Adaptación del IDS/IPS Suricata para que se pueda convertir en una solución empresarial.

Juan Astudillo Herrera [1], Fernando Ortiz Flores [2], Alberto Jiménez Macías [3], Alfonso Aranda [4]
Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil, Ecuador
juanastu@espol.edu.ec [1], fermaort@espol.edu.ec [2], albjimen@espol.edu.ec [3], jaranda@espol.edu.ec [4]

Resumen

Los Sistemas de Prevención y Detención de Intrusos (IPS/IDS) proporcionan un nivel adicional de seguridad en las redes de datos previniendo vulnerabilidades que los firewall simplemente no pueden. La mayoría de estos sistemas son pagados y costosos, por lo que resultan muy difíciles de implementar en pequeñas y medianas empresas. Suricata es un motor IPS/IDS de código abierto bajo licencia GPLv2 relativamente nuevo pero con muy buenas características siendo la más importante su arquitectura multi-hilos. El objetivo principal del proyecto es montar una solución IPS/IDS empresarial con Suricata como motor de detección. En primer lugar se definió el estado de arte del proyecto Suricata y se estudió los requerimientos necesarios para diseñar una solución IPS/IDS de uso empresarial que no se quede tan atrás con relación a soluciones comerciales existentes. Se adaptó un módulo de detección de anomalías para poder detectar amenazas que por su método de atacar son difíciles de identificar mediante reglas, como por ejemplo: gusanos y ataques de denegación de servicio. Se sometió al Suricata a pruebas de estrés para determinar su alcance y limitaciones. Finalmente se desarrolló una interfaz web para administración del IPS/IDS. Al final del documento se proponen mejoras futuras a la solución actual.

Palabras Clave: *Anomalías, Suricata, IDS, IPS.*

Abstract

Intrusion Prevention and Detection systems provide and additional level of security in networks preventing from vulnerabilities than firewalls cannot. Most of these systems are paid and expensive and because of that there are difficulties in their implementation in small and medium companies. Suricata is an open source IPS/IDS engine with GPLv2 license, relatively new but with very good features being the most important of all, its multi-threading architecture. The main objective of this project is to implementate an Enterprise IPS/IDS with Suricata as its detection engine. First we defined the state of art of the project Suricata y we studied the requirements to design an enterprise solution that don't stay behind existent commercials IPS/IDS solutions. We adapted an anomaly detection engine to detect network menaces that because of the nature of their attack the get hard to detect with pattern and rules, for instance: worms and DOS attacks. Suricata was stressed tested to determinate its limitation. Finally we developed a web front-end for administration of the IPS/IDS. At the end of this document we propose future improvements for the actual work.

Key Words: *Anomaly, Suricata, IDS, IPS, Security, Network, Information, Ourmon.*

1. Introducción

Los IDS/IPS actualmente constituye una herramienta indispensable para la seguridad de la red de una empresa ya sea pequeña, mediana o grande. Suricata es un proyecto joven de código abierto que promete ser un IDS/IPS tan efectivo como los productos con licencia.

Por esta razón, y dado que Suricata trabaja actualmente de acuerdo a los avances tecnológicos actuales, se planteó contribuir con dos aspectos fundamentales faltantes: una interfaz de administración, que facilite su gestión como solución empresarial, y un módulo de detección de anomalías.

2. Planteamiento

2.1 Definición del problema

Debido al constante avance de la tecnología, se ha ha dado paso a nuevos tipos de ataques en la red con los cuales se necesita estar protegidos no solo con protección a las computadoras con antivirus y firewalls, también se necesita tener una herramienta que detecte ataques en el límite de la red y de la LAN con posibilidad de respuesta a tiempo real, tanto para ataques existentes como aquellos que son nuevos.

En este caso, Suricata presenta deficiencias respecto a nuevos tipos de ataque debido a que su sistema de detección y prevención de intrusos está soportada a base de reglas que son descargadas diariamente en la página web oficial de Emerging Threats, sin embargo previene de ataques que siguen patrones generales definidos en las mencionadas regals. Además Suricata carece de una interfaz de administración, siendo sólo manejado mediante líneas de comandos que requieren de profesionales con un expertise moderadamente alto en el manejo de los IDS/IPS.

2.2 Objetivos

El objetivo general es desarrollar una solución IDS/IPS para empresas usando a Suricata como motor de detección y prevención.

Los objetivos específicos en línea con el objetivo general se detallan a continuación:

Describir el estado de arte del proyecto de Suricata, en vista de que es una herramienta nueva se planteó conocer completamente sus funcionalidades y definir tanto fortalezas como debilidades en su funcionamiento. Otro objetivo es el de someter al motor IDS/IPS a pruebas de rendimiento para definir sus limitantes, se utilizaron varias herramientas de simulación de tráfico y medición para un completo análisis tanto del motor, como de los recursos que son utilizados por el servidor que lo aloja. Otro objetivo fue el de desarrollar un módulo de detección de anomalías y autoaprendizaje, debido a la limitante de no conocer a tiempo real la posibilidad de nuevos tipos de ataque que no han sido conocidos o puestos en el repositorio de Emerging Threats. Y por último, para complementar el objetivo general, desarrollar una interfaz de administración para la solución para manejar a tiempo real el motor de IDS/IPS.

