ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"PLANTEAMIENTO Y VALIDACIÓN DE UN MÓDULO DE APRENDIZAJE PARA EL IDS/IPS SNORT"

TESINA DE SEMINARIO

Previa a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES ESPECIALIZACION SISTEMAS INFORMACIÓN

Presentada por:

ABDÓN ANDRÉS CARRERA RIVERA

MANUEL ANTONIO CASTILLO GUTIÉRREZ

JUAN CARLOS QUIZHPI ORDÓÑEZ

Guayaquil - Ecuador
2011

AGRADECIMIENTO

A Dios, que nos ha conservado con vida y salud.
A nuestros padres, quienes han sido y son un pilar fundamental en nuestras vidas.
A nuestros profesores, por todos los conocimientos y consejos transmitidos

DEDICATORIA

A Dios
A nuestros padres
A nuestros familiares
A nuestros amigos

TRIBUNAL DE SUSTENTACIÓN

Ing. Alfonso Aranda

PROFESOR DEL SEMINARIO

Ing. Albert Espinal

PROFESOR DELEGADO POR EL DECANO

DECLARACIÓN

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Abdón Andrés Carrera Rivera

Manuel Antonio Castillo Gutiérrez

Juan Carlos Quizhpi Ordóñez

RESUMEN

En este trabajo se presenta la implementación de un módulo de optimización para el IDS/IPS SNORT basando su mitigación no sólo en firmas, sino también mediante, estadísticas y aprendizaje.

El funcionamiento del módulo de optimización se basa en tres procesos que son modo aprendizaje, modo detección y limpieza de anomalías. El primero se centra en aprender los patrones del tráfico común, almacenándolo en una base de datos, luego se inicia el modo detección para comparar el tráfico aprendido previamente y detectar cualquier anomalía, enviando las conexiones anómalas al módulo de limpieza, lo cual permite a este último proceso eliminar única y exclusivamente la conexión maliciosa.

El documento está dividido en 5 capítulos que comprenden el planteamiento del problema, el marco teórico, el análisis del problema con su respectivo diseño de la solución, la implementación y análisis de los resultados, las pruebas realizadas y posteriormente las conclusiones y recomendaciones.

En el primer capítulo se detalla los aspectos básicos de la tesis, como son: antecedentes, objetivos, justificación y metodología para la realización de este proyecto.

En el segundo capítulo se pretende dar una noción general y especifica que solo puede brindar el soporte teórico para la implementación de todo

proyecto, donde se detalla principalmente los conceptos e información recogida acerca de la herramienta Snort, IDS/IPS y Firewall.

El tercer capítulo explica el problema y el diseño de la solución. En este capítulo se establecen las responsabilidades que tendrá cada uno de los módulos en los que se ha sido dividido el proyecto (modo aprendizaje, detección y limpieza), así como escaneos de tráfico de una red y elaboración de perfiles de conocimiento.

En el cuarto capítulo se explican los detalles considerados en el código usado para la implementación y el funcionamiento del sistema. Se enfatiza también los modos de operación que posee Snort y la manera de ejecutarse como un proceso más del sistema operativo.

El quinto capítulo describe las pruebas realizadas utilizando los módulos implementados de detección y aprendizaje, estas pruebas están basadas en análisis y tablas estadísticas

Finalmente se presentan las conclusiones obtenidas y se plantean recomendaciones para futuros trabajos relacionados.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los últimos años ha sido evidente el rápido crecimiento del internet y de las redes informáticas alrededor del mundo, las cuales almacenan, procesan y transportan información de suma importancia para las organizaciones.

Muchas de éstas redes sin la seguridad informática adecuada, están siempre propensas a intrusiones o intentos de conexiones no autorizadas por parte de agentes externos a la red, como por ejemplo los hackers; lo que hace necesario reforzar la seguridad para combatir las nuevas formas de ataques que surgen día a día.

Los IDS/IPS son un avance en la seguridad perimetral, debido a que estos sistemas permiten no solo monitorear, sino también ejercer un control de acceso en una red informática, protegiéndola de posibles ataques o accesos irregulares, siendo esto de gran ayuda para los administradores de red.

Debido a las grandes ventajas que nos proporcionan estos sistemas de seguridad informática, se plantea el mejoramiento del IDS/IPS SNORT basándose en la detección de anomalías, es decir posibles ataques basados en comportamientos fuera del tráfico normal de la red.

CAPÍTULO 1

1. PLANTEAMIENTO

1.1 Definición del problema

Entre los sistemas más importantes y reconocidos con los que se cuenta actualmente para la protección de las redes y los sistemas informáticos, se encuentran los IDS (Sistemas de Detección de Intrusos) y los IPS (Sistemas de Prevención de Intrusos). Los IDS/IPS suponen el análisis de eventos de un sistema o el tráfico que pasa por una red determinada, con el fin de descubrir o detectar actividades potencialmente maliciosas, permitiendo al administrador del sistema o red obtener información que ayude a tomar las medidas pertinentes para evitar futuras nuevas intrusiones.

Hoy en día, los IDS/IPS generalmente usan la detección basada en firmas, es decir comparan patrones de ataques ya conocidos con los datos que analizan, siendo el principal problema que solo detectan ataques conocidos, lo que imposibilita la detección de nuevos ataques o intrusiones. Este problema se incrementaría debido al crecimiento de nuevos ataques y actualmente es posible encontrar en internet información o programas que podría hacer que un usuario poco

experimentado obtenga algunos conocimientos básicos para poder acceder de forma poco honesta a algún tipo de sistema o red.

1.2 Objetivos

El presente trabajo tiene como objetivo general aprovechar las fortalezas y oportunidades que nos brinda el IDS/IPS Snort, optimizando su funcionamiento para lograr un mejor control de posibles ataques a una red.

La planificación de los siguientes objetivos específicos en el contexto del objetivo principal se detalla a continuación:

El funcionamiento se divide en tres etapas principales: Modo de aprendizaje, Detección de anomalías y Limpieza de anomalías. En el modo de aprendizaje, Snort capta y guarda todo el tráfico de una red, aprendiendo cada detalle de esta. La detección de anomalías se apoya en un proceso donde se compara el tráfico normal que Snort aprendió de una red, alertando posibles sospechas de nuevas anomalías. Las detecciones maliciosas pasan al módulo de limpieza o bloqueo para poder evitar posibles ataques.

Desarrollar una interfaz gráfica de administración y monitoreo que permita mostrar información importante que respalde la toma de decisiones en tiempo oportuno.

1.3 Justificación

Este proyecto contribuye a mejorar la protección de las redes informáticas de las nuevas y crecientes amenazas a las que están expuestas actualmente y que han surgido de la mano con las nuevas tecnologías.

En la actualidad hay varios IDS/IPS que realizan detecciones basadas en anomalías y/o estadísticas pero tienen costos. Por ejemplo el IDS/IPS de Cisco.

Entre las fortalezas del IDS/IPS Snort se pueden destacar:

- Es libre y de código abierto. Brinda la oportunidad de modificarlo y mejorarlo de acuerdo a las necesidades que se presenten en una organización.
- Es multiplataforma. Se ejecuta en diversos sistemas operativos, no se limita a uno en específico.
- Es escalable. Su arquitectura modular permite agregar nuevos módulos.

Las debilidades del IDS/IPS Snort son:

- Detecta ataques basados en firmas y en consecuencia es vulnerable a nuevos o desconocidos ataques.

- No posee una interfaz amigable. Snort trabaja bajo comandos, lo que ocasiona que sea complejo su uso.

Estas debilidades justifican el proyecto, permitiendo optimizar el funcionamiento de Snort y así aprovechar al máximo las bondades del mismo.

1.4 Alcance

Proteger las redes de posibles ataques conocidos por las firmas del IDS/IPS Snort o ataques detectados como anomalías basadas en comportamientos “no normales” de la red.

Provee información importante para el administrador de red como:

Consulta de alertas generadas en el proceso de detección, datos de conexiones de red. Por ejemplo: ip fuente, ip destino, puertos, protocolos.

Análisis parcial de cada conexión basado en estadísticas, incluyendo un índice de riesgo que define que tan peligrosa puede ser una conexión.

Seguridad de una red a través del firewall bloqueando conexiones maliciosas.

CAPÍTULO 2

2. MARCO TEÓRICO

El objetivo del estudio de este capítulo es abarcar de manera general los fundamentos teóricos para la comprensión del funcionamiento de Snort.

2.1 Snort

Snort es un Sistema de Detección de Intrusiones basado en red (NIDS o IDS de red) de código abierto que puede descargarse desde la página web www.snort.org. Es capaz de realizar un análisis en tiempo real del tráfico de una red así como registrar los paquetes que conforman dicho tráfico.

Snort implementa un motor para la detección de ataques que permite registrar, alertar y responder ante cualquier anomalía previamente definida. Para ello utiliza patrones (reglas) que corresponden a ataques, barridos, intentos de aprovechar alguna vulnerabilidad, análisis de protocolo, etc. Además existen una serie de herramientas que pueden utilizarse con Snort, ya sean herramientas para simplificar el uso del mismo o herramientas que trabajan conjuntamente y lo hacen más potente.

Snort utiliza un lenguaje flexible de reglas para describir el tráfico e indicar qué paquetes pueden pasar y cuáles no (son considerados potencialmente peligrosos).

Además de realizar el análisis del tráfico en tiempo real, Snort tiene un módulo con capacidad de generar alertas en tiempo real, registrando dichas alertas en ficheros de texto ASCII, UNIX sockets, bases de datos. (1)

Esquema de Snort

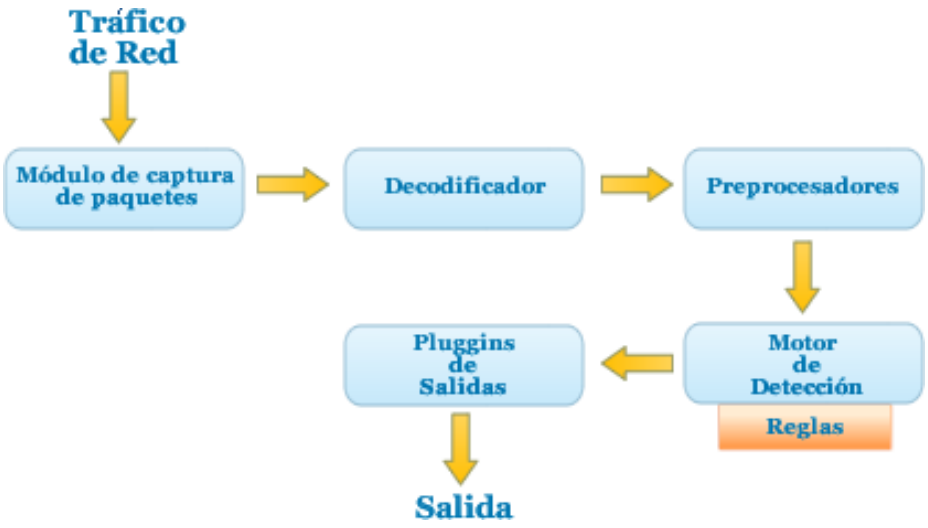

Figura 2-1 Esquema de Snort

En la Figura 2-1 se grafica el esquema de Snort detallando sus módulos a continuación:

Módulo de Captura de Paquetes

Se encarga de la captura de paquetes. Para ello usa las librerías libpcap (en Linux) y WinPcap (en Windows).

Decodificador

Decodifica los paquetes. Finalizada esta fase, Snort tiene la información de los protocolos en los lugares adecuados para un posterior análisis.

Preprocesadores

El pre procesamiento permite a Snort analizar posteriormente los paquetes más fácilmente. Genera alertas, clasifica paquetes o los filtra.

Motor de Detección

Analiza los paquetes de acuerdo a reglas predefinidas. Puede poseer módulos de detección auxiliares para análisis más específicos.

Pluggins de Salidas

Cuando se produce una alerta, Snort tiene diversos métodos de salida como los logins, bases de datos y syslogs. (2)

2.2 Sistemas de Detección de Intrusos / Sistemas de Prevención de Intrusos (IDS / IPS)

Un IDS (Intrusion Detection System) es un sistema que intenta detectar o monitorizar eventos dentro de un equipo o a una red en busca de accesos no autorizados.

Un IPS (Intrusion Prevention Systems) es un sistema de protección que defiende de las intrusiones, establece políticas de seguridad para proteger el equipo o la red de un ataque. Son una mejora a los firewalls, ya que toman decisiones de control de acceso basados en los contenidos del tráfico, en lugar de direcciones IP o puertos.

Se puede decir que un IPS brinda una protección proactiva y un IDS ofrece una protección reactiva. (3)

2.3 IDS/IPS con Detección de Anomalías

Los IDS/IPS basados en anomalías se especializan en buscar actividad sospechosa dentro de un sistema, pero apoyándose en una fase inicial de aprendizaje que construye un perfil del sistema que se está monitoreando para luego mediante técnicas estadísticas comparar la información recibida en cada instante con el perfil creado. El aprendizaje se lleva a cabo durante un tiempo conveniente con el

fin de considerar las actividades dentro del perfil como comportamiento normal y legítimo.

La Figura 2-2 muestra el esquema para elaborar un perfil

Figura 2-2 Creación de Perfil

Las medidas para la creación de un perfil podrían incluir una serie de parámetros como la carga de CPU, número de conexiones de red en una unidad de tiempo, número de procesos, entre otros.

Las ventajas de la detección de anomalías residen en la capacidad de detectar ataques en los cuales no se posee conocimiento específico, también produce información que puede ser utilizada para definir firmas o patrones.

Las desventajas de la detección de anomalías radican en la generación de gran número de falsos positivos debido a los comportamientos no predecibles de usuarios o redes, y también requieren grandes conjuntos de entrenamiento para caracterizar el comportamiento normal. (4)

La Figura 2-3 resume las diferencias entre un IDS/IPS basados tanto en firmas como en anomalías.

