Moreno, M. (15 de 04 de 2010). Introducción a la esteganografía (I). Obtenido de http://www.securityartwork.es/2010/04/15/introduccion-a-la-esteganografia-i/

Orovengua, J. (27 de 07 de 2016). ¿Cómo instalar Suricata sistema de detección de intrusos en Linux? Obtenido de http://www.linux-party.com/26-hackers/9548-como-instalar-suricata-sistema-de-deteccion-de-intrusos-en-linux

Puchades, A. (01 de 12 de 2008). https://riunet.upv.es. Obtenido de https://riunet.upv.es/bitstream/handle/10251/13179/Tesina.pdf?sequence=1

Ramírez, A. (01 de Julio de 2009). Estudio de una plataforma de deteccio´n de intrusos Open Source. Obtenido de https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=25&cad=rja&uact=8&ved=0ahUKEwikuIO2iObTAhXLSyYKHWo6DqY4FBAWCDwwBA&url=http%3A%2F%2Fupcommons.upc.edu%2Fbitstream%2Fhandle%2F2099.1%2F7483%2FPFC_Alan_Ramirez.pdf%3Fsequence%3D1&usg=AFQjCNG2Pjugq

Ramos, A. (01 de 07 de 2012). Cálculo tamaño optimo de la muestra. Obtenido de https://es.slideshare.net/maule/guia-tamao-de-la-muestra

Rediris. (12 de 11 de 2008). Sistemas de detección de intrusos . Obtenido de https://www.rediris.es/cert/doc/unixsec/node26.html

Rivero, J., Ribiero, B., & Ortiz, K. (01 de Noviembre de 2016). Comparación de algoritmos para detección de intrusos en entornos estacionarios y de flujo de datos. Obtenido de Revista Universidad y Sociedad: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202016000400004

Sanz, J. M. (28 de 07 de 2015). Bro ids: “el ojo que todo lo ve…”. Obtenido de https://www.securityartwork.es/2015/07/28/bro-ids-el-ojo-que-todo-lo-ve/

Urbina, J. (01 de Enero de 2004). Analisis y Evaluacion de Sistemas para Deteccion de Intrusos en Redes de Computadoras. Obtenido de https://www.researchgate.net/publication/40738801_Analisis_y_Evaluacion_de_Sistemas_para_Deteccion_de_Intrusos_en_Redes_de_Computadoras

Vialfa, C. (6 de 12 de 2017). Sistema de detección de intrusiones (IDS). Obtenido de http://es.ccm.net/contents/162-sistema-de-deteccion-de-intrusiones-ids

Writing Bro Scripts. (04 de 08 de 2017). Obtenido de Bro 2.5.3 documentation: https://www.bro.org/sphinx/scripting/index.html

ANEXOS

Anexo A. Instalación de Snort

Para la instalación de Snort, se han definido los siguientes pasos de acuerdo con la versión 2.9.9.

Una vez ingresado a la consola de Centos 7, se procede con la ejecución de los siguientes comandos:

Logearse como superusuario o root

su –

Descarga e instalación DAQ

yum install https://www.snort.org/downloads/snort/daq-2.0.6-1.centos7.x86_64.rpm // descarga de DAQ

Descarga e instalación de complementos

yum install ftp://rpmfind.net/linux/mageia/distrib/6/x86_64/media/core/release/lib64nghttp2_14-1.9.2-1.mga6.x86_64.rpm // descarga de complementos de Snort

Descarga e instalación de Snort

yum install https://www.snort.org/downloads/snort/snort-2.9.9.0-1.centos7.x86_64.rpm // descarga de Snort

Descarga de las reglas

mkdir /usr/local/src/snortrules // creación del directorio snortrules
cd /usr/local/src/snortrules //accede al directorio

wget https://www.snort.org/rules/snortrules-snapshot-2990.tar.gz?oinkcode=af551540944a3df742f978174996a54c8b489b7a // descargar las reglas de Snort

mv snortrules-snapshot-2990.tar.gz\?oinkcode\=af551540944a3df742f978174996a54c8b489b7a snortrules-snapshot-2990.tar.gz // renombrar el archive descargado

tar -zxvf snortrules-snapshot-2990.tar.gz // descomprimir el directorio

cd rules //acceso al directorio de las reglas de snort descargado

*mv *.rules /etc/snort/rules/ //mover las reglas al directorio de snort*

mv so_rules /etc/snort/ //mover el directorio so_rules al directorio de snort

mv preproc_rules /etc/snort/ // mover el directorio preproc_rules al directorio de snort

mv etc /etc/snort/ mover el etc al directorio de snort

touch /etc/snort/rules/white_list.rules // creación del archivo de reglas white_list.rules

touch /etc/snort/rules/black_list.rules // creación del archivo de reglas black_list.rules

touch /etc/snort/rules/local.rules // creación del archivo de reglas local.rules

Para obtener el oinkcode, se debe registrar en la página oficial de Snort https://www.snort.org.

Configuración de archivos

cd /etc/snort //acceso al directorio

gedit snort.conf // editar el archivo de configuración de Snort

//archivo snort.conf

ipvar HOME_NET X.X.X.X/24 //red analizar

var RULE_PATH /etc/snort/rules/rules //ruta del directorio de reglas

var SO_RULE_PATH /etc/snort/rules/rules/so_rules //ruta del directorio de reglas

var PREPROC_RULE_PATH /etc/snort/rules/rules/preproc_rules //ruta del directorio de reglas

var WHITE_LIST_PATH /etc/snort/rules/rules //ruta del directorio de reglas

var BLACK_LIST_PATH /etc/snort/rules/rules //ruta del directorio de reglas

*preprocessor reputation: *

*memcap 500, *

*priority whitelist, *

*nested_ip inner, *
*whitelist $WHITE_LIST_PATH/white_listip.rules, *
blacklist $BLACK_LIST_PATH/black_listip.rules
output unified2: filename snort.log, limit 128'
include $RULE_PATH/local.rules

Al finalizar guardar las configuraciones

mkdir /usr/local/lib/snort_dynamicrules //creación del directorio snort_dynamicrules
ls //listar
mkdir /var/log/snort // creación del archivo de logs
sudo chmod -R 5775 /etc/snort // permisos al directorio /etc/snort
sudo chmod -R 5775 /var/log/snort // permisos al directorio /var/log/snort
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules // permisos al directorio 5775 /usr/local/lib/snort_dynamicrules
sudo chown -R snort:snort /etc/snort //permisos al directorio y usuario snort
sudo chown -R snort:snort /var/log/snort //permisos al directorio y usuario snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules //permisos al directorio y usuario snort

Configuración del sensor

Se tienen que configurar los archivos snort.service y Snort, para ver el nombre de la interfaz de red a utilizar, se puede utilizar el comando ifconfig, para editar el archivo de configuración se debe utilizar la siguiente línea de comandos: *gedit /usr/lib/systemd/system/snort.service,. ver* **Figura 1.**

```
[Unit]
Description=Snort NIDS Daemon
After=syslog.target network.target
[Service]
Type=simple
ExecStart=/usr/sbin/snort -u snort -g snort -c /etc/snort/snort.conf -i enp0s3 // ruta del servicio de Snort
[Install]
WantedBy=multi-user.target
```

Figura 1: Configuración del servicio de Snort
Realizado por: Eduardo Arteaga

gedit /etc/sysconfig/snort // configuración del archivo
configurar la INTERFACE=enp0s3 // interfaz de red

Configuración de una alerta local

Para la creación de alertas locales o alertas propias del administrador, es necesario la edición del archivo local.rules, utilizando el siguiente comando.

gedit /etc/snort/rules/local.rules

En la **Figura 2**, se muestra un ejemplo de una regla local, que genera alertas al momento de que un equipo del HOME_NET, está recibiendo paquetes ICMP (ping) desde otro equipo.

