MAESTRIA EN TECNOLOGÍAS DE LA INFORMACIÓN

TECNOLÓGICO
DE ESTUDIOS SUPERIORES
DE CUAUTITLÁN IZCALLI
TESCI

Diseño y optimización de reglas IDS Snort con Python.

TESIS

QUE PARA OBTENER EL GRADO DE:
MAESTRA EN TECNOLOGÍAS DE LA INFORMACIÓN.

PRESENTA:

VILLASEÑOR JIMENEZ ARTURO

DIRECTOR(A) DE TESIS:

M. en C. MARÍA DEL CONSUELO MACIAS GONZÁLEZ

AUTORIZACIÓN

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de esta tesis. En primer lugar, me gustaría dedicar unas palabras a mi familia, cuyo apoyo incondicional ha sido fundamental a lo largo de este arduo pero gratificante camino académico.
A mi pareja, por su paciencia, comprensión y apoyo emocional. Gracias por estar a mi lado en cada etapa de este proceso, por entender mis ausencias y por celebrar cada pequeño logro como si fuera propio. Tu amor y compañía han sido un refugio en tiempos de estrés y agotamiento.
Gracias, familia, por ser mi mayor fuente de inspiración y mi red de apoyo incondicional. Este logro es tan suyo como mío.

Con cariño y gratitud.

DEDICATORIA

Dedico esta tesis a mi familia, cuyo amor y apoyo incondicional han sido la piedra angular de mi vida y mis logros.

A mi pareja, Cinthya Pérez Aldana, por su paciencia, comprensión y amor inquebrantable. Gracias por estar a mi lado, por celebrar mis éxitos y por ser un refugio de paz y tranquilidad en tiempos de estrés y agotamiento.

A mis hijos Carlo Arturo y Marcos Alejandro Villaseñor Pérez, por los momentos de alegría y comprensión que han aligerado este viaje.

A todos aquellos que han creído en mí y me han apoyado de diversas maneras, mi más profundo agradecimiento. Este logro es tanto suyo como mío, y me siento profundamente honrado de compartirlo con ustedes.

ÍNDICE

LISTA DE ABREVIATURAS Y TABLA DE SÍMBOLOS

RESUMEN

La seguridad en las tecnologías de la información es de vital importancia derivado al gran volumen de datos y su clasificación que se maneja cada empresa. Una de las herramientas de seguridad que disponen las grandes y pequeñas compañías son los llamados Sistemas de Detección de Intrusos o IDS.

Por ello en esta tesis pretendo realizar el estudio de las reglas del Sistema de Detección de Intrusos (Snort) un software open source con mayor relevancia en el mercado, por lo cual esta tesis se queda en el diseño (Primera etapa), pero a lo largo del tiempo continuar con este proyecto.

En la primera etapa se identifica el diseñó una interfaz con Python 3.9 para generar reglas y conectarnos al IDS y poder actualizarlo. Mientras que en la segunda etapa se realiza un diseño de una red neuronal capaz de determinar qué tipo de amenaza se está presentado en tiempo real. Por lo anterior en la tercera etapa, se aborda el entrenamiento de la red neuronal con Ciberataque para la validación de su funcionamiento. Para la última etapa se incorpora la red neuronal a la interfaz para validar que una vez la red neuronal detecte en tiempo real el ataque, y pueda generar la regla correspondiente en el IDS de Snort.

En resumen, el diseñado es principalmente plasmar una herramienta sencilla y útil por lo que tendrá que ser muy liviana para ayudar a las pequeñas y medianas empresas a evitar la carga de las costosas licencias, que a menudo tienen implicaciones importantes para ellas. Se cree que el éxito y la sostenibilidad a largo plazo de esta iniciativa requieren la participación activa de una comunidad de desarrolladores con ideologías Open Source.

Palabras Clave:
Interfaz gráfica (GUI)
Python
Snort
Sistema de Detección de Intrusos (IDS)
Reglas de configuración
Seguridad informática
Monitoreo de red
Ciberseguridad
Automatización
Reducción de tiempos
Open Source
Ataques DoS
Desbordamiento de búfer
Escaneo de puertos
Captura de paquetes
Parámetros de origen y destino
Archivos .txt
Visualización de reglas
Administración de red
Aprendizaje automático
Evolución de IDS

ABSTRACT

Security in information technology is of vital importance due to the large volume of data and its classification that each company handles. One of the security tools available to large and small companies are called Intrusion Detection Systems or IDS.

Therefore in this thesis I intend to study the rules of the Intrusion Detection System (Snort) an open source software with greater relevance in the market, so this thesis remains in the design (First stage), but over time continue with this project.

The first stage identifies the design of an interface with Python 3.9 to generate rules and connect to the IDS and be able to update it. In the second stage, we designed a neural network capable of determining what type of threat is being presented in real time. Therefore, in the third stage, the training of the neural network with cyberattack is addressed for the validation of its operation. For the last stage, the neural network is incorporated to the interface to validate that once the neural network detects the attack in real time, it can generate the corresponding rule in the Snort IDS.

In summary, the designed is primarily to embody a simple and useful tool so it will have to be very lightweight to help small and medium sized companies avoid the burden of costly licenses, which often have significant implications for them. It is believed that the success and long-term sustainability of this initiative requires the active participation of a community of developers with Open Source ideologies.

INTRODUCCIÓN

Desarrollo de una Interfaz Gráfica en Python para la Configuración de Reglas en Snort: Una Apuesta por la Eficiencia en Seguridad Informática

Como estudiante de la Maestría en TICs, he podido constatar la creciente necesidad de herramientas accesibles que ayuden a fortalecer la seguridad en las redes. En este sentido, el diseño de una interfaz gráfica en Python para la configuración de reglas en Snort representa una solución práctica e innovadora que busca optimizar el proceso de detección de intrusiones. Este proyecto tiene como propósito facilitar la creación de reglas en el sistema IDS Snort, reduciendo los tiempos de configuración mediante una interfaz amigable, intuitiva y funcional.

El proyecto parte de un análisis del funcionamiento del IDS Snort, un sistema ampliamente reconocido por su eficacia y su carácter open source. Snort permite a los administradores de red monitorear el tráfico en tiempo real, identificar amenazas como ataques DoS, desbordamientos de búfer o escaneos de puertos, y actuar rápidamente mediante alertas. Sin embargo, su configuración puede llegar a ser compleja para usuarios con poca experiencia. Por ello, se propone desarrollar una GUI en Python que permita capturar parámetros como IP de origen y destino, puertos, protocolos y otros elementos clave para generar reglas de forma sencilla y visual.

Esta interfaz también integrará un visor que permitirá visualizar cómo se estructura la regla antes de guardarla en un archivo ".txt", lo cual facilita tanto su revisión como su implementación directa en Snort. Además, el sistema estará diseñado para trabajar sobre cualquier entorno de red compatible con Snort, brindando flexibilidad y escalabilidad.

Cabe resaltar que esta propuesta se enmarca en la evolución que han tenido los IDS, desde sus orígenes en los años 80 hasta la actualidad, donde la inteligencia artificial y el aprendizaje automático han comenzado a integrarse en la detección de amenazas. Hoy, herramientas como Snort no sólo identifican patrones, sino que también permiten adaptar sus reglas a nuevos vectores de ataque. Por ello, contar con herramientas que simplifiquen su uso es una necesidad real.

En conclusión, este proyecto busca no solo facilitar la administración de reglas en Snort, sino también fomentar una cultura de prevención en ciberseguridad desde un enfoque técnico y accesible.

CAPÍTULO 1. MARCO CONTEXTUAL

Objetivo general:

Diseñar una interfaz en Python con el propósito de configurar los parámetros requeridos, con la finalizad de reducir tiempos en la creación de reglas para el Sistema de Detección de Intrusos (Snort).

Objetivos particulares:

1. Se llevará a cabo un estudio sobre el IDS Snort y la configuración de las reglas en este sistema.
2. Se diseñará una interfaz gráfica (GUI) utilizando el lenguaje de programación Python.
3. La interfaz te permitirá capturar los parámetros de origen y destino, así como contar con un visualizador en pantalla para la configuración de la regla.
4. La interfaz te mostrara opción de guardar la regla en un archivo ".txt".

En ese momento, la configuración de los sistemas expertos en detección de intrusos era bastante compleja y requería un alto nivel de conocimiento para generar las reglas. Por esta razón, se solicitaban consultores expertos para facilitar dicha configuración o capacitación.

El diseño se trabajó principalmente como una herramienta sencilla que contaba con tres paneles. Cada panel tenía como objetivo capturar información relevante necesaria para la configuración esencial del funcionamiento de la regla en el IDS.