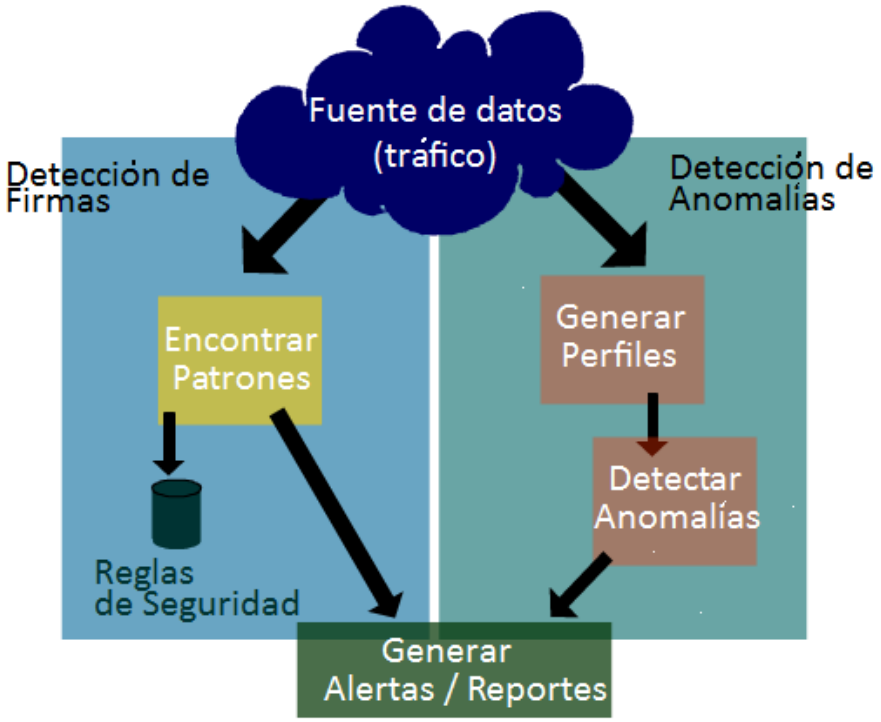

Figura 2-3 Diferencia IDS/IPS basado en firmas vs IDS/IPS basado en anomalías

2.4 ACID

ACID (Analysis Console for Intrusion Databases) es un motor de análisis basado en PHP que procesa una base de datos de incidentes de seguridad generados por IDS, firewalls y herramientas de monitoreo de red. Portátil, de código abierto y bajo licencia GPL, funciona en distintos sistemas operativos que soporten PHP como Linux y Windows.

ACID fue desarrollado por Roman Danyliw en el Centro de Coordinación CERT, en un inicio como parte del proyecto AIRCERT.

Sus características incluyen:

- Interfaz de búsqueda, encuentra alertas y su respectiva meta-información tales como las firmas, el tiempo de detección, direcciones fuente/destino, puertos, carga útil, o banderas.
- Paquete de visualización, muestra gráficamente paquetes de información de alertas registradas en las capas 3 y 4.
- Gestión de alertas, proporciona la creación de grupos de alertas, elimina alertas o falsos positivos, exporta a correo electrónico, o a archivos de alertas para transferirlos entre bases de datos.
- Genera gráficos y estadísticas basadas en el tiempo, el sensor, firma, protocolo, dirección IP, puertos TCP/UDP, o clasificación.

ACID es capaz de analizar gran variedad de eventos que están en post-procesado y para ello precisa otras herramientas como Snort o Iptables. (5)

2.5 Firewall

Un Firewall es un sistema que impone políticas de seguridad entre una red privada y el Internet. Imposibilita el acceso de extraños a una computadora desde Internet. Pueden ser de dos tipos, de software o de hardware y proveen una frontera de protección contra intrusos.

Funcionamiento de un Firewall

Un Firewall funciona, inicialmente, denegando cualquier tráfico que se produce mediante el cierre de todos los puertos de un equipo. En el instante en que un determinado servicio o programa intente acceder a la computadora lo hará saber. Se puede entonces en ese momento aceptar o denegar dicho tráfico. (6)

Iptables

Iptables es un sistema de firewall vinculado al kernel de Linux en espacio de usuario. Aplica políticas de filtrado del tráfico de una red. Está constituido de la siguiente manera:

- Regla: Permite filtrar un paquete y decide si dejarlo pasar o no.
- Cadena: Listado de reglas. Contiene la política, decide qué hacer con los paquetes que no coincidieron con alguna de las reglas.
- Tabla: Contiene las cadenas.

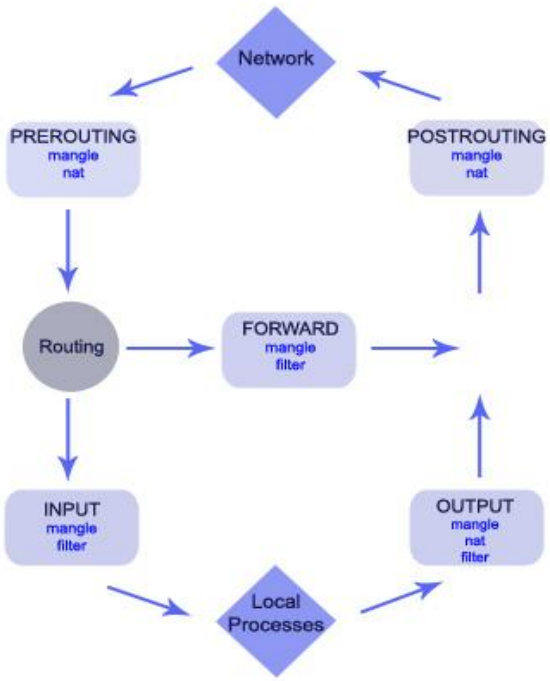

Figura 2-4 Esquema de Iptables

Existen tres tablas ya incorporadas como se aprecia en la Figura 2-4 y que se mencionan a continuación:

- **filter**, encargada del filtrado. Todos los paquetes pasan por ella. Contiene las siguientes cadenas predefinidas:
    - *INPUT* Los paquetes destinados a este sistema atraviesan esta cadena
    - *OUTPUT* Los paquetes creados por este sistema atraviesan esta cadena

- *FORWARD* Los paquetes que solo pasan por este sistema para ser encaminados a su destino recorren esta cadena

- **nat**, configura reglas de reescritura de direcciones o de puertos de los paquetes. Posee las siguientes cadenas predefinidas:
  - *PREROUTING* Los paquetes entrantes pasan por esta cadena antes de que se consulte la tabla de ruteo local.
  - *POSTROUTING* Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo.
  - *OUTPUT* Permite hacer un DNAT limitado en paquetes generados localmente

- **mangle**, se encarga de ajustar las opciones de los paquetes. Todos los paquetes pasan por esta tabla. Diseñada para efectos avanzados, por lo cual contiene todas las cadenas predefinidas. (7)

CAPÍTULO 3

3. ANÁLISIS Y DISEÑO

3.1 Análisis del problema

Cuando se analizó las debilidades actuales que presenta el IDS/IPS Snort, se encontraron oportunidades para mejorar dicha herramienta, aprovechando las ventajas que ofrece Snort debido a que está basado en código abierto y es multiplataforma.

Las cuatro debilidades más relevantes encontradas y en las que se basa este proyecto son:

1. Snort no posee una base de conocimiento, por ende no aprende ni analiza las distintas conexiones o patrones de tráfico que pueden existir en una red. Es una gran desventaja cuando se desea hallar anomalías basadas en análisis de tráficos posteriores y reales sin hacer uso de las reglas o firmas de Snort.

2. Snort solo realiza detecciones por medio de firmas, es decir patrones ya definidos para detectar ataques, esto representa un problema cuando se presentan nuevas formas de intrusiones,

haciendo que las firmas pasen a ser obsoletas si no están en constante actualización.

3. Snort asume que hay un administrador de red, por lo cual no bloquea a los intrusos automáticamente, solo mostrando al administrador que se ha activado una alarma, por lo consiguiente se deduce que Snort no es reactivo y no se encarga de bloquear la conexión maliciosa.

4. Otro punto a destacar es que Snort no proporciona una interfaz gráfica para mostrar los resultados de monitorear la red que se está protegiendo, haciendo difícil el análisis de los resultados obtenidos.

Para el problema presentado en el primer punto es muy importante poseer en una base de datos todo el tráfico de una red, de esta manera se puede analizar detalladamente los distintos patrones que pasan a través de la red, con la seguridad de que en el futuro se pueda detectar anomalías basándose en los conocimientos adquiridos previamente.

Para el segundo punto se considera importante la detección por anomalías, la cual permite mediante análisis estadístico encontrar comportamientos anormales en el tráfico de la red, con lo cual se la

protege de nuevas formas de ataques, ventaja que no podría encontrarse en la detección por firmas.

Para el tercer punto, se resalta la utilidad que nos proveen ciertos preprocesadores para el bloqueo de conexiones, lo que permite que Snort no solo sea proactivo sino también reactivo y tenga autonomía para bloquear la conexión que se considera maliciosa.

En el último punto, se considera importante que el proyecto se desarrolle como una aplicación web para mostrar de una manera amigable los resultados de monitorear la red, mediante reportes, tablas y gráficos estadísticos, que facilitan el análisis de la información obtenida a los administradores de red.

La aplicación también permite tomar decisiones en cuanto a bloqueo y desbloqueo de las conexiones antes detectadas por Snort.

3.2 Diseño de la solución

Al analizar el problema se determinó que existen cuatro etapas contenidas en la aplicación.

Monitoreo

El funcionamiento del IDS/IPS Snort es similar a los sniffers, porque el motor de análisis de Snort permite registrar y alertar ante un ataque previamente definido.

El módulo de captura de tráfico permite filtrar y capturar todos los paquetes de red,

El decodificador de paquetes se encarga de organizar los paquetes conforme vayan pasando por la pila de protocolos y cada subrutina del decodificador ordena de una forma distinta los paquetes formando una estructura de datos basada en punteros por encima del tráfico real capturado

El motor de detección analiza los paquetes de acuerdo a las firmas, el archivo de firmas (donde los ataques conocidos son definidos para su detección). Cuando llega un paquete al motor de detección, este busca en el archivo de firmas y si el paquete cumple con las opciones especificadas y si aparece alguna coincidencia no sigue buscando y se produce la alerta que es registrada en la base de datos, de donde se extrae la información que luego es mostrada mediante gráficos estadísticos, como por ejemplo las alertas producidas en un periodo determinado.

La Figura 3-1 representa como está dividida la etapa de monitoreo.

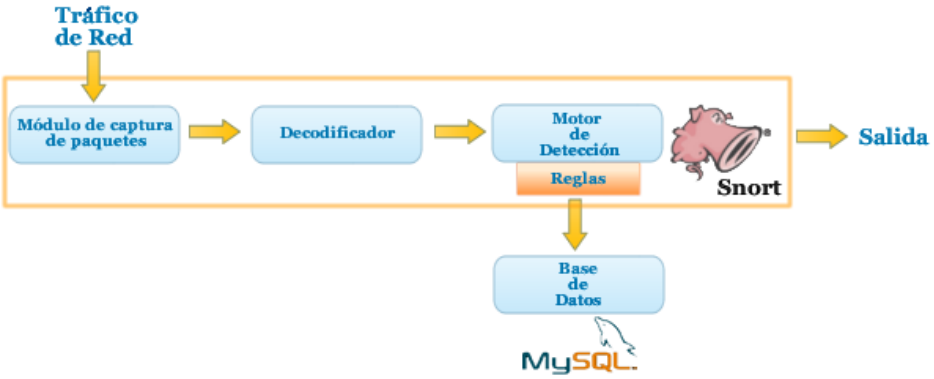

Figura 3-1 Etapa de Monitoreo y sus fases

Modo Aprendizaje

El módulo de aprendizaje, es el más importante en este proyecto. Es el núcleo para que Snort pueda adquirir información del tráfico real, analizar los paquetes que pasan por él y para proveer análisis inteligente para posibles detecciones de anomalías.

Para iniciar el modo de aprendizaje (entrenamiento), se necesita asignar la duración en horas, es decir, el tiempo que se desea aprender del tráfico “normal” de una red. Con un mínimo de 4 y un máximo de 24 horas de aprendizaje.

Cuando empieza el entrenamiento, se inicia también el IDS/IPS Snort en modo packet sniffer como un servicio o proceso más del sistema. Snort empieza a escanear todo el tráfico leyendo las cabeceras de todos los paquetes que pasen a través de la red y a su vez clasifica el

tráfico dependiendo de su protocolo: TCP, UDP, ICMP, IP para proceder a almacenar todo lo escaneado.

Todo el tráfico normal de la red es almacenado en una base de datos, la cual se convierte en la base de conocimiento para futuras detecciones. Dentro de la base de conocimiento se crean nuevos perfiles para cada sesión de aprendizaje. Se puede observar en la Figura 3-2 el contenido de cada perfil de un aprendizaje

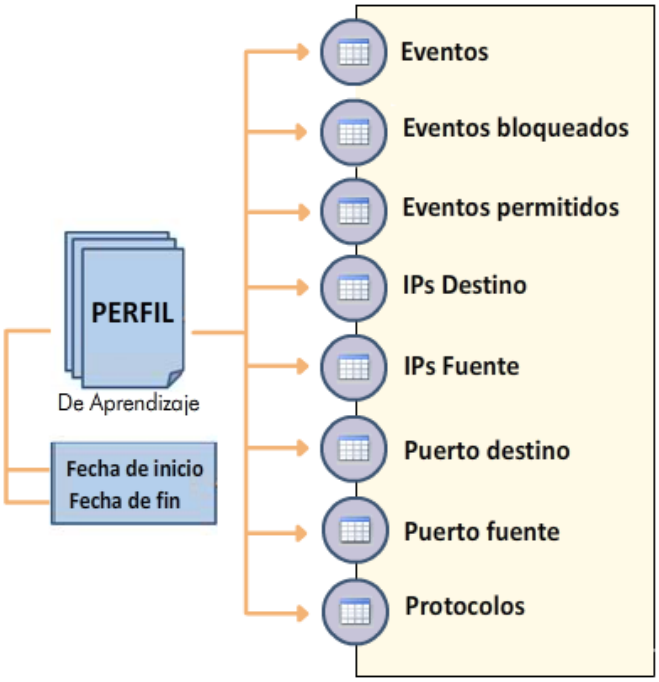

Figura 3-2 Partes del perfil de un aprendizaje

El nuevo perfil contiene un conjunto de información (tráfico y patrones) solo del tiempo establecido inicialmente; esto quiere decir que cada vez que se inicie un nuevo entrenamiento para un tráfico de una red, se crea un nuevo perfil que contiene todo el tráfico escaneado durante ese tiempo.

Una vez que es almacenado el tráfico en tiempo real, se procede a realizar el análisis estadístico creando histogramas y clasificando el tráfico por Ips Fuente, Ips Destino, Puertos fuente, Puertos destino y los eventos generales que ocurrieron en el tráfico.

Los histogramas son creados mientras se va entrenando el IDS. El factor más importante del histograma es el número de ocurrencias (en este caso el número de conexiones), así también el porcentaje de esa conexión en la red y el número de conexiones por minuto. Esto quiere decir, que cada vez que cierta dirección IP realice una conexión, su valor total de ocurrencia va a aumentar.

