Implementación y Evaluación de una Herramienta Distribuida de Anonimización de Capturas de Red

Rigazio, Emilio; Morocho, David; Abad, Cristina Ms.Sc.
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

Resumen

En la actualidad las empresas tienen redes de computadoras alimentando sus sistemas operacionales, y sobre dichas redes viaja la hoy en día invaluable información. Dicho movimiento es en muchos casos el centro del negocio, fallas, retrasos, modificaciones o robo en ese movimiento de datos puede causar graves problemas. Debido a que los archivos de log mencionados contienen la información que ha viajado por la red en un periodo de tiempo, éstos se vuelven de delicado manejo, pues pueden contener información sensible representando riesgos de seguridad. Esto genera trabas en el análisis de dichos logs, pues las organizaciones evitan o prohíben el manejo de esos archivos, dificultando la posibilidad de generar mejores soluciones a las problemáticas que las redes presentan. Existe un proceso de anonimización que consiste en pre-procesar los archivos de log de modo que la información sensible de los mismos quede "enmascarada". Sin embargo, dicho procesamiento se vuelve extenso puesto que los archivos, dado que son capturas del tráfico moviéndose por una red, pueden llegar a ser muy pesados. Este trabajo busca resolver la problemática planteada mediante el procesamiento distribuido de datos, generando la posibilidad de anonimización de grandes archivos en menores tiempos.

Palabras claves: *anonimización, logs, procesamiento distribuido de datos.*

Abstract

Nowadays, companies have computer networks feeding their operational systems, on these networks travels the invaluable information. In some cases that movement is the center of the business. Failures, delays, modifications or robbery can cause serious problems in the mentioned data movement. Due to mentioned log files contain the information that has traveled through the network in a certain period of time, they become delicate to handle because they may contain sensible information which represents security risks. This obstructs the analysis of these logs, because the organizations avoid the management of these archives, difficulting the possibility to generate best solutions to the problems networks present. It exists an anonymization process that consists in the pre-processing of log files in order for the sensible information they contain could be hidden. However, the processing becomes extensive because data that we probably are working with might be very heavy. This work is looking forward to resolve the problem set out during the distributed data processing, generating the possibility of the anonymization of a large amount of data in less time.

Keywords: *anonymization, logs, distributed data processing.*

Introducción

En la actualidad, las empresas tienen redes de computadoras alimentando sus sistemas operacionales, y sobre dichas redes viaja la hoy en día invaluable información. Dicho movimiento es en muchos casos el centro del negocio, fallas, retrasos, modificaciones o robo en ese movimiento de datos puede causar graves problemas.

Es por esto que cada día más las empresas invierten en mantener sus redes, tanto a nivel operacional como a nivel de seguridad.

Parte de ésta inversión consiste en el análisis de tráfico de la red, realizado por especialistas que estudian el comportamiento de la red en base a archivos de log que son una representación de la información que viaja por la red (1).

Existe un proceso de anonimización que consiste en pre-procesar los archivos de log de modo que la información sensible de los mismos quede "enmascarada".

Sin embargo, dicho procesamiento se vuelve extenso puesto que los archivos, dado que son capturas del tráfico moviéndose por una red, pueden llegar a ser muy pesados.

Este trabajo busca resolver la problemática planteada mediante el procesamiento distribuido de datos, generando la posibilidad de anonimización de grandes archivos en menores tiempos.

1 Planteamiento

1.1 Definición del Problema

A medida que los sistemas computacionales se han convertido en el centro operativo de pequeñas y grandes empresas, y que dichos sistemas son cada día más interconectados, la investigación forense de incidentes de seguridad se ha vuelto necesaria, y cada vez más común (2).

Si bien es conocido que el intercambio de logs es útil e importante (3), es muy difícil concretarlo incluso entre organizaciones pequeñas debido a lo complejo que resulta establecer relaciones de confianza organizacionales.
Una solución a ésta problemática consiste en la eliminación de información sensible de los logs previo a la entrega de los mismos, sin embargo, no es una solución ideal puesto que dicha eliminación conlleva a su vez la eliminación de información fundamental para el análisis de seguridad requerido, y puede fácilmente dar como resultado archivos carentes de valor para el análisis final.