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000087; rev:001;)
```

Figura 2: Alerta de Snort
Realizado por: Eduardo Arteaga

Verificar la configuración de Snort

Para verificar la configuración de Snort y que está generando las alertas correspondientes se ejecuta el siguiente comando.

snort -T -c /etc/snort/snort.conf

Y a continuación se desplegará el mensaje correspondiente

mensaje "Snort successfully validated the configuration!
Snort exiting"

Arrancar los servicios

Para arrancar los servicios de Snort se utilizó la siguiente secuencia de comandos:

systemctl enable snort // habilitar Snort
systemctl start snort // iniciar Snort

Ejemplos Snort

Para obtener la rúbrica de TCP/IP en informe, Puedes ver los IP dirección con esta opción, tipo

 # Snort – v, Ver **Figura 3.**

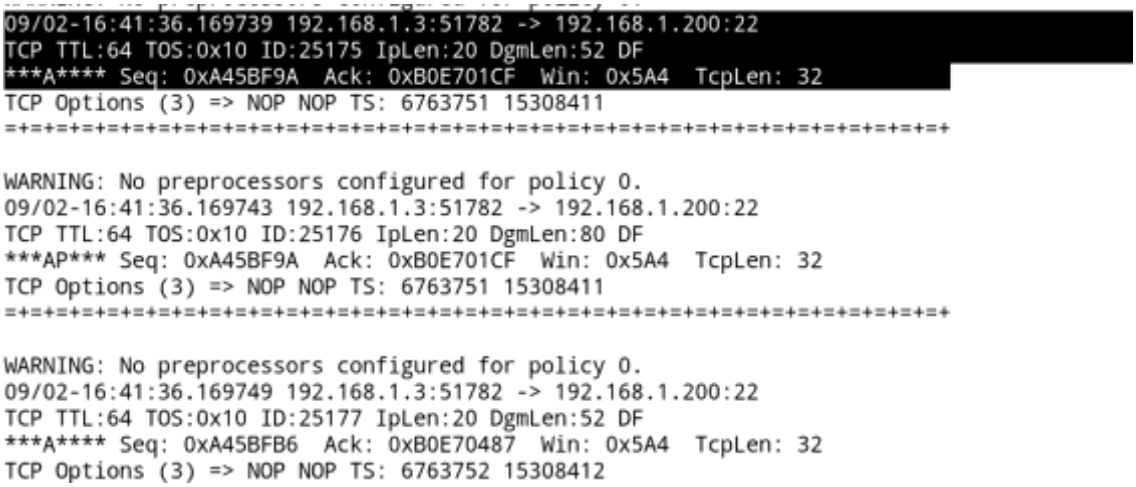

Figura 3: Rubrica TCP/IP

Realizado por: Eduardo Arteaga

Para obtener información de capa de aplicación a lo largo con IP relacionados con la información, tipo

 # Snort -vd, Ver **Figura 4.**

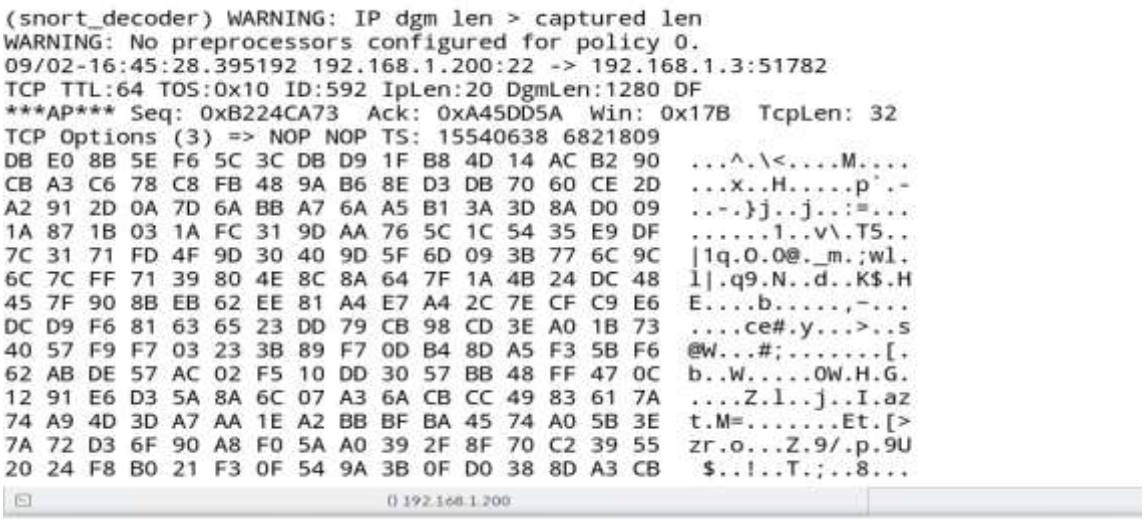

Figura 4: Información capa de aplicación

Realizado por: Eduardo Arteaga

Para librarse del maleficio en la salida, tipo

 # Snort – vdC,

-C opción eliminará hexagonal valorada de salida, ver **Figura 5**.

```
09/02-16:48:52.328431 192.168.1.200:22 -> 192.168.1.3:51782
TCP TTL:64 TOS:0x10 ID:14871 IpLen:20 DgmLen:664 DF
***AP*** Seq: 0xB31740D7  Ack: 0xA45FFDE  Win: 0x17B  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15744572 6872794
?....Y..&"=..q..*M....-........Q.........`...u...P..5j..K`:y\t..
.o.v.|Cb...r.C..c..a{%.i4......-..1/.>.........&/...-.$j.O.....i
..]..--....Q..`X..d.?9t..F.=.+Eg..:.k.O.....1.cG..{f..)..;.V.c@.
?...@>.c....i2.|. .f?.Z..i.`|.".1K...Z../.....*. >. Vc@.:..."...
b....6..W.}..i.....'....."..?Zb..-.....}..uPt......#.^h;C..sB..b
L..`.k..b{.x.mp.\.E+eFz..PG>.]..VP.I..v..<.|..Jt...._....w..5...
....)..4.K.)...P..._..........OXU?e.q.:.......A..J:.4.....vFC..N
.M/..........#...X...<......._....8.Z...s.ZV.....:.G...28Y]._..
.?..zH5-..U.4".....D............L>F@..K.z.......-..UE\\9D.4MY..
CeIA1.1mz...T%z.o+....F.._.b.M...#A.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

WARNING: No preprocessors configured for policy 0.
09/02-16:48:52.328668 192.168.1.3:51782 -> 192.168.1.200:22
TCP TTL:64 TOS:0x10 ID:33290 IpLen:20 DgmLen:80 DF
***AP*** Seq: 0xA45FFDE  Ack: 0xB317433B  Win: 0x396A  TcpLen: 32
```

Figura 5: Valoración de salida

Realizado por: Eduardo Arteaga

En la **Figura 6**, se puede ver la información de todos 6 Capas de modelo OSI excepto capa 1, para cual se debe ejecutar el siguiente comando:

 # Snort **-** *vdeC*

```
192.168.1.3:51782 -> 192.168.1.200:22 TCP TTL:64 TOS:0x10 ID:39945 IpLen:20 DgmLen:52 DF
***A**** Seq: 0xA4623C2  Ack: 0xB40B0BF3  Win: 0x396A  TcpLen: 32
TCP Options (3) => NOP NOP TS: 6906540 15879559