Por ejemplo, al iniciar el modo aprendizaje, puede darse el caso de que cierto servidor externo con dirección ip 192.188.59.33 a través del puerto 80 se comunique con una estación de trabajo de la red con ip 192.168.64.10 por el mismo puerto y después de cierto lapso de tiempo, en el sistema se puede ver que esta acción se ha repetido un determinado número de veces como se puede ver en la Tabla 3-1.

Ip Fuente	Puerto Fuente	Ip Destino	Puerto Destino	Protocolo	Ocurrencia
192.188.59.33	80	192.168.64.10	80	TCP	6

Tabla 3-1 Ejemplo de una dirección IP en un lapso corto de tiempo

Todo tráfico y conexión que pasa por el packet Sniffer de Snort se almacena en nuestro perfil dentro de la base de conocimiento. Al finalizar el modo aprendizaje se podría obtener la información que se muestra en la Tabla 3-2:

Ip Fuente	Puerto Fuente	Ip Destino	Puerto Destino	Protocolo	Ocurrencia
192.188.59.33	80	192.168.64.10	80	TCP	18
192.168.52.1	137	192.168.52.2	135	UDP	25
192.168.52.1	137	192.168.52.255	138	UDP	32
224.0.0.252	132	192.168.64.10	132	UDP	6
192.168.64.10	80	192.140.33.20	80	TCP	27
(...)					
192.180.255.2	80	192.168.64.10	3306	TCP	6

Tabla 3-2 Ejemplo de direcciones IP escaneadas en packet sniffer de snort en un lapso de tiempo

Con un total de 360 conexiones detectadas por el sistema durante el modo de aprendizaje y gracias a la información del histograma, se

puede estimar el porcentaje del umbral, el cual se encarga de proporcionar el límite de conexiones consideradas como tráfico normal en el momento que se inicia la detección.

Es decir que para la conexión tomada como ejemplo y graficada en la Tabla 3-3 se tiene:

Ip Fuente	Puerto Fuente	Ip Destino	Puerto Destino	Protocolo	Ocurrencia
192.188.59.33	80	**192.168.64.10**	80	TCP	18

Tabla 3-3 Ejemplo del No. de conexiones de una dirección IP en un lapso de tiempo

Con los resultados obtenidos se puede estimar que el valor aproximado del umbral es de 0.05, que es el resultado de la división de la frecuencia relativa del histograma con el número total de conexiones o recurrencias de todo el tráfico durante el modo de aprendizaje; es decir que el 5% de ésta acción está presente en el tráfico normal que se efectuó durante la etapa de aprendizaje.

En base a los histogramas, se presentan los datos del tráfico actual en tiempo real y a su vez el sistema actualiza estos datos de manera que siempre se muestre la última información adquirida del tráfico.

Cuando el aprendizaje cumple con el tiempo normal determinado previamente, el IDS/ IPS Snort finaliza su ejecución, presentando el

informe final del tráfico aprendido con su respectivo análisis de histogramas y conexiones por direcciones ips.

No obstante, el entrenamiento del tráfico puede ser finalizado antes de terminar el tiempo establecido inicialmente, de esta manera se actualiza el perfil con la hora inicial que se ejecutó el aprendizaje y la hora final en la cual se detuvo.

La Figura 3-3 muestra el proceso de aprendizaje

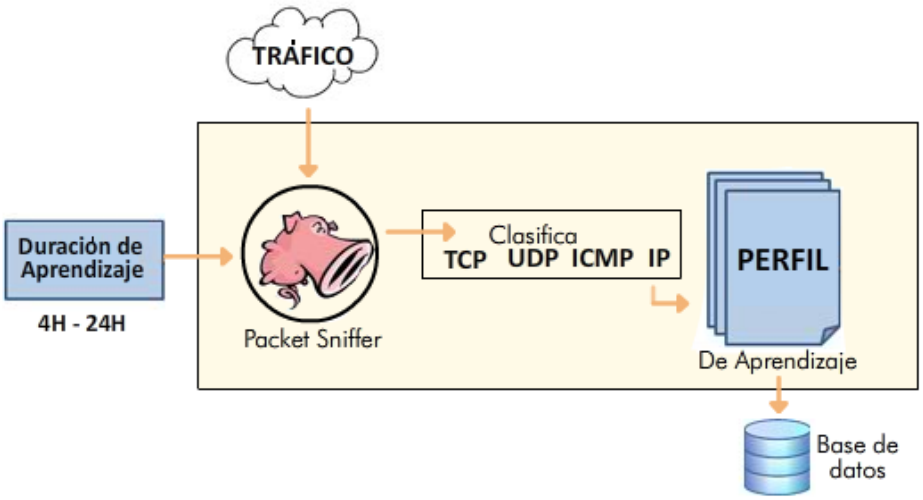

Figura 3-3 Modo Aprendizaje

Módulo de Detección

El modo detección es el siguiente paso para encontrar y reportar anomalías en una red. Este modo se basa en los conocimientos anteriormente adquiridos por Snort. Para iniciar la detección, primero

se debe tener un perfil o conocimiento previo, el cual ha sido obtenido por el entrenamiento del IDS/ IPS Snort en su fase de aprendizaje. Es por eso que antes de empezar a detectar anomalías, se presentan todos los perfiles con fechas y duración de entrenamiento de Snort.

Ambos modos de operaciones (Aprendizaje, Detección) cumplen las mismas funcionalidades; es decir, en el modo detección también se guarda todo el tráfico que se escanea en la red y también se los clasifica de la misma forma que en el modo aprendizaje, de esta manera se realizan comparaciones más rápidas y exactas mediante los histogramas y umbrales de la base de conocimiento.

Si al comparar las frecuencias de los histogramas obtenidos en la fase de aprendizaje con las frecuencias generadas en la fase de detección, se obtiene una desviación estándar mayor al umbral establecido por la frecuencia del aprendizaje, se considera algo anormal del tráfico común; es decir que en el modo de detección dicha anomalía sobrepaso el umbral de referencia que se tiene del tráfico aprendido.

En base al ejemplo anterior, de lo obtenido en el modo aprendizaje se muestra los datos en la Tabla 3-4:

Ip Fuente	Puerto Fuente	Ip Destino	Puerto Destino	Protocolo	Ocurrencia
192.188.59.33	80	**192.168.64.10**	80	TCP	18

Umbral	Porcentaje en tráfico
0.05	5%

Tabla 3-4 Ejemplo del umbral de una dirección IP en el modo aprendizaje

Pero en el transcurso del modo detección obtuvimos la información presentada en la Tabla 3-5:

Ip Fuente	Puerto Fuente	Ip Destino	Puerto Destino	Protocolo	Ocurrencia
192.188.59.33	80	**192.168.64.10**	80	TCP	35

Umbral	Porcentaje en tráfico
0.097	10%

Tabla 3-5 Ejemplo del umbral de una dirección IP en el modo detección

Internamente se analizan los datos de las ocurrencias obtenidas en la detección y se los compara con los datos del entrenamiento, donde el valor de número de conexiones actuales (35) sobrepasa el umbral del tráfico común aprendido (18); casi duplicándolo, lo que quiere decir que se ha encontrado una anomalía, algo fuera de lo común del tráfico normal que luego se debe analizar para que se tomen las medidas correspondientes. La Figura 3-4 muestra en un histograma las ocurrencias de conexiones y su respectivo umbral

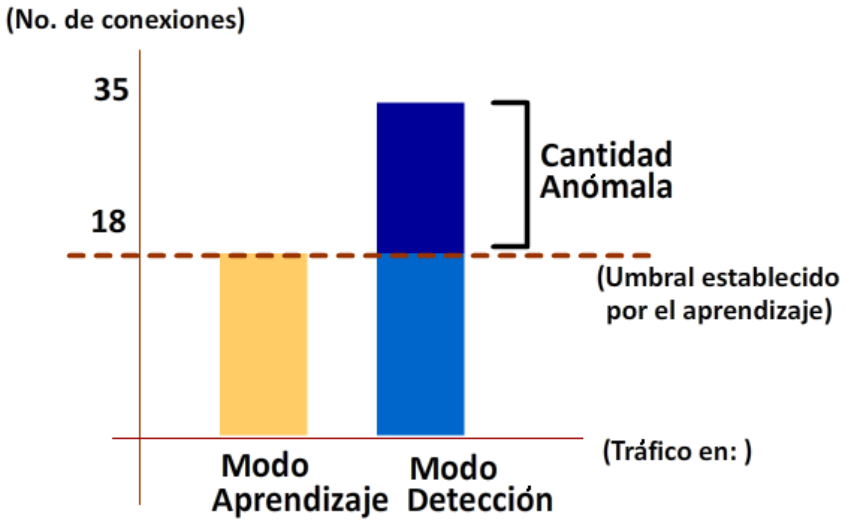

Figura 3-4 Histograma del tráfico en los modos aprendizaje y detección vs el número de conexiones

En tiempo real mientras se ejecuta la detección, se muestran las conexiones que violan la regla de sobrepasar el umbral de conexiones o por tener un valor mayor al porcentaje establecido en el histograma.

Todas estas direcciones consideradas como anómalas, pasan a disparar alertas y en ese caso se debe tomar la decisión de eliminar o aceptar dichas direcciones.

La Figura 3-5 muestra las fases del módulo de detección

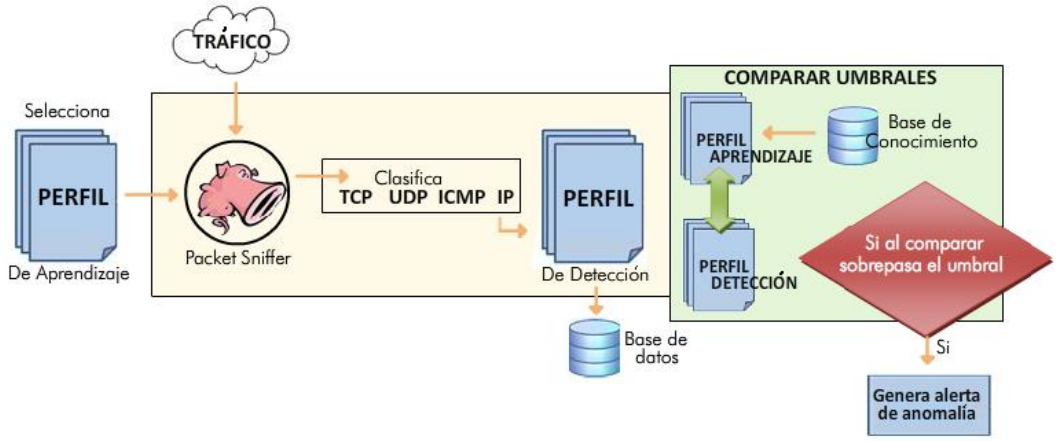

Figura 3-5 Módulo detección y sus fases

Limpieza de anomalías

El último paso de todo el proceso es eliminar las conexiones que generaron alertas en el modo de detección de anomalías.

Para eliminar y bloquear las conexiones anómalas se hace uso de un firewall, en este caso se usa iptables para escribir los protocolos, direcciones fuente, destino, puertos fuente y destino que se desean bloquear.

Puede darse el caso de que alguna dirección haya sido etiquetada como anómala y no lo sea, es decir, un falso positivo. Para estos casos se pueden aceptar las direcciones como si fueran falsos positivos procediendo a añadir excepciones a esas direcciones, esto con el fin de que en próximas detecciones no se consideren a las mismas como amenazas; lo cual ayuda a realizar detecciones más efectivas reduciendo la abundancia de falsos positivos.

La Figura 3-6 representa cómo funciona el Modo de Limpieza

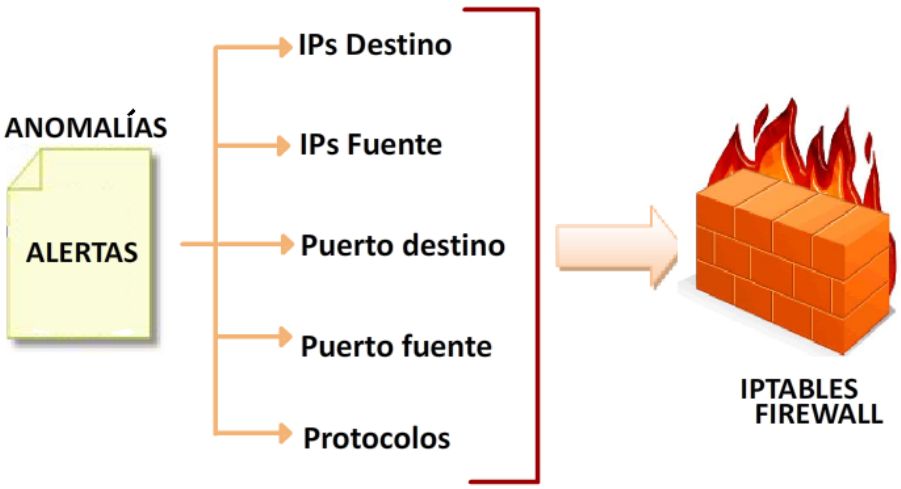

Figura 3-6 Modo de limpieza

Presentación

Para una mejor visualización de la información se implementó una aplicación web que permite un fácil acceso desde cualquier lugar y momento.

CAPÍTULO 4

4. IMPLEMENTACIÓN

En este capítulo se detalla el desarrollo de la implementación, la cual ha sido dividida en los siguientes subcapítulos:

4.1 Instalación y configuración

Para la instalación de Snort se descargó, la versión 2.9.0 de la página oficial de Snort (www.snort.org)

Una vez obtenida la versión de Snort y dentro de su directorio, se empieza a configurar el IDS desde la consola de Linux de la siguiente manera como muestra la Figura 4-1:

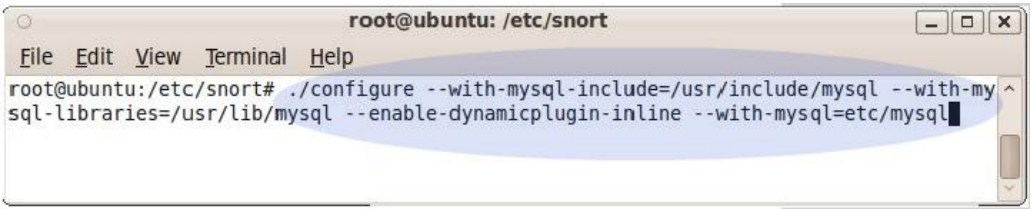

Figura 4-1 Instalación de Snort desde la consola linux

./configure --with-mysql-include=/usr/include/mysql --with-mysql-libraries=/usr/lib/mysql --enable-dynamicplugin-inline --with-mysql=etc/mysql

Se configura Snort para utilizar una base de datos, en este caso es MySQL (--with-mysql), además se habilitan opciones para que Snort trabaje con Plugins Dinámicos (--enable-dynamicplugin) y en modo Inline para trabajar en conjunto con iptables(--enable-inline).