Toma importancia la anonimización de archivos, siendo éste un proceso de enmascaramiento de información, logrando que la información sensible contenida en logs se ofusque sin que esto implique perder en información valiosa para análisis de seguridad.

Sin embargo, la anonimización de logs implica un procesamiento muchas veces masivo de datos, puesto que las grandes organizaciones, por su alta carga transaccional, generan logs del orden de los Gigabytes en cortos periodos de tiempo.

1.2 Justificación

La anonimización de archivos de log es una necesidad creciente debido a los riesgos de seguridad que implican la compartición de logs para el análisis de los mismos puesto que la presencia de información sensible y/o confidencial es un hecho en estos archivos.

La alta transaccionalidad de un organismo puede resultar en archivos muy grandes, cuyo procesamiento de anonimización puede tardar mucho más tiempo del requerido, lo cual representa una traba a la investigación de logs deseada.

Con la implementación de un mecanismo expandible y escalable de anonimización de archivos PCAP, se espera generar una solución ideal a la problemática expuesta, sentando un precedente para mecanismos de anonimización no solo de archivos PCAP, sino de logs propios de cada organización.

1.3 Alcances

La solución propuesta proveerá mecanismos de anonimización a los campos IP (origen y destino) de archivos de captura de tráfico de red en formato PCAP.

Se implementaran los algoritmos más comunes de anonimización para campos IPv4, y se proveerá un mecanismo de personalización mediante un archivo de configuración.

Se desarrollarán librerías de algoritmos, las que serán cargadas en la solución dinámicamente, y será posible la expansión de algoritmos de anonimización a través de inclusión de librerías a modo de plugins, configurables también mediante el mecanismo de personalización.

2 Marco Teórico

2.1 PCAPS

Los archivos PCAP son logs que contienen información capturada del tráfico de una red permitiendo obtener información de las comunicaciones que en la misma ocurren.

Son esencialmente copias de estados de un sistema dentro de un punto en particular en el pasado, del tráfico ocurrido en una red. (4)

Los datos de los paquetes más comúnmente usados son fechas o time stamps, direcciones ip de origen y destino, direcciones MAC de origen y destino, y los datos propios de la comunicación.

Los archivos son binarios formateados prácticamente ubicuos, interpretados por aplicaciones especializadas como WireShark, jpcapdumper, omnipeek, etc. (5).

2.2 Map/Reduce

Map/Reduce es un esquema diseñado y desarrollado por colaboradores de Google para procesar y generar grandes cantidades de datos (6).

La fase Map genera duplas (clave, valor) de los datos de entrada generando una lista de valores intermedios.
Una fase combiner intermedia fusiona todos los valores asociados a una clave común y al finalizar la ejecución se obtiene una lista por cada clave.

En la fase Reduce se reciben las listas generadas, y se procesa la información generando los resultados finales según el requerimiento lo especifique

2.3 Hadoop

Hadoop es una plataforma que permite desarrollar y ejecutar aplicaciones distribuidas en grandes clústeres abstrayendo al desarrollador de los detalles de hardware, conexiones de red, distribución de datos, entre otros pormenores que la plataforma maneja y administra.

Hadoop cuenta también con un sistema distribuido de almacenamiento (HDFS por sus siglas en inglés) que permite el alojamiento de datos a procesar en los nodos.

Tanto el paradigma map/reduce como el HDFS fueron diseñados para ambientes propensos a fallas en los nodos, de modo que errores en los nodos no impliquen errores en el resultado de la ejecución. (6)

2.4 Algoritmos de anonimización

La anonimización consiste en el proceso de transformación de la información haciendo que la misma se vuelva en “no identificable” dentro de su entorno.

Existen algoritmos de anonimización de datos que en distintos grados buscan hacer que datos se conviertan en inidentificables por terceros, y para esto se basan en diversos modelos, dependiendo en su mayoría fuertemente del tipo de dato que se intenta anonimizar.

Los algoritmos deben tener un balance de seguridad y utilidad, pues generalmente cuando se anonimizan datos con algoritmos muy seguros, los datos de salida pierden mucha información que quizás no era necesario eliminar. En cambio, los algoritmos que conservan la mayor cantidad de información suelen ser los más fáciles de descifrar, y por tanto, no son seguros.