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

09/02-16:51:07.315868 08:00:27:E3:E4:95 -> F4:6D:04:EE:05:AD type:0x800 len:0x28E
192.168.1.200:22 -> 192.168.1.3:51782 TCP TTL:64 TOS:0x10 ID:28986 IpLen:20 DgmLen:640 DF
***AP*** Seq: 0xB40B0BF3  Ack: 0xA4623A6  Win: 0x17B  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15879559 6906540
&.m........%.H..!. ..\f.T...y.zz]..r.q.zF........-..k=.u....._.[
..z.........F..W.."+d...J.../c.$a6a.\. U....... ...j`u<.3O...i.O
.6....$...[..._.@..rm....^E.Wj........_t.t..A.w.......129f14..T.
...Vt.#-....I...d}....%.@.pX.p.S...O...B..o.:...k{....a.R.P.3...
.._..L....h4L.!-._..Sz._.%...}..5.x.jFP....q..x.e...;...OI&a...n
.80[)r.....r.Q.2=.L...1!.U.d...o.MG...w.s....X.1.........5O(#.O.
.d....Q.v.....+.A........I-..|-.U..&;(..C<.a).o...@.d,Q.tP@.....
 ..?r...|O...e......&..2.U........Un#O....2..6Sv^z/.R.Vg.K.k..s.
ug...h>E..j...-.s..n.....).....oi..T....V...h..o8..1...^!..ta...
nt(A.B......
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

^C*** Caught Int-Signal
09/02-16:51:07.316147 08:00:27:E3:E4:95 -> F4:6D:04:EE:05:AD type:0x800 len:0x5EA
192.168.1.200:22 -> 192.168.1.3:51782 TCP TTL:64 TOS:0x10 ID:28987 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xB40B0E3F  Ack: 0xA4623C2  Win: 0x17B  TcpLen: 32
TCP Options (3) => NOP NOP TS: 15879559 6906540
```

Figura 6: Información capas del modelo OSI

Realizado por: Eduardo Arteaga

Anexo B. Instalación Barnyard2 para Snort

Para la instalación de Barnyard2 para Snort se utilizan los siguientes pasos, previa instalación de Snort 2.9.9.

Instalación de los complementos y herramientas

Para la instalación de Barnyard2 es necesario realizar la instalación de las siguientes dependencias:

yum install wget man make flex bison zlib zlib-devel libpcap libpcap-devel pcre pcre-devel tcpdump gcc-c++ libtool perl-libwww-perl perl-Archive-Tar perl-Crypt-SSLeay git gcc libxml2 libxml2-devel libxslt libxslt-devel curl-devel httpd httpd-devel apr-devel apr-util-devel libXrender fontconfig libXext ruby-devel unzip xz openssl-devel readline-devel // instalación de dependencias

yum groupinstall "Development Tools" -y // instalación de herramientas de programador

yum install mariadb mariadb-devel mariadb-server urw-fonts libX11-devel libXext-devel fontconfig-devel libXrender-devel xorg-x11-server-Xvfb libyaml libyaml-devel gdbm-devel db4-devel libffi-devel ethtool ImageMagick ImageMagick-devel curl libcurl libcurl-devel libmnl-devel jansson-devel libnet-devel libnetfilter_queue-devel mysql-devel // instalación de dependencias

Descarga e instalación

Para este proyecto de investigación considerando la versión del Snort se utilizó Barnyard2 versión 2-1-14.

cd /usr/src // acceder al directorio
wget https://github.com/firnsy/barnyard2/archive/7254c24702392288fe6be948f88afb74040f6dc9.tar.gz \-O barnyard2-2-1.14-336.tar.gz // descarga de barnyard2 // descargar de Barnyard2
tar zvxf barnyard2-2-1.14-336.tar.gz // descomprimido
mv barnyard2-7254c24702392288fe6be948f88afb74040f6dc9/ barnyard2 // renombrar directorio
cd barnyard2/ //accede al directorio
autoreconf -fvi -I ./m4 //autocofigurar

./autogen.sh // crear Shell script
./configure --with-mysql -with-mysql-libraries=/usr/lib64/mysql/ // configuración de make
make // compilación
make install // instalación de la compilación

Iniciar servicios

service mariadb start // iniciar Mariadb
service mariadb status // estado de Mariadb

Creación de la base de datos y esquema snort

Para la creación del esquema de la base de datos de Snort, se utilizó la consola de Mariadb y la siguiente secuencia de comandos.

mysql -u root –p // accediendo a Mariadb

Enter password:

// Welcome to the MariaDB monitor. Commands end with; or \g.
// Your MariaDB connection id is 8
// Server version: 5.5.52-MariaDB MariaDB Server
// Se crea la contraseña para el usuario root:
mysqladmin -u root password sesamo
// Se entra en la consola de MySQL:
mysql -u root -psesamo
// Se crea la base de datos snort:
MariaDB [(none)]> create database snort;
// Se crea el usuario snortuser con los permisos necesarios para trabajar con la base de datos creada:
MariaDB [(none)]> GRANT ALL PRIVILEGES ON snort. TO 'snortuser'@'localhost' IDENTIFIED BY 'sesamo';*
//Se recargan los privilegios:
MariaDB [(none)]> flush privileges;
MariaDB [(none)]> use snort

//Database changed

MariaDB [snort]> source /usr/src/barnyard2/schemas/create_mysql

MariaDB [snort]> show tables;