A continuación se instala Snort con OUTPUT para usar MySQL como base de datos:

Make

Make install

Una vez instalado Snort, se puede revisar su contenido en el directorio /etc/snort, para constatar que la instalación se realizó con éxito como lo muestra la Figura 4-2.

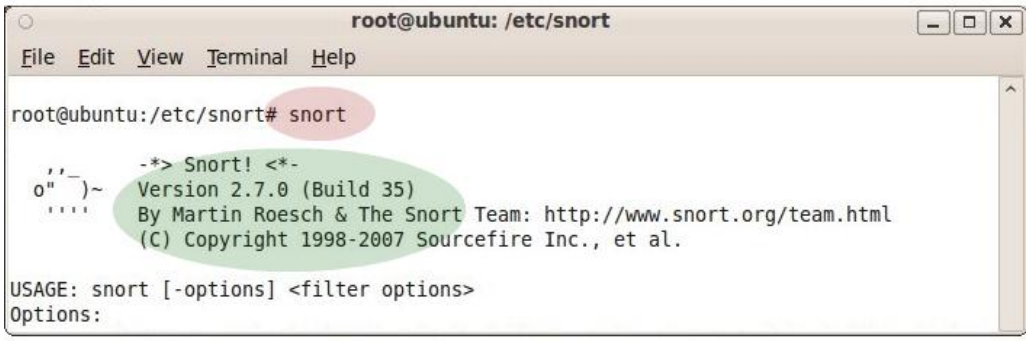

Figura 4-2 Ejecución del comando Snort en una terminal linux

Para ejecutar Snort con patrones de firmas y reglas se puede descargar las últimas reglas actualizadas correspondientes a los

ataques más comunes desde la página oficial de Snort (www.snort.org/rules).

Configuración

El archivo snort.conf es el más importante, ya que aquí se encuentran todas las configuraciones que Snort usa cuando se ejecuta.

Las configuraciones utilizadas en este proyecto son:

Var HOME_NET any

Donde HOME_NET es la red que se monitorea, por default el valor de any, para que Snort escanee la red inmediata.

Para configurar la base de datos, se aumentan líneas en la configuración para que se almacenen las salidas de las alertas que produce Snort. En este caso, se realizaran 2 salidas de alertas, una es almacenada en registros (logs) y otra en una base de datos llamada Snort en MySQL.

Output database: log, mysql, user=root password=password dbname=snort host=localhost

Output database: alert, mysql, user=root password=password dbname=snort host=localhost

Para utilizar Snort en monitoreo y detección de ataques mediante reglas o patrones, se modifica el archivo snort.conf añadiendo la referencia de dichas reglas previamente descargadas.

```
Include $RULE_PATH/aprendizaje.rules
```

Donde $RULE_PATH es la referencia al directorio donde se encuentran las reglas, seguido del nombre de la regla.

4.2 Operaciones de Snort

Las operaciones más importantes que se utilizan en este proyecto son:

Modo Sniffer

Snort puede utilizarse como un sniffer del tráfico que pasa por una red. El programa lee los paquetes y los muestra por consola según los va capturando.

Algunas acciones que se pueden realizar en este modo son:

- Mostrar por consola las cabeceras de los paquetes TCP/IP

```
./snort –v
```

La Figura 4-3 muestra los paquetes que se pueden observar en modo sniffer.

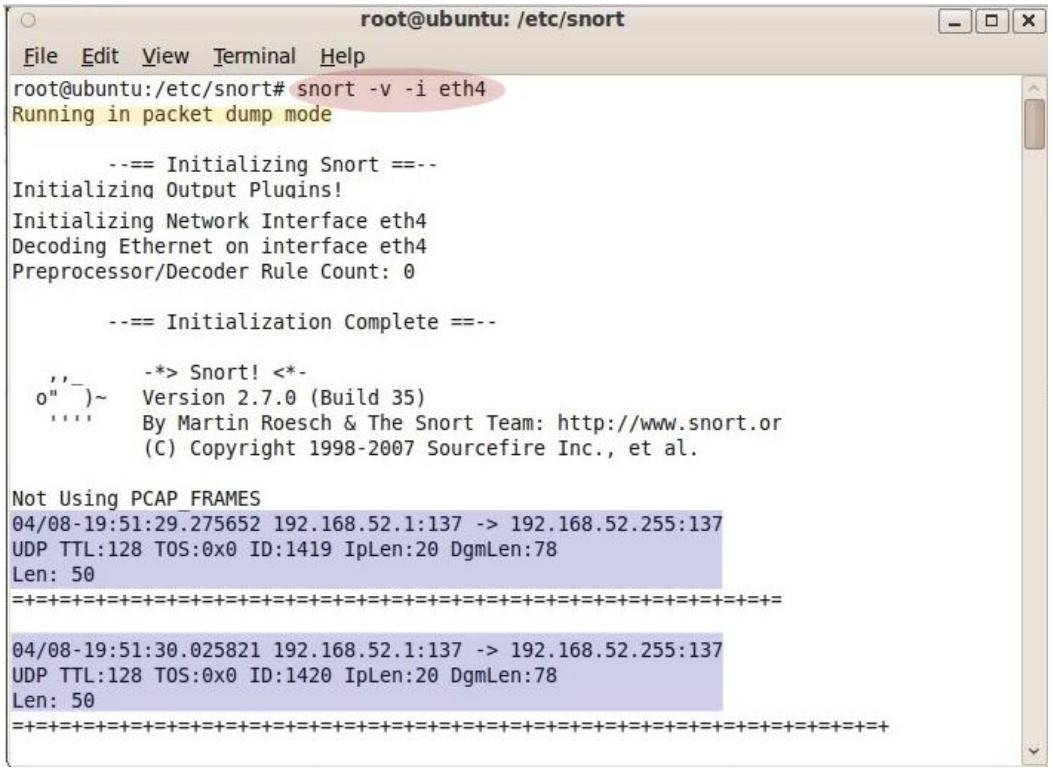

Figura 4-3 Comando ./snort –v para mostrar cabeceras

- Mostrar por consola las cabeceras de los paquetes TCP/IP, UDP, ICMP como se puede observar en la Figura 4-4.

./snort–vd

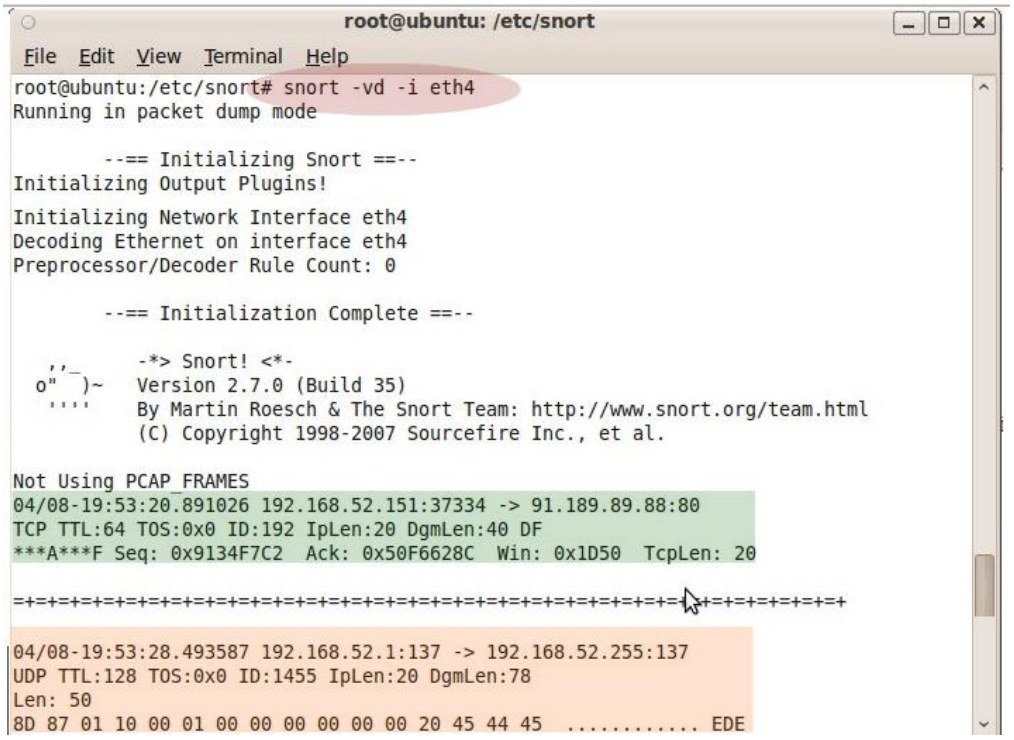

Figura 4-4 Comando ./snort –vd para mostrar cabeceras TCP IP UDP

Modo Packet Logger

Este modo de ejecución es similar al Sniffer Mode, con la diferencia de que en lugar de mostrar los paquetes capturados por consola, estos se almacenan en disco.

Algunas de las acciones que se pueden realizar en este modo son:

- Registrar los paquetes en la carpeta */log* (debe existir una carpeta llamada log en el directorio actual, en caso contrario la ejecución

del comando dará un mensaje de error. También se puede especificar la ruta absoluta de la carpeta)

./snort -dev -l ./log

- Para registrar solo los paquetes de una red en particular debemos especificar su dirección poniendo la opción -h:

./snort -dev -l ./log –I eth2

- Para mostrar por pantalla alguno de estos paquetes registrados (siendo packet.log el nombre del paquete que se muestra por consola) usamos el comando:

./snort -dv -r packet.log

Modo NIDS (Network Intrusion Detection System)

Este modo captura el tráfico de la red y lo analiza para detectar intrusiones. Para ello compara el tráfico con una serie de reglas y registra los paquetes que hagan saltar alguna alarma, es decir, que correspondan con algún comportamiento sospechoso previamente

definido. Es uno de los modos que se usan sistema, se muestra en la Figura 4-5.

Figura 4-5 Snort en modo network IDS

Para lanzar este modo de ejecución usamos el siguiente comando:

./snort -dev –I eth0 -c ./etc/snort.conf – l ./log

Siendo:

- *./log* la carpeta especificada para registrar los paquetes
- *Eth0* la interfaz que Snort va a escuchar
- *Snort.conf* el fichero de configuración de Snort (contiene información relativa a las reglas y a los preprocesadores que se usan). Si el archivo no se encuentra en el directorio de trabajo, se debe especificar la ruta absoluta dentro del sistema de ficheros.

Modo Inline

En este modo de ejecución, se obtienen los paquetes de los Iptables (reglas sobre los diferentes paquetes de una red usadas por el administrador del sistema) y se usan nuevas reglas para determinar si se permite o no el paso de esos paquetes. Es decir, los Iptables utilizarán las reglas de Snort para determinar el paso de los paquetes.

Existen tres tipos de reglas en este modo:

- **Drop.** Los paquetes se dejan pasar o no y se registran mediante los mecanismos usuales de Snort.
- **Reject.** Se realizan las mismas acciones que con las reglas Drop y además se envía un *TCP reset* si el protocolo es el TCP o un *ICMP port unreachable* si el protocolo es el UDP.
- **Sdrop.** Se deja pasar o no a los paquetes, pero no se registra ninguno.

Además, se debe asegurar de que el módulo ip_queue está cargado en el sistema. Para ello se utiliza el siguiente comando:

```
iptables -A OUTPUT -p tcp --dport 80 -j QUEUE
```

Con ello, se podrá utilizar Snort en modo inline mediante el siguiente comando:

snort -QDc ../etc/snort.conf -l /var/log/snort

4.3 Snort como servicio del sistema

Para ejecutar Snort como un servicio más del sistema operativo Linux, se debe crear un script en la ubicación: /etc/init.d. En este lugar se encuentran todos los servicios que se pueden ejecutar en el sistema.

Para este proyecto se elabora un script llamado: snortstartAp. El script contiene toda la información de cómo se debe ejecutar Snort automáticamente y la configuración correspondiente.

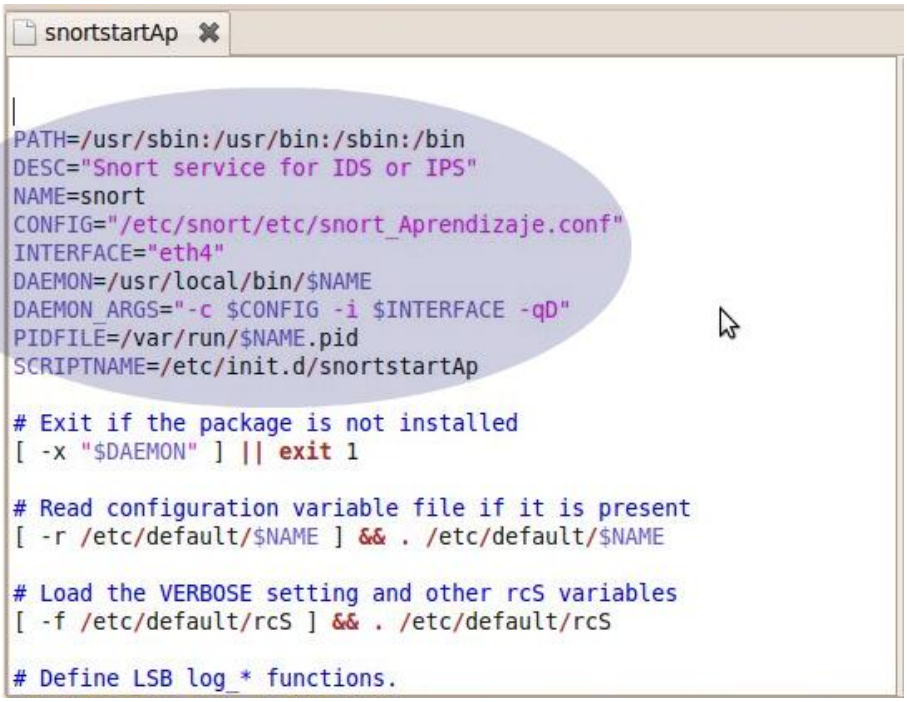

Figura 4-6 Código de Snort como proceso del sistema

En la Figura 4-6 se muestra sección del código que permite que Snort funcione como un proceso del sistema operativo Linux.