CAPÍTULO 2. MARCO TEÓRICO

Dentro de los conceptos abordados en la investigación, se encontraba el término inteligencia artificial (IA), el cual tenía múltiples interpretaciones. Se podía entender la conexión entre la IA y la seguridad informática como un vínculo significativo entre ambas, ya que la informática se ocupaba de examinar formas de crear máquinas con capacidades de inteligencia equivalentes.

La inteligencia artificial empleaba diversas técnicas matemáticas, estadísticas y algoritmos para el aprendizaje. El desarrollo de la IA se clasificaba en tres categorías: Superinteligencia, Inteligencia Artificial General, Inteligencia Artificial Restringida y Aprendizaje Automatizado, tal como se ilustraba en la figura siguiente. [1]

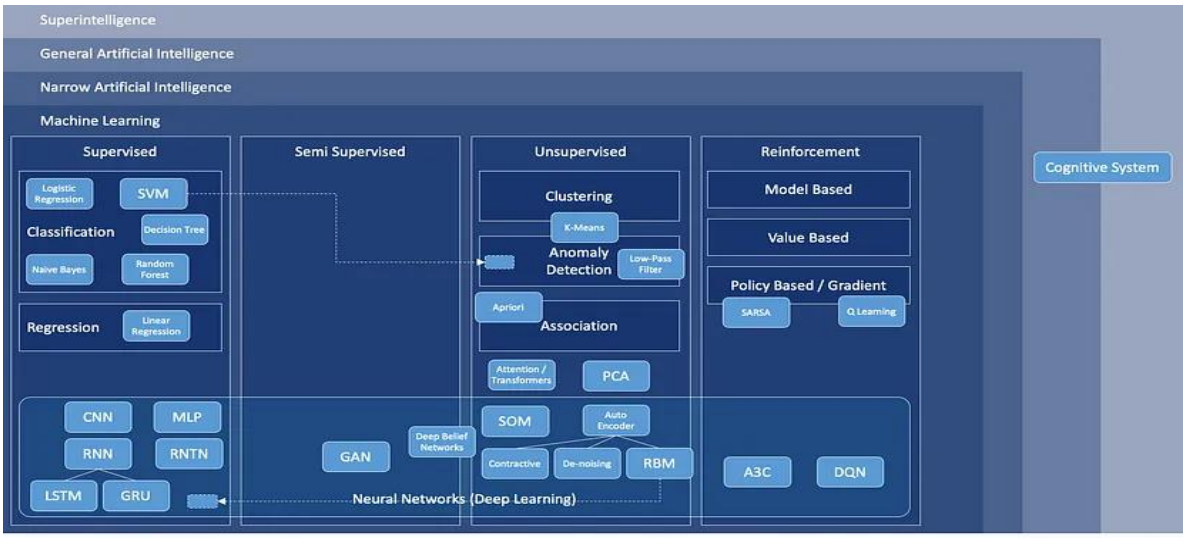

Ilustración 1"Descripción general de alto nivel de IA y aprendizaje automático" de Nils Ackermann tiene licencia Creative Commons CC BY-ND 4.0

Otra manera de referirse a las categorías de los sistemas de inteligencia artificial (IA) era para comprender, aprender y realizar cualquier tarea inteligente que los humanos pudieran llevar a cabo. La IA poseía una amplia gama de habilidades cognitivas y tenía la capacidad de adaptarse a diversas situaciones, similar a la mente humana.

[1] (Science, 2018)

La inteligencia artificial era un subcampo de la informática que se centraba en construir sistemas capaces de realizar tareas que requerían inteligencia humana. Este campo combinaba la informática, la estadística, las matemáticas y otras disciplinas para crear algoritmos y modelos que pudieran aprender, planificar, razonar o resolver problemas de forma independiente.

Se tomó como base una Red Neuronal, que es un tipo de IA dentro del aprendizaje automático (Machine Learning). Se contaba con una variedad de técnicas para resolver problemas en el ámbito del aprendizaje automático, como se podía observar en la figura siguiente.:[2]

Ilustración 2 Aplicaciones de Machine Learning (Autor Trycore)

En el ámbito de la seguridad informática, se incluía el hecho de que los recursos del sistema de información de la organización (hardware o software) se utilizaran dentro de una arquitectura específica, y que la información contenida en ellos, así como sus cambios, fueran accesibles únicamente para personas autorizadas y dentro de los límites establecidos.

Se podía inferir una definición de seguridad informática como el respeto a la confidencialidad, integridad y disponibilidad, es decir, los tres pilares de la seguridad informática. La confidencialidad requería que solo las personas autorizadas tuvieran acceso a la información. La integridad aseguraba que la información permaneciera intacta en caso de un accidente o un intento malicioso, lo que significaba que solo podía ser modificada de manera controlada por personas autorizadas. Por último, la disponibilidad garantizaba que los sistemas informáticos siguieran funcionando

[2] https://www.dropbox.com/sh/d6plg2q4oydvpcs/AAABDyuHyyKPG7iK7QV5m5MUa/Archivos%20Infolibros%20ES/Temas/390%20Inteligencia%20Artificial/04.%20Inteligencia%20Artificial%20autor%20Facultad%20de%20Ingenieria%2C%20UNAM.pdf?dl=0

sin obstaculizar el acceso, proporcionando a los usuarios autorizados los recursos que necesitaban cuando los necesitaban.

Dentro de los elementos más importantes de la seguridad informática se encontraban las políticas de seguridad. En cualquier entorno empresarial donde existieran datos o información que debían protegerse, no solo era crucial contar con herramientas de código abierto disponibles para salvaguardar esa información, sino también con una infraestructura adecuada.

Las políticas de seguridad se implementaban como una herramienta organizativa que proporcionaba un estándar para la protección de datos, sensibilizando a toda la organización sobre la importancia de la información e incorporando los servicios críticos que permitían a las empresas crecer y mantenerse competitivas.

Historia de la Evolución

La inteligencia artificial y la estadística constituían la base del aprendizaje automático. El libro *The Elements of Statistical Learning* (Hastie, Tibshirani y Friedman, 2009) era considerado el más influyente en cuanto a modelos de aprendizaje estadístico, proporcionando un marco teórico para su comprensión y aplicación.

Aplicaciones en el Mundo Real

El aprendizaje automático había demostrado ser muy beneficioso en diversos entornos prácticos. La capacidad de analizar grandes volúmenes de datos facilitaba el desarrollo de modelos predictivos para el diagnóstico temprano de enfermedades en el ámbito médico (Obermeyer & Emanuel, 2016). Esta técnica había mejorado significativamente la precisión diagnóstica y había provocado avances revolucionarios en la atención sanitaria.

En el ámbito financiero, se ofrecía una visión integral de los algoritmos de aprendizaje automático y las estrategias de inversión en el libro *Advances in Financial Machine Learning* (López de Prado, 2018). Los inversores podían utilizar modelos predictivos basados en datos históricos para mejorar su gestión de riesgos y su capacidad de toma de decisiones.

Cuestiones de Ética y Desafíos

A medida que el aprendizaje automático se volvía omnipresente, también surgían preocupaciones éticas y sociales que debían ser consideradas. Según la investigación presentada en "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification", publicada por Buolamwini y Gebru (2018), la inteligencia artificial y el aprendizaje automático podían perpetuar los sesgos existentes en los datos. En este trabajo, se argumentaba que los algoritmos debían

tener en cuenta tanto factores éticos como sociales para prevenir la discriminación y promover la igualdad.

Ziad Obermeyer (2016) señalaba que la ciencia se encargaba de realizar pedidos basándose en datos. En lugar de depender de técnicas de programación para resolver necesidades específicas, el aprendizaje automático buscaba desarrollar algoritmos genéricos capaces de extraer patrones de diversas fuentes de datos. La clasificación de números e imágenes escritas a mano, como distinguir entre perros y gatos en un programa de aprendizaje automático, se basaría en la presencia de algoritmos especializados que clasifican los datos etiquetados.

La idea de que el aprendizaje automático es una tarea simple podría ser errónea, ya que el científico de datos debía llevar a cabo tareas específicas, como descubrir la fuente de los datos, desensamblarlos, decodificar datos fuertemente vinculados (o identificar fuentes de sesgo) y realizar normalizaciones para corregir cualquier información sesgada.

Deep Learning

Las redes neuronales profundas eran estructuras inspiradas en el funcionamiento del cerebro humano, compuestas por múltiples capas que permitían a la máquina aprender de grandes volúmenes de datos. Estas redes estaban formadas por nodos conocidos como neuronas, organizadas en capas que procesaban la información de manera jerárquica a medida que esta avanzaba a través de ellas.