3 Análisis y Diseño

3.1 Análisis del problema

El primer problema es la necesidad de una herramienta expansible, que pueda incorporar nuevos algoritmos de manera dinámica, sin necesidad de modificar mantenimiento en el código base de la aplicación.

Este problema implica el desarrollo de un mecanismo dinámico de carga de librerías, y la capacidad de cargar parámetros de configuración para los algoritmos, de forma que la solución brinde la flexibilidad requerida para una anonimización parametrizable.

El segundo, y más fundamental problema, es la necesidad de procesar archivos de tamaños considerablemente grandes, y que en un paradigma común de un único nodo de procesamiento, requeriría muchas horas e incluso días de ejecución para completar.

Puesto que los archivos PCAP son un conjunto de registros de comunicación entre dos puntos, la solución debe ser capaz de procesar paralelamente la extracción, anonimización y reemplazo de valores, de modo que los tiempos de ejecución dependan no solo de la magnitud de datos a procesar, sino también del número de nodos trabajando paralelamente.

Entonces se tiene la necesidad de generar una solución escalable de procesamiento distribuido para la anonimización de archivos PCAP que a su vez pueda ser funcionalmente expandido mediante librerías que actúen en forma de plugins, y configurable mediante archivos de parámetros.

3.2 Diseño de la solución

Buscando una solución distribuida que sea robusta y con tolerancia a fallos, se tomó como plataforma la solución Hadoop, que brinda herramientas basadas en el modelo de programación Map/Reduce, y abstrae al programador de tareas de manejo de hardware, distribución de información y tolerancia a fallas.

Del análisis del problema obtuvimos que podemos dividir el proceso en 3 pasos: Extracción, Anonimización y Reemplazo.

Utilizando el modelo Map/Reduce, dividimos el problema en dos etapas de Mapeo y reducción, de manera que la primera etapa cubra la extracción de datos y anonimización de los mismos, generando tablas de reemplazo que contengan los valores originales, y sus valores de reemplazo.

Figura 3-1 Etapa 1 del diseño

Como se muestra en la figura, la fase de mapeo de ésta etapa realiza el trabajo de extraer los campos de interés que se desean anonimizar.

La fase de reducción de ésta etapa recibe de manera agrupada los valores que debe anonimizar, y a partir de la información especificada en el archivo de configuración, ejecuta los algoritmos indicados para cada campo.

Finalmente la fase de reducción genera tablas de reemplazo que contienen los valores originales y su valor anonimizado correspondiente.

La segunda etapa de mapeo y reducción utiliza las tablas generadas en la primera etapa, y realiza el reemplazo de valores, generando finalmente la salida anonimizada.

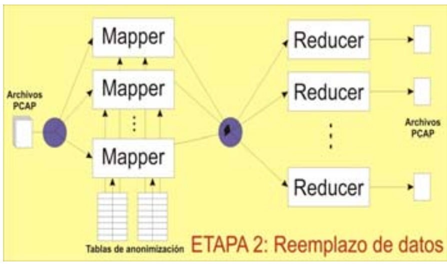

Figura 3-2 Etapa 2 del diseño

Como se observa en la figura, la fase mapper de ésta etapa recibe como datos de entrada, además de los archivos PCAP, las tablas de reemplazo o anonimización generadas en la etapa anterior.

En la fase mapper se obtienen los campos de interés, y se realiza el reemplazo de los valores utilizando la información de las tablas de reemplazo.

La fase de reducción de ésta etapa obtiene los datos anonimizados y los agrupa por archivo original de modo que se obtenga tantos archivos de salida como archivos de entrada se tuvieron. Estos archivos tendrán la información de los registros anonimizada.

4 Implementación

4.1 Implementación de algoritmos

Para la anonimización de logs existen algoritmos de variadas características que brindan diversos niveles de enmascaramiento, y los mismos están diseñados para adaptarse a ciertos tipos de campos.

Para la solución propuesta en éste trabajo, se seleccionaron los algoritmos de anonimización de campos IPv4 más comunes: 1) Black Marker, 2) Truncation, 3) Prefix-Preserving y 4) Random Permutation. (7)

La implementación de estos algoritmos puede variar de herramienta en herramienta, manteniendo el concepto fundamental de los mismos.