PATH=/usr/sbin:/usr/bin:/sbin:/bin

DESC="Snort service for IDS or IPS"

NAME=snort

CONFIG="/etc/snort/etc/snort_Aprendizaje.conf"

INTERFACE="eth0"

DAEMON=/usr/local/bin/$NAME

DAEMON_ARGS="-c $CONFIG -i $INTERFACE -qD"

PIDFILE=/var/run/$NAME.pid

SCRIPTNAME=/etc/init.d/snortstartAp

PATH: es la ubicación del ejecutable de Snort, que se crea en la dirección /usr/bin cuando Snort se ha instalado correctamente.

DESC: es la descripción del servicio del sistema

NAME: nombre del servicio con el cual se ejecuta el servicio como un proceso más del sistema

CONFIG: Hace referencia al archivo de configuración de Snort; que define el modo de ejecución de Snort, en este caso snort_Aprendizaje.conf

INTERFACE: es la interfaz por la cual el sensor de Snort trabaja, si no se define una, por default automáticamente se busca la interfaz inmediata

Por último se definen los usos para ejecutar el script.

Usage: $SCRIPTNAME {start|stop|restart|force-reload}

El proceso snortstartAp se puede iniciar, detener, reiniciar y forzar la recarga del servicio. La Figura 4-7 muestra como se puede ejecutar el servicio de Snort.

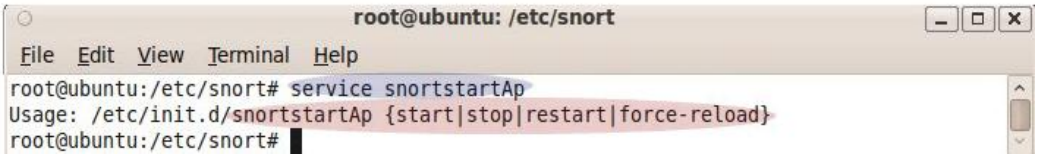

Figura 4-7 Ejecución del servicio de Snort

Para ejecutar el servicio, sólo basta con realizar una llamada al sistema como la siguiente: service snortstartAp start y automáticamente Snort empieza a trabajar y escanear el tráfico de la red.

Este script fue creado con la finalidad de automatizar la ejecución de Snort, evitar el uso de la consola de Linux y el uso de Snort con comandos demasiados complejos. Sólo basta con iniciar el proceso y Snort trabaja de manera automática ya que todo se encuentra configurado. Además es esencial para la aplicación web.

4.4 Implementación modo aprendizaje

El modo aprendizaje al igual que todo el proyecto esta implementado en php debido a que el presente trabajo se deriva de la aplicación acid de la cual ya se hizo referencia en el capítulo 2.

El archivo principal del entrenamiento es aprendizaje.php, el cual contiene el código de las funcionalidades, además de incluir otros archivos muy importantes para el aprendizaje. Para una mejor comprensión del contenido de aprendizaje.php se lo ha resumido en tres secciones que se presentan a continuación:

Sección 1: Creación de Aprendizaje

Dentro del código se detalla el proceso de aprendizaje, donde se debe escoger primero el tiempo de entrenamiento del sistema, la duración de este tiempo se lo selecciona de entre 4, 8, 16 y 24.

Figura 4-8 Tiempo de duración de un aprendizaje

Una vez escogido el tiempo del entrenamiento como se aprecia en la Figura 4-8, se da inicio al aprendizaje, aquí el sistema procede a ejecutar el archivo createLearning.php que se encarga de crear la tabla learnings dentro de la base de datos Snort (en caso de ser la primera vez que se utiliza la aplicación, de otro modo se insertan directamente los datos del nuevo aprendizaje en la tabla mencionada).

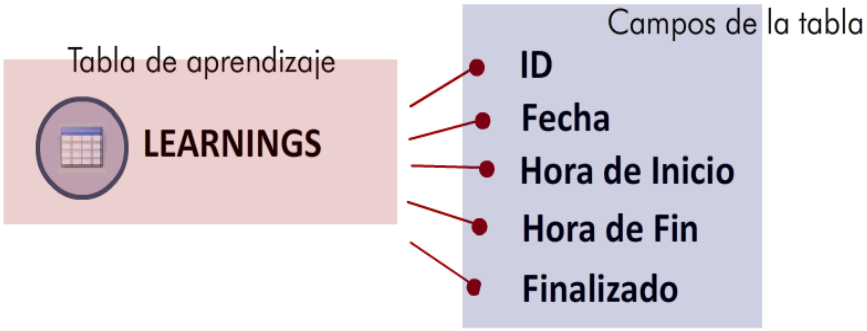

Figura 4-9 Estructura de la tabla learnings

La tabla learnings contiene los campos mostrados en la Figura 4-9 donde cada aprendizaje se lo considera un perfil, que posee un identificador único llamado id, una fecha, una hora de inicio, una hora de fin del aprendizaje y un campo denominado finalizado que inicialmente tiene el valor de cero lo cual representa que en ese momento el sistema está aprendiendo.

El ID de cada perfil es asociado con cinco tablas creadas para complementar todo el proceso del aprendizaje y en las cuales se insertan los datos recogidos por snort

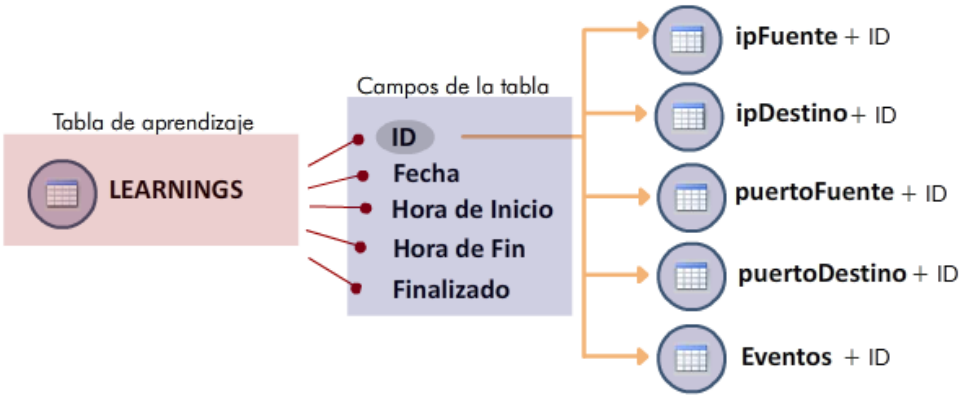

Figura 4-10 Asociación de tablas con el aprendizaje

En la Figura 4-10 se muestra como se asocian las tablas con el identificador del aprendizaje. Los nombres de las tablas son:

```
$table1 = "ipFuente".$id;

$table2 = "ipDestino".$id;

$table3 = "puertoFuente".$id;

$table4 = "puertoDestino".$id;

$table5 = "eventos".$id;
```

Donde $id es el identificador único de cada aprendizaje o perfil.

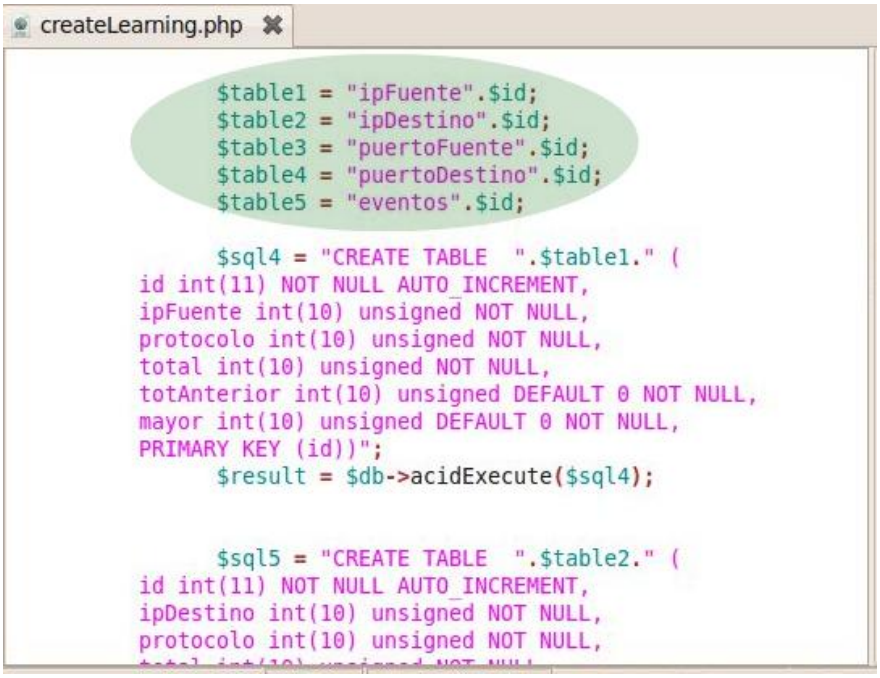

Figura 4-11 Código de elaboración de tablas

Como se aprecia en el código que se ilustra en la Figura 4-11 los nombres de las tablas que se crean con cada perfil llevan en su nombre el identificador del aprendizaje con el cual se relacionan, por ejemplo:

Si learnings posee un aprendizaje con id=27, las tablas que se crean tendrán los siguientes nombres:

"ipFuente27"

"ipDestino27"

"puertoFuente27"

"puertoDestino27"

"eventos27"

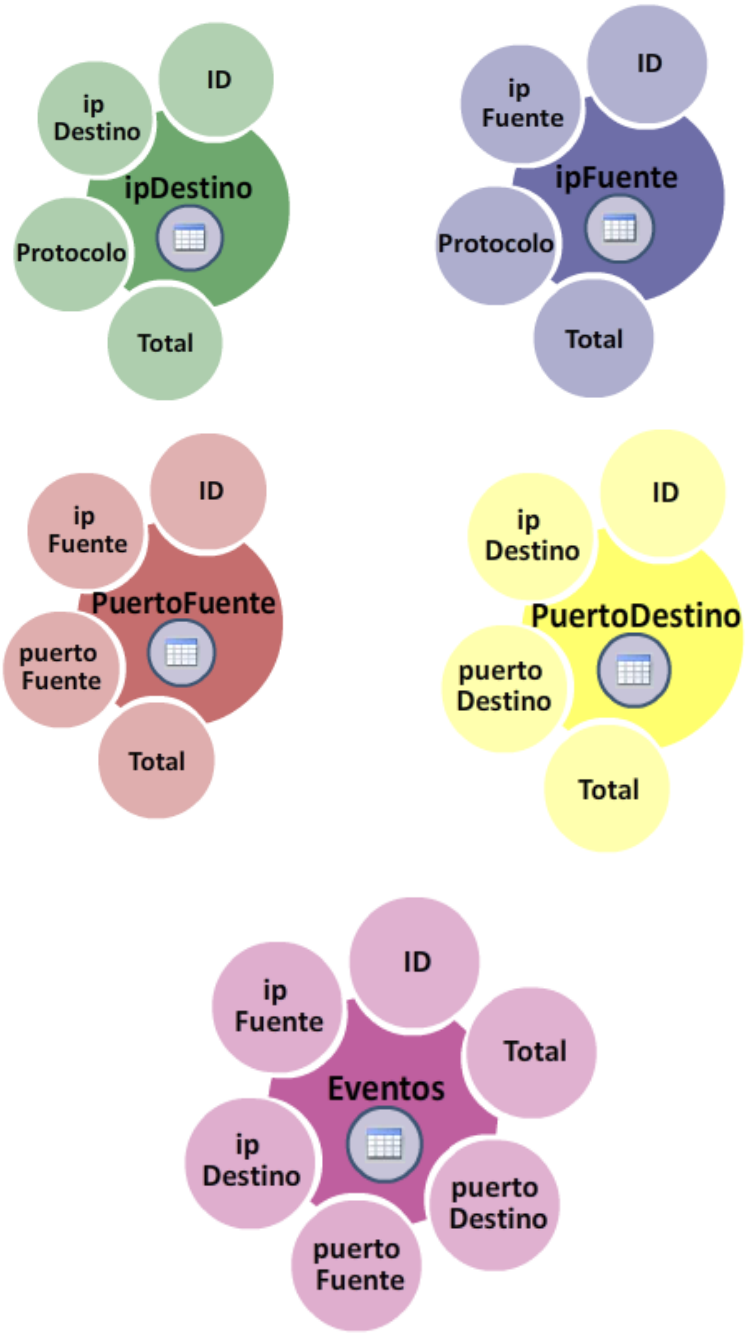

Figura 4-12 Tablas ipFuente, ipDestino, puertoFuente, puertoDestino y eventos

La Figura 4-12 muestra los campos de las cinco tablas, en las cuales se insertan o actualizan registros mediante un procedimiento almacenado que se ejecuta automáticamente antes de insertar algún registro en la tabla acid_event (tabla originaria de la aplicación acid que contiene campos como ip fuente, ip destino, protocolo, puerto fuente, puerto destino) debido a que estos registros se los redistribuye a las tablas ya mencionadas.

createLearning.php

```
$sql9 = "CREATE TRIGGER Aprendizaje BEFORE INSERT ON acid_event
FOR EACH ROW BEGIN

DECLARE num_row1 INTEGER;
DECLARE num_row2 INTEGER;
DECLARE num_row3 INTEGER;
DECLARE num_row4 INTEGER;
DECLARE num_row5 INTEGER;


SELECT COUNT(*)
INTO num_row1
FROM ".$table1."
WHERE ipFuente=NEW.ip_src and protocolo=NEW.ip_proto;
IF num_row1> 0 THEN
UPDATE ".$table1."
set total = (total +1) where ipFuente = NEW.ip_src and protocolo=NEW.ip_proto;
ELSE
INSERT INTO ".$table1."(ipFuente,protocolo,total) VALUES(NEW.ip_src,NEW.ip_proto,1);
END IF;
```

Figura 4-13 Código que crear un aprendizaje

En la Figura 4-13 se detalla la sección de código donde se crea el procedimiento almacenado incluyendo también como se inserta o actualiza según sea el caso las tablas previamente creadas.

Cada una de las cinco tablas posee un campo total que representa un contador que se inicializa o aumenta según se muestra a continuación con el ejemplo preliminar:

Para cada nuevo registro que pretende ser insertado en la tabla ipFuente27, previamente se compara en ipFuente y protocolo para determinar si existe otro registro con los mismos valores en esos campos, y de ser ese el caso se actualiza el registro ya contenido en la tabla aumentando en uno el campo total, caso contrario se inserta el nuevo registro y el campo total se inicializa en uno.

