# Sistema para Generar Gráficas a Partir de Logs Tcpdump usando Hadoop


[(1)]Ángel Cruz, [(2)]Pedro Torres, [(3)]MSc. Cristina Abad
Facultad de Ingeniería en Eléctrica y Computación
Escuela Superior Politécnica del Litoral
Campus "Gustavo Galindo V." Km. 30.5, Vía Perimetral, Apartado 09-01-5863, Guayaquil, Ecuador
[(1)] acruz@fiec.espol.edu.ec, [(2)] ptorres@fiec.espol.edu.ec, [(3)] cabad@fiec.espol.edu.ec


## Resumen


*El tamaño de los Logs a procesar es relativamente grande, por lo que se necesita un procesamiento masivo de estos datos, para ello utilizamos el esquema MapReduce a través del Framework Hadoop. Un sistema que genere gráficas a partir de un procesamiento y análisis de logs tcpdump. Analizamos los datos a ser procesados por el sistema, se realiza una breve descripción del formato de Logs así como la conversión de los mismos. Planteamos el diseño de la solución, las herramientas que utilizamos para desarrollar el sistema, además explica cómo trabaja las fases MapReduce en nuestra solución. Detallamos la implementación y el funcionamiento del sistema, se describe las pruebas realizadas en los servicios web de Amazon así como los resultados obtenidos. Finalmente, se presentan las conclusiones obtenidas con el desarrollo de sistema con el esquema MapReduce.*



**Palabras Claves:** *hadoop, mapreduce, log tcpdump.*


## Abstract


*The size of the logs to be processed is relatively large, so we need a massive processing of these data, we use the schema for MapReduce through Hadoop Framework. A system that generates graphs from a processing and analyzing tcpdump logs. We analyzed the data to be processed by the system, there is a brief description of the format of logs and convert them. We propose the design of the solution, the tools we use to develop the system, it explains how MapReduce works phases in our solution. We detail the implementation and operation of the system described tests on Amazon Web services and the results obtained. Finally, we present the conclusions obtained with the development of the scheme MapReduce system.*



**Keyword:** *hadoop, mapreduce, log tcpdump.*

# 1. Introducción

Los ataques constantes de spammers o crackers, hacia redes reales, han puesto en marcha la conformación de mecanismos que ayuden de alguna manera a reducir estos posibles ataques.

Uno de estos mecanismos son las denominadas honeypots de alta interacción o Honeynet, los cuales resultan ser anzuelos para posibles ataques que provienen del exterior. El estudio y análisis de estos ataques se convierten en una ventaja de seguridad, al tener conocimiento de la forma de actuar dichos intrusos, se podrán desarrollar políticas que apunten a tener un menor impacto en las redes reales.

Estos ataques son monitoreados en la red, que a su vez se van almacenando en registros o logs, formando una base de datos del tráfico. Estos datos masivos guardados en los logs, pueden ser de mucha ayuda, si se obtienen conocimiento de ellas a través de diferentes tipos de gráficos.

La complejidad del tratamiento de estos datos masivos, hacen que su procesamiento se vea afectado. Una alternativa es el paradigma MapReduce que permite realizar de manera eficiente este procesamiento masivo de logs, logrando de esta manera abarcar grandes cantidades de datos y poderlos representar gráficamente:

# 2. Descripción del problema

La capacidad de procesamiento y análisis de logs tcpdump del orden de Gigabytes en programas tradicionales tales como Wireshark a medida que la data comienza a incrementar de tamaño, los tiempos van siendo significativamente mayores y tiende a ponerse lento en el procesamiento. Además de estar limitada a la memoria RAM de la PC en la que se está ejecutando.

# 3. Marco referencial de trabajo

## 3.1. Paradigma MapReduce

MapReduce es un paradigma de programación propuesto por Google cuya implementación ha sido realizada para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadores (clústeres). Posterior a la publicación del trabajo de Google, han surgido varias otras implementaciones de MapReduce, utilizando diversos lenguajes de programación. En el presente proyecto utilizamos la versión libre con mayor aceptación en la actualidad: Apache Hadoop.

Para la implementación de MapReduce es necesario especificar dos funciones como son Map y Reduce.