3 Antecedentes

3.1 Descripción y características de Suricata

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

Entre algunas características de Suricata, las más representativas son las siguientes:

- Multi-threading
- Estadísticas de Rendimiento
- Detección de Protocolos automáticos
- Fast IP Matching
- IP Reputation
- Graphic Cards Acceleration

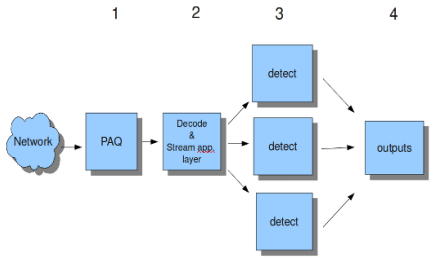

Figura 1. Proceso de los 4 módulos de Multi-hilos en Suricata

La figura 1 muestra el procesamiento de un paquete y los módulos de suricata que intervienen:
PAQ se refiere a la adquisición de paquetes.
Decode se refiere a la decodificación de paquetes.
Stream app. Layer realiza el seguimiento del flujo y reconstrucción.
Detect, compara firmas y Output procesa todos los eventos y alertas.

3.2 Comparación con otras soluciones IDS/IPS

La Tabla 1 compara algunas características de suricata con otras soluciones IPS propietarias en general. La ventaja de Suricata al ser una solución de código abierto, es la flexibilidad de agregar nuevas funcionalidades al motor, según vayan siendo requeridas por las comunidades [1].

Características	Soluciones Comerciales IDS/IPS	Suricata
Multi-Threading	x	Si
Soporte para IPV6	Cisco, IBM, Stonesoft	Si
IP Reputation	Cisco	Si
Detección Autamática de Protocolos	No	Si
Aceleración con GPU	No	Si
Variables Globales/Flowbit	No	Si
GeoIP	No	Si
Análisis Avanzado de HTTP	No	Si
HTTP Access Logging	No	Si
SMB Access Logging	No	Si
Anomaly Detection	Sí	No
Alta Disponibilidad	Si	No
GUI de Administración	Si	No
GRATIS	No	SI

Tabla 1. Comparación de Suricata con IPS/IDS's propietarias

La tabla 2 compara característica de suricata con Snort y Bro (como IDS) [2].

Características	Bro	Snort	Suricata
Multi-Threading	No	x	Si
Soporte para IPV6	Si	Sí	Si
IP Reputation	Algo	No	Si
Detección Autamática de Protocolos	Si	No	Si
Aceleración con GPU	No	No	Si
Variables Globales/Flowbits	Si	No	Si
GeoIP	Si	No	Si
Análisis Avanzado de HTTP	Si	No	Si
HTTP Access Logging	Si	No	Si
SMB Access Logging	Si	No	Si

Tabla 2. Comparación de Suricata con IPS/IDS's de código abierto

3.3 Configuración General de Suricata

Los requerimientos de Hardware para Suricata varían dependiendo de cuánto tráfico se espera que el sensor esté monitoreando, así como también de la cantidad de firmas de seguridad que serán cargadas.

A mayor throughput vamos a considerar tener un mejor procesador con mayor cantidad de núcleos. La memoria RAM por otro lado va a tener que ser mayor conforme queramos aumentar el número de firmas de seguridad y también dependerá del throughput.

Si se ejecuta Suricata en modo IDS basta con una tarjeta de red para escuchar el tráfico de red y es recomendable una segunda interfaz de administración. En modo IPS por otro lado se necesitan mínimo 2 interfaces de red, las cuales estarán puenteadas (IPS es un dispositivo de capa 2) y es muy recomendable y casi necesario una interfaz más para administración.

En cuanto a los requerimientos de Software, sin importar la plataforma sobre la cual se va a instalar suricata, es necesario tener las siguientes herramientas instaladas:

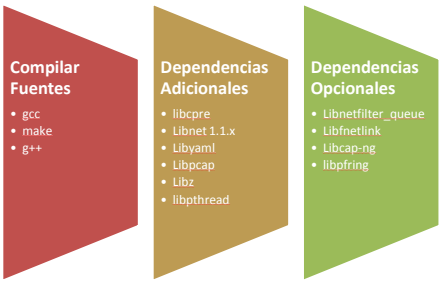

Figura 2. Dependencias y fuentes de Suricata

3.4 Futuras mejoras en próximas versiones de la herramienta Suricata

Actualmente Suricata se encuentra en la fase 2 de su proceso desarrollo. Los puntos a mejorar en esta etapa son:

- Aceleración con CUDA GPU.
- Reputación IP y DNS.
- Salida por medio de Sockets en UNIX.
- GeoIP Keyword.
- Estadísticas de Rendimiento.
- Preprocesador de Anomalías.
- Salida a Mysql/Postgress/Sguil.

Actualización de Módulos sin reiniciar todo suricata.

Embeber un sistema para prueba de reglas escritas por el administrador [3].

4. Diseño e implementación de la solución IDS/IPS empresarial

4.1 Diseño de interfaz Gráfica de Administración

Para la interfaz que diseñamos, utilizamos CodeIgniter, desarrollado por EllisLab, un framework de PHP que nos brinda herramientas para desarrollar aplicaciones web dinámicas.