Se implementó estos algoritmos en forma de librerías, de modo que brinden su funcionalidad mediante llamadas a métodos públicos.

4.1.1 Black Marker

El algoritmo tiene como finalidad enmascarar una parte o todo un dato de modo que éste no pueda ser legible.

La implementación de este algoritmo se desarrolló de manera que se enmascare los bits indicados por un parámetro, el cual puede tomar los valores de 8, 16, 24 y 32. Adicionalmente se recibe un valor de blanqueamiento que puede ser 1 ó 0, y será dicho valor el que reemplace los valores a enmascarar.

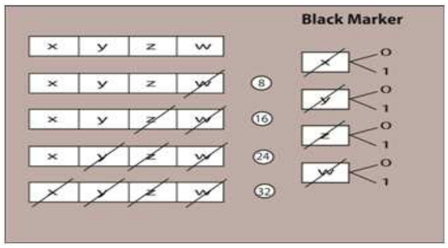

Figura 4-1 Implementación de algoritmo Black Marker

El algoritmo black marker fue desarrollado para campos numéricos, sin embargo el concepto aplica para otros tipos de campos.

4.1.2 Truncation

El algoritmo realiza un desplazamiento del dato, truncando la data que desborda del desplazamiento realizado. La implementación de este algoritmo se la realizó de modo binario y decimal de tal manera que el usuario pueda escoger que tipo de desplazamiento desea realizar.

En el modo binario, el desplazamiento se realiza a nivel de bits, de acuerdo a parámetros de entrada que indiquen la dirección y magnitud del desplazamiento.

En el modo decimal, el desplazamiento se realiza a nivel de dígitos, también considerando parámetros de entrada como el modo binario.

Puesto que la implementación del algoritmo contempla la anonimización de direcciones IP, el desplazamiento y truncamiento de desborde se hace octeto por octeto de manera individual.

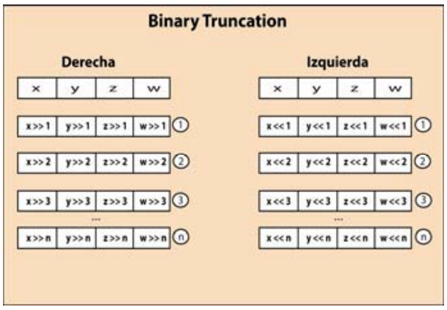

Figura 4-2 Implementación de algoritmo de Truncation Binario

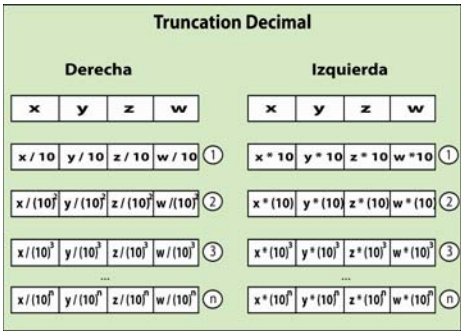

Figura 4-3 Implementación de algoritmo de Truncation Decimal

4.1.3 Prefix-Preserving

El algoritmo prefix-preserving es un algoritmo ideado para la anonimización de campos IP, y busca generar direcciones enmascaradas que no eliminen información de subredes, de modo que no se pierda importante información en el proceso de anonimización.

El algoritmo asegura que siempre que dos IPs tengan los mismos n primeros dígitos iguales, los valores anonimizados de esas IPs también tendrán n dígitos iniciales iguales.
Esto nos asegura (1) que las direcciones IP tienen una relación 1 a 1 con los valores anonimizados, y (2) se mantiene la información de subredes sin que se revele los valores reales de las mismas.

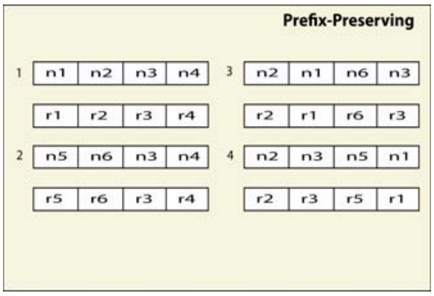

Figura 4-4 Implementación de algoritmo Prefix-Preserving

4.1.4 Random Permutation

El algoritmo realiza reemplazo de valores con valores generados de manera aleatoria, de modo que se obtengan valores de salida no mapeables a los valores de entrada.