### 3.1.1. Map

La función Map o Mapeo recibe un ítem de datos de entrada en forma de pares (k1, v1) y produce una lista de valores pares intermedios (k2, v2) por cada llamada a esta función. Posteriormente junta todos los valores que tengan una misma clave y los agrupa teniendo una lista de valores por cada clave.

Map (k1, v1) -> list (K2, v2)

### 3.1.2. Reduce

La función Reduce recibe la lista de valores y claves enviadas por el Mapeo, produce una colección de valores para cada dominio. Esto lo hace en cada llamada al Reduce, el retorno de todas esas llamadas se recoge como la lista de resultado deseado.

Reduce (K2, list (v2)) -> list (v2)

## 3.2. Framework Hadoop

Hadoop es una plataforma que nos permite desarrollar aplicaciones distribuidas que tengan que tratar con grandes cantidades de datos del orden de los Peta bytes. Hadoop provee escalabilidad además de trabajar muy bien en un clúster de servidores, ofrece ventajas de la programación distribuida aunque eso no signifique que el programador tenga que preocuparse por el paralelismo y tolerancia a fallos de las aplicaciones que desarrolla ya que Hadoop lo implementa.

## 3.3. Amazon Web Services

Para el desarrollo de nuestra aplicación utilizamos los servicios de Amazon Web Services que permiten, entre otras cosas, levantar clústeres Hadoop bajo demanda. A continuación detallamos los principales puntos del diseño:

- S3 (Simple Storage Service)

Es el servicio de almacenamiento masivo, transparente y de alta disponibilidad de Amazon. Aquí almacenamos nuestro dataset que son los Logs (input) y los resultados del proceso MapReduce (output).

- EC2 (Elastic Compute Cloud)

Este servicio de Amazon provee los recursos computacionales necesarios para correr nuestra aplicación MapReduce. En EC2 se implementa la computación paralela dado por Hadoop, donde los Map y Reduce se ejecutarán.

- Host

Es la máquina donde reside la aplicación gráfica donde se mostrará los gráficos.

- Request

Es el pedido que realiza el host a través del Internet a los servicios de Amazon. Cuando se requiere los datos para un gráfico en particular se genera un request.

- Response

Es la respuesta que el servicio de Amazon le provee al host a través del internet. En los request vienen los datos necesarios para un gráfico en particular.

Cuando la aplicación en el host reciba el response con los datos requeridos para realizar la gráfica, mediante la librería JFreeChart interpretamos los datos a gráficas.

- Elastic MapReduce

Es un servicio que ofrece Amazon en el que se puede programar la ejecución de tareas en EC2 y S3. Este servicio nos ayuda en la ejecución automática de los procesos MapReduce y la transferencia de archivos desde el S3 al EC2 y viceversa.

## 3.4. JNetStream

JNetStream es una librería de Java que nos permite capturar y enviar paquetes. Además se pueden desarrollar aplicaciones que capturen paquetes desde una interface de red, visualizarlos y analizarlos desde Java. La mayoría de los funciones de JNetStream están en el área de decodificar la estructura de los paquetes que están en representación binaria.

La librería JNetStream ha sido probada para Microsoft Windows (98/2000/XP/Vista), Linux (Fedora, Mandriva, Ubuntu), Mac OS X, FreeBSD, and Solaris, con lo cual se cubre un amplio espectro de sistemas operativos a utilizar.

## 3.5. JFreeChart - Framework para las graficas

JFreeChart es una librería para gráficos escrita en Java que facilita generación de gráficos de calidad profesional en nuestras aplicaciones, ya sean Web o de escritorio. Entre sus características principales tenemos:

- Un API consistente y bien documentado con soporte para un amplio rango de tipos de gráficas.
- Un diseño flexible fácilmente extendible, y la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) y de Cliente (Swing, por ejemplo).
- JFreeChart está distribuido bajo la licencia LGPL

# 4. Diseño e implementación de la solución

## 4.1. Diseño

El diseño de la solución está basado en dos aspectos importantes: Procesamiento del data set en Hadoop y la aplicación de los servicios de Amazon para realizar dicho procesamiento.

### 4.1.1 Procesamiento en Hadoop

Para el análisis de archivos pcap fue necesario realizar un pre-procesamiento con la finalidad de tener un data set de entrada en formato tal que lo podamos manejar en Hadoop. El pre-procesamiento fue realizado con la librería de java JNetStream, accedemos a los bits del archivo binario con el objetivo de recuperar información relevante que nos servirá para realizar las gráficas. Los archivos de texto plano generados después del pre-procesamiento son los nuevos datos de entrada para el sistema.