Nos provee varias librerías para las tareas que se necesitan comúnmente y además una interfaz simple con una estructura lógica para acceder a dichas librerías. Codeigniter utiliza la arquitectura MVC (Modelo Vista Controlador).

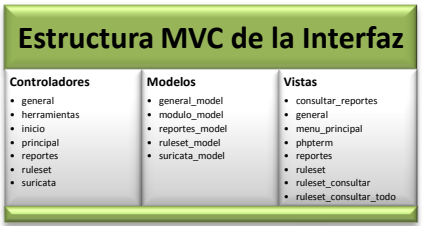

Figura 3. Estructura MVC de la Interfaz

La interfaz posee algunas pestañas de navegación.

Figura 4. Pestañas de Navegación de la Interfaz

General: Muestra información general de Hardware, interfaces de red y sistema operativo del servidor:

- Información de Procesador.
- Información de Memoria.
- Información de Red.
- Interfaces Puenteadas.
- Información de Sistema Operativo.

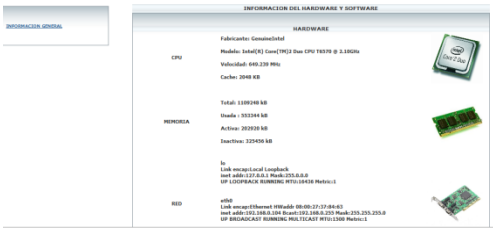

Figura 5. Página de Información General del Servidor

Suricata: Página para ver y configurar opciones del Sensor IDS/IPS.

Figura 6. Página de Información y configuración del Motor Suricata

Rulesets: Pagina para ver, habilitar y deshabilitar las reglas cargadas para un Perfil.

Internamente crea un directorio de reglas para cada perfil creado por el administrador. Cualquier cambio que se haga afectará únicamente a las reglas creadas para dicho perfil. Un perfil de reglas debe ser obligatoriamente escogido antes de iniciar Suricata, caso contrario tomará el perfil por defecto con reglas por defecto.

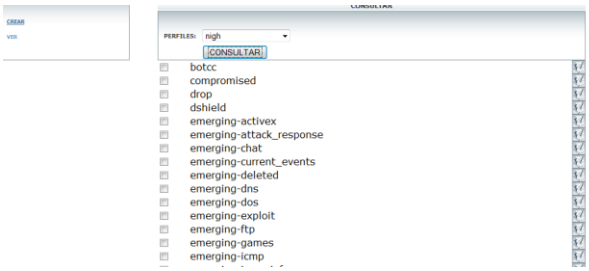

Figura 7. Página de configuración de Rulesets

Alertas: Página para ver las alertar generadas por el sensor. Se ha integra una interfaz existente llamada BASE, actualmente desarrollada para ver las alertas generadas por Snort pero que es 100% adaptable para Suricata.

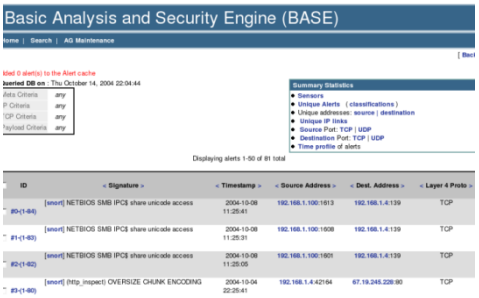

Figura 8. Página de monitoreo de Alertas usando BASE

Herramientas: Posee un Shell para configuración más específica del sensor.

Figura 9. Terminal para configuración Interna del Sensor

Configuración de Variables: Para ingresar los valores correspondientes a las variables que se utilizan en las reglas de Emerging Threats.

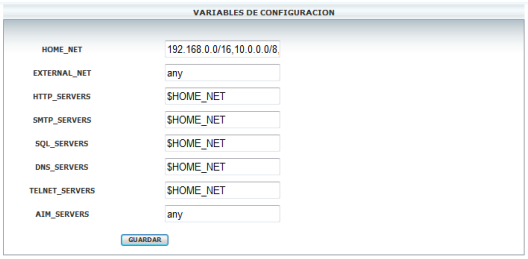

Figura 10: Terminal de variables de configuración de reglas

Graficas de monitoreo: Generadas cada minuto con información de estadísticas parseadas del archivo stats.log generado por Suricata. También hay información de rendimiento del equipo sacando la información de diversos comandos de Linux.

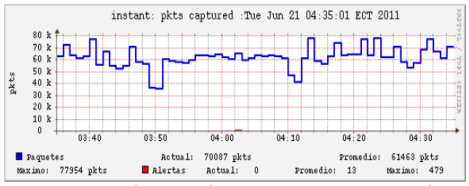

Figura 11 Monitoreo de paquetes capturados

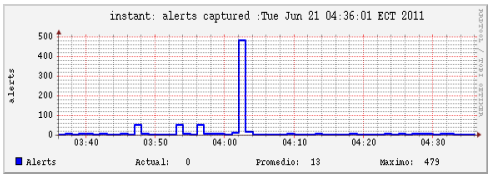

Figura 12 Monitoreo de alertas

4.2 Módulo de recuperación de fallos

Este módulo es importante para la ejecución de la Solución en modo IPS. En modo inline el tráfico pasa a través de dos interfaces de red puenteadas. Mediante reglas de iptables se redirige el tráfico deseado a Suricata para su análisis, sin estas reglas puestas, el tráfico pasaría directamente de una interfaz a otra sin pasar por Suricata. Cuando esto sucede decimos que Suricata está cortocircuitado.