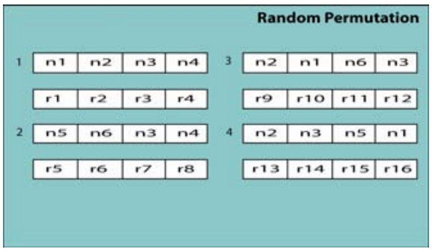

Figura 4-5 Implementación de algoritmo Random Permutation

4.2 Implementación de la herramienta distribuida

Para seguir el diseño previamente generado, se dividió la implementación en las etapas definidas en el diseño. Se explicará a continuación la implementación de las mismas.

El primer paso requerido para las ambas etapas fue la implementación de un mecanismo de lectura de archivos PCAP. Para esto, se reutilizó el código generado en un trabajo de graduación de la FIEC (8).

Con esto se pudo obtener un InputFormat que interpreta archivos PCAP, y genera información que alimenta las clases mappers de las dos etapas.

La implementación de la fase de mapeo de la primera etapa, donde se realiza la extracción de campos de interés se realizó utilizando los métodos provistos por la librería JNetStream (9) que brindan una interfaz para la obtención de valores de los campos de cada paquete obtenido.

Puesto que el alcance de éste trabajo es la anonimización de los campos IP, el método mapper obtiene los valores de los campos IP origen e IP destino, y genera duplas del tipo <[tipo de campo], [valor del campo]>, donde el primer elemento representa la clave, y el segundo elemento representa el valor.

```
Mapper1( key, value )
{
        paquete = obtenerPaquete(value)
        collect( "SrcIP", paquete.obtenerSrcIP( ) )
        collect( "DstIP", paquete.obtenerDstIP( ) )
}
```

Figura 4-6 Pseudocódigo del mapper (fase 1)

Estas duplas son agrupadas por claves por la plataforma Hadoop en una fase que es abstracta para el usuario y que es conocida como combiner (10). El resultado de esa combinación es pasado a las clases reducers.

El método reducer de la primera etapa es en donde se realiza la ejecución de la anonimización, y es por esto que en ésta fase se hace la lectura del archivo de configuración y se carga la personalización seleccionada por el usuario.

Mediante el mecanismo conocido como reflection (11), se carga dinámicamente las clases de las librerías de anonimización explicadas anteriormente, y sus parámetros son tomados del archivo de configuración.

```
# Nivel de logeo (debug | info)
loglevel message

# Configuracion de algoritmo para ip origen
# srcIP [nombre de clase]
srcIP truncation.Truncation

# Configuracion de algoritmo para ip destino
# dstIP [nombre de clase]
dstIP prefixpreserving.PrefixPreserving

# CONFIGURACION ALGORITMOS
# [nombre de clase] [nombre de parametro] [valor de parametro]

# Configuracion blackmarker
# blackmarker.BlackMarker [nom] [val]
blackmarker.BlackMarker bytesToMark 8
blackmarker.BlackMarker replaceVal 0

# Configuracion prefix preserving
# prefixpreserving.PrefixPreserving [nom] [val]
prefixpreserving.PrefixPreserving randomFx 1

# Configuracion random permutation
# randompermutation.RandomPermutation [nom] [val]

# Configuracion truncation
# truncation.Truncation [nom] [val]
truncation.Truncation numShifts 1
truncation.Truncation base 10
truncation.Truncation dir 1
```

Figura 4-7 Formato de archivo de configuración

La clase, reducer recibe una colección del tipo <[clave], {valor1, valor2, … valor n}> donde la clave representa el tipo de campo que la clase reducer va a anonimizar (en éste caso IP origen o destino), y la lista de valores con las direcciones IP obtenidas en la fase anterior.

