ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

INFORME DE MATERIA DE GRADUACIÓN

"MÓDULO DE GENERACIÓN DE REPORTES GRÁFICOS DE UNA HONEYNET A PARTIR DE LOS LOGS TCPDUMPS."

Previa a la obtención del Título de:

**INGENIERO EN COMPUTACIÓN
ESPECIALIZACION SISTEMAS TECNOLÓGICOS**

**INGENIERO EN COMPUTACIÓN
ESPECIALIZACION SISTEMAS INFORMACIÓN**

Presentada por:

DENISSE AURORA CAYETANO CARVAJAL

CHRISTIAN ALEJANDRO RIVADENEIRA ZAMORA

Guayaquil - Ecuador
2009

AGRADECIMIENTO

A Dios, todopoderoso y eterno.
A nuestras familias que nos han apoyando
siempre en el camino de nuestra vida,
a la Ing. Cristina Abad que ha sido nuestra
guía y sin su apoyo no hubiésemos podido
lograr el presente trabajo.

DEDICATORIA

A todos aquellos que aportan cosas productivas a la sociedad.

TRIBUNAL DE GRADO

DIRECTORA DEL PROYECTO DE GRADUACIÓN

____

MsC. Cristina Abad R.

PROFESORA DELEGADA POR EL DECANO

____

Ing. Ana Tapia

DECLARACIÓN EXPRESA

"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)

Denisse Aurora Cayetano Carvajal

Christian Alejandro Rivadeneira Zamora

RESUMEN

En este trabajo se presenta la implementación de un Módulo de Generación de Reportes Gráficos de una Honeynet a partir de los logs tcpdumps. El documento está dividido en 7 capítulos que comprenden el planteamiento del problema, el análisis conceptual, la plataforma cloud computing, el análisis de la solución, el diseño del módulo, la implementación del módulo, y las pruebas y análisis de resultados obtenidos al utilizar técnologias escalables de procesamiento masivo y escalable de datos para la interpretación y generación de reportes gráficos a partir de los logs tcpdumps.

En el Capítulo 1 se describe los detalles que indican la necesidad de una herramienta de gran escalabilidad. Se plantean los objetivos, el alcance, limitaciones y la justificación del presente proyecto.

En el Capítulo 2 se realiza una breve descripción del marco teórico utilizado, como los Honeypots, Honeynet y los Logs tcpdump/pcap. Posteriormente se centra en el formato de los archivos bajo estudio.

En el Capítulo 3 se describe la plataforma de desarrollo, herramientas y los servicios. Más adelante se describen cada uno de estos servicios y su utilidad dentro del proyecto, centrándose en el papel del modelo de programación map/reduce dentro del procesamiento masivo y escalable de datos.

Para una correcta selección de la arquitectura a implementar se llevó a cabo un análisis del problema, llegando a la descripción de los diferentes reportes que se grafican, detallados en el Capítulo 4.

En el Capítulo 5 se detalla el diseño general de la arquitectura para el desarrollo del módulo, y del front-end para la visualización de los diferentes reportes gráficos.

En el Capítulo 6 se realiza una descripción detallada de los pasos seguidos para la implementación del módulo de lectura y procesamiento de archivos pcap, acoplándolos al ambiente Hadoop.

A partir del módulo implementado se realizaron pruebas que permitan tener una referencia de la efectividad y eficiencia del trabajo realizado en contraste con herramientas existentes. Estas pruebas se encuentran detalladas en el Capítulo 7.

ÍNDICE GENERAL

ÌNDICE DE GRÁFICOS

ABREVIATURAS

GB Gigabyte

TB Terabyte

FIEC Facultad de Ingeniería en Electricidad y Computación

CIB Centro de Información Bibliotecaria

PCAP Packet Captured

IaaS Infrastructure as a Service

AWS Amazon Web Services

EC2 Amazon Elastic Compute Cloud

S3 Amazon Simple Storage Service

HDFS Hadoop Distributed FileSystem

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ARP Address Resolution Protocol (Protocolo de Resolución de Direcciones)

TCP Transmission Control Protocol (Protocolo de Control de Transmisión)

UDP User Datagram Protocol

ICMP Internet Control Message Protocol

HTTP HyperText Transfer Protocol (Protocolo de Transferencia de).

FTP File Transfer Protocol (Protocolo de Transferencia de Archivos).

Telnet Teletype Network.

SSH Secure Shell (Intérprete de Comandos Seguro)

SMTP Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo).

POP3 Post Office Protocol (Protocolo 3 de Correo).

IP Internet Protocol (Protocolo de Internet).

XML Extensible Markup Language (Lenguaje de marcas extensible).

ESPOL Escuela Superior Politécnica Del Litoral.

MXML Macromedia eXtensible Markup Language

INTRODUCCIÓN

En la actualidad, la mayoría de las redes mantienen conexión con el ancho mundo de la Internet. Ésta conexión representa un constante peligro debido a la vulnerabilidad hacia los ataques que realizan los hackers. Ésta es en efecto la razón principal por la cual en muchas redes, donde es primordial mantener la confidencialidad y consistencia de sus datos, se ejecutan algunos tipos de mecanismos de seguridad sobre sus conexiones.

Estudios han demostrado que las redes más atacadas son las empresariales [10], otro blanco muy común son las redes universitarias. Debido a esto, los administradores de redes tratan de manera cotidiana de controlar las vulnerabilidades a través de mecanismos que sirven de defensa para las redes en general. Una de estos mecanismos de defensa es el uso de una Honeynet con la cual podemos analizar los medios que se usan para realizar ataques.

En el mercado existen varias herramientas que ayudan al análisis de, lo que podría ser, un posible ataque en la red; sin embargo, a pesar de que dichas herramientas se encuentran disponibles tanto de manera gratuita como propietaria, no abastecen la demanda para el análisis de una cantidad grande de información (en el orden de los GB y TB).

El presente trabajo describe una herramienta escalable y distribuida para el procesamiento de logs de tráfico de red (en formato pcap) y la generación de reportes gráficos a partir de dichos logs, de tal manera que dichos reportes pueda ser utilizado como parte de procesos de análisis forense de seguridad informática.

CAPÍTULO 1.

1 PLANTEAMIENTO DEL PROBLEMA

1.1 Definición del Problema

El Capítulo Ecuador del Proyecto Honeynet ha montado una red que permite generar logs con datos obtenidos de las conexiones que se realizan a esta red. Considerando que sobre esta red se han montado "servicios falsos", todo tipo de requerimientos que se realice a uno de estos servicios es considerado como ataque. A partir de esta información se pueden generar reportes de gran utilidad.