Así mismo, para las demás tablas del ejemplo se lleva de manera similar el proceso, pero con la diferencia en los campos de comparación que se presentan a continuación con su respectiva tabla:

Tabla ipDestino27, ipDestino y protocolo

Tabla puertoFuente27, puertoFuente e ipFuente

Tabla puertoDestino27, puertoDestino e ipDestino

Tabla eventos27, ipFuente, ipDestino, puertoFuente y puertoDestino

Por otro lado, Snort recoge los datos del tráfico del cual está aprendiendo mediante la ejecución del servicio snortstartAp:

```
exec('sudo service snortstartAp start',$salida);
```

La página se actualiza cada minuto mostrando los resultados

Sección 2: Detener Aprendizaje

También se presenta la opción de detener el aprendizaje, proceso que se establece mediante el archivo stopLearning.php cuyas funciones son finalizar el aprendizaje, actualizar el campo del correspondiente registro de la tabla learnings a finalizado=1, y eliminar el procedimiento almacenado creado inicialmente. La acción que se ejecuta para detener el servicio es la siguiente:

```
exec('sudo service snortstartAp stop',$salida);
```

Sección 3: Graficar Tablas

Muestra las cinco tablas de alertas del último aprendizaje e incluyen también los siguientes archivos que permiten graficar dichas tablas:

tablaEventos.php

Dependiendo del aprendizaje y siguiendo con el ejemplo previo, se extrae de la base de datos los registros de la tabla eventos27 cuyas ip fuentes, puertos fuentes, ip destinos y puertos destinos no consten en las tablas eventosBloqueados y eventosPermitidos. Una vez obtenidos

los registros se los incorpora a la interfaz para mostrarlos en la tabla Alertas de Eventos, que contiene los datos de:

IP Fuente, correspondiente al campo ipFuente de cada registro obtenido.

Puerto Fuente, correspondiente al campo puertoFuente de cada registro obtenido.

IP Destino, correspondiente al campo ipDestino de cada registro obtenido.

Puerto Destino, correspondiente al campo puertoDestino de cada registro obtenido.

De igual forma, las demás tablas de los archivos php tienen semejanza, pero presentan diferencias en los campos que extraen de sus respectivas tablas, además de las tablas de bloqueados y permitidos como se observa a continuación:

tablaIpFuente.php

Campos: ipFuente y protocolo,

Tabla de extracción de registros: ipFuente27,

Otras tablas necesarias: ipFuenteBloqueados e ipFuentePermitidos

tablaIpDestino.php

Campos: ipDestino y protocolo,

Tabla de extracción de registros: ipDestino27,

Otras tablas necesarias: ipDestinoBloqueados e ipDestinoPermitidos

tablaPuertoFuente.php

Campos: puertoFuente e ipFuente,

Tabla de extracción de registros: puertoFuente27,

Otras tablas necesarias: puertoFuenteBloqueados y puertoFuentePermitidos

tablaPuertoDestino.php

Campos: puertoDestino e ipDestino,

Tabla de extracción de registros: puertoDestino27,

Otras tablas necesarias: puertoDestinoBloqueados y puertoDestinoPermitidos

Además, todas las tablas que se presentan en la interfaz, también muestran los siguientes campos:

Ocurrencias en Aprendizaje, corresponde al campo total, lo que representa el número de conexiones de cada registro.

Ocurrencias en Aprendizaje %, es la relación porcentual entre el valor del campo total de cada registro y la suma de los valores de los campos total de todos los registros, es decir, es el porcentaje del número de conexiones de cada registro relativo al número de conexiones de todos los registros.

MNCM (Máximo Número de Conexiones por Minuto), corresponde al campo mayor de cada registro, siendo equivalente al mayor resultado de la diferencia de los campos total y totAnterior(campo que contiene el valor que tuvo el campo total un minuto antes) durante cada minuto. En otras palabras, se trata del número máximo de conexiones por minuto que puede alcanzar un registro.

Es posible restaurar o bloquear las ip que han sido bloqueadas o permitidas respectivamente durante todo el proceso de aprendizaje o de detección con la opción Control de Ip que se encuentra en el menú principal. Esta alternativa se ayuda del archivo ipControl.php para presentar un control tanto de permitidos como de bloqueos.

En el Control de Permitidos se incluyen los archivos php con su correspondiente tabla de la base de datos snort:

tablaEventosPer.php, tabla eventosPermitidos

tablaIpFuentePer.php, tabla ipFuentePermitidos

tablaPuertoFuentePer.php, tabla puertoFuentePermitidos

tablaIpDestinoPer.php, tabla ipDestinoPermitidos

tablaPuertoDestinoPer.php, tabla puertoDestinoPermitidos

En el Control de Bloqueos se incluyen los archivos php con su correspondiente tabla de la base de datos snort:

tablaEventosBlo.php, tabla eventosBloqueados

tablaIpFuenteBlo.php, tabla ipFuenteBloqueados

tablaPuertoFuenteBlo.php, tabla puertoFuenteBloqueados

tablaIpDestinoBlo.php, tabla ipDestinoBloqueados

tablaPuertoDestinoBlo.php, tabla puertoDestinoBloqueados

4.5 Implementación modo de detección

A continuación se presenta como fue elaborado el modo de detección. En esta sección se destaca la comparación entre perfiles y las alertas que se disparan cuando se encuentra alguna anomalía en la red.

Para preparar la detección de anomalías de una red, primero se debe seleccionar un perfil de entrenamiento o aprendizaje creado previamente. Es por eso que se presentan todos los perfiles disponibles con los cuales se va a comparar el tráfico común.

Figura 4-14 Perfiles disponibles para la detección

Inicialmente y como se grafica en la Figura 4-14 se muestran todos los perfiles con la fecha de creación, la hora de inicio y la duración del

tráfico escaneado en horas. Además se presenta la opción que tiene el administrador de controlar el bloqueo de las anomalías de manera manual o automática, la cual se detalla en el próximo subcapítulo.

deteccion.php

```
<?php
        $sql = 'show tables';
        $result = $db->acidExecute($sql);
        $num = $result->acidRecordCount();
        $flag = 0;
        for ($i = 0; $i < $num; $i++)
        {
                $myrow = $result->acidFetchRow();
                if($myrow[0]=='learnings')
                        $flag=1;
        }
        if($flag==1)
        {
                $sql = 'select * from learnings';
                $result = $db->acidExecute($sql);
                $num = $result->acidRecordCount();
                if($num>0){
                $result = mysql_query("select id from learnings order by id DESC
                limit 1");
                $resultado = mysql_result ($result, 0);
                $id = $resultado;
                $sql = 'select fecha,horainicio,horafin,id from learnings WHERE
                finalizado=1 order by id desc';

                $result = $db->acidExecute($sql);
                $num = $result->acidRecordCount();
```

Figura 4-15 Código detección.php

En el código del archivo detección.php que se observa en la Figura 4-15 se muestra como se cargan los perfiles, buscándolos y extrayéndolos desde la base de datos de Snort.

Se extrae información de la tabla learnings, en la cual se encuentra el aprendizaje. Los datos son: fecha, hora de inicio, hora fin y el identificador.

Existe un campo denominado finalizado, que tiene un valor de 1 cuando el aprendizaje ha terminado con éxito y tomar el valor de 0 cuando el aprendizaje no ha terminado, debido a que puede permanecer aprendiendo mientras se desea detectar anomalías.

```
$sql = ‘select fecha, horainicio, horafin, id from learnings WHERE finalizado = 1 order by desc’
```

Por ende solo se cargan los perfiles que hayan culminado con éxito su tiempo en aprendizaje ya sean las 4, 8, 16 o 24 horas.

Una vez seleccionado el aprendizaje, se da inicio a la detección. Se realiza de manera similar a como se crea un aprendizaje. Dentro del código de createDetection,php se crea un perfil de detección, que compara el tráfico común y el aprendizaje de una manera más rápida y exacta. El único parámetro necesario para createDetection.php es el perfil base, que contiene el identificador ID del campo de la tabla learnings y el tipo de limpieza que se desea realizar en caso de surgir alertas de anomalías. De esta manera, con el ID del aprendizaje el sistema procede a comparar los tráficos de entrenamiento con los que recibe en tiempo real.

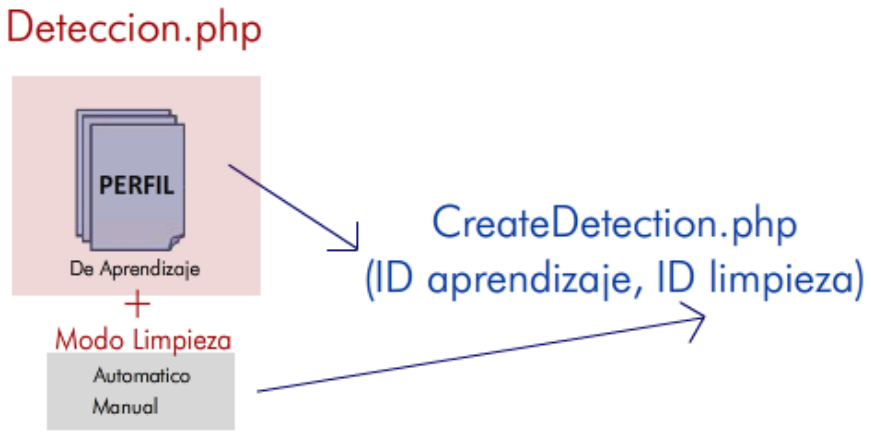

Figura 4-16 Parámetros para iniciar la detección

Así como en la Figura 4-16, CreateDetection.php con sus parámetros de perfil y de limpieza procede a crear el denominado perfil de detección, un perfil muy similar al perfil de entrenamiento. Aquí se almacena todos los datos del tráfico en distintas tablas para proceder a compararlas.

Se crea la tabla detección

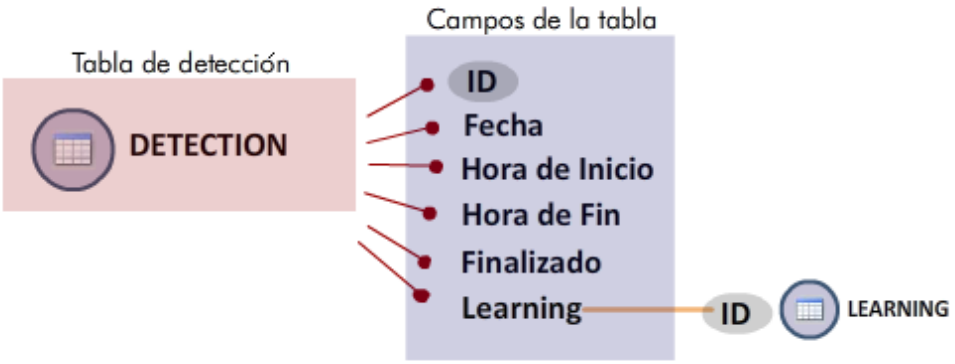

Figura 4-17 Estructura de la tabla detección

Como se detalla en la Figura 4-17, los campos de la tabla detección son: id, fecha, hora de inicio, hora de fin, finalizado que determina si la detección sigue en ejecución, y el parámetro más importante, el ID, que se extrae desde que inicia el proceso de detección, y que tiene de referencia al aprendizaje de la base de conocimientos con el cual se compara, proveniente y almacenado en la tabla learnings de la base de Snort.

Figura 4-18 Código que crea la estructura de una detección

Una vez creada la tabla general de detección, la cual se observa en su código en la Figura 4-18, se procede a crear tablas dentro de la base de datos de snort, para almacenar el tráfico que se detecta, donde se lo clasifica por ips fuente, ips destino, puertos fuente, puertos destino, protocolos y eventos generales.

$table1 = "ipFuenteDet".$id;

$table2 = "ipDestinoDet".$id;

$table3 = "puertoFuenteDet".$id;

$table4 = "puertoDestinoDet".$id;

$table5 = "eventosDet".$id;

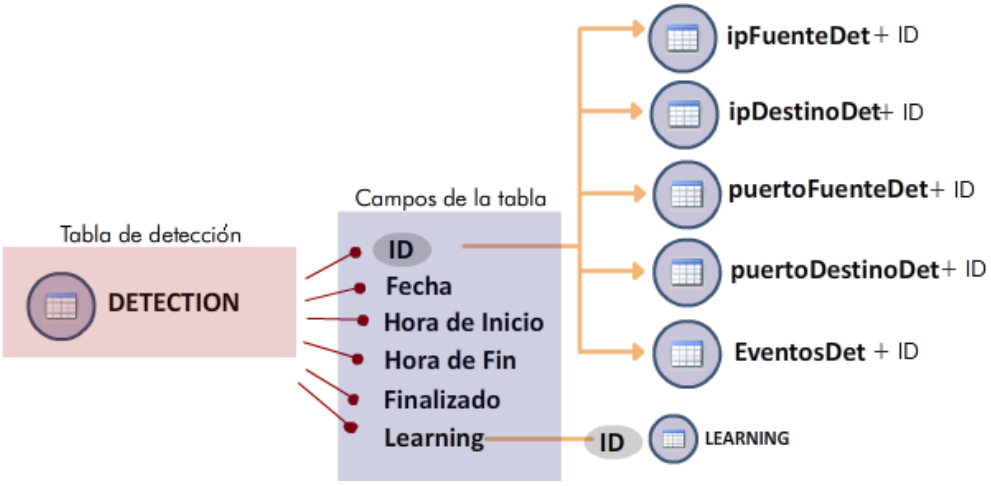

Figura 4-19 Estructura general y elementos que posee la detección

Luego de resumir la estructura global de la detección como se muestra en la Figura 4-19, se detalla cómo funciona el proceso de comparación que se lleva a cabo de acuerdo cada archivo php explicado a continuación:

tablaEventosDet.php

Por un lado se extrae la información de la tabla eventosDet que fue creado en el momento de iniciar la detección y por otro se extraen los

registros de la tabla eventos del perfil escogido con el cual luego se realiza la comparación. Para ambos casos se determina un promedio como se detalla a continuación:

Promedio de detección, es la relación porcentual entre el campo total de un registro y la suma de los campos total de todos los registros de la tabla eventosDet, es decir, es el porcentaje del número de conexiones de un registro con respecto al número total de conexiones de todos los registros.

Promedio de aprendizaje, es la misma relación porcentual que el promedio de Detección con la diferencia de que los registros son extraídos de la tabla eventos.