Una vez cargados los valores, se ejecuta la anonimización de los mismos, y se produce un archivo de salida donde se definen los pares <[valor original] , [valor anonimizado]>. Esto concluye la primera etapa, y genera las tablas de reemplazo requeridas en la siguiente etapa.

```
Reducer1( key, listaDeValores )
{
        collect( "Tabla de anonimizacion de", key )
        listaTemporal = { }
        algoritmo = config.getAlgoritmo( key )
        params = config.getParams( algoritmo )

        while( listaDeValores.hasNext( ) )
        {
                ip = listaDeValores.next( )
                if( ! listaTemporal.contiene( ip|)
                {
                        ipAnon = anonimizar( algoritmo, params, ip)
                        collect( ip, ipAnon)
                }
        }
}
```

Figura 4-8 Pseudocódigo del reducer (fase 1)

La implementación de la segunda etapa inicia utilizando nuevamente el InputFormat utilizado en la etapa previa.

En la fase mapper de ésta etapa realiza una carga de los archivos o tablas generadas en la etapa anterior para utilizar dichos datos en el reemplazo de valores.

Los paquetes de red son leídos, y como en la fase mapper de la etapa anterior, se obtiene los valores de los campos IP origen y destino, los mismos que son buscados en su respectiva tabla de reemplazo, y se generan duplas de la forma <[identificador de archivo], [data del paquete con campos anonimizados]>.

```
Mapper2( key, value )
{
        tablaMapeoSrc = cargarTabla("src")
        tablaMapeoDst = cargarTabla("dst")

        paquete = obtenerPaquete(value)
        ipSrcAnon =
tablaMapeoSrc.getValue(paquete.obtenerSrcIP( ) )
        paquete.reemplazarSrc(ipSrcAnon)

        ipDstAnon =
tablaMapeoDst.getValue(paquete.obtenerDstIP( ) )
        paquete.reemplazarDst(ipDstAnon)

        collect( getOriginFilename() , paquete )
}
```

Figura 4-8 Pseudocódigo del mapper (fase 2)

La fase reducer de ésta etapa realiza la generación de archivos de salida que son finalmente presentados como resultado de la anonimización de los archivos de entrada.

Para esto, la fase hace una recuperación de los datos del registro, y recolecta la información que finalmente es escrita en los archivos de salida.

Puesto que cada instancia de ésta fase reducer escribe un archivo de salida, es necesario que se instancien tantos reducers como archivos de entrada se leen.

5 Resultados

5.1 Metodología de las pruebas

Con el objetivo de evaluar la herramienta propuesta y desarrollada, se planificó un conjunto de pruebas que permitan un análisis de la solución.

Puesto que para ejecutar la anonimización distribuida se tiene dos variables (número de nodos del clúster y tamaño total de la carga a anonimizar), se planteó hacer dos pruebas, haciendo en cada una un muestreo de una variable, dejando la otra con un valor fijo.

En cada prueba se procedió a levantar un clúster en los AWS con el número de nodos requerido para la prueba, y se ejecutó en paralelo 4 trabajos de anonimización. Cada trabajo realizó la anonimización de cargas iguales.

Se procedió a medir el tiempo del trabajo de mayor duración, y se tomó dicho valor como el tiempo de ejecución de la prueba.

Se realizaron 10 repeticiones de cada prueba, y se promedió el valor para el análisis correspondiente. Se calculó así mismo la desviación estándar del muestreo tomado para cada prueba.

5.2 Análisis de Resultados

El primer tipo de prueba se realizó modificando el valor de nodos en el clúster, ejecutando en él 4 trabajos de anonimización con una carga de 4GB distribuidos entre los 4 trabajos.

Se realizó pruebas con 1, 2, 5, 10 y 15 nodos, y los resultados (en minutos) se muestran en la figura 5 1.

Figura 5-1 Promedio de pruebas con número de nodos variable

De la figura 5-1, se puede observar que existe una notable mejora en los tiempos de anonimización cuando se paso de 1 nodo a 2 y posteriormente a 5 nodos. El tiempo promedio de ejecución con 1 nodo (50 minutos) se redujo en un 38% al utilizar 2 nodos en el clúster de procesamiento, y en un 50% al utilizar 5 nodos en el clúster.

Sin embargo, pasar de 5 a 10 nodos solo mejoró en un 16% el tiempo de anonimización, y finalmente subir a 15 nodos mejoró en un 8% el tiempo logrado con 10 nodos.