El diseño de la solución en Hadoop corresponde al diseño de las dos funciones principales: Map y Reduce que implementa el framework MapReduce.

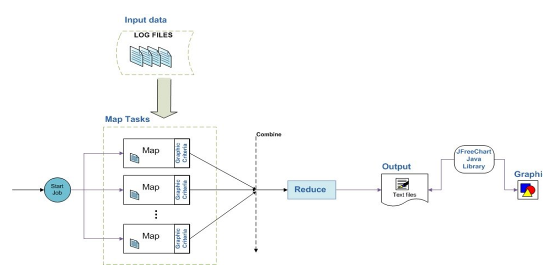


**Figura 1.** Diseño procesamiento Hadoop.

En la Figura 1 observamos el esquema general del procesamiento realizado en Hadoop.

A continuación detallaremos los aspectos más importantes del diseño.

- ***Proceso Map:*** Hadoop particiona el data set en splits que se los tratará como texto. Creamos nuestra propia clase MyPcap que es el contenido de un paquete de nuestro log. Lo que recibe el Map son la el tipo de protocolo (TCP, IP, ICMP, ARP) y el valor es el objeto MyPcap que contiene los atributos más importantes de un paquete (Figura 2).

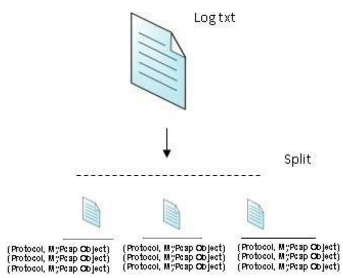


**Figura 2.** Split de un log en records (clave, valor)

La función de Mapeo recibe los pares de (Protocolo, MyPcap Object), sobre estos valores se extrae los datos dependiendo de los criterios del gráfico a mostrar. Cuando el usuario ejecuta su requerimiento se envía un parámetro que guarda el número de gráfico deseado. Se recupera este parámetro en el Map para saber qué criterios debe aplicar y de esa manera obtener los datos que se requieren. Estos datos requeridos son enviados en forma de (graphic parameter, one), donde la clave es él o los valores que se consideran para necesarios para realizar la gráfica y one corresponden un número 1 para denotar la concurrencia de la misma (Figura 3).

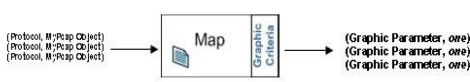


**Figura 3.** Diseño del Map a la solución

- ***Proceso Combine y Reduce:*** Los valores intermedios que produce el Map son ordenados por clave y una lista de valores que tienen en común a la clave. El proceso Reduce toma esta lista de valores con su clave y la cuenta teniendo la concurrencia total del mismo. Este proceso se realiza en los mismos nodos donde realiza el Map, y se le da el nombre de combine. El resultado del combine lo recibe el Reduce que realiza nuevamente la sumatoria y sus resultados son los finalmente los esperados (Figura 4).

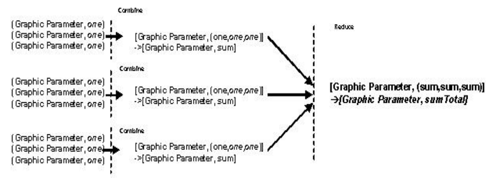


**Figura 4.** Diseño del combiner y reduce de la solución

### 4.1.2 Aplicación de Servicios de Amazon al procesamiento de Hadoop

Los servicios de Amazon nos proveen la infraestructura de clúster, en la cual podemos realizar el procesamiento en Hadoop. Como vimos anteriormente los servicios básicos que utilizaremos son S3 (Simple Storage Service), EC2 (Elastic Cloud Computing) y Elastic MapReduce.