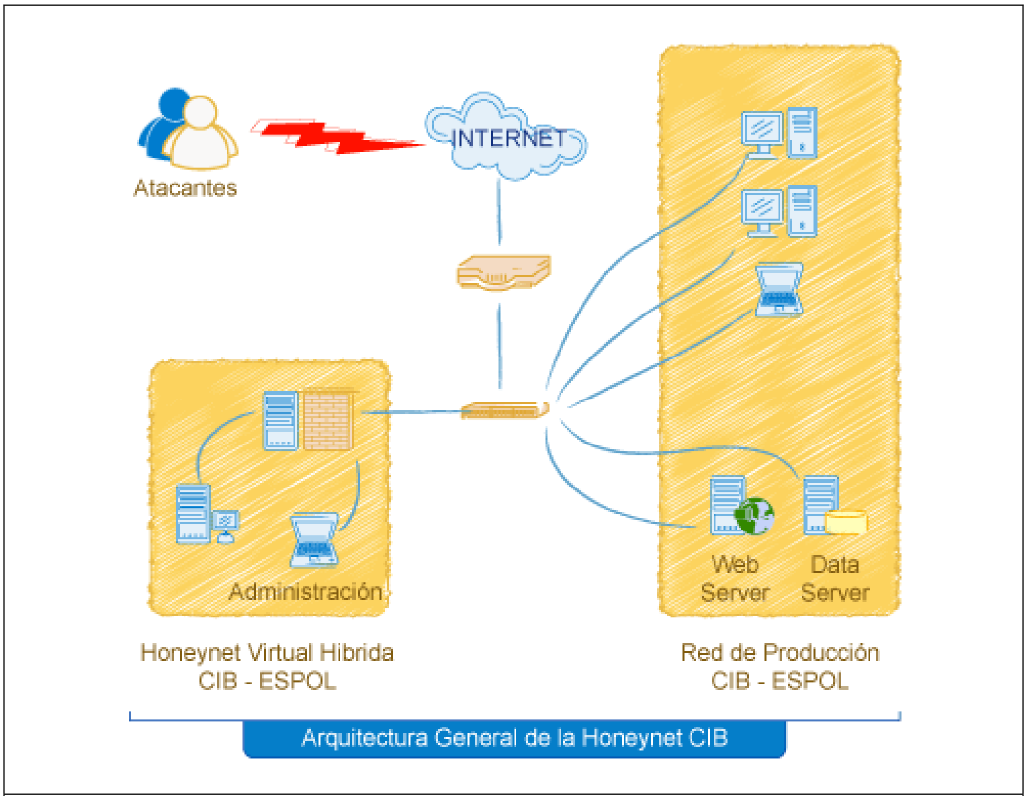

Figura 1.1.1 Arquitectura General de la Honeynet CIB

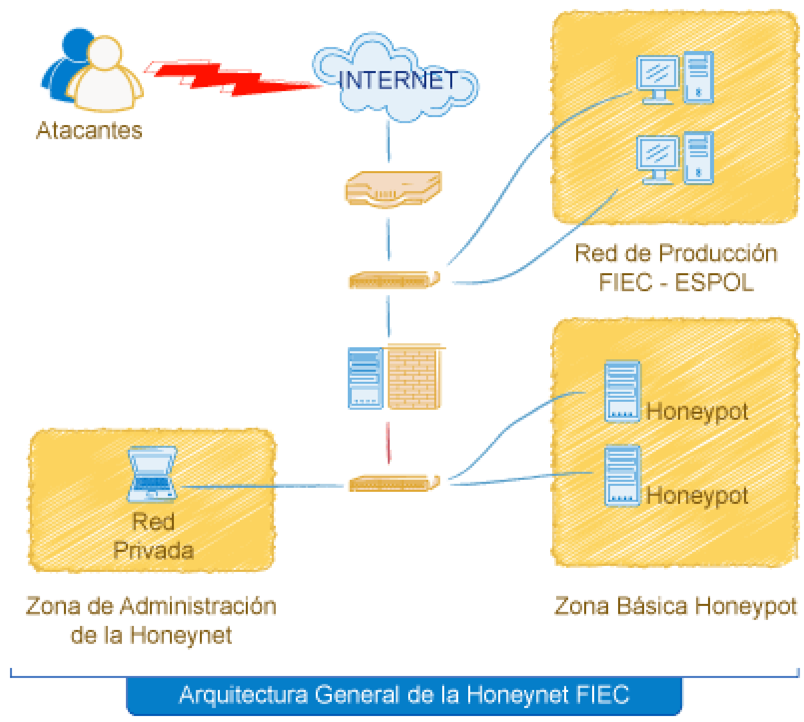

Figura 1.1.2 Arquitectura General de la Honeynet FIEC

Como parte de las actividades de este Capítulo, se realizó un estudio que generó un historial de ataques que se recibieron en dos honeynets instaladas dentro del campus universitario Gustavo Galindo de la ESPOL. Ambas arquitecturas se las puede visualizar en la Figura 1.1-1 y en la Figura 1.1-2. Durante aproximadamente 4 meses (de Agosto a Noviembre de 2008), se monitoreó y registró todo el tráfico recibido por estas dos honeynets. A partir del tráfico capturado se realizaron varios análisis forenses y se identificaron ciertos patrones de ataque.

Lastimosamente, no se pudo realizar gráficas resumiendo el tráfico de todo el periodo analizado, ya que los registros de los logs al ser muy grandes (mayores a 4 GB) saturaban a las diversas herramientas disponibles en el mercado para este propósito. Un procesamiento tradicional de estos logs tampoco fue realizado, debido al largo tiempo que esto hubiera tomado. Es por esto que surge la necesidad de integrar tecnología de alta escalabilidad, capaz de procesar una gran cantidad de datos y proporcionar prontos resultados.

1.2 Objetivos generales

La presente tesis tiene como objetivo enriquecer el conjunto de datos almacenados en archivos tcpdumps pcap, a través de Reportes Gráficos que proporcionen información oportuna, precisa y concisa, que pueda ser analizada e interpretada por los administradores de red.

1.3 Objetivos específicos

Para llegar al objetivo general se definieron los siguientes objetivos específicos:

- Apoyar la integración de reportes gráficos que describan un comportamiento específico de requerimiento malicioso, como parte del Capítulo Ecuador del Proyecto Honeynet.

- Demostrar la evolución o cambio de la Honeynet en relación a los ataques registrados, una vez que éstos han sido agregados a los logs correspondientes.

- Graficar el comportamiento de distintos ataques en base a la información generada por la Honeynet, con el objeto de conocer debilidades y fortalezas de las redes y tomar decisiones que permitan un proceso de mejora contínua.

- Implementar un módulo de transformación de de archivos pcap en una estructura de datos definida (parseo), que se acople a Hadoop y sea escalable para otros usos.

1.4 Justificación

La necesidad de una herramienta que permita el procesamiento de datos en gran escala, y que además permita el filtrado de datos para generar reportes, ofreciendo una visión global y realista de los posibles tipos de ataques a los que la red es susceptible; es la razón principal para enriquecer el conjunto de datos almacenados en archivos tcpdumps pcap, y poder mostrarlos a través de reportes gráficos que proporcionen información oportuna, precisa y concisa, que pueda ser analizada e interpretada por los administradores de red.

Con la integración de una herramienta de gran escalabilidad, los datos registrados dentro de los log podrán ser procesados y generaran un resultado que apoyará la visualización sistemática de los datos registrados en los archivos tcpdumps.

La implantación de éste módulo (PcapReports), pondrá a disposición de la comunidad, una herramienta que permita la visualización a través de reportes gráficos del comportamiento de las actividades capturadas a lo largo del Proyecto Honeynet.

1.5 Alcances y limitaciones

Este módulo básicamente procesará los datos registrados en los logs del Proyecto Honeynet generando un resultado que será visualizado en un reporte gráfico, y permitirà detallar la información desde lo más general hasta lo más particular posible.

Los datos con los que se cuenta son los proporcionados por el proyecto Honeynet Capítulo Ecuador que corresponden a la Facultad de Ingeniería en Electricidad y Computación (FIEC) y al Centro de Información Bibliotecaria (CIB).

Cabe recalcar, que el presente módulo no se limita a los archivos históricos recolectados hasta el momento. Por el contrario, se espera que conforme se siga monitoreando la red, se pueda continuar retroalimentando el historial de información a tal punto de poder seguir de cerca algún comportamiento malicioso que pueda afectar de alguna manera la seguridad de los datos que se encuentren viajando, y la integridad de los servicios que se encuentren disponibles en la red.

CAPÍTULO 2.

2 ANÁLISIS CONCEPTUAL

2.1 Honeypots

Un Honeypot o "tarro de miel" en el campo de la seguridad en redes de información se define como un recurso de la red que se encuentra voluntariamente vulnerable para que el atacante pueda examinarla y/o atacarla [1]. Directamente no es solución a ningún problema; su función principal es recoger información importante sobre el atacante que permita prevenir estas incursiones dentro del ámbito de la red real en casos futuros.

La información recogida en la honeypot se almacena en logs tcpdump/pcap de manera continua durante el tiempo funcional de la misma.

2.2 Honeynet

Una Honeynet es una red entera de gran interacción diseñada para ser atacada con la intención de almacenar información sobre el tráfico ocurrido en cada uno de los Honeypot que la integran. Implementada con Sistemas Operativos reales y corriendo servicios reales, nos ayuda a identificar nuevas técnicas de ataque y analizar el “modusoperandi" de los intrusos.

2.3 Logs tcpdumps/pcap

El propósito de los archivos log es el de registrar actividades durante un periodo de tiempo en particular, guardando todas las peticiones (requests) y servicios entregados por el honeypot. La información se encuentra en formato binario en extensiones de archivos pcap.

Estos archivos pcap (Packet Captured) mantienen un formato definido, el cual permite conocer toda la información (Quién, Qué, Cuándo, Dónde, Cuánto) correspondiente a los distintos paquetes que se transmitieron desde y hacia el Honeypot.

El archivo se compone de una Cabecera Global, la cual contiene información global seguida de cero ó más registros por cada paquete capturado, tal como se muestra en la Figura 2.3-1:

Figura 2.3.1 Formato de los Archivos PCAP

El Formato de los archivos pcap nos será de gran utilidad al momento de interpretar los datos registrados y poderla procesar masivamente para generar los diferentes reportes que ayudaràn a la mejora contÍnua en el control preventivo de acceso hacia la red bajo estudio.

CAPÍTULO 3.

3 PLATAFORMA DE CLOUD COMPUTING

El procesamiento de un gran volumen de datos, como los almacenados en los archivos pcap, es costoso considerando la infraestructura que se necesita para poder llevarla a cabo.

Cloud Computing, o computación en las nubes, es un modelo de prestación de servicios orientada hacia la escalabilidad. En la actualidad existen muchos proveedores de servicios de cloud computing, específicamente en el área de IaaS (Infrastructure as a Service). Para PcapReports se ha seleccionado al proveedor Amazon, a través de sus Amazon Web Services (AWS), debido a que (1) en la actualidad es la empresa líder en este tipo de servicios, y (2) nos ha proporcionado acceso gratuito a sus servicios a través de su programa de fondos AWS in Education. AWS presta varios servicios de cloud computing, entre los cuales encontramos el Amazon Elastic Compute Cloud (Amazon EC2) y el Amazon Simple Storage Service (Amazon S3).

3.1 Amazon EC2

Es un servicio Web que proporciona un tamaño variable de capacidad de cómputo dentro de una infraestructura que soporta un modelo de prestación de servicios. Proporciona herramientas útiles para desarrollar y ejecutar aplicaciones flexibles, aislando los escenarios de fallo.

El cluster de Amazon EC2 es un ejemplo de infraestructura necesaria para ejecutar la implementación del módulo, en combinación con las implementaciones open source de MapReduce provista por el proyecto Hadoop.