Basándonos en este principio, se ha elaborado un programa que continuamente envía paquetes ligeros a través de las colas donde Suricata recibe paquetes.

Si no se detecta estos paquetes generados por el programa en un intervalo de tiempo definido, se cortocircuita Suricata hasta que este se recupere del congestionamiento o cualquier situación que impidió que se lean los paquetes de monitoreo.

La siguiente figura muestra el algoritmo usado para la recuperación de fallo.

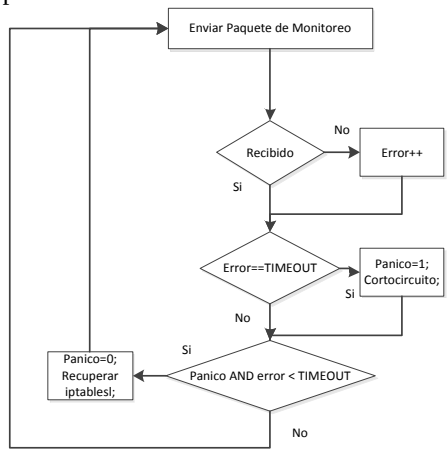

Figura 13 Diagrama de estados del módulo de recuperación de fallos

4.3 Instalador de la Solución Empresarial

Para crear un instalador para nuestra Solución Empresarial decidimos convertir a Suricata en paquete (RPM) e integrándolo a una distribución Linux CentOS, de esta manera cuando el usuario instale CentOS, dentro de los paquetes se le instalará, estará Suricata con Libpcap incluido.

5 Diseño e integración del módulo de detección de anomalías.

5.1 Antecedentes de módulos de detección de anomalías

Denning [4] formalizó el concepto de detección de anomalías asumiendo que las violaciones de seguridad pueden ser detectados inspeccionando aquellas direcciones IP que solicitaban utilizar recursos en dimensiones anormales, como resultado, estas técnicas de detección de anomalías buscan establecer actividades dentro del tráfico “normal” de una red mediante métricas, y en caso de ocurrir un

ataque, dicho comportamiento de tráfico será completamente distinto al que el dispositivo ya ha estudiado.

De acuerdo a Axelsson [5], la detección de anomalías se lo considera como auto aprendizaje debido a que estos sistemas forman una opinión acerca del comportamiento normal del tráfico. Se pueden aplicar varias técnicas de detección de anomalías, tales como información estadística de comportamiento, entrenamiento del motor de detección para saber cómo es el tráfico de red normalmente, entre otros.

5.2 Herramienta Ourmon

Ourmon analiza los datos utilizando varias instancias de BPFs (Berkeley Packet Filter) y los muestra a través de herramientas gráficas tales como RRDTOOL, histogramas y reportes ASCII.

Ourmon además utiliza varios algoritmos para detectar posibles ataques sospechosos, debido a la recolección de datos que es almacenado en tuplas y está basado en las direcciones IP host donde se puede proveer información de ataques coordinados y gusanos [6].

Algunas cosas que Ourmon puede hacer son las siguientes:

- Monitorear flujos TCP y UDP
- Ayuda a medir el tráfico de manera estadística
- Atrapa servidores de correo en modo relay.
- Atrapa "botnets"
- Descubre infecciones con malware "zero-day"
- Descubre ataques desde adentro o desde afuera.
- Observa que protocolos están ocupando el mayor ancho de banda.

5.3 Algoritmo de detección de anomalías en tráfico TCP

El algoritmo de de anomalías en tráfico TCP ordena los paquetes por tuplas SYN. Estas tuplas consisten en una agrupación de atributos básicos con una dirección IP que es la que será monitoreada, la tupla SYN tiene la siguiente forma:

(IP Source address, SYNS, SYNACKS, FINSSENT, FINSBACK, RESETS, ICMP ERRORS, PKTSSENT, PKTSBACK, port signature data)

La clave lógica en esta tupla es que el IP Source address, SYNS, FINS y RESETS sean contadores de los paquetes TCP.

SYNS son contadores de paquetes SYN enviados desde el IP fuente.
SYNACKS es un subset de aquellos SYN enviados con la bandera ACK.
Los **FINS** son una bandera que notifica que no hay más datos que mandar desde la fuente.
RESETS son contados cuando son enviados de regreso a la IP fuente.
ICMP Errors se refiere a ciertos errores de ICMP que son inalcanzables.
PKTSSENT cuenta los paquetes que han sido enviados por la IP fuente.
PKTSBACK son los paquetes que retornados a la fuente IP.

Existen dos "pesos" asociados a la tupla SYN, la cual Ourmon la llama "peso de trabajo" y "peso de gusano". El peso de trabajo es computado por IP fuente con la siguiente fórmula:

$$(S_S + F_s + R_r)/T_{sr}$$

Siendo la terminología la siguiente:

S = SYN	s = sender
F = FINS	r = receiver
R = RESETS	
T = TCP	

El resultado de esta expresión está dado en porcentaje. La idea es controlar los paquetes para ser usados de una manera anómala y dividir ese contador por el total de paquetes TCP. Si da 100%, es una mala señal e implica que una anomalía de un tipo se está dando.