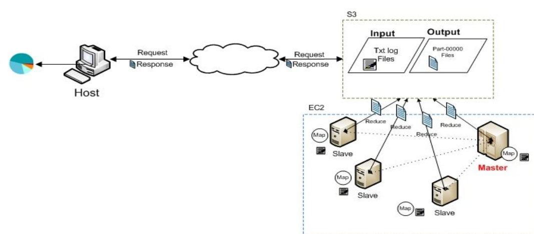


**Figura 5.** Diseño Aplicación Servicios de Amazon para la solución

En la Figura 5 podemos observar como interactúa el usuario final o host junto al S3 y EC2 de Amazon. En primer lugar los archivos de texto plano generados en el pre-procesamiento son cargados en el S3 en una carpeta a la que llamamos input. En cada petición o request que realiza el usuario cuando elige una gráfica en particular, se levanta un jobflow que no es nada más que una secuencia de pasos previamente configurados con Elastic MapReduce. Estos pasos son:

a. Copia de datos del S3 a los nodos del Ec2.
b. Procesamiento MapReduce con Hadoop en cada nodo del EC2.
c. Copia de resultados obtenidos en el EC2 al S3 a la carpeta llamada output.

Finalmente, se recupera esta información del S3 de la carpeta output a través de un response hacia la máquina donde se encuentra el usuario final. Este interpreta los resultados utilizando la librería JFreeChart que nos dará como resultado la gráfica solicitada.

### 4.2. Implementación

La creación del sistema consistió en la realización de algunas partes importantes como el pre-procesamiento de la data set, procesamiento MapReduce y la recuperación de los resultados para mostrar las gráficas a través de una GUI.

#### 4.2.1 Pre-Procesamiento

Se realizó un análisis exhaustivo al formato binario de los paquetes de los logs pcap. Esto nos sirvió para acceder a los bits necesarios que contienen los datos relevantes para nuestro análisis.

```
if (linkType.equals("Ethernet") == true) {

    int etherProtocol = in.readUnsignedShort();

    // Now check if its IP protocol
    if (etherProtocol == 0x800) {
        int version = in.readBits(4);
        int hlen = in.readBits(4);
        int precedence = in.readBits(3);
        int delay = in.readBits(1);
        int throughtput = in.readBits(1);
        int reliability = in.readBits(1);
        in.readBits(2);     // Reserved 2 bits

        int length = in.readUnsignedShort();
        int id = in.readUnsignedShort();
        in.readBits(1);     // Reserved 1 flag bit

        int doNotFragment = in.readBits(1);
        int moreFragments = in.readBits(1);
```

**Figura 6.** Acceso los paquetes con JNetStream

En la Figura 6 observamos como accedemos a los bits de los campos del paquete, dependiendo del tipo de protocolo accedemos a sus distintos campos. Imprimimos en un archivo de texto los campos que obtuvimos con el siguiente formato:

> Año, mes, día, hora, ip fuente, ip destino, puerto fuente, puerto destino, protocolo

Esta conversión se realizó debido a que su formato binario dificultaba de gran manera la manipulación en Hadoop. Es por eso que optamos por realizar una previa conversión de los archivos pcap a archivos de texto plano, el cual nos beneficia en el sentido de que ahora podemos manipular los datos de los paquetes dándoles un formato específico para nuestros intereses. No obstante realizar este proceso de conversión conllevaría más tiempo que tratar de procesar directamente con los archivos pcap.

#### 4.2.2 Procesamiento MapReduce

Este procesamiento consiste en la aplicación de las dos funciones que son Map y Reduce. Con la función Map realizamos la obtención de datos específicos para la gráfica. Obtenemos estos datos basándonos en el parámetro enviado –graphic #grafico. De esta forma nos aseguramos de obtener los datos correctos para la gráfica correcta. Los datos de entrada al Map son (Protocolo, MyPcap Object), donde Protocolo es la clave y está en Texto, y MyPcap Object es el objeto que creamos para almacenar los datos de un paquete, podemos decir que el objeto MyPcap es un paquete personalizado con información relevante y de interés a nuestros propósitos. En la Figura 7 se muestra la clase MyPcap en java donde muestra los datos del paquete que guarda.

```
public class MyPcap implements
Writable{
        private int anio;
        private String mes;
        private int dia;
        private String hora;
        private String src_ip;
        private String dst_ip;
        private String src_port;
        private String dst_port;
        private String protocolo;
```

**Figura 7.** Clase MyPcap en Java

Una vez que sabemos lo que recibe cada Map procedemos a obtener los datos. Para ello tenemos que saber los criterios que debemos considerar. Como hemos dicho los criterios depende de la solicitud del usuario, esta solicitud la sabemos obteniendo el valor del parámetro –graphic.