3.2 Amazon S3

Es un servicio de almacenamiento escalable con una interfaz simple de servicio Web. AWS ha integrado S3 con EC2, de tal manera que transferir datos entre ambos servicios esea eficiente y no represente un costo alto para el cliente.

3.3 Procesamiento Masivo y Escalable de Datos

La cantidad de datos obtenidos de los tcpdumps de una Honeynet crece muy rápido, lo cual hace que su procesamiento en un solo computador resulte imposible (por restricciones de memoria y debido a la gran cantidad de tiempo que tomaría procesarlos). Este tipo de problemas cae dentro de un área que en la actualidad se denomina “data intensive scalable computing” ó “procesamiento masivo y escalable de datos”.

En la actualidad, existen algunas plataformas para el procesamiento masivo y escalable de datos, pero de todas ellas, Hadoop es la que más acogida a tenido, debido a estar basada en principios desarrollados por Google [11] y a contar con el apoyo de gigantes como Yahoo! y Facebook.

3.4 Hadoop

Apache Hadoop, ofrece una alternativa al procesamiento masivo de datos de manera distribuida, permitiendo el análisis de información en nodos distribuidos de cantidades ingentes de datos.

Aprovechando el modelo de programación Map/Reduce, el cual permite implementar programas paralelos definiendo cómo el procesamiento puede ser dividido en pequeños fragmentos de trabajo, podemos segmentar juegos de datos de gran tamaño y poder distribuirlos de un modo fraccionado en los distintos nodos que se encuentran bajo un clúster de análisis, en que pueden ubicarse computadores de distinta potencia indistintamente [9].

Trabajando bajo Hadoop Distributed FileSystem (HDFS) podemos almacenar Petabytes de datos a través de miles de nodos, esta plataforma nos asegura que los datos están siempre disponibles, incluso si los nodos subyacentes fallan [9].

3.5 Modelo de Programación Map/Reduce

Map/Reduce es uno de los elementos estructurales de Google [11]. Este modelo de programación se basa en una estructura muy simple de trabajos que "mapean", y trabajos que "reducen", de forma que los primeros buscan la información en forma de pares clave/valor, y los segundos las unen, simplifican y procesan.

Bajo este modelo de programación se paralelizan las tareas, se controlan los trabajos y la comunicación entre ellos, se controlan errores; básicamente, se simplifica el trabajo que puede ser llevado a cabo en un tiempo excesivo (en algunos casos).

CAPÍTULO 4.

4 ANÁLISIS DE LA SOLUCIÓN.

4.1 ¿Qué estamos resolviendo?

El estudio realizado por el Capítulo Honeynet del Ecuador en una red específica dentro de la ESPOL, dejó como resultado logs tcpdump de aproximadamente 4 GB de tamaño, los mismos que contienen información de los distintos paquetes que fueron emitidos y recibidos por la honeypot a través de la red.

Con la integración de Hadoop, los datos registrados dentro de los logs tcpdump son procesados en gran escala, permitiendo conocer de manera sistemática cuáles son los patrones de ataques sufridos, los tipos de ataques, y las vulnerabilidades a las que la red fue susceptible durante ese periodo de tiempo.

Cabe recalcar, que el módulo desarrollado no se limita a los archivos históricos recolectados hasta el momento. Por el contrario, se espera que conforme se siga monitoreando la red, se pueda continuar retroalimentando el historial de información a tal punto de poder seguir de cerca algún comportamiento malicioso que pueda afectar de alguna manera la seguridad de los datos que se encuentren viajando, y la integridad de los servicios que se encuentren disponibles en la red.

4.2 ¿Por qué lo estamos resolviendo?

Este módulo busca poner a disposición de la comunidad una herramienta que permita visualizar a través de Reportes Gráficos el comportamiento de las actividades capturadas a lo largo del Proyecto Honeynet, permitiendo ir de lo general a lo particular.

La finalidad de estos Gráficos es poder analizar de manera detallada las vulnerabilidades que se presentan en los servidores actuales y que los distintos hackers están explotando. Este tipo de información será de utilidad para mejorar de manera contínua la seguridad en las redes actuales.

4.3 ¿Cómo lo estamos resolviendo?

Hemos Hadoop como herramienta para el procesamiento masivo de datos, a través de clústers levantados bajo demanda utilizando los servicios de Amazon Web Services. Con esta plataforma se procesa los archivos pcaps directamente en su formato binario, y se generan reportes en formato XML, los cuales pueden ser graficados utilizando una interfaz Web implementada con Adobe Flex. Entre los reportes que se generan, se encuentran:

- **Por Protocolo por cada mes**

En este reporte se puede visualizar el comportamiento mensual de los distintos protocolos, ya sean estos bajo la Capa de Red (IPv4, IPv6, ARP), bajo la Capa de Transporte (TCP, UDP, ICMP), ó bajo la Capa de Aplicación (HTTP, FTP, Telnet, SSH, SMTP, POP3).

- **Tráfico por País**

Debido a que los ataques provienen de cualquier punto geográfico del planeta, la identificación del origen de los mismos da una idea más clara sobre el nivel de fuentes maliciosas que existen alrededor de nuestro entorno. Esto es factible gracias al reconocimiento de origen de la IP fuente del ataque.

- **Tráfico por IP.**

El conocimiento de origen de los ataques a través de la IP fuente, ayuda a que los controles de acceso se encuentren con filtros cada vez más restrictivos, manteniendo bajo estudio aquellas IP que mantengan un comportamiento malicioso.

- **Cantidad de Bytes por día de la semana y hora específica.**

El volumen de datos que se transfieren a través de la red, pueden llegar a ser incalculables en algunos casos. Sin embargo, el poder tener un control sobre la cantidad de ataques que se reciben en una Honeypot durante el tiempo funcional de la misma en la red, ayuda a regularizar el tráfico sobre la misma, y a prevenir el ingreso de ataques específicos que provengan, por ejemplo, de alguna IP determinada.

CAPÍTULO 5.

5 DISEÑO DEL MÓDULO.

5.1 Diseño General

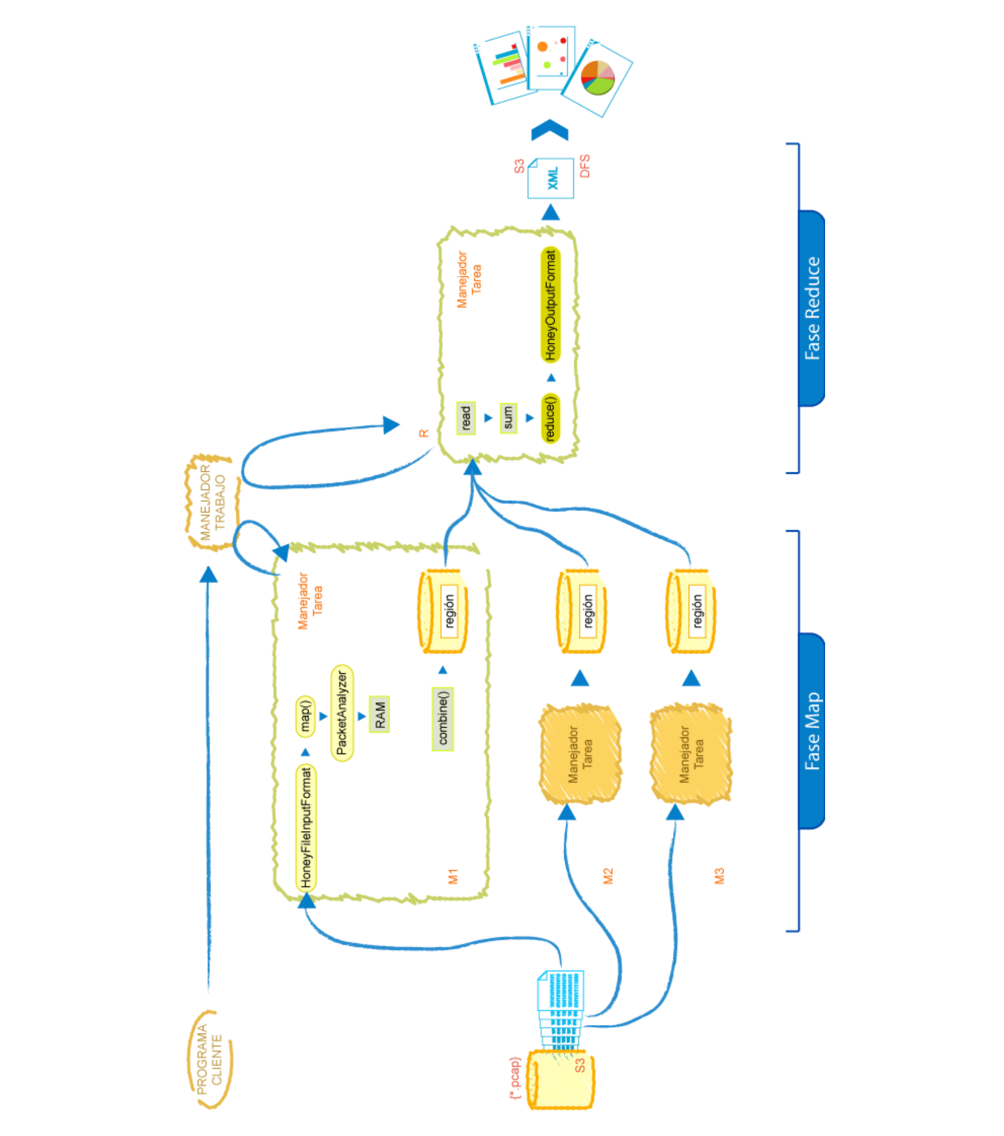

Figura 5.1.1 Diseño Arquitectónico del Módulo

5.1.1 Archivos de entrada

Cada uno de los archivos tcpdumps se almacenan en Amazon S3 para su posterior procesamiento. Hadoop nos provee de métodos para leer archivos de texto. Sin embargo, para poder procesar cada uno de los pcap almacenados en formato binario, fue necesario crear una clase que extienda de FileInputFormat[1] y que pueda leer los datos almacenados en binario. A esta clase se la denominó HoneyFileInputFormat y permite leer paquetes almacenados en archivos pcap, sin permitir que estos sean divididos en el proceso.