Para calcular el peso de gusano, se aplica la siguiente fórmula:

$$S_s - F_r > C$$

Donde se extrae el FINS retornado por los SYNS enviados y solo almacena la tupla si C (que es una constante) es mucho mayor que una constante configurable manualmente. Ourmon pone como default la constante C como 20 de manera intuitiva justificando que dicha constante debe ser lo suficientemente grande que cualquier IP fuente esté generando más SYNS que FINS.

5.4 Anomalías en paquetes UDP

El front-end coge paquetes UDP en forma de tuplas durante el periodo de recolección de datos, aproximadamente ocurre cada 30 segundos. Las tuplas UDP producidas por el periodo de recolección de datos son llamados UDP Port Report. La tupla tiene la siguiente forma:

(IPSRC, WEIGHT, SENT, RECV, ICMPERRORS, L3D, L4D, SIZEINFO, SA/RA, APPFLAGS, PORTSIG).

La clave lógica en esta tupla es la dirección IP IPSRC (IP Source).
SENT y RECEIVED son contadores de los paquetes UDP enviados desde/hasta el host en cuestión.
ICMPERRORS es un contador de tipos distintos de errores ICMP que pueden haber, normalmente la mayoría de los errores se dan porque el destino es inalcanzable.
L3D es el contador de la capa 3 de direcciones IP destino durante el periodo recolectado.
L4D es el contador de los puertos UDP destino.
SIZEINFO es un histograma de tamaño de paquetes enviados a nivel capa 7.
SA/RA es un promedio de la carga útil del tamaño de los paquetes UDP en capa 7 enviados por el host (SA) y recibidos por el host (RA).
APPFLAGS es un campo programable basado en expresiones regulares y es usado para inspeccionar el contenido en capa 7.

El peso máximo es un peso determinado por el administrador de redes.

Para calcular el Work Weight se utiliza la siguiente fórmula:

*Peso de trabajo = (SENT * ICMPERRORS) + REC*

En un período de recolección de datos, contamos los paquetes UDP enviados por el host y se lo multiplica por el número total de errores ICMP que han sido retornados al host.

5.5 Integración final de la solución IDS/IPS Empresarial

Con un script creado por nosotros, se realizan lecturas de este archivo cada minuto, se accede a la información relevante del tcpworm.txt que es la dirección IP y en caso de existir un posible ataque en curso, se crea un nuevo set de reglas para ser leído por Suricata en modo IDS que será guardado en un archivo llamado 'Ourmon-anomaly.rules' donde estarán todas las IPs aprendidas por Suricata desde Ourmon.

```
alert ip -SUSPICIOUS_IP- any -> $HOME_NET any (
msg:"Anomaly Detected, possible worm attack";
classtype: anomaly-ourmon; sid:1100000; rev:1;)
```

Figura 14. Regla generada para alertar el host sospechoso detectado por Ourmon

El administrador recibirá una alerta cuando se notifique de un posible ataque anómalo ya sea un gusano o un scanner, dependerá del administrador y su expertise, tomar medidas ya sea ubicando a esa IP alertada por nuestra solución en un Black List o si conoce que dicha IP es segura, ubicarlo en un White List.

Las reglas generadas por este método tienen acción ALERT, esto debido a que este algoritmo está sujeto a falsos positivos, si bien es cierto son sospechosas no implica que siempre será un ataque en la red. Por lo tanto hemos considerado para una detección eficiente de posibles ataques anómalos, dentro de nuestra Solución.

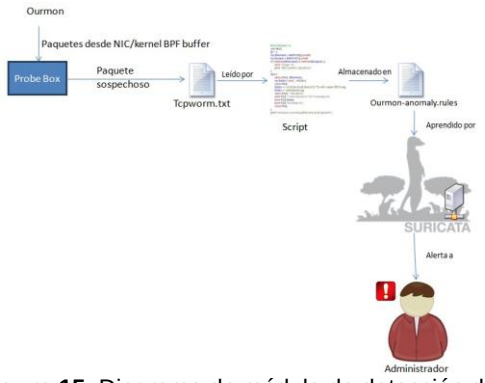

Figura 15. Diagrama de módulo de detección de anomalías y auto aprendizaje

6 Diseño y ejecución de pruebas de Rendimiento de Suricata

6.1 Diseño de pruebas de rendimiento de la solución empresarial

El hardware utilizado para las pruebas fueron los siguientes:

Motherboard: Intel Corporation DP55WB
Procesador: Intel(R) Core(TM) i7 CPU 870@2.93GHz
Memoria RAM: 8 Gigas
Tarjetas de Red: 3 Intel 82572EI Gigabit Ethernet

Adicionalmente se usaron 3 computadoras adicionales de menores recursos para la generación de tráfico.

Todas las computadoras tienen tres interfaces de red, dos para comunicación entre ellas y una interfaz para administración. Las dos interfaces de comunicación están conectadas a un Switch Cisco Catalys C3560G. Se crearon dos vlans para forzar que el Suricata esté en el medio de la comunicación entre dos computadoras de prueba (Sin las Vlans, las comunicación únicamente pasaría por el Switch al ser el dispositivo de capa 2 que está antes de Suricata).