Para realizar la comparación primero se plantea un índice de riesgo, el cual determina el peligro que podría albergar una conexión, y que inicialmente tiene un valor de cero.

Dentro del código, la variable del índice de riesgo tiene el nombre de $porcentajeC, la cual puede tomar diferentes valores de acuerdo a ciertas condiciones excluyentes que se explican a continuación:

Si el promedio de aprendizaje es mayor a cero y el promedio de detección es mayor al promedio de aprendizaje entonces el $porcentajeC = porcentajeC + 0.3;

Si el campo total del registro correspondiente a la tabla eventosDet es mayor al campo total del registro correspondiente a la tabla eventos entonces el $porcentajeC = porcentajeC + 0.3;

Si el número de conexiones por minuto del registro correspondiente a la tabla eventosDet es superior al número de conexiones por minuto del registro correspondiente a la tabla eventos entonces el $porcentajeC = porcentajeC + 0.4;

En otras palabras, los valores mencionados del promedio, del campo total y del campo mayor del aprendizaje representan umbrales que al ser superados por los correspondientes valores de detección producen que el índice de riesgo aumente y de acuerdo a ello se podrán bloquear manual o automáticamente ips y puertos respectivos.

El procedimiento anterior se generaliza para los cuatro restantes archivos php y sus correspondientes tablas de detección y de aprendizaje de la base de datos como se muestra:

tablaIpFuenteDet.php, tabla ipFuenteDet, tabla ipFuente

tablaIpDestinoDet.php, tabla ipDestinoDet, tabla ipDestino

tablaPuertoFuenteDet.php, tabla puertoFuenteDet, tabla puertoFuente

tablaPuertoDestinoDet.php, tabla puertoDestinoDet, tabla puertoDestino

4.6 Implementación limpieza de anomalías

El proceso en el cual se bloquean o permiten ciertas conexiones, se denomina modo de limpieza.

Mientras se realiza el proceso de entrenamiento o aprendizaje en tiempo real, la aplicación muestra como Snort va adquiriendo conocimiento del tráfico, clasificando toda la información acorde a su dirección ip, puertos, protocolos o eventos generales como se muestra en la Figura 4-20

Figura 4-20 Conexiones durante el aprendizaje con opciones de bloqueo o permitir

Existe la opción de bloquear o permitir una conexión. En este caso se realiza de manera opcional, debido a que se puede dar el caso de que

durante el entrenamiento el administrador de la red desee omitir cualquier ip para que no sea contada durante el entrenamiento y el sistema no la tome en cuenta desde el inicio.

La limpieza de anomalías empieza cuando se va a realizar una detección. Mediante el proceso de detección se encuentran anomalías en las redes y es donde se toman decisiones.

Figura 4-21 Tipo de Bloqueo

Teniendo dos opciones primordiales para limpiar las anomalías de las redes como se presenta en la Figura 4-21. La primera alternativa es la limpieza automática, donde cualquier conexión que genere una alerta, se procede a bloquear automáticamente. Para la segunda opción, la limpieza es manual, donde un administrador de red puede observar una alerta disparada por una conexión anómala, y tomar una decisión, es decir, posee la última palabra para eliminarla.

Cuando una alerta se genera, ésta se presenta en su tabla correspondiente, mostrando toda su información. Si la alternativa seleccionada es manual, se presenta la opción de bloqueo de una conexión como se muestra en la Figura 4-22.

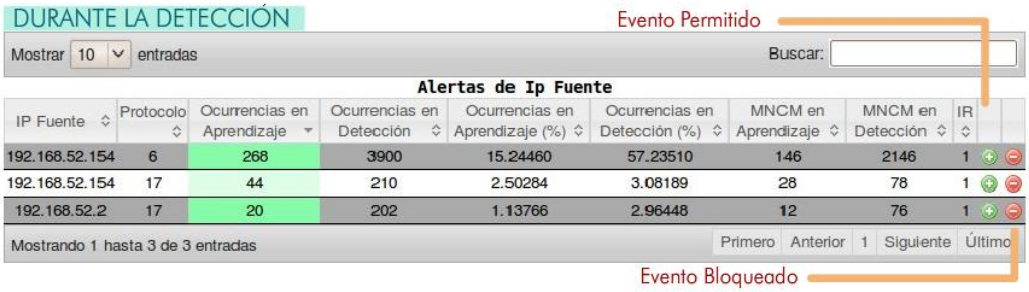

Figura 4-22 Opción de bloqueo durante el proceso de detección de anomalías

Cada botón hace referencia a una función de bloqueo o permisos dentro de las páginas permitirEvento.php y bloquearEvento.php respectivamente como se presenta en el código de la Figura 4-23.

Adicionalmente se envían todos los parámetros correspondientes de la conexión maliciosa.

```
<TD><a href="permitirEvento.php?ipFuente='.$myrow[0].'&ipDestino='.$myrow[2].'&puertoFuente='.$myrow[1].'&puertoDestino='
        [3].'&url=aprendizaje.php"><img src="edit.png" alt="permitir" border="0"/></a></TD>

<TD><a href="bloquearEvento.php?ipFuente='.$myrow[0].'&ipDestino='.$myrow[2].'&puertoFuente='.$myrow[1].'&puertoDestino='
        [3].'&url=aprendizaje.php"><img src="delete.png" alt="Bloquear" border="0"/></a></TD>
                </TR>';
```

Figura 4-23 Código de parámetros de bloqueo

El archivo bloquearEvento.php se encarga de almacenar toda la información de la conexión a la base de datos para proceder a eliminar las conexiones y tener una constancia de cuales han sido las ips que generan anomalías, y si en algún momento se desea volver a permitir dicha conexión en la red el sistema presenta la opción de control de ips.

Los datos almacenados en la tabla eventosBloqueados de la base de datos snort y que se grafican en la Figura 4-24 son: ip fuente, ip destino, puerto fuente, puerto destino.

Figura 4-24 Estructura de la tabla eventos bloqueados

Una conexión anómala pasa a ser insertada en la tabla eventos bloqueados como se muestra en el código de la Figura 4-25.

bloquearEvento.php

```
        $sql = "insert into eventosBloqueados
(ipFuente,ipDestino,puertoFuente,puertoDestino) values('$_GET
[ipFuente]','$_GET[ipDestino]','$_GET[puertoFuente]','$_GET
[puertoDestino]')";
        $result = $db->acidExecute($sql);

        $sql2 = "select ip_proto from acid_event where ip_src=".$_GET
['ipFuente']." and ip_dst=".$_GET['ipDestino']." and layer4_sport=".$_GET
['puertoFuente']." and layer4_dport=".$_GET['puertoDestino']."";

        $result2 = $db->acidExecute($sql2);

        $myrow = $result2->acidFetchRow();

        if($myrow[0]==6)
                $proto = 'tcp';
        else if($myrow[0]==17)
                $proto = 'udp';
        else
                $proto = 'icmp';
```

Figura 4-25 Código que inserta una conexión y la etiqueta como bloqueada

$SQL="insert into eventosBloqueados (ipFuente, ipDestino, puertoFuente, puertoDestino) values('$_GET[ipFuente]',' $_GET[ipDestino]',' $_GET[puertoFuente]',' $_GET[puertoDestino]')";

Luego se procede a bloquearla mediante el firewall con el uso de iptables, creando reglas y modificando la configuración del firewall.

Como parámetro se ejecuta la siguiente sentencia que escribe la regla para el firewall.

```
exec("sudo /sbin/iptables -I INPUT -s ".acidLong2IP($_GET[ipFuente])." -p ".$proto." --sport ". $_GET[puertoFuente]." -d ".acidLong2IP($_GET[ipDestino])." --dport ". $_GET[puertoDestino]." -j DROP");
```

Donde la sentencia general está dada por:

Iptables –I INPUT -s I*pFuente* -p *Protocolo* --sport *PuertoFuen* –dport *PuertoDest* –j DROP

Se escribe la regla cambiando los valores de ipFuente, proto, puertoFuente, ipDestino, puertoDestino, por los valores de la conexión maliciosa almacenada en la base de datos de snort en la tabla eventosBloqueados.

Para la administración de bloqueo y permiso de direcciones ip, el módulo de limpieza consta con una sección donde el administrador puede examinar todas las conexiones que han sido eliminadas por violar reglas de anomalías así como también se pueden observar las direcciones ip que tienen permisos y que no son contadas en la detección.

La administración de bloqueo es llamada ipControl como se muestra en la Figura 4-26.

Figura 4-26 Control de direcciones ips bloqueadas o permitidas

Como en la Figura 4-27 se puede verificar que las conexiones han sido bloqueadas revisando en las reglas de bloqueo que posee iptables; escribiendo en la consola de Linux la sentencia: iptables –nL la cual muestra todas las reglas y sus respectivas restricciones.

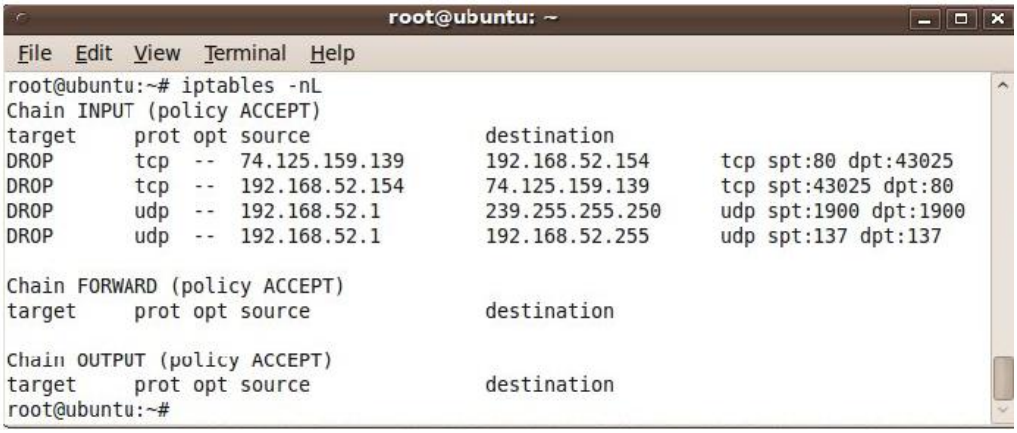

Figura 4-27 Conexiones en la reglas de iptables

CAPÍTULO 5

5. ANÁLISIS DE RESULTADOS

5.1 Metodología de las pruebas

Con el objetivo de evaluar la herramienta propuesta y desarrollada, se planificó un conjunto de pruebas que se describen en esta sección.

Para la ejecución del análisis de forma distribuida se planteo realizar dos pruebas para demostrar el uso de la herramienta en ambos modos, aprendizaje y detección.

Para realizar las pruebas, se obtuvo de la empresa TELCONET por medio del departamento CERT un tráfico real de un cliente.

Cabe recalcar que se ha firmado un acuerdo de confidencialidad acerca de los datos obtenidos del cliente por motivos de reglas y politicas de TELCONET.

El tráfico solicitado ha sido obtenido en modo sniffer, realizando las pruebas los días 19 de Abril del 2011 y el 20 de Abril del 2011. En ambos días escaneando el tráfico desde las 4 PM hasta las 5 PM al cliente con un ancho de banda de 512kbits

5.2 Resultados

Una vez concluido el primer proceso de entrenamiento en el primer día, se recolectaron los datos y se procedió a realizar los gráficos e inferir resultados a partir de los mismos.

Al finalizar el aprendizaje se muestran un top 10 de las conexiones más concurridas por varios factores: eventos generales, ip fuente, ip destino y puertos.

Alertas de Eventos

IP Fuente	Puerto Fuente	IP Destino	Puerto Destino	Ocurrencias en Aprendizaje	Ocurrencias en Aprendizaje (%)	MNCM
172.21.0.156	8193	172.21.0.157	2142	60041	35.23719	19256
172.21.0.154	8193	172.21.0.155	2142	53278	31.26808	28038
172.21.0.44	8193	172.21.0.45	2142	2872	1.68554	392
172.21.0.52	8225	172.21.0.51	2142	2196	1.28880	396
172.21.0.50	66	172.21.0.51	2142	2190	1.28528	402
172.21.0.50	2	172.21.0.51	2142	2188	1.28411	400
172.21.0.130	8193	172.21.0.131	2142	1886	1.10687	312
172.21.0.130	8225	172.21.0.131	2142	1866	1.09513	310
10.11.16.177	646	224.0.0.2	646	1032	0.60567	80
192.168.11.1	646	224.0.0.2	646	1032	0.60567	78

Tabla 5-1 Resultado de 10 conexiones de mayor ocurrencia en aprendizaje

Alertas de Ip Fuente

IP Fuente	Protocolo	Ocurrencias en Aprendizaje	Ocurrencias en Aprendizaje (%)	MNCM
172.21.0.156	17	60045	35.23954	33220
172.21.0.154	17	53278	31.26808	47943
172.21.0.50	17	4380	2.57056	2404
172.21.0.130	17	3754	2.20317	1884
172.21.0.44	17	2872	1.68554	1454
172.21.0.52	17	2196	1.28880	1194
192.168.52.1	17	1284	0.75356	652
172.26.65.65	1	1146	0.67257	576
10.11.16.177	17	1032	0.60567	516
10.33.23.4	17	1032	0.60567	518

Tabla 5-2 Resultado de 10 direcciones ip Fuente de mayor ocurrencia en aprendizaje

Alertas de Puertos Fuentes

IP Fuente	Puerto Fuente	Ocurrencias en Aprendizaje	Ocurrencias en Aprendizaje (%)	MNCM
172.21.0.156	8193	60041	35.23719	33218
172.21.0.154	8193	53278	31.26808	47943
172.21.0.44	8193	2872	1.68554	1454
172.21.0.52	8225	2196	1.28880	1194
172.21.0.50	66	2190	1.28528	1200
172.21.0.50	2	2188	1.28411	1202
172.21.0.130	8193	1886	1.10687	948
172.21.0.130	8225	1866	1.09513	934
172.26.65.65		1146	0.67257	576
10.11.16.177	646	1032	0.60567	516

Tabla 5-3 Resultado de 10 direcciones ip Fuente y puerto fuente de mayor ocurrencia en aprendizaje

Alertas de Ip Destino

IP Destino	Protocolo	Ocurrencias en Aprendizaje	Ocurrencias en Aprendizaje (%)	MNCM
172.21.0.157	17	60043	35.23836	33220
172.21.0.155	17	53278	31.26808	47943
224.0.0.2	17	31536	18.50802	15774
172.21.0.51	17	6576	3.85936	3598
172.21.0.131	17	3752	2.20199	1882
172.21.0.45	17	2872	1.68554	1454
239.255.255.250	17	1072	0.62914	552
10.164.191.253	17	970	0.56928	572
172.27.9.50	1	964	0.56576	486
10.20.200.151	6	790	0.46364	768