```
+---------------------+
| Tables_in_snort     |
+---------------------+
| data                |
| detail              |
| encoding            |
| event               |
| icmphdr             |
| iphdr               |
| opt                 |
| reference           |
| reference_system    |
| schema              |
| sensor              |
| sig_class           |
| sig_reference       |
| signature           |
| tcphdr              |
| udphdr              |
```

MariaDB [(none)]> exit

Preparación de snort para Barnyard2

Configuración de snort.conf después de la línea 520, insertando la siguiente línea de código

output unified2: filename snort.u2, limit 128

Creación y Configuración de archivos para ejecutar Barnyard2

cd /usr/src/barnyard2 // acceder al directorio
cp etc/barnyard2.conf /etc/snort // copiar el archivo de configuración
mkdir /var/log/barnyard2 // creación del directorio

chown snort.snort /var/log/barnyard2 // permisos al archivo barnyard2
touch /var/log/snort/barnyard2.waldo // creación del archivo barnyard2.waldo
chown snort.snort /var/log/snort/barnyard2.waldo // permisos al archivo barnyard2.waldo
touch /etc/snort/sid-msg.map // creación del archivo sid-msg.map

Configuración de Barnyard2 para conectar con la base de datos de MySQL

Editar el archivo /etc/snort/barnyard2.conf y añadir la siguiente línea de código con la siguiente información de acuerdo con la configuración del servidor de mysql:

output database: log, mysql, user=snortuser password=sesamo dbname=snort host=localhost

configuración de permiso del archivo barnyard2.conf file, para prevenir que otros usuarios lo puedan leer:

chmod o-r /etc/snort/barnyard2.conf

Testeo de Barnyard2

Ejecutar Snort en modo de alerta

/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3 // ejecución de Snort

Para verificar que Snort está emitiendo alertas, se realizó un ping a la interface de red del servidor desde otro computador, después se procedió a verificar la alerta del ping en el archivo de logs de Snort con los siguientes comandos.

cd /var/log/snort //accede al directorio de los logs
ls -l /var/log/snort/ listar los logs

Seguidamente se visualizará el siguiente contenido

-rw-r--r-- 1 snort snort 0 Nov 11 14:07 barnyard2.waldo
-rw------- 1 snort snort 744 Nov 11 13:49 snort.log.1415710140
-rw------- 1 snort snort 1360 Nov 11 14:10 snort.u2.1415711432

Ejecutar Barnyard2

Para iniciar Barnyard2, se ejecuta la siguiente línea de comandos*: barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 –w /var/log/snort/barnyard2.waldo -g snort -u snort // iniciar barnyard2*

Para verificar que Barnyard2 está escribiendo los eventos alertados por Snort, en la base de datos Mysql se puede utilizar el siguiente comando

mysql -u snort -psnort -D snort -e "select count(*) from event"

Creación de scripts para iniciar servicios

cd /usr/src/barnyard2
cp barnyard2 /etc/init.d/

chmod +x /etc/init.d/snort /etc/init.d/barnyard2

Anexo C. Instalación de Adodb

Para la instalación de Abodb, se utilizaron los siguientes pasos, para lo cual es necesario tener instalado previamente Php5.

Descargar el paquete Adodb en su versión más reciente desde de la página web: /var/www/html

Descomprimir el paquete y copiar dentro de la carpeta /var/www/, por medio de la siguiente secuencia de comandos.

tar xvfz adodb-5.20.9.tar.gz //descomprimir el archivo
mv adodb5 adodb //renombrar la carpeta
mv ./adodb /var/www/ //mover al directorio www

Anexo D. Instalación de BASE

Para realizar la instalación de BASE antes es necesario instalar y configurar previamente otras herramientas: servidor web Apache (Httpd) y el soporte de Php5 para dicho servidor.

Para la instalación de BASE, se realizaron con los siguientes pasos:

Instalar complementos

yum install php
yum install php-mysql

Instalación

Descargar el paquete base en su versión más reciente desde de la página web https://sourceforge.net/projects/secureideas/

Descomprimir el paquete y copiar dentro de la carpeta /var/www/html, para lo cual se debe seguir la siguiente secuencia de comandos:

cd /var/www/html // acceder al directorio
tar xvfz base-1.4.5.tar.gz // Descomprimir los archivos de base
mv base-1.4.5 base // renombrar la carpeta
mv ./adodb /var/www/html //mover carpeta

Configuraciones de archivo

Se realizaron las siguientes configuraciones:

cd /var/www/html/base
cp base_conf.php.dist base_conf.php
gedit base_conf.php
$BASE_urlpath = '/base'; # line 50
$DBlib_path = '/var/adodb/'; # line 80
$alert_dbname= 'snort'; # line 102
$alert_host= 'localhost';

$alert_port= '3306';

$alert_user= 'snortuser';

$alert_password = sesamo; # line 106

Una vez finalizado las configuraciones, se accede a través del navegador con el siguiente utilizado la siguiente dirección “http://localhost/base” y pulsar sobre el link “setup page”. Esto nos lleva a una página donde debemos de pulsar “Create BASE AG” (Arriba a la derecha) y por último pulsar sobre el link “Main page”, que nos llevará a la página principal y poder ver las alertas generadas por Snort, ver **Figura 7**.

Una vez finalizado las configuraciones de los archivos se tiene que otorgar los permisos de solo lectura al directorio.

chmod 550 /var/www/html/base -R

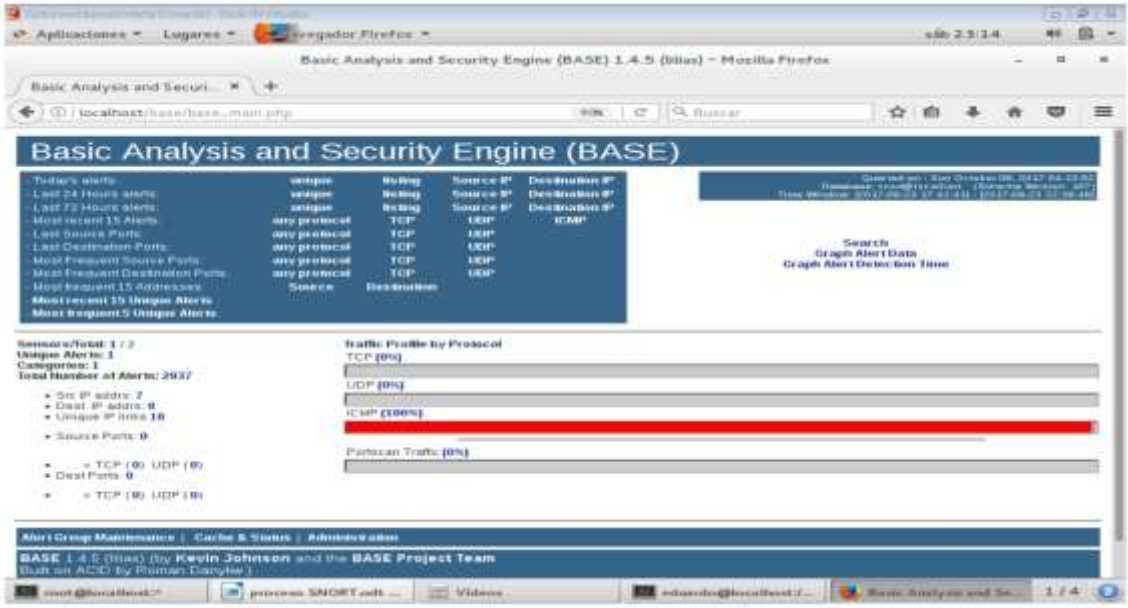

Figura 7: Información capas del modelo OSI

Realizado por: Eduardo Arteaga

Anexo E. Instalación de Suricata

Para la implementación de Suricata, es necesario realizar los siguientes pasos, en los que consta la descripción de la ejecución de comandos del proceso de instalación de Suricata.

Instalar dependencias

Para la instalación de Suricata, se realizó la instalación de las siguientes dependencias, mediante la ejecución del siguiente comando:

yum install libpcap-devel libnet-devel pcre-devel wget gcc-c++ automake autoconf libtool make libyaml-devel zlib-devel file-devel jansson-devel nss-devel // instalación de dependencias

Instalación de Suricata

Para la instalación de Suricata, es necesario descargar el último código fuente desde la página web http://suricata-ids.org/download/, y construirlo. Para realizar el proceso de instalación de Suricata se utilizó la siguiente secuencia de comandos:

wget http://www.openinfosecfoundation.org/download/suricata-2.0.8.tar.gz // descargar el código fuente de Suricata
tar -xvf suricata-2.0.8.tar.gz // descomprimir los archivos
cd suricata-2.0.8 //acceder al directorio
./configure --sysconfdir=/etc --localstatedir=/var //configuración de archivos

Después se presentará la configuración de Suricata de la siguiente forma, ver **Figura 8**.

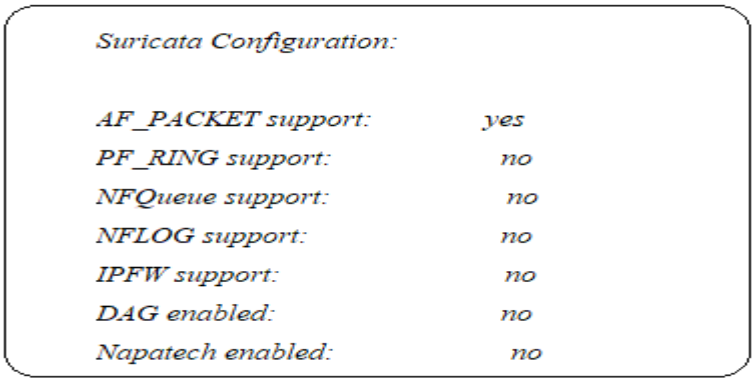

Figura 8: Información capas del modelo OSI

Realizado por: Eduardo Arteaga

Después de la configuración se procede la compilación e instalación respectivamente con los siguientes comandos.

make // compilación para obtención del make
make install // instalación de la compilación

El código fuente de Suricata viene con archivos de configuración por defecto, para la instalación de los archivos de configuración se utilizó el siguiente comando.

sudo make install-conf // instalación de archivos de configuración de Suricata