Las herramientas que se utilizaron para las pruebas fueron las siguientes:

- **Tomahawk:** Es una herramienta opensource para realizar pruebas de throughput y capacidades de bloqueo para sistemas de prevención de Intrusos.
- **Hping3:** Es una herramienta multi-usos para generar paquetes TCP/IP.
- **Httperf:** Herramienta desarrollada para realizar pruebas de desempeño en los servidores HTTP.
- **Siege:** Simula tráfico real hacia servidores web generando peticiones constantes desde varios navegadores web virtuales.
- **Iperf:** Sirve para medir el desempeño máximo de ancho de banda para tráfico TCP y UDP.

La topología utilizada para las pruebas fue la siguiente

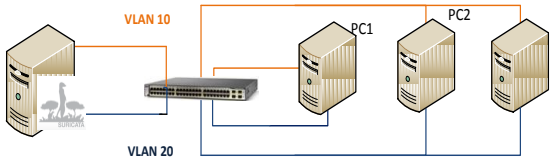

Figura 16. Topología para las pruebas

Se instaló perf en las tres computadoras de prueba, Apache2 en una de las computadoras de prueba para responder peticiones solicitadas por httperf y Siege y Tomahawk en 2 computadoras y el resto de herramientas en las 3 por igual.

6.2 Implementación y análisis de resultados de las pruebas de rendimiento.

En modo IDS, Suricata analiza los paquetes capturados con la librería libpcap, dado que los paquetes analizados son solo una copia del tráfico de red se pudo analizar hasta 1 Gbps de Throughput sin ningún inconveniente.

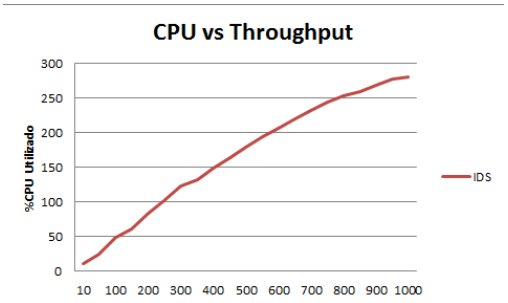

Figura 17. Utilización de CPU vs Througput en modo IDS

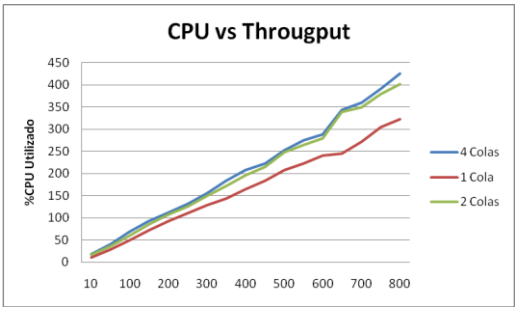

Figura 18. Utilización de CPU vs Througput en modo IPS

Se aprecia que al utilizar más colas aumenta el procesamiento del servidor, sin embargo aumenta el Througput máximo del motor. La diferencia entre 2 colas y 4 colas es casi indiferente aunque se aprecia un aumento de throughput máximo en este último.

```
Tasks: 148 total,   1 running, 147 sleeping,   0 stopped,   0 zombie
Cpu0  :  9.8%us, 23.4%sy, 36.6%ni, 30.2%id,  0.0%wa,  0.0%hi,  0.0%si,
Cpu1  : 43.9%us, 18.0%sy,  0.0%ni, 38.0%id,  0.0%wa,  0.0%hi,  0.0%si,
Cpu2  : 36.1%us,  6.5%sy,  0.0%ni, 57.4%id,  0.0%wa,  0.0%hi,  0.0%si,
Cpu3  : 35.5%us,  4.7%sy,  0.0%ni, 59.9%id,  0.0%wa,  0.0%hi,  0.0%si,
Cpu4  :  4.3%us, 13.7%sy,  0.0%ni, 61.2%id,  0.0%wa,  0.0%hi, 20.7%si,
Cpu5  :  6.4%us, 10.6%sy,  0.0%ni, 61.4%id,  0.0%wa,  0.0%hi, 21.5%si,
Cpu6  :  9.7%us, 19.1%sy,  0.0%ni, 67.7%id,  0.0%wa,  0.0%hi,  3.5%si,
Cpu7  : 11.0%us, 20.7%sy,  0.0%ni, 64.9%id,  0.0%wa,  0.0%hi,  3.4%si,
Mem:   8174536k total,  2340340k used,  5834196k free,   133192k buffers
Swap:  1052248k total,        0k used,  1052248k free,   770272k cached

21634 root      20   0 1607m 1.1g 1732 S  338 13.8  37:26.52 suricata
   33 root      20   0     0    0    0 S    0  0.0   6:43.84 events/6
    1 root      20   0  3852  668  568 S    0  0.0   0:03.30 init
  867 root      20   0     0    0    0 S    0  0.0   0:04.82 kjournald
```

Figura 19. Procesamiento de Suricata por procesador con 4 colas

6.3 Pruebas del módulo de detección de anomalías.

Hemos puesto al módulo de detección de anomalías en tráfico real para evaluar su efectividad en capturar posibles ataques de gusano y además probar el anomaly detection.