Se observa entonces que entre 5 y 10 nodos parece ser un número adecuado de nodos para un buen rendimiento sin necesidad de un clúster muy numeroso.

El segundo tipo de prueba se realizó modificando la carga proporcionada a los trabajos de anonimización, manteniendo un clúster de 10 nodos a partir del análisis de los resultados de las pruebas anteriores.

Se realizaron pruebas con cargas de 130MB, 410MB, 1GB, 2GB, 4GB y 12GB repartidos en cuatro trabajos corriendo en el mismo clúster simultáneamente.

Los resultados de las pruebas (en minutos) se resumen en la Figura 5-2.

Figura 5-2 Resultados promedio de pruebas con carga variable

Se observa que existe un fuerte aumento en los tiempos promedio de ejecución cuando se incrementa la carga hasta de información a ser anonimizada hasta 2 GB.

Sin embargo es interesante notar que el tiempo de ejecución para la anonimización de 4 GB es apenas 4% menor que el tiempo de ejecución para la anonimización de 12 GB.

Esto indica que la solución presenta alta escalabilidad, pues mayores cargas implican mínimos aumentos de los tiempos totales de procesamiento.

Así mismo se analizó la efectividad de los resultados de la ejecución de la anonimización en las pruebas antes expuestas, con el afán de dar validez a la solución.

Se observó un promedio de 90% de efectividad en la anonimización de registros. El 10% corresponde a problemas puntuales que se presentaron en ciertas ejecuciones de las pruebas.

5.3 Comparación de tiempos con herramientas existentes

Con el fin de medir el comportamiento obtenido con la herramienta frente a soluciones no distribuidas de anonimización de logs, se realizó una prueba de anonimización utilizando la herramienta FLAIM (7).

Se ejecutaron anonimización de un archivo de 4 GB utilizando la herramienta, y se promedió un tiempo de ejecución 40 minutos.

La Figura 5-3 muestra de manera visual la mejora en los tiempos de procesamiento que se obtuvo con la herramienta distribuida.

Figura 5-3 Tiempos de ejecución de herramientas

6 Trabajo Futuro

6.1 Escritura de archivo PCAP

Se investigó buscando librerías de escritura de archivos PCAP, y se realizó pruebas preliminares con las librerías jpcap (12) y jnetpcap (13), las que se basan en la librería libpcap (Linux) o winpcpap (Windows).

Sin embargo, no se pudo completar el desarrollo de dicha funcionalidad, y en cambio se implementó la escritura de archivos de texto que contienen la información básica de los paquetes, luego de la anonimización realizada.

Si bien dichos archivos de texto brindan la información requerida para análisis de comportamientos y características de redes, habrá quienes requieran utilizar herramientas especializadas, en cuyo caso se deseará la información en formato PCAP.

Es por esto que se propone como trabajo futuro el desarrollo de un mecanismo de escritura de archivos PCAP a partir de los datos anonimizados, que debe ser incorporado en la fase Reducer de la segunda etapa de la herramienta presentada, tal como se define en el diseño de la misma.

6.2 Anonimización de otros campos

Si bien el alcance del presente documento definía la anonimización de campos IPv4, es posible realizar anonimización de otros campos de los paquetes capturados, con el fin de mejorar el resultado final y brindar mayor funcionalidad al usuario.

Existe una gran cantidad de protocolos de comunicación que son usados en las redes actuales, y cada uno cuenta con diversos campos con información que en ciertos casos puede ser catalogada como sensible.

Se propone generar funcionalidad adicional para la anonimización de campos de interés en diversos protocolos, de acuerdo a las necesidades planteadas por los potenciales usuarios.

6.3 Anonimización de archivos de log personalizados

Si bien el presente trabajo presenta una solución de anonimización de archivos de captura de tráfico de red, no es muy difícil proyectar la investigación realizada fuera de los límites aparentes.

Así por ejemplo, sería muy sencillo implementar mecanismos similares para la anonimización de otros archivos de log que pudiesen contener información sensible.

Los logs de la mayoría de aplicaciones son un conjunto de registros ordenados compuestos cada uno de campos con información puntual.

A partir de dicho concepto, el mecanismo propuesto en éste trabajo es fácilmente aplicable a la anonimización de archivos de log, y puede ser personalizado de acuerdo al tipo de log que se desee anonimizar.