Considerando que Suricata es inútil sin los conjuntos de reglas del IDS, es importante realizar la instalación de las reglas actualizadas del IDS, para instalar reglas de Suricata, se ejecutó el siguiente comando.

sudo make install-rules // instalación de las reglas de Suricata.

Las reglas de Suricata, se descargan desde la instantánea actual de los conjuntos de reglas disponibles en la comunidad de EmergingThreats.net, y se almacenan en el directorio /etc/suricata/rules, para ver el conjunto de reglas instaladas se puede ejecutar el siguiente comando.

ls /etc/suricata/rules // directorio de las reglas de Suricata

En la **Figura 9**, se muestra el listado de las reglas instaladas para el funcionamiento de Suricata.

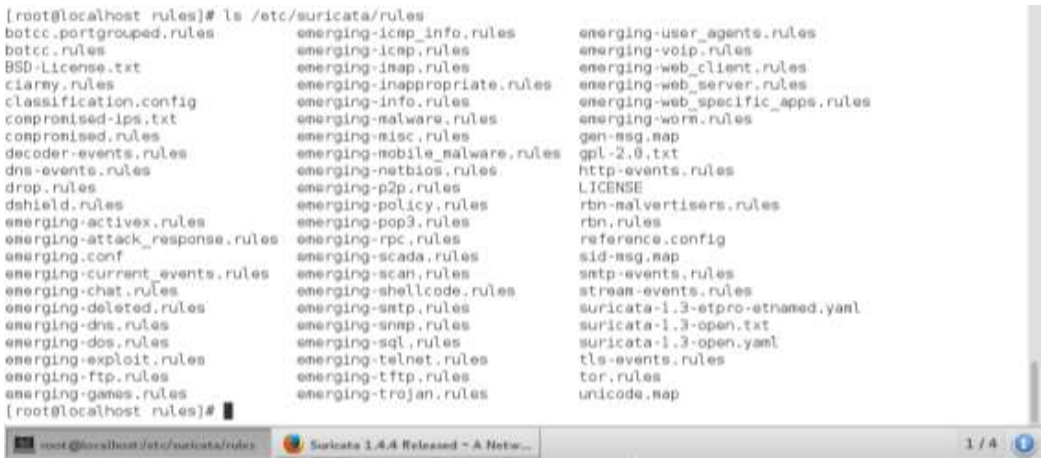

Figura 9: Reglas Suricata

Realizado por: Eduardo Arteaga

Configurar Suricata

Para la configuración de Suricata es necesario la edición del archivo de configuración que se encuentra en **/etc/suricata/suricata.yaml.**, para la edición y configuración del archivo se ejecutaron los siguientes comandos.

vi /etc/suricata/suricata.yaml //editar el archivo de configuración

Las reglas de configuración básica de Suricata se las puede identificar con la palabra clave "default-log-dir", estas deben apuntar a la ubicación de los archivos de registro Suricata.

default-log-dir: /var/log/suricata/ //archivos logs de Suricata

Variables para configurar en el archivo suricata.yaml

HOME_NET: "[192.168.122.0/24]" // red local
EXTERNAL_NET: "! $HOME_NET" // cualquier otra red
HTTP_PORTS: "80" // número de puertos de servicios
SHELLCODE_PORTS: "!80"
SSH_PORTS: 22 // puerto ssh

Ejecución

Para probar la ejecución del IDS Suricata, se puede utilizar el modo de captura de .pcap, para lo cual es recomendable desactivar cualquier característica de paquetes offloead (por ejemplo, LRO / GRO) en la tarjeta de red, que está escuchando Suricata, ya que esas funciones pueden interferir con la captura de paquetes en vivo de la red.

Para desactivar las características de LRO/GRO en la interfaz de red se tiene que utilizar el siguiente comando:

sudo ethtool -K enp0s3 gro off lro off // desactivar LRO/GRO

Es importante tener en cuenta que, dependiendo de la tarjeta de red, es posible que aparezca el siguiente mensaje de advertencia, que se puede pasar por alto. Simplemente significa que la tarjeta de red no es compatible con LRO.

Cannot change large-receive-offload // No se puede cambiar a recibir-offload

Para ejecutar Suricata en modo pcap, se utiliza la siguiente línea de código.

sudo /usr/local/bin/suricata -c /etc/suricata/suricata.yaml -i enp0s3 --init-errors-fatal, para ver la ejecución de Suricata, ver **Figura 10.**

Figura 10: Ejecución Suricata

Realizado por: Eduardo Arteaga

Anexo F. Instalación de Snorby

Para empezar con la instalación de Snorby es necesario primeramente instalar Develoment Tools y las dependencias, para lo cual se utilizó la siguiente secuencia de comandos:

yum groupinstall "Development Tools" –y //instalación de las herramientas de desarrollo

Instalación de las dependencias de Snorby

yum install openssl-devel readline-devel libxml2-devel libxslt-devel urw-fonts libX11-devel libXext-devel git fontconfig-devel libXrender-devel unzip wget xorg-x11-server-Xvfb libyaml libyaml-devel gdbm-devel db4-devel libffi-devel ethtool mariadb mariadb-devel mariadb-server httpd httpd-devel ImageMagick ImageMagick-devel curl libcurl libcurl-devel libmnl-devel gcc zlib-devel jansson-devel libnet-devel libnetfilter_queue-devel java-1.8.0-openjdk –y // instalación de las dependencias de Snorby

Compilar Ruby, para lo cual se procede a descargar en el directorio correspondiente y posterior su instalación utilizando la siguiente secuencia de comandos.

cd /usr/src //acceder al archivo src
wget -c https://cache.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-p551.tar.gz //descargar el paquete comprimido de ruby versión 1.9.3
tar zxvf ruby-1.9.3-p551.tar.gz // Descomprimir el paquete
cd ruby-1.9.3-p551/ //accede a la carpeta de archivos
./configure --prefix=/usr //configuración de archivos necesarios para make
make // compilación de los archivos de Ruby
make install // instalación de la compilación de Ruby.

Finalmente se pueden verificar la versión de Ruby instalada

ruby –v // verificar la versión de Ruby instalada

ruby 1.9.3p551 (2014-11-13 revision 48407) [x86_64-linux]

Habilitar los servicios de Apache y Mariadb

systemctl enable httpd //inciar servidor web
systemctl enable mariadb // iniciar Mariadb

Instalación manual de la dependencia wkhtmltox-0.12, utilizando la siguiente secuencia de comandos

cd /usr/src/
wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.4/wkhtmltox-0.12.4_linux-generic-amd64.tar.xz
tar Jxvf wkhtmltox-0.12.4_linux-generic-amd64.tar.xz
cd wkhtmltox/
./configure
cd bin/
mv wkhtmltopdf /usr/bin/wkhtmltopdf
mv wkhtmltoimage /usr/bin/wkhtmltoimage

Instalación de rails y rake

gem install bundler rails
gem install rake --version=0.9.2

Descarga de Snorby en el directorio correspondiente y copiado de archivos

cd /var/www/html/ // acceder al directorio
git clone git://github.com/Snorby/snorby.git // clonar los archivos de Snorby
cd snorby/config/ // acceder al directorio
cp database.yml.example database.yml // renombrar los archivos
cp snorby_config.yml.example snorby_config.yml // renombrar los archivos

Configuración de archivos de Snorby

vim /var/www/html/snorby/config/snorby_config.yml // archivo de configuración de Snorby
#/var/www/snorby/html/config/snorby_config.yml // ruta del archivo de configuración
production:
baseuri: ''
domain: 'snorby.prueba.com.br' // dominio
wkhtmltopdf: /usr/bin/wkhtmltopdf // ruta del wkhtmltopf

ssl: false
mailer_sender: 'douglas.q.santos@gmail.com' // mail
geoip_uri:
"http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz"
rules:
- ""
authentication_mode: database
timezone_search: true
time_zone: 'America/Guayaquil'

Para el acceso remoto de Suricata editamos el siguiente archivo y procedemos a editar la dirección correspondiente.

vim /etc/my.cnf
[mysqld]
[...]
bind-address = 0.0.0.0

Iniciar servicios de Mariadb

systemctl start mariadb // iniciar Mariadb

Creación del password de root para Mariadb

mysqladmin -u root password 'password' //creación de la clave para el usuario root

Creación de la base de datos Snorby y usuarios respectivamente, para lo cual se debe ingresar a la consola de Mariadb.

mysql -u root –p // acceso a Mariadb con el usuario root
CREATE DATABASE snorby; // creaación de la base de datos snorby
// creación del usuario snorby y sus privilegios
GRANT ALL PRIVILEGES ON snorby. TO snorby@'%' IDENTIFIED BY 'claveuser'; //*
GRANT ALL PRIVILEGES ON snorby. TO snorby@'localhost' IDENTIFIED BY 'claveuser';*
FLUSH PRIVILEGES; //refrescar
exit //salir de la consola de Mariadb

Configuración del archivo de conexión de a la base de datos de Snorby database.yml

vi /var/www/html/snorby/config/database.yml //creación del archivo de configuración

Configuraciones del archivo database.yml

// Definición de la cadena de conexión