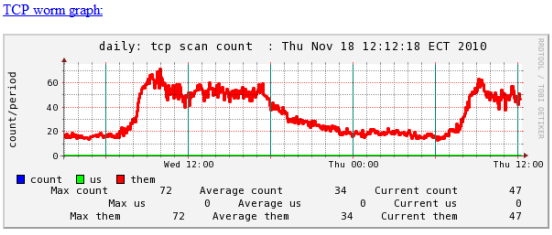

Figura 20. Grafo de detección de gusanos utilizando Ourmon.

Ourmon está mostrando un grafo de gusano en base al peso de trabajo que está ocurriendo en un lapso de 24 horas que hemos puesto a Ourmon en prueba, a mayor peso de trabajo, más es la

probabilidad de que un escaneo o ataque de gusano se esté dando.

En cuanto al rendimiento de procesador y memoria, Ourmon no representa mucho consumo. La figura 5.11 muestra una captura del comando 'top' mostrando ourmon en ejecución y su consumo de recursos.

```
top - 23:23:18 up 15 days,  6:36,  3 users,  load average: 0.21, 0.14, 0.10
Tasks: 168 total,   1 running, 167 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.6%us,  0.1%sy,  0.0%ni, 98.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   8301656k total,  2766700k used,  5534956k free,   310040k buffers
Swap:  5406716k total,        0k used,  5406716k free,  1835844k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2657 root      20   0  2020  448  372 S  0.3  0.0   9:38.22 hald-addon-stor
 2819 mysql     20   0 2082m 356m 5560 S  0.3  4.4  51:42.87 mysqld
11449 root      20   0 13920  11m  880 S  0.3  0.1   0:03.09 ourmon
15553 root      20   0  2380  964  704 S  0.3  0.0   0:02.71 top
    1 root      20   0  2116  596  504 S  0.0  0.0   0:07.71 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
```

Figura 21 Rendimiento de procesador y memoria de Ourmon

6.4 Conclusiones de los resultados de las pruebas.

Ya habíamos mencionado que en modo IPS Suricata va a tener un througput limitado, no por la capacidad de la interfaz y medio de transmisión sino por qué tan rápido el motor pueda procesar los paquetes. Este valor máximo de througput va a variar de red en red debido al flujo de paquetes y también al tipo de tráfico. Por ejemplo, una red con gran cantidad de tráfico TCP va a necesitar que Suricata consuma más memoria manteniendo el estado de las sesiones.

Logramos también con el módulo de detección de anomalías integrado a nuestra solución empresarial detectar ciertas direcciones IP que estaban produciendo tráfico anómalo dentro de una red a tiempo real, esto sirvió para el motor Suricata aprender de ellas y generar alertas y acciones idóneas contra este tipo de ataque que sin ayuda de la detección de anomalías no hubiese sido posible encontrar a tiempo.

Lastimosamente Suricata debe reiniciarse cada cierto tiempo para tener conocimiento de las nuevas reglas añadidas por el detector de anomalías así que no se puede hablar de un autoaprendizaje en tiempo real.

7 Propuestas de mejoras para la solución empresarial actual.

7.1 Interfaz gráfica de gestión y control remoto

La idea es ser capaz de controlar, administrar y monitorizar no sólo uno sino varios sensores remotos. Para esto se necesita una aplicación que posea una arquitectura centralizada, es decir que se pueda administrar varios motores de suricata desde la misma aplicación.

Una recomendación para el sistema propuesto es montar un Servicio Web o una aplicación de cliente en el lado de los Suricata que se comunique con una aplicación remota (podría ser un aplicación web).

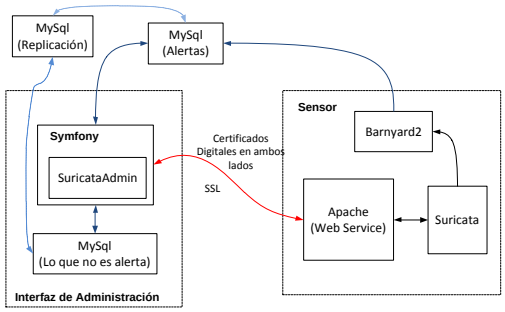

Figura 22. Diagrama de bloques para una interfaz centralizada

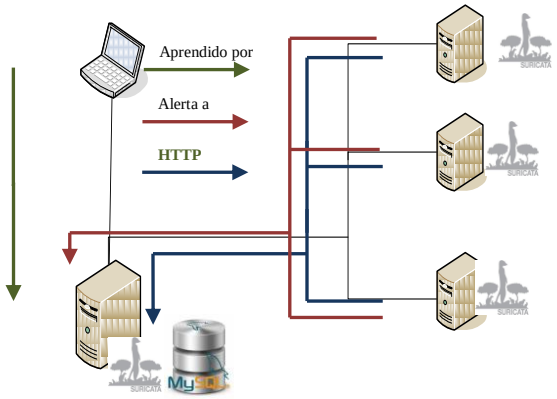

Figura 23. Diagrama Web Service para una interfaz centralizada

La comunicación con la interfaz será a través de http. Desde la interfaz de administración remota el usuario es capaz de ver todos los servidores con suricata activos en la red, identificando tanto al principal como a los secundarios. Es posible en cada uno ver las características del servidor, iniciar y detener el servicio de suricata, observar las alertas y administrar las firmas de seguridad.