Finalmente el Map envía los datos necesarios que cumple los criterios. Estos son enviados en la forma (clave, valor), donde la clave es el o los parámetros requeridos (Figura 8) y el valor es un entero uno bajo el tipo de dato IntWritable.

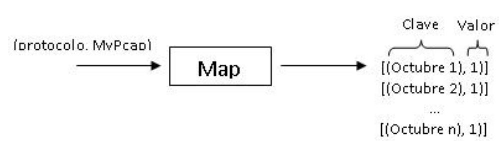


**Figura 8.** Implementación del Map en Hadoop

La fase de Reduce se encarga del conteo de los valores que recibe por parte del Map. Este conteo lo hace a través de una sumatoria de unos que tienen una clave en común. Cuando este Reduce se lo hace en el mismo nodo en que se hace el Map toma el nombre de combine. En la Figura 9 se muestra la clave y valor que recibe el Reduce y lo que produce como resultado.

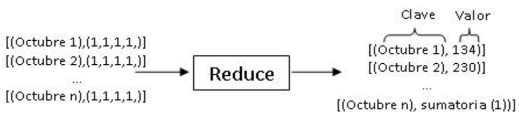


**Figura 9.** Implementación del Reduce en Hadoop

### 4.2.3 Creación de la GUI

Para el cumplimiento de la finalidad de nuestro proyecto como es el generar gráficamente reportes a partir de los logs tcpdump, realizamos un GUI (Graphical User Interface) que facilite al usuario final la interacción con los resultados obtenidos en Hadoop.

- ***Interfaz Desktop:*** Diseñamos una interfaz desktop para el manejo del Jobflow que nos provee Elastic MapReduce. Esto facilita al usuario final ya que no debe de preocuparse de claves y usuarios que son necesarios para acceder a los servicios de Amazon (Figura 10).

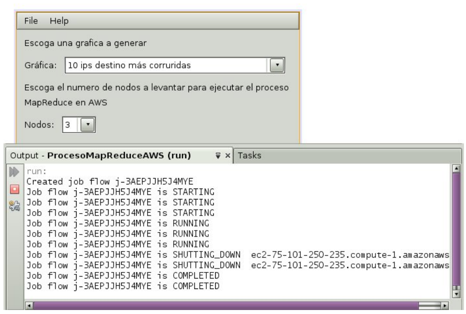


**Figura 10.** Interfaz Desktop en ejecución

- ***Interfaz Web:*** Una vez realizado la ejecución en la interfaz desktop, los resultados generados ya han sido guardados en una carpeta destinada para los mismos y que se encuentran en el S3 de Amazon. La interfaz web realiza una extracción de estos resultados desde el S3 y con la utilización de la librería gráfica JFreeChart generamos la gráfica correspondiente (Figura 11).


**Figura 11.** Interfaz Web de la Aplicación

## 5. Pruebas Realizadas

Para las pruebas que se realizaron en el sistema, se tomó en consideración dos factores importantes como son: el número de nodos que conforman el clúster y el tamaño de la data.

**Tabla 1.**Tiempos obtenidos con 1.4 GB de data

| | Tiempo | | | |
|---|---|---|---|---|
| **No. Nodos** | **Step 1 "S3 to HDFS"** | **Step 2 "MapReduce"** | **Step 3 "HDFS to S3"** | **Total** |
| **5** | 66 minutos | 62 segundos | 21 segundos | 67 min 23 seg |
| **8** | 66 minutos | 41 segundos | 21 segundos | 67 min 2 seg |
| **10** | 65 minutos | 35 segundos | 23 segundos | 65 min 58 seg |

En la Tabla 1 podemos observar que el tiempo total no tiene mucha diferencia, esto se debe a que vemos el tiempo de manera global. Es por eso que está separado en los tiempos que toma cada paso. Enfocándonos en el paso 2 que es el tiempo de MapReduce, observamos que a medida que aumenta el número de nodos el tiempo de ejecución disminuye. A continuación en la Tabla 2 podemos apreciar solo el tiempo de MapReduce.