```
snorby: &snorby
  adapter: mysql //base datos
  username: snorby //usuario
  password: "claveuser" // clave de acceso
  host: localhost // host
production:
  database: snorby // nombre de la base de datos
<<: *snorby
```

Despliegue de Snorby

cd /var/www/html/snorby/config/ //
bundle install //instalar components snorby
bundle exec rake snorby: setup RAILS_ENV=production //ejecución de Snorby

Instalación de Passenger gem ruby

gem install passenger

Instalación del módulo de Passenger para Apache

passenger-install-apache2-module -a

Creación del módulo de configuración de Apache.

vim /etc/httpd/conf.modules.d/passenger.conf
LoadModule passenger_module /usr/lib/ruby/gems/1.9.1/gems/passenger-5.0.6/buildout/apache2/mod_passenger.so //configuración del módulo de passenger para apache

Creación del archivo de configuración para el módulo de Passenger

vim /etc/httpd/conf.d/passenger.conf

<IfModule mod_passenger.c>

PassengerRoot /usr/lib/ruby/gems/1.9.1/gems/passenger-5.0.6 // rutas del módulo de passenger

PassengerDefaultRuby /usr/bin/ruby // ruta de Ruby

</IfModule>

Creación del host virtual para el uso de Snorby

vim /etc/httpd/conf.d/snorby.conf

*<VirtualHost *:80> // Host virtual*

ServerName snorby.prueba.com.br //nombre del host

DocumentRoot /var/www/html/snorby/public //ruta

RailsEnv production

<Directory /var/www/html/snorby/public>

AllowOverride all

Options -MultiViews

</Directory>

ServerSignature Off

LogLevel info

CustomLog /var/log/httpd/snorby.prueba.com.br-access.log combined

ErrorLog /var/log/httpd/snorby.prueba.com.br-error.log

</VirtualHost>

Permisos al directorio de Snorby

chown -R apache:apache /var/www/html/snorby //permisos del directorio de Snorby

Reiniciar Apache

systemctl restart httpd // reiniciar servidor web

Ahora se pueden chequear los archivos de los logs de Snorby

cd /var/log/httpd/ //acceso al directorio de los logs
tail -f /var/log/httpd/snorby.prueba.com.br- //verificación de los últimos logs almacenados*

Creación del servicio para iniciar y parar Snorby

vim /usr/lib/systemd/system/snorby.service //creación del archivo snorby.service para permitir iniciar o parar Snorby

[Unit] // Descripción del servicio
Description=Snorby ConfiServ
After=syslog.target

[Service] // Configuración del servicio de inicio y apagado respectivamente
Type=oneshot
RemainAfterExit=yes
ExecStart=/etc/snorby/snorby-start
ExecStop=/etc/snorby/snorby-stop

[Install] //servicios necesarios de nivel 3
WantedBy=multi-user.target

Habilitación de Snorby

systemctl enable snorby

Figura 11: Ingreso a Snorby

Realizado por: Eduardo Arteaga

Ahora ya podemos acceder a Snorby a través del navegador utilizando el siguiente url http://localhost:3000, y lo siguientes datos de ingreso al sistema email: snorby@example.com password: snorby, Ver **Figura 11.**

Anexo G. Instalación de Barnyard2 para Suricata

Para proceder con la instalación de Barnyard2, es necesario primero instalar las siguientes dependencias:

yum install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev libnet1-dev mysql-client libmysqlclient16-dev autoconf automake libtool // instalación de dependencias

Clonación de los archivos

cd /srv /ruta de descarga
git clone https://github.com/firnsy/barnyard2.git //clonación de los archivos
cd barnyard2/ //acceso a los archivos
autoreconf -fvi -I ./m4 //ejecución automática
./autogen.sh // creación del shell script
../configure --with-mysql -with-mysql-libraries=/usr/lib64/mysql/ //configuración de los archivos con liberias de mysql y obtención de make
make // realizar la compilación
make install //instalar la compilación

Copiamos el archivo de configuración

cd barnyard2/etc //abrir la ruta del archivo de configuración
cp barnyard2.conf /etc/suricata/ //copiar el archivo de configuración

Editamos el archivo de configuración de barnyard2.conf considerando los siguientes parámetros

config classification_file: /etc/suricata/classification.config
//activar ruta de los archivos de configuración
config reference_file: /etc/suricata/reference.config
//activar ruta de los archivos de configuración
config sid_file: /etc/suricata/rules/sid-msg.map
//activar ruta de las reglas de configuración
config gen_file: /etc/suricata/rules/gen-msg.map
//activar ruta de las reglas de configuración

output database: log, mysql, user=snorbyuser password=PASSWORD123 dbname=snorby host=localhost //descripción de usuario, password y base de datos de Mariadb para la conexión

sensor_name=enp0s3 //nombre de la tarjeta de red

Creación de la carpeta de logs de Barnyard2 y Suricata

mkdir /var/log/barnyard2