7.2 Módulo de monitoreo para Centro de Operaciones de Redes.

Este módulo debería mostrar gráficas en tiempo real con reportes periódicos configurables. Los reportes deben ser almacenados para futuras referencias y a su vez enviadas por correo

electrónico, o cualquier otro mecanismo, al administrador.

Debería ser capaz también de configurar alertas prioritarias que alerten inmediatamente al administrador cuando sean activadas.

7.3 Módulo de Detección Reactiva para Suricata en modo IDS

Suricata en modo IDS actualmente funcionan de una manera pasiva, es decir que únicamente alerta sobre el evento sospechoso y el administrador de red es quién debe tomar acciones oportunas.

Algunos IDS son capaces de reaccionar a los eventos mediante reseteo de conexiones TCP (en caso de escaneos de red o ataques de inundación), e incluso enviar señales a los Firewalls para bloquear o redirigir paquetes en base a su dirección IP origen.

7.4 Otras propuestas de mejoras

Suricata por ser relativamente nuevo tiene aún mucho camino por recorrer, especialmente en lo que a código se refiere, pues la mayoría de las aplicaciones desarrolladas, por no decir todas, son aplicaciones que en primer lugar fueron desarrolladas para trabajar en conjunto con snort, y que gracias a la compatibilidad de Suricata han podido adaptarse a este motor IDS/IPS.

Como propuestas adicionales de mejoras pueden ser:

Integración del módulo de Detección de Anomalías al motor.

Integración del módulo de Recuperación de Fallos y alta Disponibilidad al motor.

Análisis y optimización del código para aumentar el througput máximo en modo IPS limitado por la latencia que produce el motor.

8.1 Conclusiones

1. Se estableció patrones de medición de rendimiento utilizando libcaps de tráfico pesado para evaluar la capacidad de Suricata de analizar dicho tráfico sacando provecho la tecnología multi-hilos que lo caracteriza.
2. A mayor througput mayor uso de procesador y memoria necesaria. Este requerimiento adicional de hardware se puede atenuar usando tarjetas de red especiales para captura y procesamiento de datos que si bien aumentan el costo de implementación también aumentan el desempeño de la solución.
3. El modo IPS demanda más recursos que el modo IDS. Es importante para el modo IPS trabajar únicamente con las reglas necesarias y optimizar el hardware a nivel de red.
4. Suricata necesita aun trabajo para llegar a competir con soluciones comerciales para grandes redes, pero no deja de ser una muy buena opción para pymes según las pruebas de rendimiento realizadas.
5. Todas las herramientas utilizadas en esta tesina de seminario de grado han sido de código abierto, demostrando la compatibilidad entre ellas y consistencia durante el desarrollo de esta tesina de seminario. No se presentaron problemas de conexión y nos permitió conocer más a fondo sobre las bondades del código abierto.
6. Se conoció algoritmos de detección de anomalías propuesto por Ourmon para contrarrestar scans intensivos y posibilidades de infección de gusanos, que pueden ser utilizadas por Suricata como oportunidad de mejora para una protección más completa del negocio.

8.2 Recomendaciones

1. Para poder trabajar con el sistema, es necesario definir políticas y procedimientos de seguridad dentro de la empresa.
2. Se recomienda capacitar al personal encargado del mantenimiento de la solución IDS/IPS de la empresa, pues se requiere conocimientos más allá de saber manejar un Firewall.
3. Se recomienda mantener actualizado los rulesets debido a que cada día se puede registrar un nuevo ataque de diferente moderación. Cada vez que exista un ataque en la red, EmergingThreats se encargará de neutralizarlo con la regla correspondiente. Para estar al día, EmergingThreats, se requiere un costo adicional, caso contrario se tiene que esperar un tiempo para disponer de nuevas reglas.
4. Se debe concientizar más a los empresarios sobre la existencia y las bondades de la Seguridad Informática y de la amenaza constante que existe en las redes hacia la información, que es el mayor activo de la empresa.
5. Se debe realizar nuevas pruebas de rendimiento conforme los avances tecnológicos sigan llegando al mercado, debido a que Suricata, siendo un IDS/IPS que basa su óptimo trabajo en hilos, puede

tomar ventaja en estos y seguir siendo la mejor opción en cuanto a los IDS/IPS de código abierto.

6. Esta solución es recomendable para pequeñas y medianas empresas que no disponen de capital para invertir en un IPS comercial, las capacidades de Suricata, siendo de código libre, cubre muy bien las necesidades de protección de red para este tipo de empresas.

9. Bibliografía

[1] McRee, "Suricata in Toolsmith: Meet the Meerkat" http://holisticinfosec.blogspot.com/2010/08/suricata-in-toolsmith-meet-meerkat.html. [Cited:2010].

[2] Karen Scarfone, "Guide to Intrusion Detection and Prevention Systems". National Institute of Standards and Technology. [Cited: 2007].

[3] Gerber, "Suricata: A Next Generation IDS/IPS Engine" http://blog.securitymonks.com/2010/01/05/suricata-a-next-generation-idsips-engine/. [Cited: 2010]

[4] Denning, "An Intrusion Detection Model", IEEE. [Cited: 1987].

[5] Axelsson**, "**The Base-rate fallacy and the difficulty of Intrusion Detection System". Tissec. Cited [2000].

[6] Binkley**.** "Ourmon – Network Monitoring and Anomaly detection System". http://ww.ourmon.sourceforge.net. 2005.