5.1.2 El ambiente EC2

Lo componen básicamente dos tareas, la ejecución de los mappers y la ejecución de los reducers.

Cada mapper recibe un archivo de entrada y lo parsea de tal modo que pueda obtener los datos de interés. Es decir, en cada mapper se toma cada uno de los paquetes que se encuentran registrados en el archivo de entrada y se le extraen los datos necesarios para el procesamiento. Como se manejan cuatro reportes diferentes, se ha definido cuatro mappers independientes para cada uno de ellos, descritos a continuación.

---

[1] El FileInputFormat es la clase base del API de Hadoop para la lectura de datos de entrada que provienen de archivos.

Para los Reportes Gráficos por Protocolo por cada mes, es necesario conocer el protocolo de comunicación que se utilizó durante la transmisión de datos. De igual forma, la fecha de captura del paquete, y el tamaño del mismo. Cada uno de estos datos permitirá en lo posterior realizar un correcto filtrado de información.

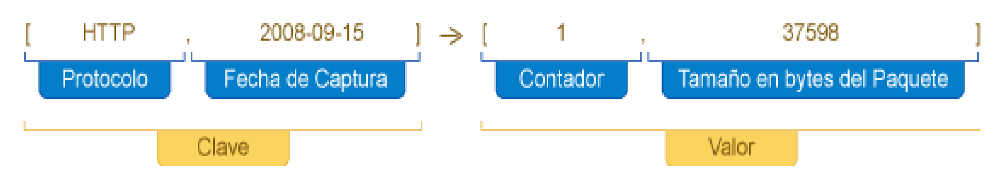

Figura 5.1.2.1 Formato de salida del Mapper de Protocolo por cada mes

Por cada paquete que se parsea en el mapper, se forma un registro con el formato que se presenta en la Figura 5.1.2-1, el cual detalla como está formado el modelo clave/valor. Dentro de la clave, se encuentran el protocolo y la fecha de captura, mientras que en el valor se encuentran un contador y el tamaño en bytes del paquete.

En el caso de los Reportes Gráficos de Tráfico por País, se necesita conocer primero la IP desde la cual se envió dicho paquete. Una vez que se obtiene la IP, se procede a buscar el País correspondiente con ayuda de la librería InetAddressLocator, capaz de transformar la dirección IP en un número entero, y de esta manera poder verificar la procedencia del paquete, adicional al dato del País de procedencia, también se extraen la fecha de captura y el tamaño del paquete.

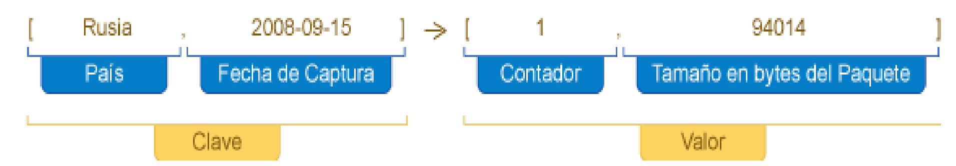

Figura 5.1.2.2 Formato de salida del Mapper de Tráfico por País

Por cada paquete que se parsea en el mapper, se forma un registro con el formato que se presenta en la Figura 5.1.2-2, el cual detalla como está formado el modelo clave/valor. Dentro de la clave, se encuentran el país de procedencia y la fecha de captura, mientras que en el valor se encuentran un contador y el tamaño en bytes del paquete.

Por otra parte, para los Reportes Gráficos de Tráfico por IP, es requerido conocer la IP fuente del paquete, así como la fecha de captura y el tamaño del mismo.

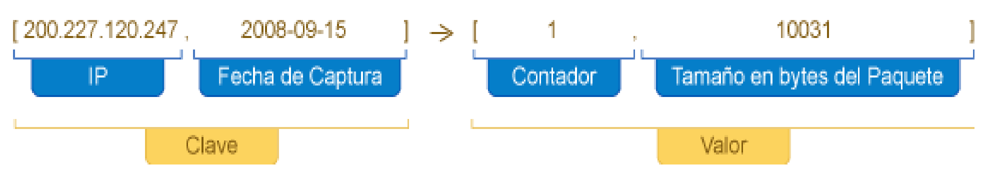

Figura 5.1.2.3 Formato de salida del Mapper de Tráfico por IP

Por cada paquete que se parsea en el mapper, se forma un registro con el formato que se presenta en la Figura 5.1.2-3, la cual detalla como está formado el modelo clave/valor. Dentro de la clave, se encuentran la IP de origen y la fecha de captura, mientras que en el valor se encuentran un contador y el tamaño en bytes del paquete.

Y por último, para los reportes de cantidad de bytes por día de la semana y hora específica, se requieren del día de envío del paquete, la hora de captura y el tamaño del mismo.

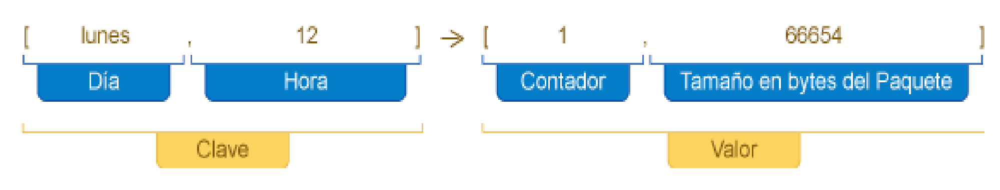

Figura 5.1.2.4 Formato de salida del Mapper de Cantidad de bytes por día de la semana y hora del día

Por cada paquete que se parsea en el mapper, se forma un registro con el formato que se presenta en la Figura 5.1.2-4, la cual detalla como está formado el modelo clave/valor. Dentro de la clave, se encuentran el día de la semana y la hora de captura, mientras que en el valor se encuentran un contador y el tamaño en bytes del paquete.

El proceso de parsear cada archivo de entrada se lo realiza gracias a la librería JNetStream, la misma que está orientada a la interpretación de paquetes de red de manera primitiva. Esta librería no posee métodos definidos para la extracción de datos, razón por la cual se implementó una clase PacketAnalyzer capaz de satisfacer las necesidades con respecto a la información requerida para la correcta generación de cada reporte.

A medida que el modelo clave/valor se va generando a partir de cada mapper, para cada tipo de reporte se va ejecutando el correspondiente reducer, cada uno de ellos tiene la finalidad de agrupar de manera correcta cada valor con su respectiva clave.

Una vez que todos los mappers han entregado sus correspondientes porciones de datos, el reducer procesa dichos datos y los agrupa en un solo resultado final.

5.1.3 El ambiente de los archivos de salida

El resultado del reducer (para nuestro caso) debe de almacenarse en un archivo XML. Para poder realizar el correspondiente almacenamiento, se debió de crear una clase que extienda de FileOutputFormat[1] y que sobrescriba los métodos necesarios para que el resultado final se almacene como archivo XML en Amazon S3.

[1] El FileOutputFormat es la clase base del API de Hadoop para la escritura de datos de salida en archivos (de texto o binario, dependiendo de la subclase).

5.2 Diseño del Front-End

Figura 5.2.1 Diseño Arquitectónico del Front-End

5.2.1 Adobe FLEX

La interacción entre los reportes y el usuario debía ser muy práctica, por lo que el nivel de presentación se optó por desarrollarlo con Adobe FLEX, el cual trabaja usando lenguaje XML.

Mediante Gráficos Estadísticos como Barras, Pastel y Burbujas se puede visualizar, interpretar y analizar el comportamiento de la red, pudiendo detectar posibles ataques por parte de hackers. Cada uno de estos gráficos es el resultado del procesamiento de datos masivos almacenados en archivos tcpdumps.

CAPÍTULO 6.

6 IMPLEMENTACIÓN DEL MÓDULO

Para la implementación del módulo se heredó de la clase InputFormat que viene por defecto en Hadoop, tal y como se lo indicó en el capítulo anterior. Los archivos que recibimos como entrada son de tipo binario, y de gran tamaño (en el orden de los GB). Por tal motivo, la división de los archivos se la debía manejar de manera diferente.

HDFS utiliza un tamaño de bloque en el sistema de archivos por defecto de 64MB[1]. El mismo valor se lo mantiene para cada segmento de tarea chunk (espacio en se almacena una división de trabajo listo para los mappers). Sin embargo, con el propósito de reducir el costo de búsqueda en disco de la información en comparación al de lectura y escritura, se lo definió en 512MB tanto el tamaño del bloque como el segmento de tarea chunk.