De forma jerárquica y no lineal, a medida que la información pasaba por las capas, la red ajustaba automáticamente sus pesos internos mediante un proceso conocido como propagación, lo que mejoraba su capacidad para hacer predicciones o tomar decisiones. Un ejemplo real de aprendizaje automático era el reconocimiento de voz en asistentes virtuales como Siri y Google.

Estos sistemas utilizaban redes neuronales profundas para procesar el habla natural, entender la intención del usuario y proporcionar respuestas o realizar acciones. El reconocimiento de voz requería el análisis de patrones en el audio y su transformación en texto o comandos, aprendiendo de una gran cantidad de ejemplos para mejorar la precisión.

La mayoría de los métodos de aprendizaje utilizaban una arquitectura de redes neuronales, razón por la cual los modelos de aprendizaje profundo a menudo se denominaban redes neuronales profundas.

El término "profundo" a menudo se refería a la cantidad de capas ocultas en una red neuronal. Mientras que las redes neuronales tradicionales contenían solo dos o tres capas ocultas, las redes profundas podían comenzar con 50 capas y llegar a tener más de 150 capas.

Los modelos de aprendizaje profundo se entrenaban utilizando conjuntos de datos ricos y etiquetados, y las arquitecturas de redes neuronales se entrenaban directamente sobre los datos sin necesidad de extracción manual de características.

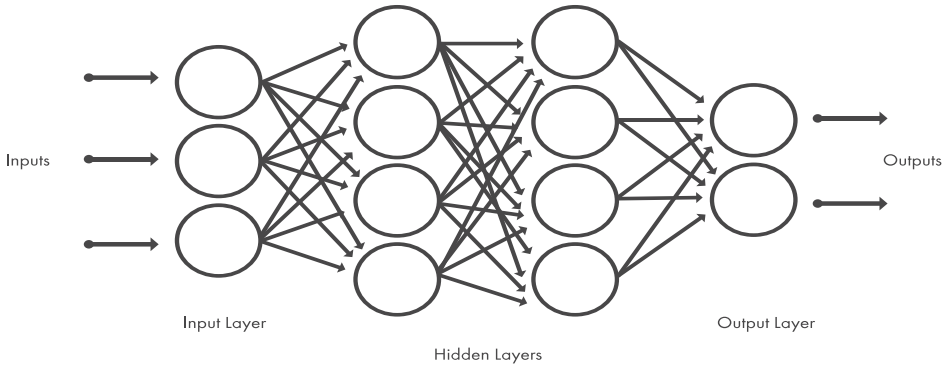

Ilustración 2 Las redes neuronales están organizadas en capas que constan de muchos nodos interconectados.

Gracias a los avances que permitió el Deep Learning en las técnicas de aprendizaje automático, se logró que las redes neuronales profundas, con sus múltiples capas, realizaran automáticamente la extracción de características de la información suministrada como parámetros de entrada.

Al utilizar una red neuronal profunda, se podía determinar si en una imagen de entrada había un perro o un gato. Para ello, se proporcionaba a la red una gran cantidad de imágenes de diferentes razas de perros, así como una cantidad similar de imágenes de gatos. A este proceso se le conocía como entrenamiento de la red neuronal profunda. Una vez que la red neuronal estaba preen trenada, era capaz de distinguir cualquier imagen de entrada como un perro o un gato con un 97% de precisión. En contraste, en el proceso tradicional de aprendizaje automático, era necesario extraer las características manualmente, lo que resultaba en un bajo nivel de precisión.

En el siguiente caso, se entrenó la red neuronal con imágenes de motos deportivas.

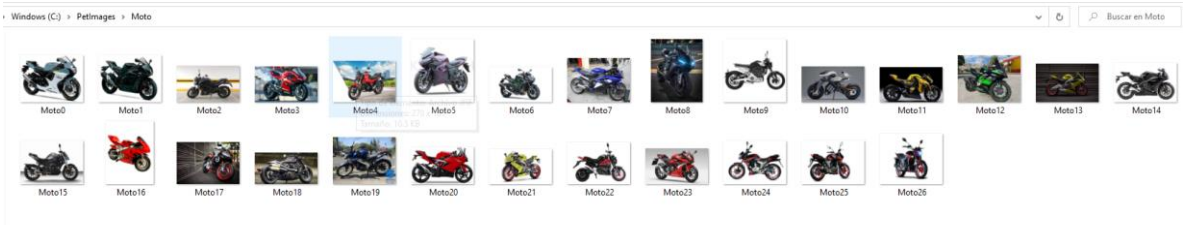

Ilustración 3 Base de imágenes para entrenar una red neuronal

En la siguiente imagen se podían observar tres paneles. En el primer panel se mostraba el código en Python, en el segundo panel se visualizaban gráficos que

recorrían cada imagen, y en el tercer panel de la terminal se generaba la salida correspondiente a cada imagen.

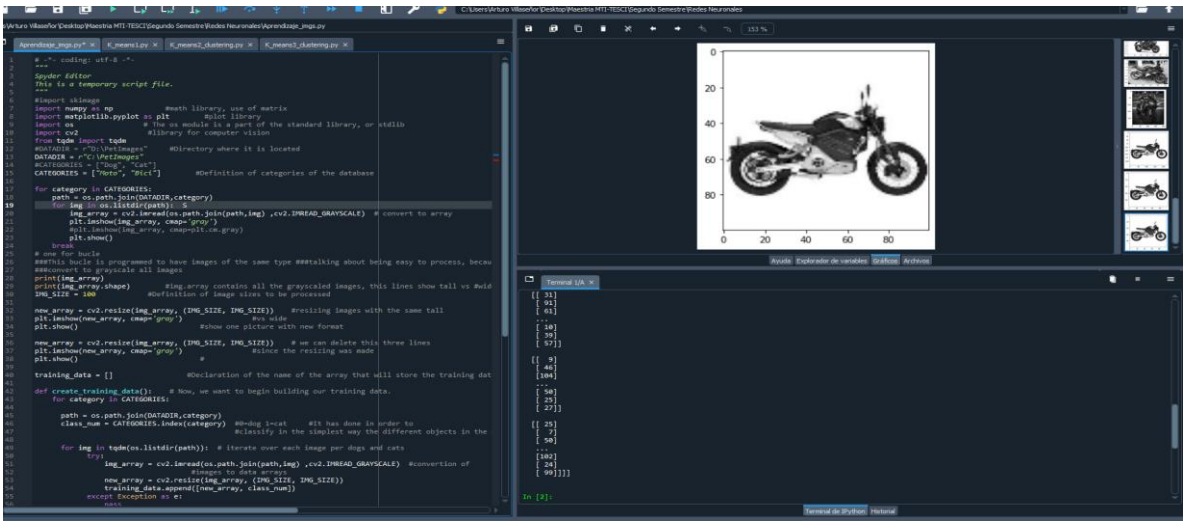

Ilustración 4 Red neuronal en proceso de entrenamiento

Snort Sistema de detección de Intrusos (IDS)

El Pasado de los Sistemas de Detección de Intrusos (IDS). Los sistemas de detección de intrusos, conocidos por su acrónimo en inglés IDS, habían experimentado mejoras notables desde sus primeras implementaciones.

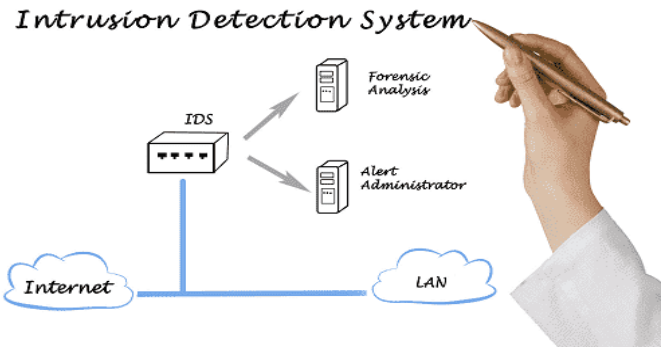

Ilustración 5 Sistemas de Detección de Intrusos (Autor - comodo.com)

La creciente necesidad de proteger las redes informáticas impulsó la evolución hacia sistemas más avanzados y adaptativos. En la década de 1980, los IDS eran todavía rudimentarios, y su principal objetivo consistía en identificar patrones de tráfico sospechosos o ataques ya conocidos. Varios de los primeros sistemas utilizaban firmas o patrones específicos para reconocer riesgos potenciales.
Durante la década de 1990, se introdujeron los primeros IDS basados en red y en host. Los IDS basados en host se encargaban de monitorear sistemas individuales,

mientras que los IDS vinculados a la red analizaban el tráfico general y detectaban cualquier anomalía.
A medida que la conciencia sobre la seguridad aumentaba, los IDS comenzaron a adoptar métodos más sofisticados, como la detección de anomalías y el análisis heurístico. A principios de los años 2000, estos sistemas evolucionaron hacia lo que se conoció como sistemas de prevención de intrusiones (IPS), capaces de detectar y monitorear activamente las amenazas.