Tabla 5-4 Resultado de 10 direcciones ip Destino de mayor ocurrencia en aprendizaje

Alertas de Puertos Destinos

IP Destino	Puerto Destino	Ocurrencias en Aprendizaje	Ocurrencias en Aprendizaje (%)	MNCM
172.21.0.157	2142	60043	35.23836	33220
172.21.0.155	2142	53278	31.26808	47943
224.0.0.2	646	26706	15.67336	13354
172.21.0.51	2142	6576	3.85936	3598
224.0.0.2	1985	4830	2.83466	2428
172.21.0.131	2142	3752	2.20199	1882
172.21.0.45	2142	2872	1.68554	1454
239.255.255.250	1900	1056	0.61975	540
172.27.9.50		964	0.56576	486
10.20.200.151	4209	790	0.46364	768

Tabla 5-5 Resultado de 10 direcciones ip destino y puerto destino de mayor ocurrencia en aprendizaje

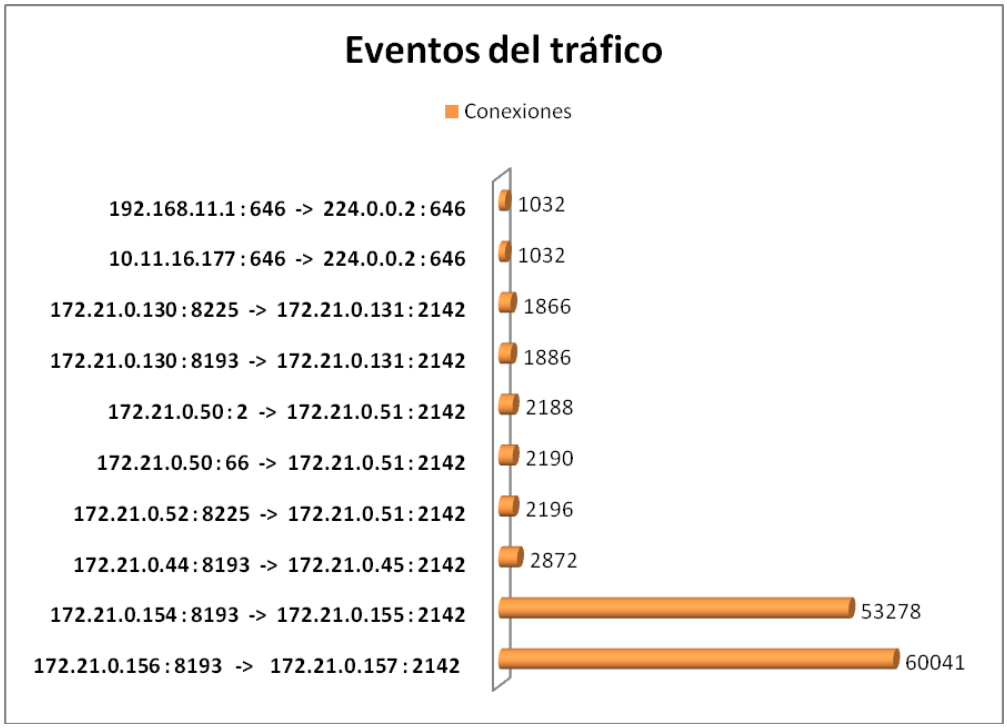

Figura 5-1 Conexiones sobresalientes del tráfico

De la figura 5-1 se puede observar que la dirección fuente 172.21.0.156 : 8194 hacia la dirección destino 172.21.0.157 : 2142 posee el mayor número de ocurrencias en la red, con un valor de 60041 es la mayor cantidad aprendida por snort. Cabe recalcar el exceso de esta conexión al ser uno de los nodos de máximo uso del cliente.

Existen un sin número de conexiones que snort aprendió de los paquetes que han ingresado. Aproximadamente unas 1800 conexiones distintas, ya sean estas variaciones por puertos e IPs. El

número de paquetes aprendidos se eleva aproximadamente a 1’233.775.

Acorde con esta información se construye un umbral de referencia para el proceso posterior de detección y de ser un tráfico normal, no debe sobrepasar en exceso las conexiones mostradas en las tablas.

Bajo las mismas condiciones del tráfico aprendido, se procedió el siguiente dia a realizar la misma prueba, con la diferencia de que se buscó anomalías en la red.

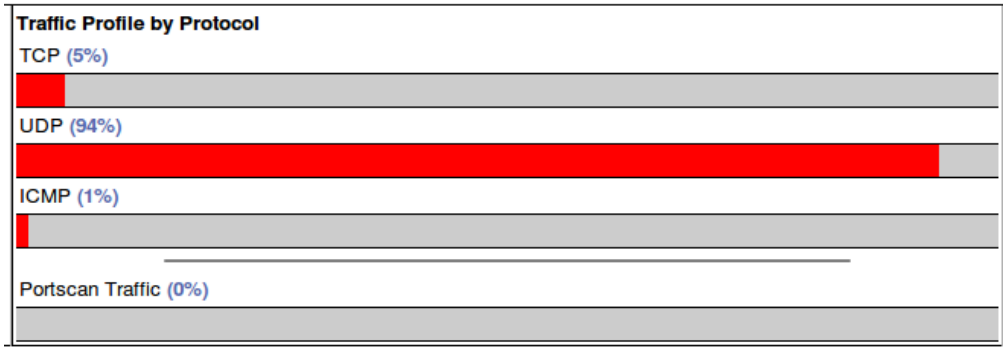

Figura 5-2 Tráfico escaneado en modo detección

Como breve resumen, se puede observar el perfil del tráfico general, dividido en un 5% TCP, 94% UDP y un mínimo de 1% para el tráfico ICMP; bajo estas condiciones se presentan las siguientes detecciones de anomalías

Alertas de Eventos

IP Fuente	Puerto Fuente	IP Destino	Puerto Destino	Ocurrencias en Aprendizaje	Ocurrencias en Detección	Ocurrencias en Aprendizaje (%)	Ocurrencias en Detección (%)	MNCM en Aprendizaje	MNCM en Detección	IR
192.168.52.1	55597	172.21.0.156	3702	57	214	0.60567	1.10687	46	93	1
192.168.23.3		190.95.190.148		38	80	0.00117	0.01740	28	78	1
192.168.52.1	60309	239.255.255.250	1900	120	96	0.07043	0.41750	12	24	0.7
192.168.52.1	138	192.168.52.255	138	72	42	0.04226	0.18266	2	10	0.7
172.26.2.211	138	172.26.2.255	138	18	10	0.01056	0.04349	2	6	0.7

Tabla 5-6 Resultado de las anomalías detectadas

La tabla 5-6 muestra las anomalías registradas en el tráfico acorde con lo que Snort aprendió un día anterior.

Con un índice de riesgo (IR) igual a 1 se muestran las anomalías más destacadas e importantes las cuales se procedieron a eliminar por ser más graves, así también se tiene anomalías con un índice de riesgo de 0.7 de gran importancia.

En las Figuras 5-2,3 se puede apreciar porque la dirección fuente 192.168.52.1 genera una alerta de anomalía hacia la dirección ip 172.21.0.156

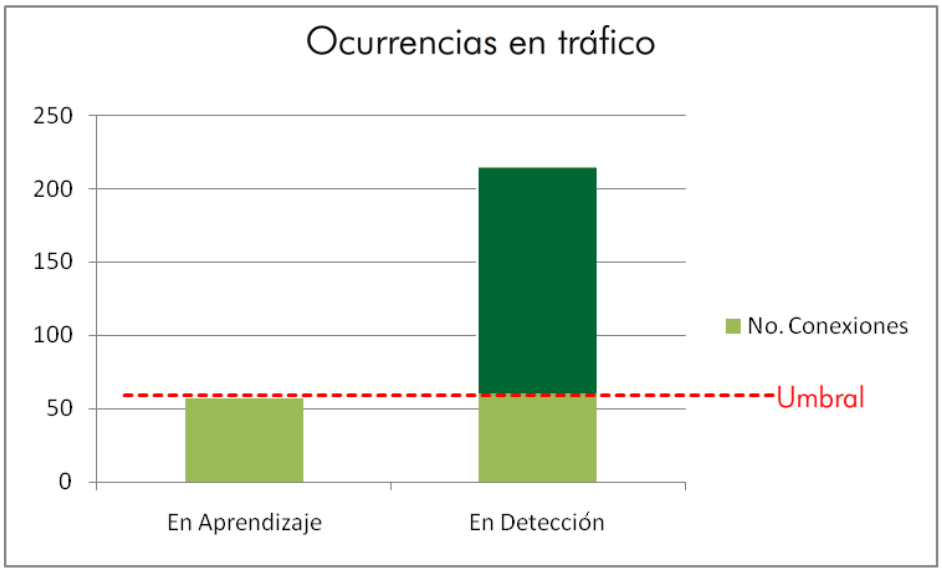

Figura 5-3 Comparación de umbrales en ocurrencias de tráfico

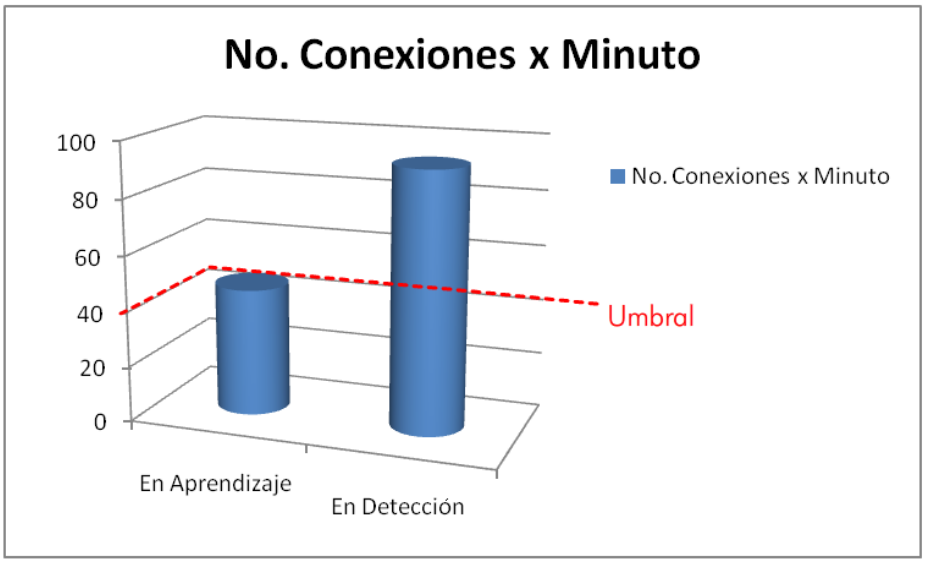

Figura 5-4 Comparación de umbrales en No. conexiones por minuto de tráfico

En ambos casos la conexión 192.168.52.1 : 55597 → 172.21.0.156 : 3702 supera el umbral establecido por el entrenamiento, una vez que sobrepasa las condiciones de los umbrales de número de conexiones máximas,

porcentaje en el tráfico de la red y el mayor número de conexiones por minuto, se puede afirmar que dicha conexión es anómala con un índice de riesgo de 1.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Con las apariciones de nuevas formas de ataque no basta con la prevención de un IDS/IPS basado solo en firmas o patrones conocidos, esta implementación basada en comportamientos permite sólo una protección adicional a la red.

2. Utilizar este módulo no asegura que una alerta generada sea efectivamente un ataque a la red, puede que se genere un falso positivo.

3. Establecer correctamente los factores para formar el índice de riesgo reduce el número de falsos positivos generados u obtenidos por este módulo.

4. Otra forma de reducir falsos positivos es realizar el aprendizaje en momentos en que la red se encuentre en su mayor uso y así no detectar como alertas algo normal y común en la red.

5. El uso de este módulo no reducirá a 0% el riesgo de ataques en la red pero si se tendrá un mejor control de posibles ataques.

RECOMENDACIONES

1. Establecer políticas de seguridad con respecto al comportamiento de la red contribuye a una mejor detección de posibles ataques.

2. Aumentar los factores para el índice de riesgo y así reducir falsos positivos.

3. Para tráficos de redes de enormes dimensiones se debe tener en cuenta el almacenamiento de conexiones en la base de datos por lo cual es recomendable eliminar el tráfico en desuso que se encuentra en la base.

4. Entrenar el tiempo suficiente el IDS/IPS Snort para que adquiera un mejor conocimiento del tráfico, de 8 a 24 horas es lo más recomendado.

5. El aprendizaje puede tener sus falencias, debido a que el tráfico con el cual se entrena puede resultar anormal y por ende perjudicar el proceso de detección, haciendo que tráfico anómalo pase como normal. Por ello se recomienda que el entrenamiento se lo realice en horas en las cuales a criterio del administrador de red el tráfico sea menos proclive a intrusiones.

6. Existen otras formas para la etapa de entrenamiento como por ejemplo las redes neuronales, las cuales representarían una aproximación más real al comportamiento de un tráfico

BIBLIOGRAFÍA

[1] **Armando Mira.** "Tutorial de Snort". http://club.telepolis.com/websecure/tutoriales/tutorial_snort.pdf [Online] [Cited 2005]

[2] **Carlos Jiménez Galindo.** "Proyecto de Fin De Carrera". http://www.adminso.es/wiki/images/d/d0/Pfc_Carlos_cap3.pdf [Online] [Cited 2007]

[3] **Pablo Martínez.** "Análisis de Snort". http://pmartinez.files.wordpress.com/2007/07/analisis_snort.pdf [Online] [Cited 2006]

[4] **Diego González Gómez.** "Sistema de Detección de Intrusiones". http://www.dgonzalez.net/pub/ids/trans/IDStCOL.pdf [Online] [Cited Julio 2003]

[5] **Roman Danyliw.** "Analysis Console for Intrusion Databases". http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html [Online] [Cited 2003]

[6] **José Madrigal Garcia, Mateo Luna Luna.** "Firewalls". http://www.monografias.com/trabajos14/firewalls/firewalls.shtml [Online] [Cited 2007]

[7] **Enrique González Rodríguez, Diego Trujillo García.** “Firewall, Iptables, Proxy”. http://serdis.dis.ulpgc.es/~a013775/asignaturas/ii-aso/curso0607/trabajos/seguridad/filtros/filtros_t.pdf [Online] [Cited Febrero 2007]