Cada archivo de entrada mantiene un conjunto de paquetes que no necesariamente mantienen un tamaño fijo. Por tal motivo se considera que del tamaño total de los archivos de entrada, se genera una perdida de aproximadamente 10%. Esta perdida se genera debido a que al

[1] Este tamaño es de 128MB por defecto en las configuraciones de la empresa Cloudera (http://www.cloudera.com).

realizar las respectivas divisiones algunos paquetes son truncados, formándose restos de paquetes tanto al inicio como al final de cada chunk, dichos restos no son considerados al momento de extraer la información. Los detalles de cómo se generan los InputSplits pueden encontrarse revisando la implementación del método GetSplitsForPaths de la clase InputFormat del API de Hadoop.

Lo mappers son los que acoplaran cada uno de los chunks bajo la estructura clave/valor. Para obtener cada uno de los reportes, fue necesario extraer de los paquetes almacenados en los archivos binarios, la información correspondiente a fechas, tamaños, protocolos, direcciones IP, entre otras. Cada reporte tiene asociado su tarea map, por ejemplo, en el caso del Reporte de tráfico por País, la parte principal del mapper se encarga de obtener la fecha de captura y el País de procedencia, tal como se muestra en la porción de código incluida a continuación.

```
while((packet = decoder.nextPacket()) != null)
{
    int num_paquetes = packet.getHeaderCount();
    for(int i = 0 ; i < num_paquetes ; i++)
    {
            header = packet.getHeader(i);
            protocolo = pa.obtenerProtocolo(header);

            if(i==0)
            {
                    ipSRC = pa.sourceIP(packet);
                    fechaCaptura = pa.obtenerFechaCaptura(packet);
                    fechaCaptura = fechaCaptura.substring(0,7);
                    tamanoPaquete = pa.obtenerTamanoPaquete(packet);
                    StringBuilder key_value=new StringBuilder("");
                    StringBuilder value_value=new StringBuilder("");
                    if(!ipSRC.equals("desconocida")){
                            s=InetAddressLocator.getLocale(ipSRC);
                            if(s.getDisplayCountry().equals("EU") ||
                            s.getDisplayCountry().equals("") ||
                            s.getDisplayCountry().equals("**"))
                            key_value.append("NNN,DESCONOCIDO");
```

```
                else{
                        key_value.append(s.getISO3Country());
                        key_value.append(",");
                        key_value.append(s.getDisplayCountry());
                }
        }
        else{
                key_value.append("NNN,DESCONOCIDO");
        }
        //key_value.append(ipSRC);

        value_value.append("1,");
        value_value.append(tamanoPaquete);
        output.collect(new Text(key_value.toString()), new Text(value_value.toString()));
    }
}
```

Para el Reporte por Protocolo mensual, el mapper asociado busca el protocolo correspondiente al paquete que fue transmitido por la red, cuya implementación se incluye a continuación.

```
while((packet = decoder.nextPacket()) != null) {
        int num_paquetes = packet.getHeaderCount();

        for(int i = 0 ; i < num_paquetes ; i++)
        {
                header = packet.getHeader(i);
                protocolo = pa.obtenerProtocolo(header);
                fechaCaptura = pa.obtenerFechaCaptura(packet);
                fechaCaptura = fechaCaptura.substring(0,7);
                tamanoPaquete = pa.obtenerTamanoPaquete(packet);
                StringBuilder key_value = new StringBuilder("");
                StringBuilder value_value = new StringBuilder("");
                key_value.append(protocolo);
                key_value.append(",");
                key_value.append(fechaCaptura);
                value_value.append("1,");
                value_value.append(tamanoPaquete);
                output.collect(new Text(key_value.toString()), new Text(value_value.toString()));

        }
}
```

En el caso del Reporte de Tráfico por IP, la tarea map se enfoca en extraer la IP de procedencia del paquete, como se detalla a continuación.

```
while((packet = decoder.nextPacket()) != null)  {
        int num_paquetes = packet.getHeaderCount();
        for(int i = 0 ; i < num_paquetes ; i++)
        {
                header = packet.getHeader(i);
                protocolo = pa.obtenerProtocolo(header);
```

```
                    if(i==0)
                    {
                              ipSRC = pa.sourceIP(packet);
                              fechaCaptura = pa.obtenerFechaCaptura(packet);
                    fechaCaptura = fechaCaptura.substring(0,7);
                    tamanoPaquete = pa.obtenerTamanoPaquete(packet);
                    StringBuilder key_value = new StringBuilder("");
                    StringBuilder value_value = new StringBuilder("");
                    key_value.append(ipSRC);
                    key_value.append(",");
                    key_value.append(fechaCaptura);
                    value_value.append("1,");
                    value_value.append(tamanoPaquete);
                    output.collect(new Text(key_value.toString()), new Text(value_value.toString()));
                    }
          }
     }
```

Finalmente, en el caso del Reporte de cantidad de bytes por día de la semana y hora específica, el mapper se encarga de obtener el día y la hora en el cual el paquete fue receptado, parte del código detallada a continuación.

```
while((packet = decoder.nextPacket()) != null) {
          int num_paquetes = packet.getHeaderCount();

          for(int i = 0 ; i < num_paquetes ; i++)
          {
                    header = packet.getHeader(i);
                    protocolo = pa.obtenerProtocolo(header);

                    if(i==0)
                    {
                              fechaYhora = pa.obtenerFechaCaptura(packet);
                              fechaCaptura = fechaYhora.substring(0,10);
                              dia = fechaCaptura.substring(8,10);
                              dia = dia.trim();
                              mes = fechaCaptura.substring(5,7);
                              mes = mes.trim();
                              anio = fechaCaptura.substring(0,4);
                              anio = anio.trim();
                              Calendar cal = Calendar.getInstance();
                              cal.set(Integer.parseInt(anio),Integer.parseInt(mes),Integer.parseInt(dia));
                              horaCaptura = fechaYhora.substring(11,13);
                              tamanoPaquete = pa.obtenerTamanoPaquete(packet);
                              StringBuilder key_value = new StringBuilder("");
                              StringBuilder value_value = new StringBuilder("");
                              key_value.append(String.valueOf(cal.get(Calendar.DAY_OF_WEEK)));
                              key_value.append(",");
                              key_value.append(horaCaptura);
                              value_value.append("1,");
                              value_value.append(tamanoPaquete);
                              output.collect(newText(key_value.toString()),new
                              Text(value_value.toString()));
                              }

                    }
               }
```

Los Reducers se encargan de reconstruir la información mapeada, de tal forma que cada clave este asociada a un valor determinado, y ya no a un conjunto de valores. Para nuestro proyecto, al igual que cada reporte esta asociado a un mapper, están asociados a un reducer en particular, el cual unira las distintas salidas de los mappers en un solo conjunto de datos, obteniendo tamaños totales y contadores de interés.

Por ejemplo, en el caso del Reporte de Tráfico por IP, tenemos la siguiente muestra de reducer:

```
while (values.hasNext()) {
            value_value = values.next().toString();
            String[] campos = value_value.split(",");
            numero = numero + Long.parseLong(campos[0].trim());
            total = total + Long.parseLong(campos[1].trim());
}

String[] campos = key.toString().split(",");
returnKey.append("ip=\"");
returnKey.append(String.valueOf(campos[0].trim()));
returnKey.append("\" ");
returnKey.append("fecha=\"");
returnKey.append(String.valueOf(campos[1].trim()));
returnKey.append("\"");

returnValue.append("cantidad=\"");
returnValue.append(String.valueOf(numero));
returnValue.append("\" ");
returnValue.append("tamanio=\"");
returnValue.append(String.valueOf(total));
returnValue.append("\"");
```

En el código podemos observar que vamos acoplando la información recolectada bajo un esquema XML.

Los reportes gráficos se construirán a partir de archivos de entrada con extension XML. Para generar los archivos de entrada en este formato, fue necesario heredar del OutputFormat que viene por defecto en Hadoop, completando el acoplamiento ya iniciado en la fase del Reducer.

6.1 Ejemplo de Reportes Gráficos

En el capítulo anterior se detalló brevemente cada uno de los reportes que se pondrán como ejemplo, detallando el enfoque de análisis que cada uno de ellos mantiene.

Cada uno de estos reportes gráficos es un ejemplo de lo que se puede generar a partir de la información recolectada por los archivos pcap en formato binario. Cabe recalcar que el tamaño total de información recolectada por las Honeynets es de 4.43GB en total.

La generación de estos reportes requirió que el archivo de entrada se encuentre bajo la estructura de XML. Se utiliza XML como formato de intercambio de datos considerando que es un estándar y existen diferentes librerías y aplicaciones que ya pueden parsearlo de forma nativa. Un ejemplo de estas herramientas es FLEX, que ha sido usado en nuestro proyecto para representar cuatro reportes generados en vistas que pueden servir para el posterior análisis de los datos procesados por Hadoop.

Cada uno de los reportes muestra un dashboard con tres diferentes perspectivas de análisis: el Pastel, las Barras y las Burbujas, con excepción del Reporte de Tráfico por País, que presenta un mapamundi en lugar del Pie. Dicho mapamundi presenta un degradé de colores que representan la concurrencia de ataques provenientes del País visualizado.