Evolución de los Sistemas de Detección de Intrusos (IDS). Con el paso del tiempo, la correlación de eventos adquirió mayor relevancia para mejorar la precisión y reducir los falsos positivos. En el año 2010, la aparición de la inteligencia artificial y el aprendizaje automático impactó significativamente en el avance de los IDS. La detección de comportamientos inusuales se utilizó cada vez más para identificar amenazas desconocidas. En ese momento, los IDS comenzaron a emplear una combinación de tecnología de firmas, análisis de comportamiento, inteligencia artificial (IA) y análisis de amenazas para enfrentar un panorama de ciberseguridad en constante evolución.

En el ámbito de la tecnología de la información, la seguridad del sistema se convirtió en uno de los aspectos más cruciales a considerar. Sin una estrategia sólida de seguridad, los datos, que son una de las partes más importantes y sensibles del sistema, podían verse gravemente comprometidos. Los componentes que ayudaban a los administradores a abordar la seguridad incluían sistemas de detección de intrusos (IDS). En la actualidad, existen diversos programas que permiten implementar IDS, destacando Snort por su flexibilidad, ya que ofrece la capacidad de almacenar registros en archivos de texto o en bases de datos.
La incorporación de IA al IDS (Snort) se reflejó en el G2 Grid® para sistemas de prevención y detección de intrusos (IDPS). Por esta razón, se optó por este IDS debido a su naturaleza Open Source.

Se recomienda consultar el G2 Grid® para conocer los mejores productos en sistemas de prevención y detección de intrusiones (IDPS). G2 califica tanto productos como proveedores basándose en las reseñas recopiladas de su comunidad de usuarios, así como en datos agregados provenientes de fuentes en línea y redes sociales. Estos puntajes se representan en su G2 Grid® patentado, que puede utilizarse para comparar productos, agilizar el proceso de compra e identificar rápidamente las mejores opciones según las experiencias de los usuarios.

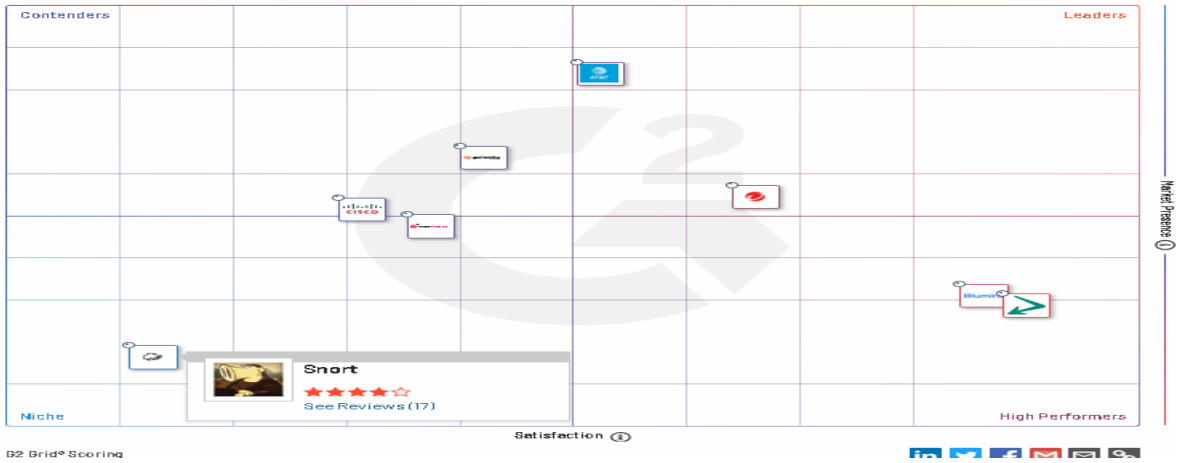

Ilustración 6 G2 Grid® for Intrusion Detection and Prevention Systems (IDPS)

Funciones y Clasificaciones de los Sistemas de Detección de Intrusos (IDS). En un contexto más especializado dentro de la seguridad informática, el IDS cumplía funciones similares a las de un sniffer de red. Sin embargo, el IDS había pasado por procesos de optimización que le permitieron seleccionar el tráfico de red deseado. Gracias a esta funcionalidad, podía analizar exclusivamente las reglas configuradas sin afectar el rendimiento del sistema. Esto facilitaba la recolección de datos para el análisis, lo que a su vez permitía generar nuevas reglas basadas en los resultados obtenidos.

"Un sniffer es una herramienta de software o hardware que permite al usuario supervisar su tráfico en Internet en tiempo real y capturar todo el tráfico de datos que entran y salen de su equipo."[3]

Existían diferentes clasificaciones de los IDS, como los Network IDS (NIDS) y los Host IDS (HIDS). En este texto, se centrará exclusivamente en los NIDS.
Snort se destacaba como uno de los líderes en este tipo de tecnología, según la opinión del autor basada en el "G2 Grid® for Intrusion Detection and Prevention Systems (IDPS)". Esta herramienta era preferida por ser Open Source, lo que la hacía accesible y flexible para su implementación.

Introducción a Snort

Desarrollo y Funcionalidades de Snort. Esta herramienta fue desarrollada por Marty Roesch en 1998, inicialmente como un sniffer o analizador de protocolos. Con el tiempo, Snort incorporó diversas funcionalidades adicionales y, en la actualidad, cuenta con un preprocesador, un complemento de base de datos, múltiples opciones para enviar alertas, diferentes esquemas de evaluación de paquetes, un conector para Windows, gráficos en la consola de administración, entre otros.

[3] https://www.avast.com/es-es/c-sniffer

Snort es un software gratuito y de código abierto que también puede funcionar como rastreador de paquetes para el monitoreo del sistema en tiempo real. Los administradores de red podían utilizarlo para supervisar todos los paquetes entrantes y determinar cuáles eran perjudiciales para el sistema.

Se basa en una herramienta de captura de paquetes disponible en la biblioteca, lo que permite crear y ejecutar reglas de manera sencilla. Estas reglas se pueden implementar en cualquier sistema operativo y en diversos entornos de red. La principal razón por la que este IDS se volvió más popular que otros fue su naturaleza gratuita y de código abierto, lo que permitía a cualquier usuario utilizarlo según sus necesidades.

Snort se fundamentaba en libpcap, una biblioteca ampliamente utilizada para analizadores y rastreadores de tráfico TCP/IP. Al analizar protocolos y buscar contenido específico, Snort detectaba vectores de ataque que incluían ataques de denegación de servicio, desbordamientos de búfer, ataques CGI, escaneos de puertos ocultos y sondas SMB. Cuando se identificaba actividad sospechosa, Snort enviaba alertas en tiempo real al registro del sistema, a un archivo separado denominado "alerta" o a una ventana emergente.

Tipos de Ataques Detectados por Snort
Ataque de Denegación de Servicio (DoS): Este tipo de ataque consistía en inundar una red o servicio con un gran volumen de solicitudes, lo que provocaba la sobrecarga de recursos y dejaba el servicio inutilizable.

Desbordamiento de Búfer: Esta vulnerabilidad ocurría cuando un programa recibía más datos de los que podía manejar. Un atacante podía aprovechar esta debilidad enviando datos maliciosos que superaban la capacidad del búfer, causando daños en la memoria y permitiendo la ejecución de código malicioso.

Ataque CGI: Los ataques a través del Common Gateway Interface (CGI) apuntaban a vulnerabilidades en programas y scripts CGI utilizados por servidores web. Un atacante podía manipular la entrada para ejecutar comandos no autorizados en el servidor.
Escaneo de Puertos Ocultos: Esta técnica era utilizada por atacantes para identificar puertos abiertos que no eran fácilmente detectables mediante métodos estándar. En lugar de buscar puertos comunes, realizaban análisis más profundos para encontrar puertos abiertos no documentados.

Sonda SMB: Esta técnica implicaba escanear sistemas utilizando el protocolo SMB, empleado para compartir archivos e impresoras en una red. Un atacante podía enviar solicitudes a puertos específicos para identificar sistemas vulnerables y posibles puntos de entrada.
Snort se convirtió así en una herramienta esencial para la detección y prevención de intrusiones dentro del ámbito de la seguridad informática.