touch /var/log/suricata/suricata.waldo

Habilitar el registro unified2 en suricata.yaml:

 # alert output for use with Barnyard2
- unified2-alert:
enabled: yes
filename: unified2.alert

Para iniciar Suricata, se debe ejecutar la siguiente línea de comandos: *sudo suricata -c /etc/suricata/suricata.yaml -i enp0s3 -D*

Seguidamente iniciar Barnyard2, utilizando la siguiente línea de comandos: *sudo barnyard2 -c /etc/suricata/barnyard2.conf -d /var/log/suricata -f unified2.alert -w /var/log/suricata/suricata.waldo -D*

Anexo H. Instalación de Bro

Para la instalación de Bro, se utilizaron los siguientes pasos:

Prerrequisitos:

yum install cmake make gcc gcc-c++ flex bison libpcap-devel openssl-devel python-devel swig zlib-devel //instalación de dependencias
yum group install "Development Tools" // instalación de herramientas de desarrollo

Instalar Bro versión 2.5.1

wget https://www.bro.org/downloads/bro-2.5.1.tar.gz // descarga del paquete de Bro
tar zxvf bro-2.5.1.tar.gz // descomprimir el paquete
cd bro-2.5.1/ //acceso al directorio
./configure //compilación
make && make install // instalación de Bro

Configuración

Editamos el archivo vim .bashrc, para verificar que existe el path, caso contrario de deberá insertar.

export PATH=/usr/local/bro/bin:$PATH

Luego ejecutamos el bash

source ~/.bashrc

Configuración de la interfaz de red de Bro en node.cfg

cd usr/local/ bro/etc/
vim node.cfg
interface=enp0s3

Insertar la subred en el archivo networks.cfg

cd usr/local/ bro/etc/

vim networks.cfg

192.168.1.0/24 Homenet

Anexo I. Instalación de BroControl

BroControl es una herramienta para operar las instalaciones de Bro. *BroControl* tiene dos modos de operación: un modo *independiente* para administrar una configuración Bro tradicional y de un solo sistema; y un modo de *clúster* para mantener una configuración multisistema de instancias coordinadas de Bro que equilibra la carga del trabajo en un conjunto de máquinas independientes.

Par iniciar con su instalación utilizamos el siguiente comando

broctl

Para abrir la ayuda de la consola

[BroControl] > help

Instalar y actualizar configuraciones

[BroControl] > install
[BroControl] > start // iniciar el proceso
[BroControl] > diag // diagnóstico de nodos
[BroControl] > restart // Reiniciar el proceso
[BroControl] > deploy // Check
[BroControl] > status // Estado del proceso

Para ver la ejecución de BroControl, ver **Figura 12**.

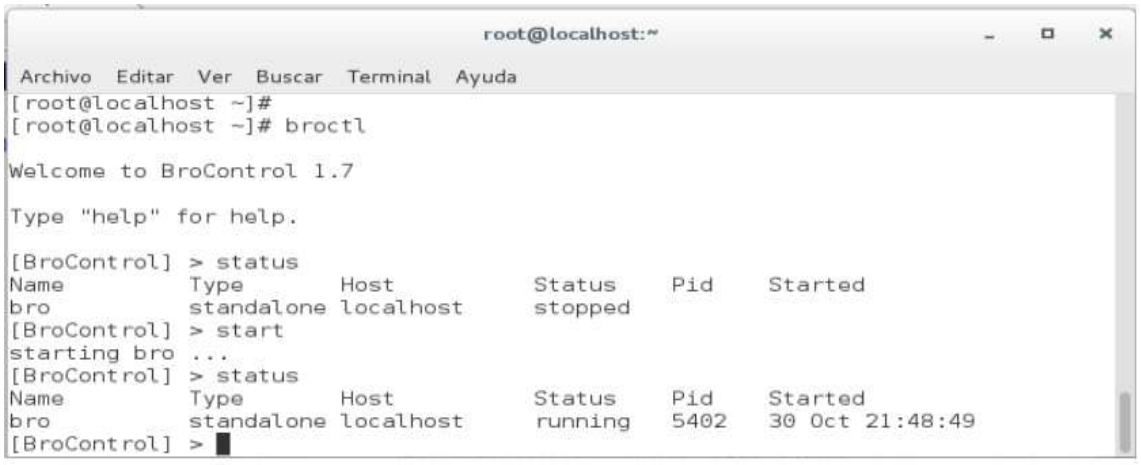

Figura 12: Ejecución de BroControl

Realizado por: Eduardo Arteaga

Una vez terminada las configuraciones de Bro, se realiza un ping al servidor y se procede a verificar los logs, a través del comando

tail –f conn.log.

La ubicación de los logs es: /usr/local/bro/logs, Ver **Figura 13**.

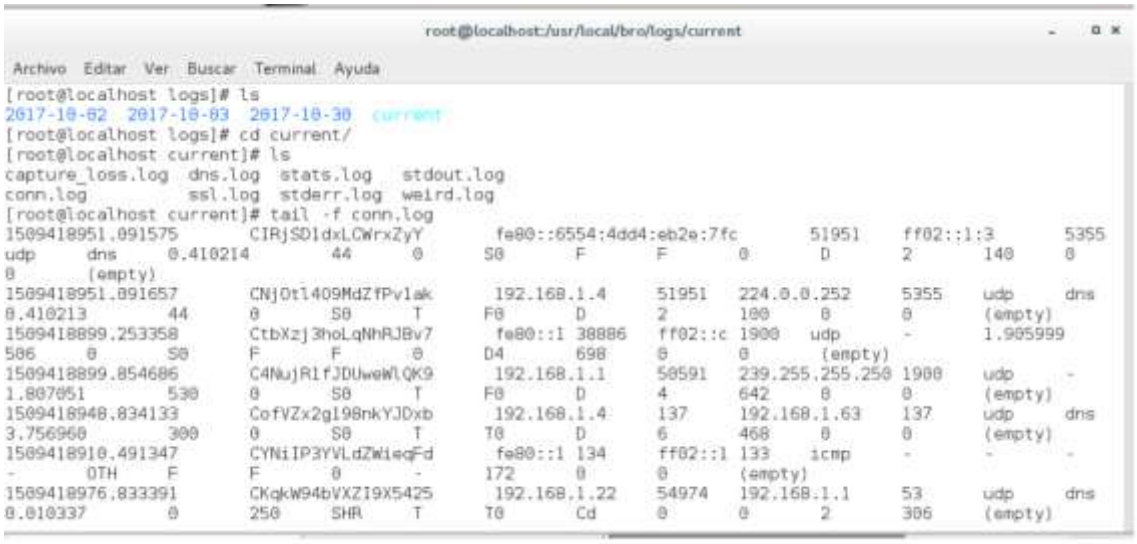

Figura 13: Exploración del log conn.log

Realizado por: Eduardo Arteaga

Anexo J. Esquema red de servidores de la Secretaría de Hidrocarburos

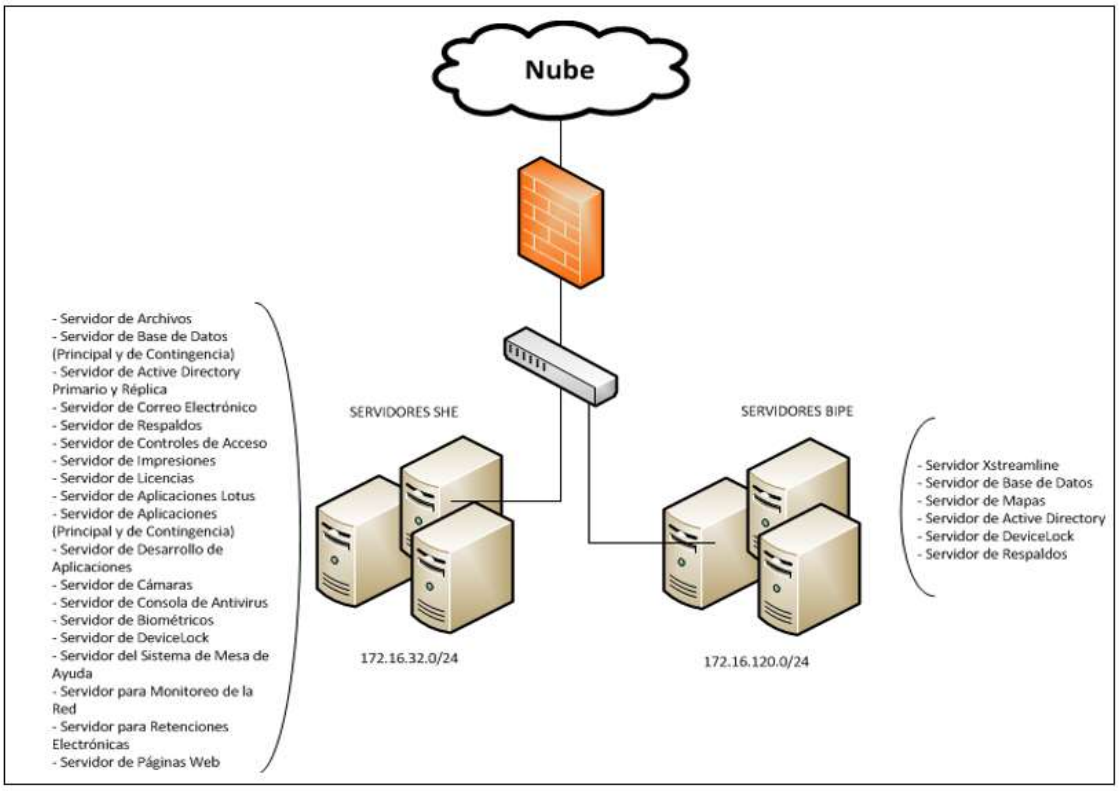

Figura 14: Esquema de la Red de Servidores

Fuente: Secretaría de Hidrocarburos

Realizado por: Eduardo Arteaga

Anexo K. Comparación de funciones que inciden directamente con la detección de intrusiones de los IDSes

Tabla 1: Comparación de funciones de que inciden directamente con la detección de intrusiones de los IDSes

Características /Descripción	Snort	Suricata
Altamente escalable Es la propiedad deseable del sistema, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.	Snort puede funcionar como: Sniffer: Permite ver a través de consola y en tiempo real el tráfico de la red. Registro de paquetes: Permite guardar los logs para su posterior análisis. NIDS: Generando alertas frente a coincidencias de patrones establecidos de tráfico anormal.	▪ Suricata tiene múltiples hilos. Permitiendo ejecutar una instancia y equilibrar la carga de procesamiento en cada procesador en un sensor que Suricata está configurado para usar. ▪ Esto permite la comodidad del hardware para alcanzar velocidades de 10 gigabits en el tráfico de la vida real sin sacrificando la cobertura del conjunto de reglas.
Protocolos de red Es la capacidad de las herramientas NIDS de detectar ciertos protocolos de red mediante palabras clave. Este criterio es importante debido a que representa: facilidad en el momento de escribir, entender o personalizar reglas de seguridad.	Un motor de detección puede aplicar a diferentes reglas en distintas partes de un paquete. Estas partes son las siguientes: ▪ Cabecera IP: Se aplica a las reglas de las cabeceras IP de un paquete. ▪ Cabecera de la capa de Transporte: Incluye las cabeceras de los protocolos UDP, TCP e ICMP. ▪ Cabecera de la capa de Aplicación: Incluye cabeceras de los protocolos FTP, SNMP, DNS y SMPT.	▪ Soporte para la decodificación de paquetes de: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, Ethernet, PPP, PPPoE, Raw, SLL, VLAN, QINQ, MPLS, ERSPAN. ▪ Decodificación de capa de aplicación de: HTTP, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP / CIP, DNP3, NFS, NTP. ▪ Soporte experimental para el desarrollo de analizadores en el lenguaje Rust, para una decodificación segura y rápida.