En el caso del Reporte de Tráfico por País, podemos visualizar en la Figura 6.1.1 la concurrencia de ataques, siendo los Países con una escala de color oscura, los que mayor ataques han enviado hacia las Honeynets implementadas en la ESPOL.

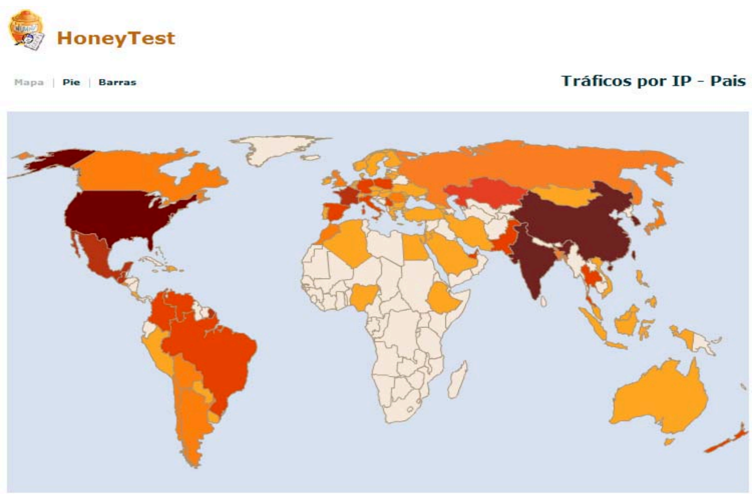

Figura 6.1.1 Reporte de Tráfico por País – Vista Heat Map

En el caso del Reporte de Tráfico por IP, se puede visualizar bajo la perspectiva de Barras la concurrencia de ataques por año-mes, tal como se muestra en la Figura 6.1.2.

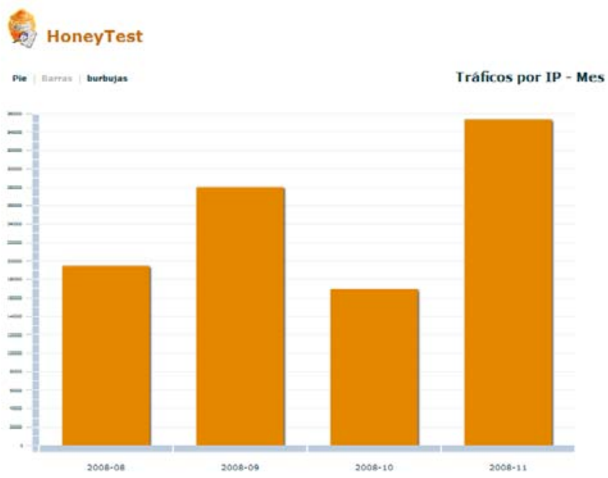

Figura 6.1.2 Reporte de Tráfico por IP – Vista Barras Nivel 0

Bajo el mismo Reporte de Tráfico por IP, a través de la funcionalidad drill-down podemos generar otro gráfico de barras a partir de la Figura 6.1.2, tal como podemos visualizar a continuación en la Figura 6.1.3.

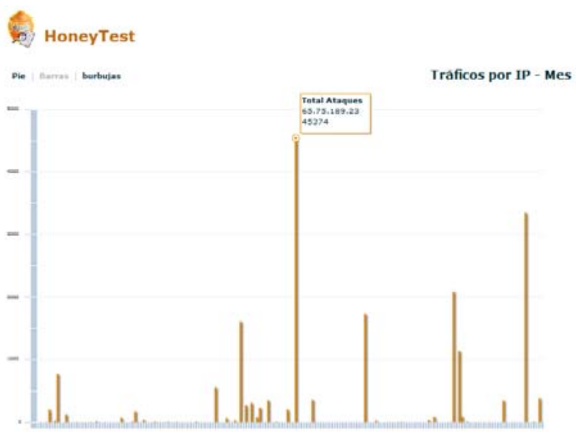

Figura 6.1.3 Reporte de Tráfico por IP – Vista Barras Nivel 1

Bajo la perspectiva de Pie, como se muestra en la Figura 6.1.4 podemos tener un comportamiento parecido al de barras, visualizando la concurrencia de ataques por año-mes que se han efectuado hacia la Honeynet.

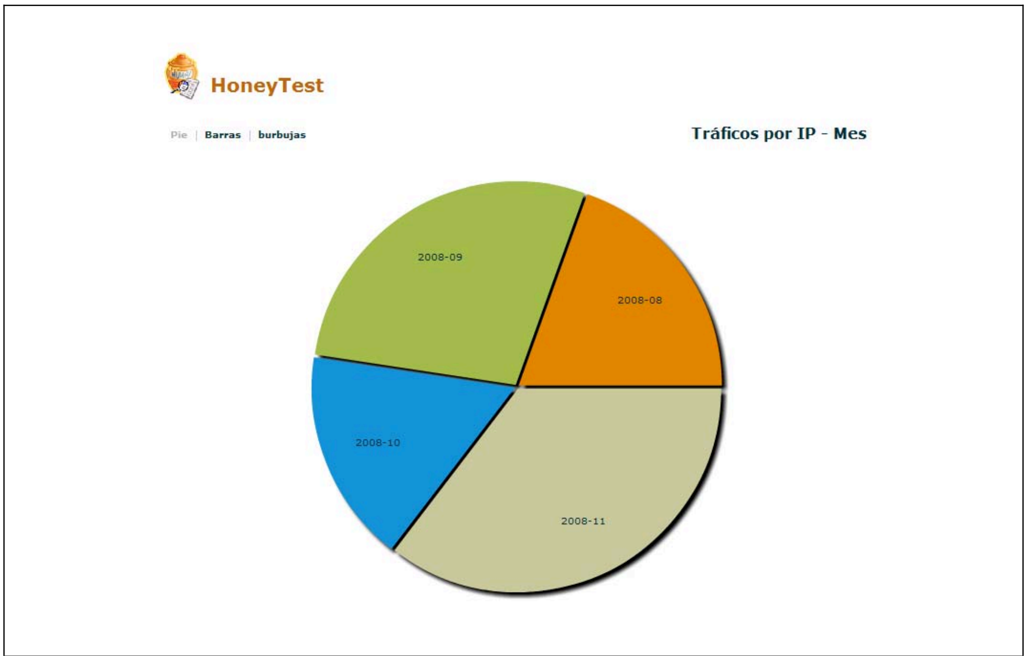

Figura 6.1.4 Reporte de Tráfico por IP – Vista Pie Nivel 0

Aprovechando la funcionalidad de navegación a lo profundo (drill-down), podemos ir de lo general a lo particular, visualizando la concurrencia por IP, de tal manera que se pueda tener una restricción de acceso con cada una de ellas, tal como se muestra en la Figura 6.1.5.

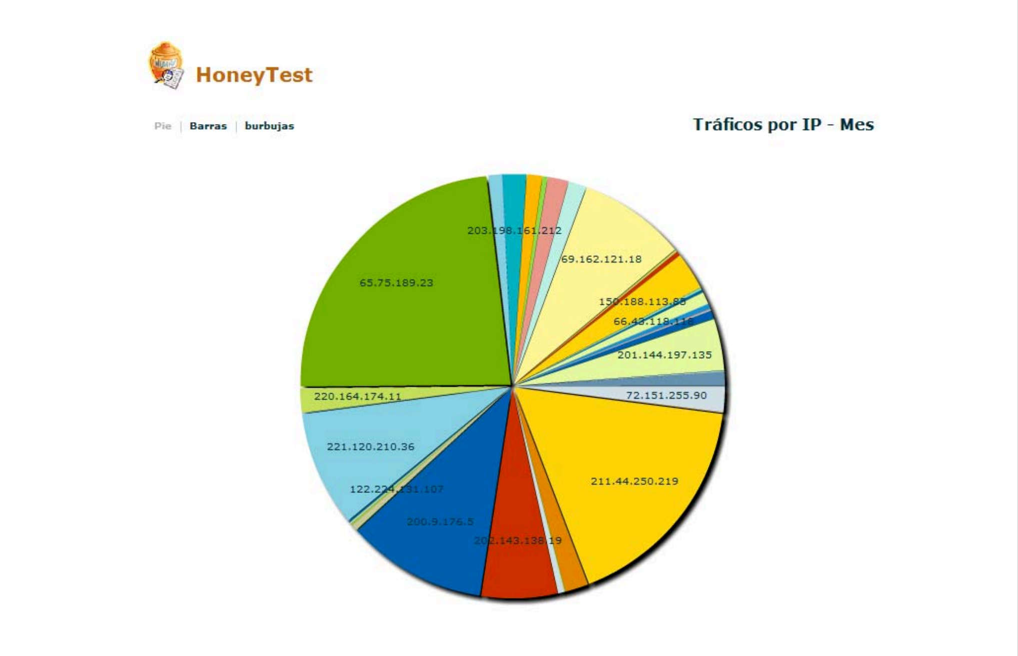

Figura 6.1.5 Reporte de Tráfico por IP – Vista Pie Nivel 1

El Reporte de cantidad de bytes por día de la semana y hora específica, permite analizar el comportamiento que presentan los ataques de manera acumulada mostrados en una semana. Tal como se muestra en la Figura 6.1.6, se puede ver que días presentan mayor tráfico hacia las Honeynets.