Características principales

Estas son las principales características de Snort:

- Monitor de tráfico en tiempo real
- Registro de paquetes
- Análisis de protocolo
- Coincidencia de contenido
- Huellas digitales del SO
- Puede instalarse en cualquier entorno de red.
- Crea registros
- Fuente abierta
- Las reglas son fáciles de implementar
- Snort funciona en Windows y en Linux.

Componentes de Snort

Snort consiste de cuatro componentes como base:

- Sniffer
- Procesadores
- Motor de detección
- Salidas

A continuación, se representa la funcionalidad con un esquema:

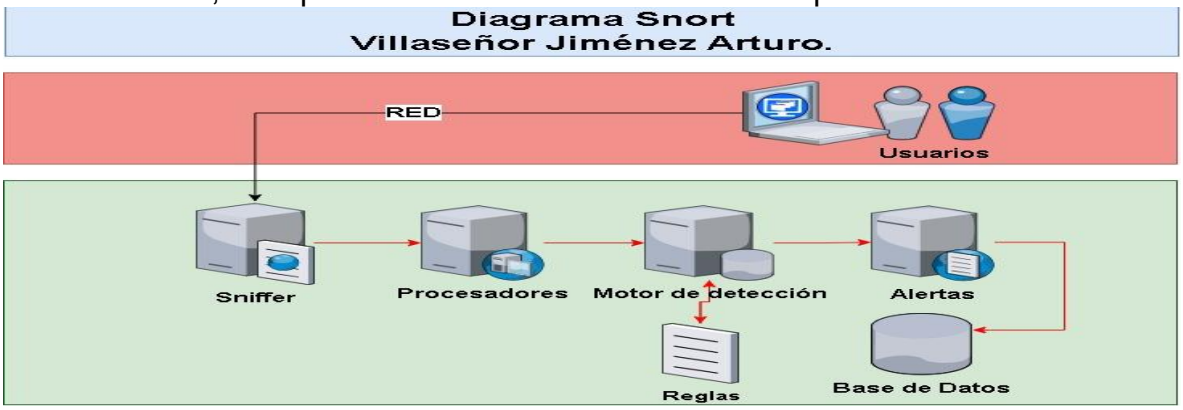

Ilustración 7 Funcionamiento grafico de Snort.

El primer componente era el Sniffer, encargado de capturar todos los paquetes para que otros componentes pudieran utilizarlos. Su función consistía en identificar el protocolo subyacente utilizado en cada paquete, así como determinar la ubicación y el tamaño de los datos contenidos en él. Esta información era luego empleada por los componentes posteriores. Además, el descodificador también buscaba anomalías en los encabezados que pudieran generar advertencias.
El segundo componente era el preprocesador. Estos elementos actuaban como complementos que organizaban o modificaban los paquetes. Por lo tanto, cada servicio (como HTTP o FTP) contaba con un preprocesador específico diseñado para verificar excepciones relacionadas con ese servicio. Su objetivo era dificultar el engaño a los mecanismos de detección.

Ejemplos de estas funciones incluían la decodificación de URI, la desfragmentación de paquetes, la detección de escaneos de puertos y la identificación de anomalías en los paquetes ARP, como la suplantación de identidad ARP.
El tercer componente, y el más importante, era el motor de detección. Este se encargaba de identificar cualquier intrusión en los paquetes. Lo hacía combinando los conjuntos de reglas especificadas en un archivo de configuración y aplicándolas a cada paquete recibido. Si un paquete coincidía con una regla, se ejecutaba la acción definida para esa regla o se descartaba el paquete.

El cuarto componente eran los sistemas de registro y alertas. Este era responsable de generar una alerta si un paquete coincidía con alguna regla. Las notificaciones y el contenido generado por este componente podían personalizarse mediante archivos de configuración. Si un paquete activaba múltiples reglas, solo se generaba la alerta correspondiente al nivel más alto.

Finalmente, cuando se generaba una alerta o un registro, este pasaba a través del módulo de salida. El propósito de este componente era controlar el tipo de señal de salida producida por el sistema plug-in, ofreciendo al usuario una gran flexibilidad y opciones de personalización.

Esto podía incluir registros simples en bases de datos, envío de capturas SNMP, generación de informes XML o incluso envío de alertas a través de sockets UNIX, lo que permitía modificar dinámicamente la configuración de la red (firewall o enrutador).
En cuanto al inicio del almacenamiento en bases de datos por parte de Snort, se utilizaban diferentes tipos, siendo MySQL la más estable en sus inicios; actualmente, se recomienda MariaDB o Postgres.

Reglas en Snort

Como se mencionó anteriormente, todos los componentes utilizaban estas reglas para identificar anomalías en los paquetes. Las reglas podían incorporarse a los encabezados de las capas de transporte y de red, así como a los paquetes de la capa de aplicación (como FTP, HTTP), entre otros. Además, era posible aplicarlas a paquetes individuales.

Cada regla constaba de dos partes: un encabezado que proporcionaba instrucciones sobre cómo actuar ante paquetes coincidentes, indicando el tipo de paquete (TCP, UDP, etc.) y las direcciones IP de origen y destino, junto con los números de puerto. La última sección, denominada Configuración de reglas, especificaba qué paquetes debían ser considerados como coincidentes, y la regla general adoptaba la siguiente forma:

- Rule Header.
- Message.
- Flow.
- Detection.

- Metadata.
- References.
- Classification.
- Signature ID.

Análisis de las Reglas de Snort en el Laboratorio Virtua, para mostrar y profundizar en las reglas de Snort, en su entorno de Laboratorio Virtual, se utilizó la distribución Ubuntu 22.04. Las reglas podían encontrarse en la siguiente ruta: /etc/snort.

```
snort@snort-virtual-machine:/etc/snort/rules$ ls
attack-responses.rules          community-smtp.rules           icmp.rules        shellcode.rules
backdoor.rules                  community-sql-injection.rules  imap.rules        smtp.rules
bad-traffic.rules               community-virus.rules          info.rules        snmp.rules
chat.rules                      community-web-attacks.rules    local.rules       sql.rules
community-bot.rules             community-web-cgi.rules        misc.rules        telnet.rules
community-deleted.rules         community-web-client.rules     multimedia.rules  tftp.rules
community-dos.rules             community-web-dos.rules        mysql.rules       virus.rules
community-exploit.rules         community-web-iis.rules        netbios.rules     web-attacks.rules
community-ftp.rules             community-web-misc.rules       nntp.rules        web-cgi.rules
community-game.rules            community-web-php.rules        oracle.rules      web-client.rules
community-icmp.rules            ddos.rules                     other-ids.rules   web-coldfusion.rules
community-imap.rules            deleted.rules                  p2p.rules         web-frontpage.rules
community-inappropriate.rules   dns.rules                      policy.rules      web-iis.rules
community-mail-client.rules     dos.rules                      pop2.rules        web-misc.rules
community-misc.rules            experimental.rules             pop3.rules        web-php.rules
community-nntp.rules            exploit.rules                  porn.rules        x11.rules
community-oracle.rules          finger.rules                   rpc.rules
community-policy.rules          ftp.rules                      rservices.rules
community-sip.rules             icmp-info.rules                scan.rules
snort@snort-virtual-machine:/etc/snort/rules$ pwd
/etc/snort/rules
```

Ilustración 8 Listado de Reglas dentro nuestro IDS Snort

En el laboratorio, se contaba con la versión 2.9.15.1 GRE del IDS Snort, tal como se ilustraba en la imagen siguiente.

```
snort@snort-virtual-machine:/etc/snort$ sudo snort -v
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "ens33".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11
```

Ilustración 9 Pantalla de la versión del IDS Snort

Análisis de una Regla Configurada en Snort, se presentó una regla configurada extraída del archivo denominado snort3-browser-chrome.rules, dentro del sistema Snort (IDS), con el propósito de explicar la estructura de las reglas.

```
EXAMPLE
Rule Header:    alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (
Message:        msg:"BROWSER-CHROME Google Chrome GURL cross origin bypass attempt";
Flow:           flow:to_client,established;
Detection:      file_data; content:"|3C|iframe",nocase;
                    content:"src=|22|https|3A 2F 2F|www.google.com|2F|accounts|2F|ManageAccount?hl=fr|22|", within 100,nocase;
                    content:"window.open|28 27|",within 500;
                    content:"alert|28|document.cookie|29 27|",within 75;
                    pcre:"/window\.open\x28\x27[\w\W]{0,35}\x3aalert\x28document\.cookie\x29\x27/ims";
Metadata:       metadata:policy max-detect-ips drop,policy security-ips drop; service:http;
References:     reference:bugtraq,39813; reference:cve,2010-1663;
Classification: classtype:attempted-user;
Signature ID:   sid:16667; rev:13;
                                                                    )
```

Ilustración 10 Regla obtenida del archivo snort3-browser-chrome.rules de Snort

Rule Header (Encabezado de la Regla): En el encabezado de la regla se incluía la acción correspondiente, el protocolo utilizado, las direcciones IP de origen y destino, la máscara de red, así como la información sobre los puertos de origen y destino.