Conclusiones

1. La anonimización de archivos de captura de red es un proceso que puede ser extenso cuando se procesa secuencialmente archivos de gran tamaño.

2. Aplicar un modelo distribuido de anonimización genera soluciones altamente escalables, y la tolerancia a fallos puede ser controlada sin problemas.

3. Se generó una herramienta que aprovecha el modelo de programación distribuida para lograr los objetivos planteados, y los resultados de las pruebas demuestran la mejora frente a otra solución, además de evidenciar la potencial escalabilidad.

4. Se pudo comprobar las ventajas de trabajar en ambientes distribuidos para el procesamiento masivo de datos, pues los tiempos de ejecución pueden ser drásticamente reducidos, y la capacidad de procesar grandes cantidades de datos supera a la de ambientes no distribuidos.

5. Existen servicios comerciales que brindan Infraestructura como un Servicio (IaaS por sus siglas en inglés), donde se oferta la capacidad de generar y trabajar con clústers customizables de

manera remota, abstrayendo al usuario de los temas relacionados con hardware.

Recomendaciones

1. Mejorar las capacidades de la herramienta propuesta mediante la funcionalidad de anonimización de otros campos de los protocolos más comunes o de interés.

2. Implementar un mecanismo de escritura de archivos PCAP de salida con la información anonimizada, de modo que la misma pueda ser visualizada en herramientas especializadas.

3. Generar módulos que permitan la compatibilidad con otros formatos de archivos de capturas de red existentes.
4. Implementar nuevos algoritmos de anonimización que puedan ser ejecutados en la herramienta propuesta.

5. Expandir el ámbito de anonimización a otros tipos de logs utilizando el modelo planteado en éste trabajo.

Bibliografía

[1] Seigneur, J y Slagell, A. "Collaborative Computer Security and Trust Management". 2009: pp. 64-80.

[2] Markoff, J., Bergman, L. "Internet Attack is called Broad and Long Lasting. New York Times", 2005.

[3] Slagell, A., Yurcik, W. "Sharing Computer Network Logs for Security and Privacy: A Motivation for New Methodologies of Anonymization". Atenas, Grecia : SECOVAL: The Workshop on the Value of Security through Collaboration, 2005.

[4] Schiller, C., Fogie, S., DeRodeff, C., y Gregg, M. "Infosecurity 2008 Threat Analysis". s.l. : Syngress Publishing 2007: pp. 314-315.

[5] Oppleman, V. "Network Defense". Hakin9 #17 2006: pp.14-25.

[6] Dean, J. y Ghemawat, S. "MapReduce: Simplified Data Processing on Large Clusters". En Memorias del Sixth Symposium on Operating System Design and Implementation. San Francisco, CA-EE.UU. OSDI 2004: pp. 137-149.

[7] Slagell, A, Lakkaraju, K y Luo, K. "FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs". Presentado en el 20th Large Installation System Administration Conference. LISA 2006: pp 101-108.

[8] Cayetano, D., Rivadeneira, C. "Módulo de generación de reportes gráficos de una honeynet a partir de los logs tcpdumps". Guayaquil, Ecuador : FIEC - ESPOL, 2009.

[9] Bednarczyk, B. "Jnetstream User Guide". [Online] Febrero 2, 2010. http://jnetstream.sourceforge.net/docs/userguide.html.

[10] White, T. "Hadoop: The Definitive Guide". s.l. : O'Reilly Media, Inc., June 2009: 175-186.

[11] "Introduccion al API Reflection de Java". [Online] [Cited: Febrero 2, 2010.] http://www.javahispano.org/contenidos/es/introduccion_al_api_reflection_reflexion_de_java.

[12] Jpcap. "A Java library for capturing and sending network packets". [Online] http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html.

[13] Sly Technologies. "jNetPcap OpenSource | a Libpcap/WinPcap Wrapper". [Online] http://jnetpcap.com.

[14] Ghemawat, J. Dean y S. "Mapreduce: Simplied data processing on large clusters". Presentadoen el USENIX Symposium on Operating Systems Design & Implementation. (OSDI) 2004: pp. 137-147.