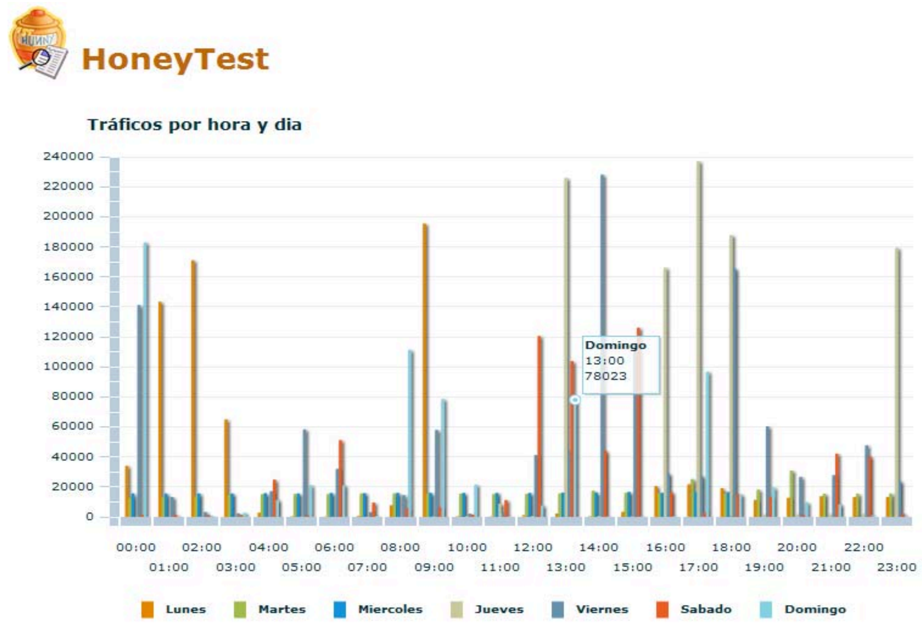

Figura 6.1.6 Reporte de Tráfico por Hora y Día

Como se mencionó anteriormente, estos reportes gráficos son un ejemplo de lo que se puede generar a partir de la información recolectada por los archivos pcap en formato binario.

CAPÍTULO 7.

7 PRUEBAS Y ANÁLISIS DE RESULTADOS

En el presente capítulo se detalla las pruebas realizadas para determinar la eficiencia y eficacia del modulo desarrollado.

7.1 Pruebas de eficacia

Para realizar estas pruebas fue necesario contrastar el trabajo realizado por el módulo de parseo de archivos *.pcap con otros programas de lectura existentes, sin embargo esto sólo fue posible con archivos en el orden de los MB, para archivos de mayor tamaño se encontraron errores en los programas probados por las limitantes que tienen al depender de las características de memoria y velocidad de procesamiento de una sola máquina.

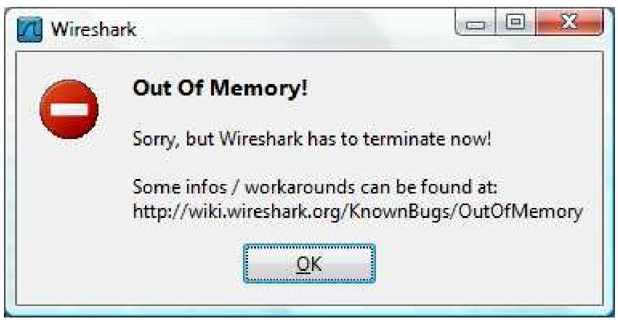

Figura 7.1.1 Error tomado al abrir archivo de tamaño 1,25GB con Wireshark V

La Figura 7.1.1 muestra el mensaje correspondiente a "Fuera de

Memoria" del programa Wireshark[1], al tratar de analizar la información contenida en el archivo de 1GB del mes de Septiembre de 2008.

Otros programas de distribución gratuita como JpcapDumper[2], simplemente se cerraron al tratar de analizar el mismo archivo.

7.2 Pruebas de eficiencia

Estas pruebas fueron realizadas en el Amazon EC2 y se han usado diferentes números de nodos para medir el tiempo que se requiere para completar la tarea.

Los resultados los describimos a continuación:

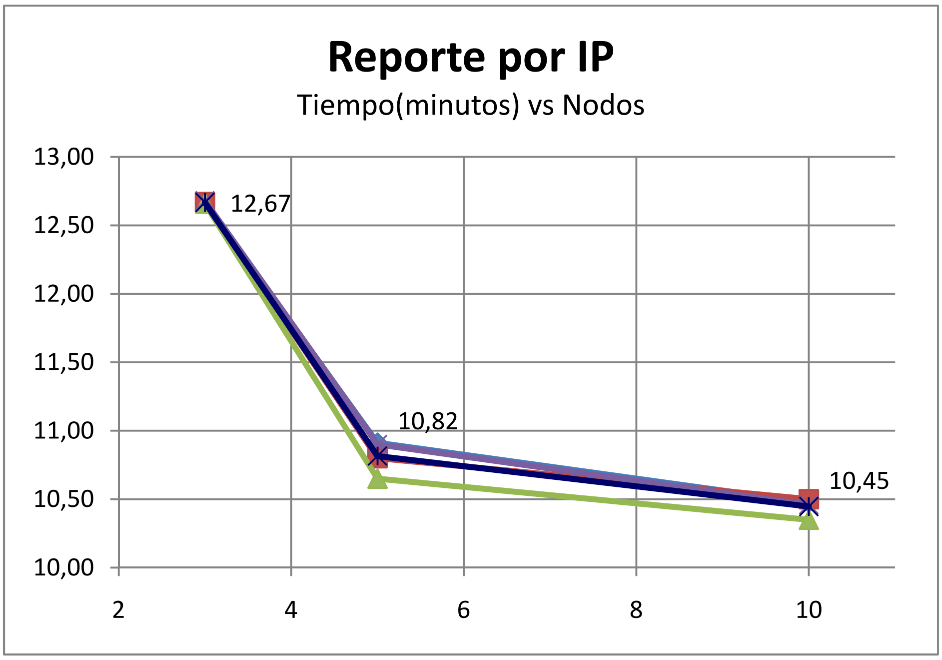

Figura 7.2.1 Reporte Gráfico por IP

[1] Wireshark puede ser descargado de manera gratuita de http://www.wireshark.org/.
[2] Disponible en http://netresearch.ics.uci.edu/kfujii/jpcapdumper/doc/index.html.

Como se muestra en la Figura 7.2.1 se realizaron tres pruebas de procesamiento masivo para el Reporte de IP, teniendo una diferencia de tiempo dentro de un intervalo de +/- 10 seg. Con tres nodos se obtuvo un tiempo de procesamiento de 12,67 seg., en cambio con cinco nodos se obtuvo un tiempo de procesamiento de 10,82 seg., mientras que con diez nodos se obtuvo un tiempo de procesamiento de 10,45 seg.

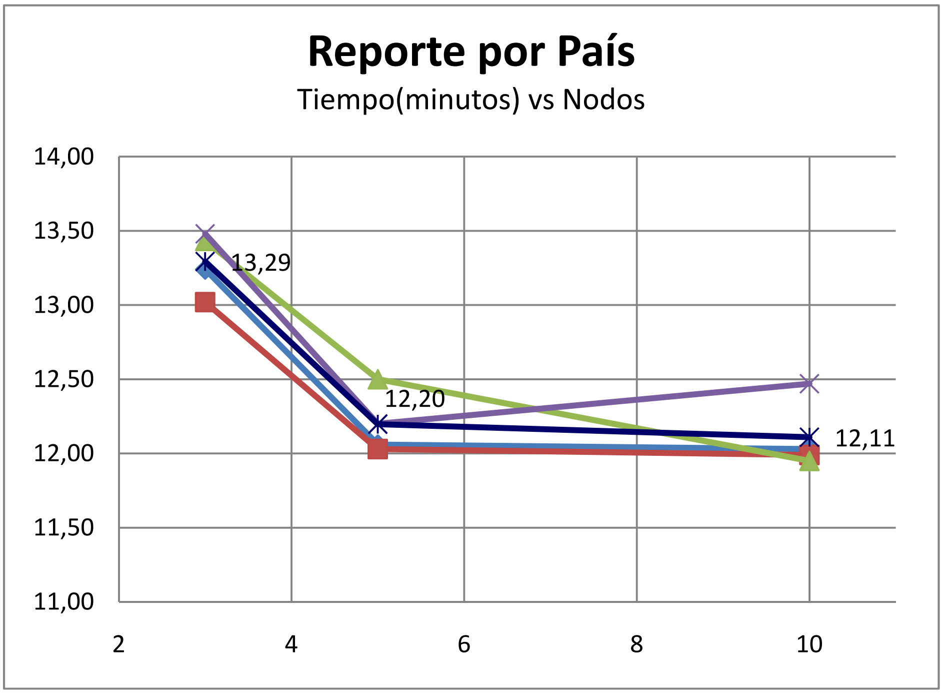

Figura 7.2.2 Reporte Gráfico por País

Como se puede observar en la Figura 7.2.2, en el caso del Reporte por País, se realizaron tres pruebas de procesamiento masivo, obteniendo una diferencia dentro de un intervalo de +/- 10 seg. Con tres nodos se obtuvo un tiempo de procesamiento de 13,29 seg., en cambio con cinco nodos se obtuvo un tiempo de procesamiento de 12,20 seg., mientras que con diez nodos se obtuvo un tiempo de procesamiento de 12,11 seg.