Alert: La acción de alerta (opcional) era el primer componente de la regla. Esta acción indicaba a Snort qué hacer al encontrar una solicitud que cumplía con los criterios establecidos por la regla, generalmente generando una alerta.
TCP: El siguiente campo de la regla correspondía al protocolo. En ese momento, Snort analizaba cuatro protocolos en busca de comportamientos sospechosos: TCP, UDP, ICMP e IP.
$EXTERNAL_NET: Esta era una variable de Snort que podía contener literalmente la dirección de origen.
$HTTP_PORTS: Esta también era una variable de Snort que podía representar los puertos de origen.
->: El operador de dirección indicaba la dirección u orientación del tráfico en relación con la regla.
$HOME_NET: Esta era otra variable de Snort que podía contener literalmente la dirección de destino.
Any: Esta variable de Snort podía representar el puerto de destino o contener un valor literal.

Opciones de Reglas

El motor de detección de intrusiones de Snort se fundamentaba en el uso de opciones de reglas, que combinaban facilidad de uso, potencia y flexibilidad. Snort diferenciaba las opciones de regla mediante un punto y coma (;), mientras que la separación de palabras clave y argumentos se realizaba utilizando dos puntos (:).
Mensaje: Los mensajes podían ser tanto confidenciales como significativos, y generalmente contenían información sobre lo que detectaba y definía la regla. La opción correspondiente a cada regla indicaba a Snort qué mostrar cuando se producía una coincidencia. Esta cadena de texto era simple, es decir, sin formato, y debía ser muy clara.

Flujo: Para que la regla funcionara correctamente, era necesario especificar la dirección en la que debía fluir el tráfico de la red. La palabra clave "flow" se utilizaba junto con el reensamblaje del flujo TCP, permitiendo aplicar las reglas solo a direcciones específicas del tráfico.
Referencias: Las palabras clave de referencia permitían incluir texto o enlaces a fuentes externas dentro de las reglas, ayudando a profundizar el contexto de los mensajes.
Classtype: La palabra clave "classtype" determinaba cómo Snort informaba o compartía los resultados de un ataque exitoso.
sid/rev: El ID dentro de Snort debía especificarse como un identificador único para cada regla. Esta información facilitaba que los complementos de salida definieran las reglas fácilmente y debía usarse junto con la palabra clave "rev" y el número de revisión.
Content: Esta característica era importante porque permitía a los usuarios establecer reglas que buscaban contenido específico en las cargas útiles de los paquetes y activar respuestas basadas en esos datos. Los datos adicionales podían incluir texto plano y datos binarios.

- **Distance / Offset**: Estas palabras clave permitían a los autores de las reglas especificar si la búsqueda comenzaba desde el inicio de la carga útil o desde el comienzo de la coincidencia del contenido.

- **Within / Depth**: Estas palabras clave permitían a cada autor determinar hasta qué punto buscar desde el final de la coincidencia anterior y, una vez encontrada una coincidencia, hasta qué punto buscar nuevamente.

Pcre: La palabra clave "pcre" permitía escribir reglas utilizando expresiones regulares compatibles con el lenguaje Perl, lo que posibilitaba coincidencias más complejas en comparación con las simples coincidencias de contenido.
Byte Test: La opción "byte_test" permitía que cada regla verificara una cantidad específica de bytes con un valor binario determinado.

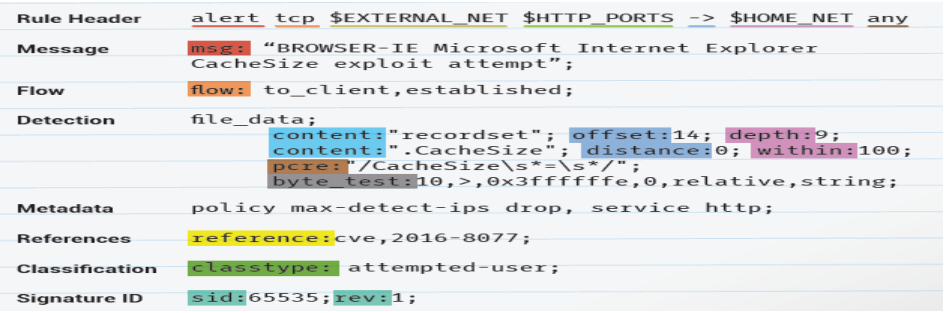

Ilustración 11 Ejemplo de una regla de snort propiedad de SNORT.ORG

Para comprender los beneficios de un encabezado de regla de este tipo, era necesario considerar dos reglas específicas que se enfocaban en detectar "clave_encriptación_secreta" en paquetes de datos. La primera regla centraba su

búsqueda en cadenas de caracteres presentes en paquetes enviados a través de los protocolos HTTP e IMAP dirigidos a un cliente específico.
Por otro lado, la segunda regla tenía como objetivo identificar la misma cadena en los paquetes SMTP que se redirigían a cualquier servidor SMTP. Este enfoque dual ofrecía una cobertura integral en diferentes protocolos y dominios, lo que mejoraba la eficiencia del sistema de detección.

```
alert tcp $EXTERNAL_NET [80,143] -> $HOME_NET any
(
    msg:"MALWARE-OTHER Win.Ransomware.Agent payload download attempt";
    flow:to_client,established;
    file_data; content:"secret_encryption_key",fast_pattern,nocase;
    service:http, imap;
    classtype:trojan-activity;
    sid:1;
)
alert tcp $EXTERNAL_NET any -> $SMTP_SERVERS 25
(
    msg:"MALWARE-OTHER Win.Ransomware.Agent payload download attempt";
    flow:to_server,established;
    file_data; content:"secret_encryption_key",fast_pattern,nocase;
    service:smtp;
    classtype:trojan-activity;
    sid:2;
)
```

Ilustración 12 Ejemplo de reglas en Snort, propiedad de Snort.org

Sin embargo, era posible combinar este par de reglas en una única alerta utilizando el formato de regla de archivo. De este modo, Snort recibiría instrucciones para buscar la presencia de "secret_encryption_key" en cualquier archivo encontrado en la red, sin importar su origen, destino o servicio asociado. Esta combinación de criterios ofrecía un enfoque más integral, permitiendo una detección más amplia y eficaz en los entornos de red.

```
alert file
(
    msg:"MALWARE-OTHER Win.Ransomware.Agent payload download attempt";
    file_data;
    content:"secret_encryption_key",fast_pattern,nocase;
    classtype:trojan-activity;
    sid:3;
)
```

Ilustración 13 Ejemplo de regla mejorada de la Ilustración 13, propiedad de Snort.org

CAPÍTULO 3. MARCO METODOLÓGICO

La metodología que se utilizó como base fue Scrum, debido a su capacidad para gestionar proyectos de manera efectiva y adaptable.

La estructura clara y flexible de Scrum facilitaba el cumplimiento de plazos y la adaptación a cambios imprevistos, como se podía observar en la Ilustración 16.
Todo el proceso comenzaba con la recopilación de requerimientos, los cuales se plasmaban como reglas de negocio en lo que se conocía como Backlog. Posteriormente, se generaba una estimación de tiempo para cada tarea dentro del Backlog, lo que permitía crear un Sprint Planning para determinar cuántas tareas se podían completar en un plazo de una semana (Sprint Backlog).

Durante el periodo del Sprint, el equipo involucrado (Scrum Team) se reunía diariamente en una junta (Daily Scrum) para notificar el estado de cada actividad, incluyendo el porcentaje de avance. En caso de que hubiera algún retraso o si una tarea dependía del resultado o avance de otra persona, se le denominaba Stopper.

Si todo transcurría sin contratiempos, los cambios se aplicaban al producto final y se revisaba la calidad del mismo. Además, se realizaban juntas de seguimiento para decidir si esa versión se consideraba estable o si era necesario hacer un rollback e iterar nuevamente desde el Backlog o planear un nuevo Sprint.