**Tabla 2.**Tiempo del Step 2 "MapReduce"

Podemos notar la diferencia que existe en la utilización de varios nodos. Cuando usamos 5 nodos el tiempo fue de 62 segundos al duplicar el número de nodos el tiempo fue de 35 segundos, esta diferencia se debe a una de las características importantes del esquema MapReduce como es la programación paralela. La programación paralela hace que más datos se estén procesando al mismo tiempo puesto que existe mayor número de nodos, de ahí a que el tiempo de procesamiento tienda a disminuir conforme a la inclusión de más nodos.

## 6. Conclusiones

El inconveniente que presentan las herramientas para el análisis de tráfico de red, es la capacidad de procesar grandes cantidades de datos. Esto limita a que no se realice un análisis completo sobre estos datos. El esquema MapReduce es una alternativa a este inconveniente, ya que implementa programación paralela en la que puede ejecutar grandes cantidades de datos sobre un grupo de computadoras o clúster.

El sistema para generar gráficas a partir de logs tcpdump implementa el esquema MapReduce a través del Framework de Apache Hadoop. Este sistema realiza un análisis en todo el dataset, obteniendo información de mayor alcance que si lo hiciéramos con una herramienta común para este tipo de análisis. El sistema ha sido desarrollado como una aplicación web con una interfaz sencilla en la que el usuario final puede interactuar sin ningún problema. Las pruebas del sistema se realizaron utilizando los servicios web de Amazon, lo cual nos ayudó mucho ya que pudimos probar el sistema con varios nodos es decir en un clúster real.

El servicio Elastic MapReduce que provee Amazon nos ayudó en la automatización de tareas como crear instancias, ejecución del proceso MapReduce, copiar el dataset de S3 a EC2 y viceversa. De esta forma el usuario final no tiene que preocuparse en los comandos necesarios para utilizar los servicios de Amazon. La utilización de la librería gráfica JFreeChart nos proporcionó un buen manejo de los datos a mostrar, además de ofrecernos diferentes tipos de gráficos de acuerdo al tipo de análisis realizado.

Cabe recalcar que esta experiencia nos ayudó a fortalecer más los conocimientos adquiridos en la realización de aplicaciones web, así como la aplicación de nuevos conocimientos como el esquema MapReduce, los servicios de Amazon que sin ninguna duda nos servirán en el ámbito profesional.

## 7. Recomendaciones

Algunas de las grandes empresas como Google, Yazoo, IBM, entre otras, han apostado a la programación paralela como plataforma para muchos de sus servicios, esto nos da a entender que cada vez más va tomando importancia la implementación de servicios en este tipo de esquema. La utilización de nuevas alternativas de desarrollo resulta asequible por su bajo costo económico y su eficiencia en temas como el procesamiento masivo de datos.

El análisis que realizamos sobre la data es muy limitado y poco profundo, pero se puede realizar un análisis más exhaustivo en los datos como por ejemplo seguir un comportamiento de una dirección ip específica, para ello la inclusión de nuevas herramientas como datamining junto con el esquema Map reduce sería una alternativa muy buena.

## 8. Referencias


[1] Pazmiño, M. y Avilés, J. "Captura y Análisis de los ataques informáticos que sufren las redes de datos de la ESPOL, implantando una honeynet con miras a mejorar la seguridad informática en redes de datos del Ecuador". Tesis de ESPOL. Guayaquil, Marzo 2009.

[2] Dean, J. y Ghemawat, S. "MapReduce: Simplified Data Processing on Large Clusters". En Memorias del Sixth Symposium on Operating System Design and Implementation (OSDI 2004). San Francisco, CA-EE.UU., Diciembre, 2004.

[3] Thakar, U., Varma, S., Ramani, AK. "Honey Analyzer – Analysis and Extraction of Intrusion Detection Patterns & Signatures Using Honeypot". Presentado en The Second International Conference on Innovations in Information Technology, 2005.

[4] Nauta, K., Lieble, F. "Offline Network Intrusion Detection: Mining TCPDUMP Data to Identify Suspicious Activity". Presentado en AFCEA Federal Database Colloquium, San Diego, CA-EE.UU., Septiembre, 1999.

[5] Bernardczyk M. "JNetStream", http://jnetstream.com.

[6] The Apache Software Foundation. "Apache Hadoop", http://hadoop.apache.org.

[7] Universidad Autónoma de México. "Bitácora Honeynet UNAM", http://www.honeynet.unam.mx/es/reports.pl.