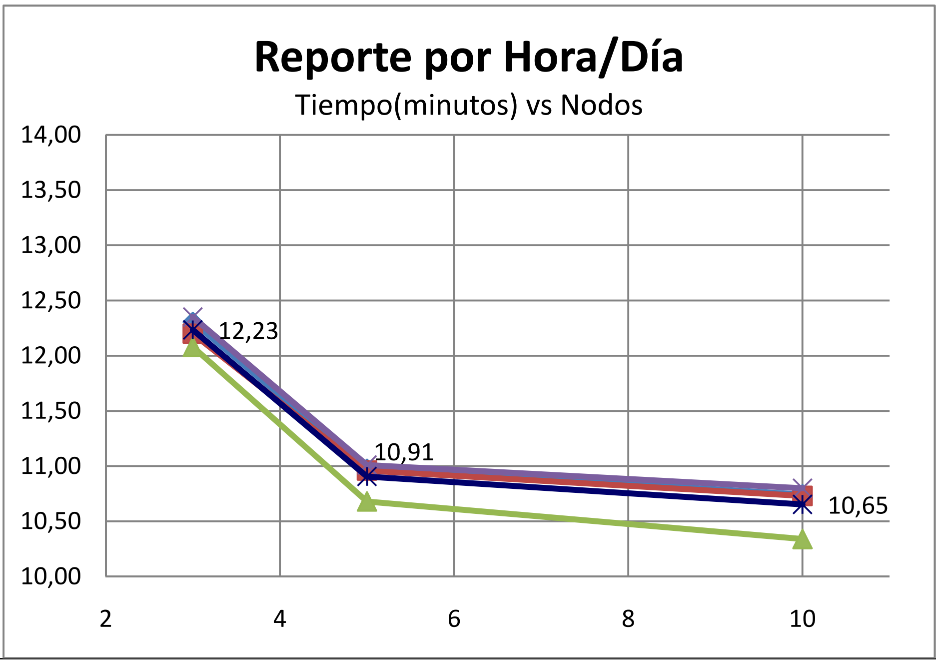

Figura 7.2.3 Reporte Gráfico por Hora y Día

Como se muestra en la Figura 7.2.3, en el caso del Reporte por Hora/Día, se realizaron tres pruebas de procesamiento masivo, obteniendo un intervalo de diferencias de +/- 10 seg. Con tres nodos se obtuvo un tiempo de procesamiento de 12,23 seg., en cambio con cinco nodos se obtuvo un tiempo de procesamiento de 10,91 seg., mientras que con diez nodos se obtuvo un tiempo de procesamiento de 10,65 seg.

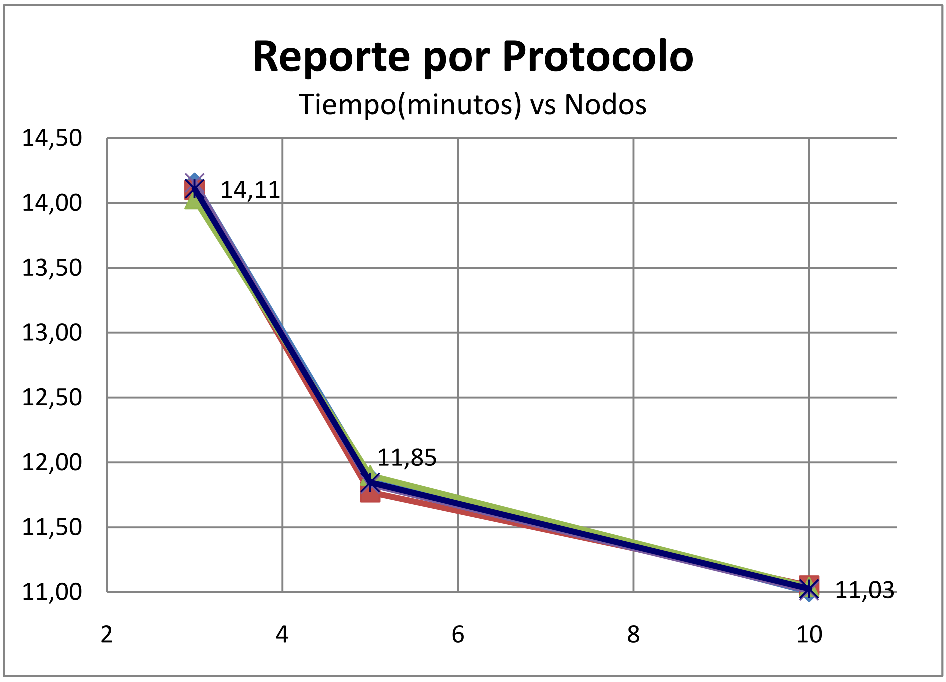

Figura 7.2.4 Reporte por Protocolo

Como se muestra en la Figura 7.2.4, se realizaron tres pruebas de procesamiento masivo para el Reporte por Protocolo, teniendo una diferencia de tiempo dentro de un intervalo de +/- 10 seg. Con tres nodos se obtuvo un tiempo de procesamiento de 12,67 seg., en cambio con cinco nodos se obtuvo un tiempo de procesamiento de 10,82 seg., mientras que con diez nodos se obtuvo un tiempo de procesamiento de 10,45 seg.

7.3 Análisis de los resultados

En cada gráfico se puede observar el comportamiento del rendimiento del módulo bajo tres condiciones de recursos. De manera general, se observa que a mayor número de nodos, se tiene un menor tiempo de procesamiento. Sin embargo, el correcto aprovechamiento del número de nodos depende del tamaño total de información a procesar, del número de mappers a utilizar y del tamaño configurado para cada chunk.

<u>CONCLUSIONES Y RECOMENDACIONES</u>

CONCLUSIONES

1. Los reportes graficos fueron generados bajo un ambiente altamente usable, pues se ha explotado varias caracteristicas visuales que provee el lenguaje utilizado para la implementacion del front end (MXML, Action Script).

2. A través de cada uno de los reportes, se puede demostrar el resultado del parseo, procesamiento y filtrado de los correspondientes logs. Pudiendo visualizar bajo varias perspectivas, diferentes enfoques de analisis, apoyando el monitoreo de las redes y controlando las debilidades y fortalezas de las mismas, dando soporte a la toma de decisiones y el mejoramiento continuo a la administración de la red.

3. Proveemos una alternativa diferente a las herramientas existentes, debido a que las actuales no brindan soporte para archivos de mayor tamaño (1GB) y mucho menos generar reportes a partir de varios archivos, si estos superan el tamaño soportado por la herramienta.

4. Se desarrolló un módulo que puede ser adaptable a las necesidades de los administradores de red para el procesamiento de los logs en formato pcap.

RECOMENDACIONES

1. Los reportes presentados son una muestra de lo que se puede realizar a partir de los archivos de salida generados por map/reduce. Sin embargo, se pueden generar reportes mas complejos, exportarlos a otros tipos de formato e inclusive visualizarlos de una forma diferente utilizando las herramientas adecuadas (muchas herramientas soportan la generación de graficos a partir de un archivo XML).

2. El analisis de archivos pcap en el ambiente Hadoop ya ha sido discutido en foros de Internet[1], donde no han encontrado una solución viable. Nosotros hemos compartido nuestra propuesta y esperamos una pronta retroalimentacion con el objetivo de mantener el mejoramiento continuo del módulo.

3. Los administradores de red podrían adaptar la propuesta que hemos detallado para generar reportes personalizados a sus necesidades, sin embargo, tendrían que familiarizarse con el ambiente distribuido que utiliza Hadoop para el procesamiento de datos.

[1] Por ejemplo, en: http://stackoverflow.com/questions/1245176/how-can-i-use-the-input-logs-pcapbinary-with-map-rreduce-hadoop.

REFERENCIAS BIBLIOGRÁFICAS

[1] Lance Spitzner, “Honeypots: Tracking Hackers”, Addison Wesley professional, 2002.

[2] Apache. Hadoop. http://lucene.apache.org/hadoop/, 2008

[3] Apache. Pig. http://incubator.apache.org/pig/, 2008

[4] Chu, L., Tang, H., Yang, T., and Shen, K. 2003. Optimizing data aggregation for cluster-based internet services. In Pro-ceedings of the Ninth ACM SIGPLAN Symposium on Prin-ciples and Practice of Parallel Programming, 2003.

[5] Jeffrey Dean, Sanjay Ghemawat: MapReduce: simplified data processing on large clusters. OSDI 2004: 137-149.

[6] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, Den-nis Fetterly: Dryad: distributed data-parallel programs from sequential building blocks. EuroSys 2007: 59-72.

[7] Fay Chang et al, Bigtable: a distributed storage system for structured data, OSDI 2006, 205-218

[8] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: The Google file system. SOSP 2003: 29-43.

[9] OpenSolaris. Hadoop. http://opensolaris.org/os/project/livehadoop/, 2008

[10] Herringshaw C., "Detecting attacks on networks", Diciembre 1997.

[11] Jeffrey Dean, Sanjay Ghemawat, "Simplified Data Processing on Large Clusters", Diciembre 2004.