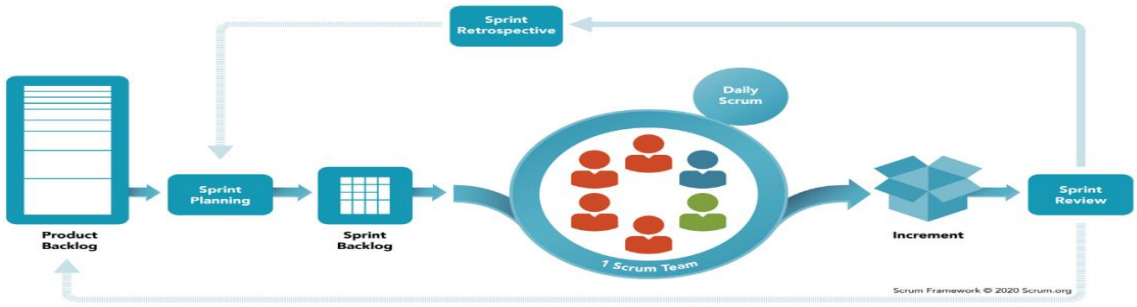

Ilustración 14 Diagrama de Proceso Scrum

CAPÍTULO 4. APLICACIÓN DE LA METODOLOGÍA Y DISCUSIÓN DE RESULTADOS

Programación de Reglas desde Python

Como se mencionó en temas anteriores, el motor de detección de intrusiones de Snort se fundamentaba en el uso de opciones de reglas. En este contexto, para ayudar a los administradores de seguridad, se realizó una conexión desde un PC al servidor y se generó una nueva regla, como se muestra a continuación.

Se presentó una nueva pantalla con cinco paneles. El primer panel contenía los "Server Parameters" (parámetros del servidor) que debían configurarse. Para la configuración inicial, se solicitó proporcionar la información necesaria.

Estos parámetros incluían el "Sistema operativo", que en ese momento estaba restringido a distribuciones de Linux, siendo "Ubuntu" una de las opciones disponibles. Se preguntaba cuáles eran las otras opciones. Además, se requería ingresar la dirección IP del servidor, así como el nombre de usuario y la contraseña necesarios para acceder al sistema. Esta información era esencial para establecer una conexión confiable y eficiente con el servidor.

Además de elegir la distribución de Linux "Ubuntu", se podían considerar otras opciones en las actualizaciones de software, según las preferencias del usuario. Al incorporar esta flexibilidad, el sistema podía adaptarse más fácilmente a diferentes entornos y gustos de los usuarios.

Si se encontraban detalles en los materiales proporcionados, era importante tener en cuenta que estos podrían no ser precisos y podrían dificultar la capacidad para acceder y operar correctamente el servidor. Por lo tanto, se recomendaba examinar detenidamente la información antes de proceder con la configuración.

Ilustración 15 Imagen del Sistema para crear Reglas en Snort - Panel 1.

La siguiente pantalla mostraba y resaltaba los parámetros de configuración necesarios para capturar y detectar paquetes anómalos en la red.Estos componentes desempeñaban un papel crucial en la identificación y análisis de problemas potenciales. Las opciones incluían:

- **Protocolo**: Indicaba el tipo de protocolo utilizado por los paquetes de datos, como TCP, UDP, ICMP, entre otros.
- **Dirección IP de origen**: La dirección IP del dispositivo que enviaba el paquete.
- **Puerto de origen**: El número de puerto utilizado por el dispositivo de origen para enviar los paquetes.
- **Ruta**: La trayectoria que debía seguir un paquete en la red para llegar a su destino.
- **Dirección IP de destino**: La dirección IP del dispositivo al que se enviaba el paquete.
- **Puerto de destino**: El número de puerto utilizado por el dispositivo de destino para recibir el paquete.
- **Flujo**: Para que la regla funcionara, era necesario especificar la dirección en la que debía dirigirse el tráfico de la red. La palabra clave "flujo" se utilizaba junto con el reensamblaje del flujo TCP, permitiendo aplicar las reglas solo a direcciones específicas del tráfico.
- **Aviso**: El contenido del paquete podía variar según el tipo de protocolo y la aplicación.
- **Clasificación**: Permitía clasificar los paquetes según su contenido, origen, destino, etc.
- **Enlaces**: Cualquier enlace o metadato adicional asociado con el paquete que pudiera ayudar en su análisis y seguimiento.

- **ID**: Un identificador único asignado al paquete para su seguimiento y uso futuro.
- **Evaluación**: Un análisis adicional del paquete que podía incluir acciones correctivas o recomendaciones para mitigar problemas potenciales.

Asegurarse de que todos estos componentes se recopilaran y analizaran de manera efectiva era esencial para mantener la seguridad y el rendimiento óptimo de la red. Cada uno de estos parámetros proporcionaba información única y valiosa que contribuía a una comprensión integral de la actividad en la red y facilitaba la detección temprana y resolución de posibles anomalías.

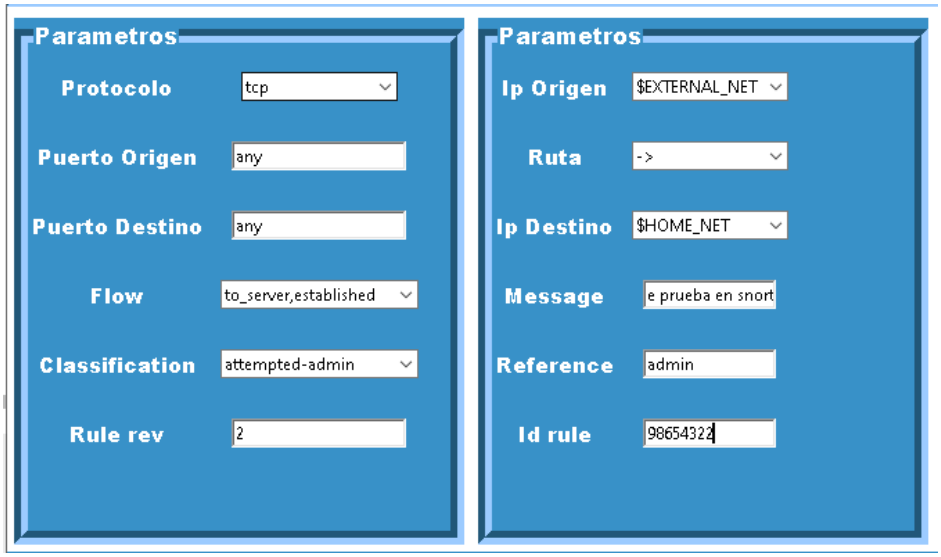

Ilustración 16 Imagen del Sistema para crear Reglas en Snort - Panel 2 y 3 parámetros.

Una vez que se aseguraron los parámetros deseados, la regla configurada aparecía en el panel de Reglas de configuración.

Esta regla se podía identificar porque comenzaba con cualquiera de los siguientes ejemplos: "alert tcp", "alert udp" o "alert icmp", y finalizaba con la versión "rev:2", tal como se ilustraba en el ejemplo proporcionado. Para ofrecer una explicación más detallada.

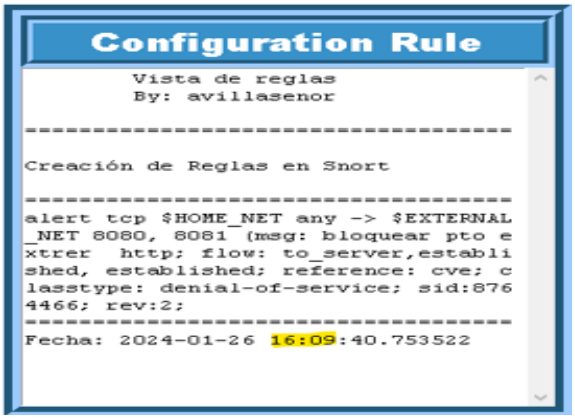

Ilustración 17 Imagen del Sistema para crear Reglas en Snort - Panel 4 Visualización de la regla

La sección de Reglas de configuración mostraba las reglas que se habían configurado con los parámetros ingresados previamente. Comprender estas reglas era extremadamente importante, ya que determinaba cómo se procesarían los paquetes que cumplían con ciertos criterios.

Las reglas de alerta eran fundamentales en los sistemas de detección de intrusiones y en el análisis de la seguridad de la red. Permitían al sistema identificar y responder a posibles amenazas o actividades sospechosas en la red.

Por ejemplo, "alert tcp" indicaba una regla configurada para alertar sobre cualquier tráfico TCP que cumpliera con ciertos criterios; "alert udp" se refería al tráfico UDP, y "alert icmp" al tráfico ICMP.La sección "rev:2" señalaba la versión de la regla, lo cual resultaba útil para hacer un seguimiento de las actualizaciones o cambios en la misma. Al comprender cómo se creaban y aplicaban estas reglas, los administradores de red podían mejorar la seguridad y el rendimiento de sus sistemas, asegurando una respuesta adecuada a amenazas potenciales y mitigando el riesgo de intrusiones no deseadas.