Gestiona distintos módulos salida Capacidad que consiste en procesar varios módulos de salida para las alertas generadas que permitan determinar mediante un previo análisis una intrusión como verdadera o falso positivo. Este parámetro se lo considera importante debido a que registra en log información referente a: cabeceras de los paquetes IP y protocolos encapsulados, datos de aplicaciones relevantes, paquetes IP en bruto.	Por su parte Snort hace uso de 8 tipos de salida para las alertas que son: alert_syslog, alert_fast, alert_full, alert_unixsock, log_tcpdump, csv, unified 2, log null. Snort es mono hilo, por lo tanto, procesador del sistema hardware no procesa más de unas tantas peticiones a la vez, ocasionando pérdida de paquetes.	▪ Registro de Eve, toda la alerta JSON y la salida de eventos. ▪ Guiones de salida Lua. ▪ Soporte Redis. ▪ Registro de solicitud HTTP. ▪ TLS handshake logging. ▪ Salida unificada2 - compatible con Barnyard2 (unified.alert). ▪ Alerta de registro rápido (fast.log). ▪ Registro de depuración de alertas: para escritores de reglas. ▪ Registro de tráfico usando registrador pcap. ▪ Soporte de preludio. ▪ Drop log - netfilter style log de paquetes caídos en modo IPS. ▪ Syslog - alert to syslog. ▪ Estadísticas - estadísticas del motor a intervalos fijos. ▪ Registro de archivos, incluida la suma de comprobación MD5. en formato JSON. ▪ Archivo extraído que se almacena en el disco. ▪ Registrador de solicitud / respuesta DNS, incluidos datos TXT. ▪ Rotación de registro basada en la señal. ▪ Registro de flujo.
Soporte IPV6 Es de importancia, debido a que, en la actualidad, varias instituciones, dentro de sus redes de información cuentan con soporte de IPV6.	▪ Soporte completo de IPv6.	▪ Soporte completo de IPv6.

Aplicaciones extendidas Se refiere a las aplicaciones externas en las que puedan ser incorporadas los NIDS y que proporcionen una representación visual de los datos de intrusión como son los NMS, SIM, SIEM.	Herramientas Open Source: ▪ Security onion (NMS). ▪ Ossim (SIEM). ▪ Prelude (SIM). ▪ Aanval (SIEM). ▪ AlienVault USM (SIEM). ▪ openIDS (NMS). Herramientas Comerciales ▪ Cisco ISE (SIEM). ▪ Splunk(SIEM).	Herramientas Open Source: ▪ Security onion (NMS). ▪ Ossim (SIEM). ▪ Prelude (SIM). ▪ Aanval (SIEM). ▪ AlienVault USM (SIEM). ▪ openIDS (NMS). Herramientas Comerciales. No hay integración.
Subherramientas para respuestas activas Se refiere a las herramientas adicionales que permitan al NIDS además de descartar el tráfico catalogado como muy peligroso, reconfigurar el firewall con el fin de bloquear el tráfico de la dirección origen del ataque por un periodo de tiempo.	Integración con: ▪ Iblock es un demonio de Linux que bloquea a los anfitriones infractores a través de iptables. ▪ SnortSam es un plugin ligero que permite a Snort trabajar conjuntamente con el firewall como: Checkpoint Firewall-1, Cisco PIX firewalls, Cisco Routers (using ACL's or Null-Routes), Former Netscreen, now Juniper firewalls, IP Filter (ipf), available for various Unix-like OS' entre otros.	Integración con: ▪ Snortsam. ▪ Iblock. Cabe mencionar que estos fueron construidos explícitamente para Snort y que Suricata se adapta a estas herramientas.
Reputación IP Funcionalidad que permite compartir información de direcciones IP que son de mala reputación con otras organizaciones y soluciones de seguridad, con la finalidad de disminuir los falsos positivos.	Preprocesador de reputación ip desde la versión 2.9.1, Este preprocesador generará alertas por cada paquete enviado o dirigido a una dirección IP incluida en dichas listas negras.	▪ Carga de grandes cantidades de datos de reputación basados en host. ▪ Coincidencia en datos de reputación en el lenguaje de reglas utilizando la palabra clave "iprep". ▪ Soporte de recarga en vivo. ▪ Admite rangos CIDR.

GeoIP Funcionalidad que permite generar archivos que contienen información de la procedencia de una alerta utilizando geo localización. Los cuales pueden ser visualizados con Google Earth o cualquier herramienta similar.	Soporte GeoIP.	Soporte GeoIP.

Fuente: Investigador

Realizado por: Eduardo Arteaga