El siguiente caso de uso presentaba una regla de ejemplo que abarcaba el tráfico TCP desde la red externa hacia la red interna, específicamente en los puertos 8080 y 8081. Esta regla tenía como objetivo bloquear dicho tráfico. Para ofrecer una explicación más detallada.

Al bloquear el tráfico TCP desde la red externa hacia la red interna en puertos específicos, se podía reducir el riesgo de seguridad y proteger los recursos internos contra posibles amenazas externas.

Los puertos 8080 y 8081 eran comúnmente utilizados en aplicaciones web y servicios de red, por lo que bloquear el tráfico en estos puertos representaba una medida proactiva para evitar el acceso no autorizado o actividades maliciosas.

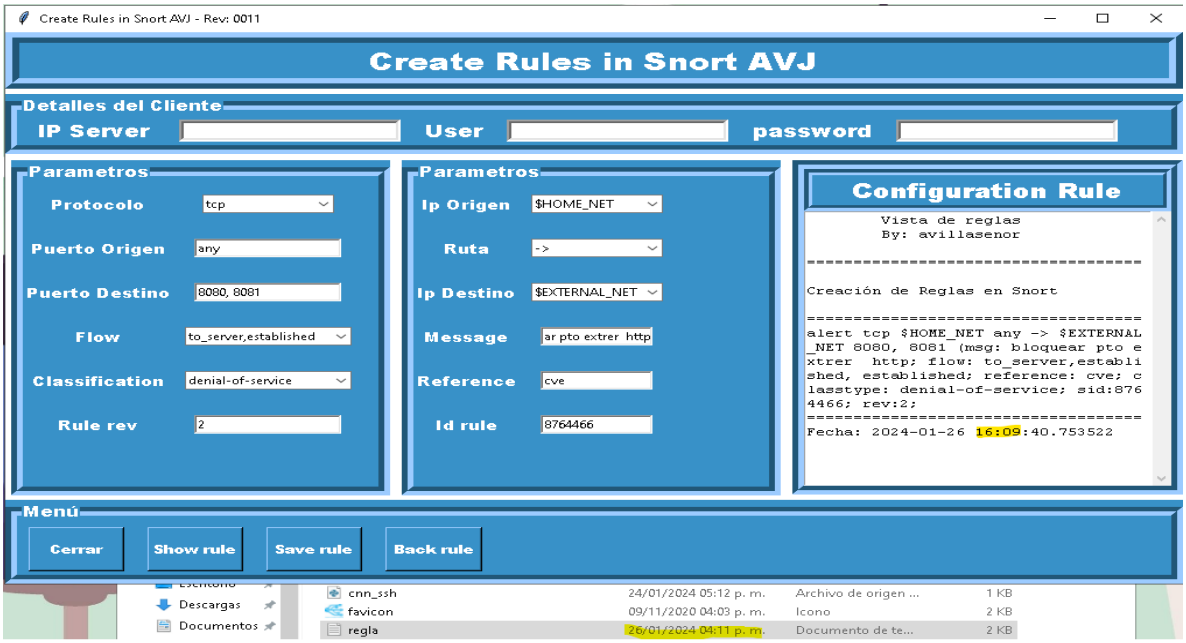

Ilustración 18 Imagen del Sistema para crear Reglas en Snort - Mostrando todos los paneles.

Una vez que la regla se mostraba en la pantalla, se guardaba en el archivo de reglas, como se ilustraba en la imagen a continuación. Una vez guardada, se podía consultar Snort IDS para ver las nuevas reglas. Para ofrecer una explicación más detallada.

La capacidad de guardar reglas en un archivo era importante para mantener un registro organizado y accesible de las políticas de seguridad aplicadas al sistema.

Esto facilitaba la administración y el mantenimiento de las reglas a lo largo del tiempo, así como su restauración cuando fuera necesario.

En IDS Snort, la posibilidad de probar y revisar nuevas reglas era fundamental para verificar su correcta implementación y funcionamiento.

Esto permitía a los administradores de seguridad comprobar rápidamente que las reglas se aplicaran y funcionaran adecuadamente, lo que contribuía a mantener un entorno de red seguro y protegido contra posibles amenazas

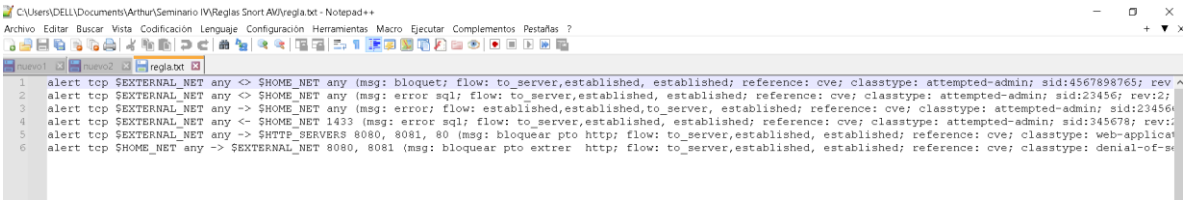

Ilustración 19 Imagen del archivo .rules donde el Sistema guardo la Reglas creada

CAPÍTULO 5. CONCLUSIONES Y PERSPECTIVAS PARA TRABAJOS FUTUROS

Se llevó a cabo una investigación sobre el sistema Snort para comprender su funcionamiento y los módulos que componen el sistema de detección de intrusiones (IDS Snort). El objetivo principal de esta investigación era aprender a crear reglas para el IDS Snort utilizando una interfaz que facilitara la generación de reglas.

En la siguiente fase, la prioridad y el objetivo serían desarrollar una red neuronal utilizando el lenguaje de programación Python (web), con el fin de entrenar la red neuronal (RN) para que pudiera absorber y obtener información de las reglas existentes en Snort. El proceso de entrenamiento de la RN sería exhaustivo y prolongado, permitiendo establecer un vínculo entre la red neuronal y el IDS.

La integración de la etapa 2 con la etapa 1 estaría diseñada para aumentar y mejorar las capacidades de detección de intrusos en la red interna. Este diseño se basaría en software de código abierto (Open-source software). Con esto en mente, el proyecto buscaría fomentar un esfuerzo colaborativo entre los miembros de la comunidad para trabajar juntos en el mantenimiento y mejora de la funcionalidad de la red neuronal en colaboración con Snort.

REFERENCIAS

Atico34, G. (s.f.). *Así es Snort, el sistema de detección de intrusos más popular*. Obtenido de ¿Qué es Snort?: https://protecciondatos-lopd.com/empresas/snort-deteccion-intrusos/#:~:text=SNORT%20es%20un%20sistema%20de,software%20gratuito%20de%20c%C3%B3digo%20abierto.

Bobadilla, J. (2020). *Machine Learning y Deep Learnig.* Madrid, España: Ra-Ma.

G2. (12 de 2023). *G2 Grid® for Intrusion Detection and Prevention Systems (IDPS)*. Obtenido de https://www.g2.com/categories/intrusion-detection-and-prevention-systems-idps?utf8=%E2%9C%93&selected_view=grid&segment=mid-market#grid

Science, T. D. (12 de diciembre de 2018). *towardsdatascience.com*. Obtenido de Artificial Intelligence Framework: A Visual Introduction to Machine Learning and AI: https://towardsdatascience.com/artificial-intelligence-framework-a-visual-introduction-to-machine-learning-and-ai-d7e36b304f87

Snort. (s.f.). *Snort 3 Rule Writing Guide*. Obtenido de https://docs.snort.org/rules/headers/file_rules

UNAM. (4 de enero de 2024). *04. Inteligencia Artificial autor Facultad de Ingenieria, UNAM*. (UNAM, Editor) Obtenido de Capitulo 4: https://www.dropbox.com/sh/d6plg2q4oydvpcs/AAABDyuHyyKPG7iK7QV5m5MUa/Archivos%20Infolibros%20ES/Temas/390%20Inteligencia%20Artificial/04.%20Inteligencia%20Artificial%20autor%20Facultad%20de%20Ingenieria%2C%20UNAM.pdf?dl=0

Ziad Obermeyer, M. a. (29 de Septiembre de 2016). *Predicting the Future — Big Data, Machine Learning, and Clinical Medicine*. (T. N. Medicine, Editor) Obtenido de nejm.org: https://www.nejm.org/doi/10.1056/NEJMp1606181

ANEXOS

ANEXO A “Nombre del documento”